STANISŁAW OSOWSKI

SIECI NEURONOWE DO PRZETWARZANIA INFORMACJI

OFICYNA WYDAWNICZA POLITECHNIKI WARSZAWSKIEJ
WARSZAWA 2000

СТАНИСЛАВ ОСОВСКИЙ

НЕЙРОННЫЕ СЕТИ ДЛЯ ОБРАБОТКИ ИНФОРМАЦИИ

Перевод с польского
И.Д. Рудинского

МОСКВА
"ФИНАНСЫ И СТАТИСТИКА"
2002

УДК 004.032.26
ББК 32.813
О-75

РЕЦЕНЗЕНТ
доктор технических наук
И.Б. Фоминых

Осовский С.
О-75 Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.

ISBN 5-279-02567-4

Представлены важнейшие разделы теории искусственных нейронных сетей. Основное внимание уделяется алгоритмам обучения и их применению для обработки измерительной информации. Дается детальный обзор и описание важнейших методов обучения сетей различной структуры, иллюстрируемые численными экспериментами с практически подтвержденными результатами.

Для аспирантов и научных работников, интересующихся методами искусственного интеллекта. Может быть полезна специалистам в области информатики, статистики, физики и технических дисциплин, а также специалистам биомедицинских отраслей знаний.

О $\frac{1402070000-150}{010(01)-2002}$ 256–2002

УДК 004.032.26
ББК 32.813

ISBN 83-7207-187-X (Польша)
ISBN 5-279-02567-4 (Россия)

Содержание

К читателю 9

Предисловие к русскому изданию 13

Предисловие 16

1. ВВЕДЕНИЕ 17

1.1. Биологические основы функционирования нейрона 19
1.2. Первые модели нейронной сети 20
1.3. Прикладные возможности нейронных сетей 22

2. МОДЕЛИ НЕЙРОНОВ И МЕТОДЫ ИХ ОБУЧЕНИЯ 25

2.1. Персептрон 26
2.2. Сигмоидальный нейрон 27
2.3. Нейрон типа "адалайн" 32
2.4. Инстар и оутстар Гроссберга 34
2.5. Нейроны типа WTA 37
2.6. Модель нейрона Хебба 40
2.7. Стохастическая модель нейрона 44

3. ОДНОНАПРАВЛЕННЫЕ МНОГОСЛОЙНЫЕ СЕТИ СИГМОИДАЛЬНОГО ТИПА 46

3.1. Однослойная сеть 47
3.2. Многослойный персептрон 50
3.2.1. Структура персептронной сети 50
3.2.2. Алгоритм обратного распространения ошибки 51
3.3. Потоковые графы и их примененне для генерации градиента 55
3.4. Градиентные алгоритмы обучения сети 60
3.4.1. Основные положения 60
3.4.2. Алгоритм наискорейшего спуска 62
3.4.3. Алгоритм переменной метрики 63
3.4.4. Алгоритм Левенберга-Марквардта 65
3.4.5. Алгоритм сопряженных градиентов 67
3.5. Подбор коэффициента обучения 68
3.6. Эвристические методы обучения сети 71
3.6.1. Алгоритм Quickprop 71
3.6.2. Алгоритм RPROP 73
3.7. Сравнение эффективности алгоритмов обучения 73
3.8. Элементы глобальной оптимизации 75
3.8.1. Алгоритм имитации отжига 78
3.8.2. Генетические алгоритмы 81
3.9. Методы инициализации весов 86

4. ПРОБЛЕМЫ ПРАКТИЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ 89

4.1. Предварительный подбор архитектуры сети ... 89
4.2. Подбор оптимальной архитектуры сети ... 92
4.2.1. Способность к обобщению ... 92
4.2.2. Методы редукции сети с учетом чувствительности ... 98
4.2.3. Методы редукции сети с использованием штрафной функции ... 103
4.3. Методы наращивания сети ... 105
4.4. Подбор обучающих выборок ... 105
4.5. Добавление шума в обучающие выборки ... 107
4.6. Прнмеры использования персептроиной сети ... 110
4.6.1. Распознавание и классификация образов ... 110
4.6.2. Нейронная сеть для сжатия данных ... 116
4.6.3. Идентификация динамических объектов ... 121
4.6.4. Прогнозирование нагрузок энергетической системы ... 124

5. РАДИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ 129

5.1. Математические основы ... 130
5.2. Радиальная нейронная сеть ... 132
5.3. Методы обучения радиальных нейроиных сетей ... 137
5.3.1. Применеиие процесса самоорганизации для уточнения параметров радиальных функций ... 139
5.3.2. Вероятностиый алгоритм подбора параметров радиальных функций ... 142
5.3.3. Гибридный алгоритм обучения радиальных сетей ... 144
5.3.4. Алгоритмы обучения, основанные иа обратном распространении ошибки ... 146
5.4. Пример использования радиальной сети ... 149
5.5. Методы подбора количества базисных функций ... 151
5.5.1. Эвриетические методы ... 151
5.5.2. Метод ортогонализации Грэма-Шмидта ... 152
5.6. Сравнение радиальных и сигмоидальных сетей ... 157

6. СПЕЦИАЛИЗИРОВАННЫЕ СТРУКТУРЫ НЕЙРОННЫХ СЕТЕЙ 159

6.1. Сеть каскадной корреляции Фальмана ... 159
6.2. Сеть Вольтерри ... 163
6.2.1. Структура и особенности обучения сети ... 166
6.2.2. Примеры использования сети Вольтерри ... 169

7. РЕКУРРЕНТНЫЕ СЕТИ КАК АССОЦИАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА 176

7.1. Введение ... 176
7.2. Автоассоциативная сеть Хопфилда ... 178
7.2.1. Основные зависимости ... 178
7.2.2. Режим обучения сети Хопфилда ... 180
7.2.3. Режим распознавания сети Хопфилда ... 182
7.3. Сеть Хемминга ... 184
7.4. Сеть типа BAM ... 189
7.4.1. Описание процесса функционирования сети ... 189
7.4.2. Модифицированный алгоритм обучения сети BAM ... 191
7.4.3. Модифицированная структура сети BAM ... 192

8. РЕКУРРЕНТНЫЕ СЕТИ НА БАЗЕ ПЕРСЕПТРОНА 200

8.1. Введение ... 200
8.2. Персептронная сеть с обратной связью ... 200
8.2.1. Структура сети RMLP ... 200
8.2.2. Алгоритм обучения сети RMLP ... 202
8.2.3. Подбор коэффициента обучения ... 204
8.2.4. Коэффициент усиления сигнала ... 204
8.2.5. Результаты компьютерного моделирования ... 205
8.3. Рекуррентная сеть Эльмана ... 210
8.3.1. Структура сети ... 210
8.3.2. Алгоритм обучения сети Эльмана ... 212
8.3.3. Обучение с учетом момента ... 214
8.3.4. Пример компьютерного моделирования сети Эльмана ... 215
8.4. Сеть RTRN ... 219
8.4.1. Структура сети и алгоритм обучения ... 219
8.4.2. Результаты вычислительных экспериментов ... 221

9. СЕТИ С САМООРГАНИЗАЦИЕЙ НА ОСНОВЕ КОНКУРЕНЦИИ 226

9.1. Отличительные особенности сетей с самоорганизацией на основе конкуренции ... 227
9.1.1. Меры расстояния между векторами ... 228
9.1.2. Нормализация векторов ... 229
9.1.3. Проблема мертвых нейронов ... 230
9.2. Алгоритмы обучения сетей с самоорганизацией ... 231
9.2.1. Алгоритм Кохонена ... 233
9.2.2. Алгоритм нейронного газа ... 233
9.2.3. Сравнение алгоритмов самоорганизации ... 235
9.3. Сеть восстановления одно- и двумерных данных ... 238
9.4. Восстановление Сэммона ... 241
9.5. Применение сетей с самоорганизацией ... 242
9.5.1. Компрессия данных ... 243
9.5.2. Диагностирование неисправностей оборудования ... 246
9.5.3. Краткосрочное прогнозирование нагрузок энергетической системы ... 249
9.6. Гибридная сеть ... 252

10. СЕТИ С САМООРГАНИЗАЦИЕЙ КОРРЕЛЯЦИОННОГО ТИПА 257

10.1. Энергетическая функция корреляционных сетей ... 257
10.2. Нейронные сети PCA ... 259
10.2.1. Математическое введение ... 259
10.2.2. Определение первого главного компонента ... 264
10.2.3. Алгоритмы определения множества главных компонентов ... 265
10.3. Нейронные ICA-сети Херольта-Джуттена ... 267
10.3.1. Предварительные пояснения ... 267
10.3.2. Статистическая независимость сигналов ... 268
10.3.3. Рекуррентная структура разделяющей сети ... 269
10.3.4. Алгоритм Херольта-Джуттена для рекуррентной сети ... 270
10.3.5. Обобщенный алгоритм обучения рекуррентной сети ... 272
10.3.6. Однонаправленная сеть для разделения сигналов ... 274

11. МАТЕМАТИЧЕСКИЕ ОСНОВЫ НЕЧЕТКИХ СИСТЕМ **279**

11.1. Операции на нечетких множествах ... 281
11.2. Меры нечеткости нечетких множеств ... 283
11.3. Нечеткость и вероятность ... 284
11.4. Нечеткие правила вывода ... 286
11.5. Системы нечеткого вывода Мамдани-Заде ... 287
11.5.1. Фуззификатор ... 290
11.5.2. Дефуззификатор ... 293
11.5.3. Модель Мамдани-Заде как универсальный аппроксиматор ... 294
11.6. Модель вывода Такаги-Сугено-Канга ... 295

12. НЕЧЕТКИЕ НЕЙРОННЫЕ СЕТИ **299**

12.1. Структура нечеткой сети TSK ... 299
12.2. Структура сети Ванга-Менделя ... 303
12.3. Гибридный алгоритм обучения нечетких сетей ... 304
12.4. Применение алгоритма самоорганизации для обучения нечеткой сети ... 308
12.4.1. Алгоритм нечеткой самоорганизации *C-means* ... 308
12.4.2. Алгоритм пикового группирования ... 310
12.4.3. Алгоритм разностного группирования ... 312
12.4.4. Алгоритм нечеткой самоорганизации Густафсона-Кесселя ... 313
12.4.5. Сеть с нечеткой самоорганизацией в гибридной структуре ... 318
12.4.6. Примеры реализации нечетких сетей ... 321
12.5. Адаптивный алгоритм самоорганизации нечеткой сети ... 327

Литература ... **330**

Предметный указатель ... **340**

К читателю

Основные тенденции развития кибернетики в начале третьего тысячелетия можно выразить двумя словами: биологизация и гибридизация. Под *биологизацией* чаще всего понимается построение и исследование моделей поведения сложных объектов и способов управления ими на основе имитации механизмов, реализованных Природой в живых существах. Такой подход обусловлен тем фактом, что многие так называемые "классические" методы обработки информации воспринимаются в настоящее время как простейшие реализации универсальных способов функционирования биологических объектов. В качестве примеров можно привести последовательные алгоритмы фон Неймана (вырожденный случай параллельной обработки информации), а также двоичную логику (частный случай нечеткой логики). С другой стороны, стремительное увеличение вычислительных мощностей и развитие математического аппарата позволили подступиться к решению таких задач, размерность которых еще 5 – 10 лет назад была непреодолимым барьером для исследователя.

Гибридизация, в свою очередь, состоит в совместном применении различных методов и/или моделей для обработки информации об одном и том же объекте Парадигма такого подхода основана на согласии с тем, что любая сколь угодно сложная искусственная модель реального объекта всегда будет примитивнее и проще оригинала, и только многоаспектное его изучение с последующей интеграцией получаемых результатов позволит обрести необходимые знания или приблизиться к оптимальному решению. Гибридный подход давно и эффективно используется в научных исследованиях (вспомним понятия корпускулярно-волнового дуализма, микро- и макроэкономические исследования одних и тех же хозяйственных систем и т.п.).

Предлагаемая вниманию уважаемого читателя книга видного польского ученого С. Осовского удачно иллюстрирует названные тенденции в одной из наиболее динамично развивающихся областей современной теории интеллектуальных вычислений (англ.: *computational intelligence*), связанной с построением и применением искусственных нейронных сетей. Сформированные в рамках этого направления многослойные сетевые модели, в качестве прототипа которых используются структуры и механизмы функционирования биологических нервных систем, все более серьезно рассматриваются в качестве методологического базиса для создания сверхскоростных технических устройств параллельной обработки информации.

Представленный в книге материал можно рассматривать с трех точек зрения. Во-первых, это своего рода справочник по наиболее важным моделям нейронных сетей, написанный очень точным и корректным с математической точки зрения языком. Обширная библиография дополнительно обогащает семантику книги, превращая ее в своего рода путеводитель по первоисточникам. Во-вторых, это конспект лекций, стиль и характер которого свидетельствуют о богатом педагогическом опыте и мастерстве автора – профессора Варшавского политехнического университета. Большое количество примеров практической реализации сетей и их детальный анализ делают настоящее издание незаменимым учебным пособием для всех изучающих, преподающих и применяющих теорию искусственных нейронных сетей. В-третьих, книга содержит результаты собственной научной деятельности автора, который в силу личной скромности умалчивает о том, что многие описываемые в ней методы, структуры и алгоритмы (в частности, методы генерации градиента в многослойной сети на основе потоковых и сопряженных графов, метод обучения персептронной сети по алгоритму переменной метрики BFGS с направленной минимизацией, гибридный алгоритм обучения радиальных сетей, алгоритм обучения многослойного персептрона с обратной связью RMLP, гибридная сеть с самоорганизующейся и MLP-компонентой, структура и гибридный алгоритм обучения нечеткой сети TSK с самоорганизацией нейронов, а также многие другие) известны мировой научной общественности по имени их создателя С. Осовского.

В разделе 1 обсуждаются биологические основы функционирования нейрона, базовая модель нейрона МакКаллока–Питса, а также виды межнейронных взаимодействий, позволяющие строить искусственную нейронную сеть.

В разделе 2 представлены наиболее широко применяемые в настоящее время модели нейронов, в том числе модель персептрона, модель сигмоидального нейрона, адалайн, модель Хебба, инстар и оутстар, а также модель WTA. Рассматриваются важнейшие алгоритмы обучения, основанные как на обучении с учителем, так и на самоорганизации.

Раздел 3 посвящен однонаправленным сигмоидальным сетям, чаще всего применяющимся на практике. Обсуждаются градиентные алгоритмы обучения, реализующие метод обратного распространения ошибки для ее минимизации при генерации вектора градиента. Представлены элементы глобальной оптимизации методами имитации отжига и генетических алгоритмов.

В разделе 4 рассматриваются проблемы практического использования неоднородных сетей, в том числе выбора оптимальной архитектуры сети, принципов формирования обучающих выборок и методов повышения эффективности обобщения. Представлены примеры использования таких сетей для задач распознавания и классификации образов, сжатия сигналов, идентификации динамических объектов и прогнозирования временных рядов.

Раздел 5 посвящен сетям с радиальными базисными функциями (RBF). Рассматриваются типовые сетевые структуры и алгоритмы обучения, основанные на обучении с учителем и на самоорганизации конкурирующих нейронов.

В разделе 6 описываются специализированные сети, структура которых проектируется исходя из необходимости достижения конкретных целей. К ним относятся сеть каскадной корреляции Фальмана и сеть Вольтерри. Особенно интересна неоднородная структура сети Вольтерри, предназначенной для адаптивной обработки сигналов в реальном времени.

В разделе 7 содержатся сведения о рекуррентных сетях, используемых в качестве ассоциативных запоминающих устройств. Представлено описание структур и методов обучения сетей Хопфилда, Хемминга и сети типа BAM. Их функционирование иллюстрируется на численных примерах.

Раздел 8 посвящен рекуррентным сетям, построенным на неоднородных персептронах, в частности рассматриваются сеть RMLP, сеть Эльмана и сеть RTRN Вильямса–Зиспера. Теоретические рассуждения подкрепляются многочисленными примерами компьютерного моделирования этих сетей.

В разделе 9 обсуждаются сети с самоорганизацией на основе конкуренции нейронов. Приводятся описания алгоритмов обучения таких сетей и принципы их применения для решения задач распознавания и классификации образов, сжатия сигналов и прогнозирования временных рядов.

В разделе 10 рассматриваются вопросы корреляционного обучения (по Хеббу). Представлены два вида нейронных сетей, в которых реализовано обобщенное правило Хебба: сети PCA, осуществляющие анализ главных компонентов, и сети ICA Херольта–Джуттена для слепого разделения сигналов. Обсуждаются важнейшие алгоритмы обучения этих сетей и принципы их практического использования.

Разделы 11 и 12 посвящены новейшим достижениям в теории нейронных сетей, связанным с применением нечеткой логики. В одиннадцатом разделе представлены математические основы нечетких систем, необходимые для понимания деятельности таких сетей. В двенадцатом разделе рассматриваются базовые структуры и принципы функционирования нечетких сетей, использующих для адаптации параметров методы обучения как с учителем, так и на основе самоорганизации.

Несколько слов о применяемой терминологии. Причины лингвистической неоднозначности, которой грешат многие русскоязычные публикации, кроются в разрозненности отечественных научных центров, занимающихся исследованием и применением искусственных нейронных сетей, а также в отсутствии русскоязычного периодического издания национального масштаба, целиком посвященного этой проблематике и способного формировать

"терминологическую политику". При переводе монографии С. Осовского мы старались соблюдать принцип стилистического единства текста (особенно в плане использования профессиональных англицизмов), а также максимально корректно применять правила англо-русской транскрипции и транслитерации. В необходимых случаях наряду с используемыми в книге переводными терминами приводятся их наиболее распространенные русифицированные аналоги.

Издание рассчитано на читателя, обладающего определенной математической подготовкой. Предполагается, что он знаком с основными понятиями линейной алгебры, дифференциального исчисления, теории оптимизации, теории вероятности и теории множеств.

Хочется верить, что публикация книги С. Осовского на русском языке станет еще одним стимулом для повышения эффективности как проводимых, так и планируемых в России исследований в области нейронных сетей.

И. Рудинский,
кандидат технических наук

Предисловие к русскому изданию

Нейронные сети – это раздел искусственного интеллекта, в котором для обработки сигналов используются явления, аналогичные происходящим в нейронах живых существ. Важнейшая особенность сети, свидетельствующая о ее широких возможностях и огромном потенциале, состоит в параллельной обработке информации всеми звеньями. При громадном количестве межнейронных связей это позволяет значительно ускорить процесс обработки информации. Во многих случаях становится возможным преобразование сигналов в реальном времени. Кроме того, при большом числе межнейронных соединений сеть приобретает устойчивость к ошибкам, возникающим на некоторых линиях. Функции поврежденных связей берут на себя исправные линии, в результате чего деятельность сети не претерпевает существенных возмущений.

Другое не менее важное свойство – способность к обучению и обобщению накопленных знаний. Нейронная сеть обладает чертами искусственного интеллекта. Натренированная на ограниченном множестве данных сеть способна обобщать полученную информацию и показывать хорошие результаты на данных, не использовавшихся в процессе обучения.

Характерная особенность сети состоит также в возможности ее реализации с применением технологии сверхбольшой степени интеграции. Различие элементов сети невелико, а их повторяемость огромна. Это открывает перспективу создания универсального процессора с однородной структурой, способного перерабатывать разнообразную информацию.

Использование перечисленных свойств на фоне развития устройств со сверхбольшой степенью интеграции (VLSI) и повсеместного применения вычислительной техники вызвало в последние годы огромный рост интереса к нейронным сетям и существенный прогресс в их исследовании. Создана база для выработки новых технологических решений, касающихся восприятия, искусственного распознавания и обобщения видеоинформации, управления сложными системами, обработки речевых сигналов и т.п. Искусственные нейронные сети в практических приложениях, как правило, используются в качестве подсистемы управления или выработки решений, передающей исполнительный сигнал другим подсистемам, имеющим иную методологическую основу. Функции, выполняемые сетями, подразделяются на несколько групп:

аппроксимация; классификация и распознавание образов; прогнозирование; идентификация и оценивание; ассоциативное управление.

Аппроксимирующая сеть играет роль универсального аппроксиматора функции нескольких переменных, который реализует нелинейную функцию вида $y = f(x)$, где x – входной вектор, а y – реализованная функция нескольких переменных. Множество задач моделирования, идентификации, обработки сигналов удается сформулировать в аппроксимационной постановке.

Для классификации и распознавания образов сеть накапливает в процессе обучения знания об основных свойствах этих образов, таких, как геометрическое отображение структуры образа, распределение главных компонентов (PCA), или о других характеристиках. При обобщении акцентируются отличия образов друг от друга, которые и составляют основу для выработки классификационных решений.

В области прогнозирования задача сети формулируется как предсказание будущего поведения системы по имеющейся последовательности ее предыдущих состояний. По информации о значениях переменной x в моменты времени, предшествующие прогнозированию, сеть вырабатывает решение о том, чему должно быть равно оцениваемое значение исследуемой последовательности в текущий момент времени.

В задачах управления динамическими процессами нейронная сеть выполняет, как правило, несколько функций. Во-первых, она представляет собой нелинейную модель этого процесса и идентифицирует его основные параметры, необходимые для выработки соответствующего управляющего сигнала. Во-вторых, сеть выполняет функции следящей системы, отслеживает изменяющиеся условия окружающей среды и адаптируется к ним. Она также может играть роль нейрорегулятора, заменяющего собой традиционные устройства. Важное значение, особенно при управлении роботами, имеют классификация текущего состояния и выработка решений о дальнейшем развитии процесса.

В задачах ассоциации нейронная сеть выступает в роли ассоциативного запоминающего устройства. Здесь можно выделить память автоассоциативного типа, в которой взаимозависимости охватывают только конкретные компоненты входного вектора, и память гетероассоциативного типа, с помощью которой сеть определяет взаимосвязи различных векторов. Даже если на вход сети подается вектор, искаженный шумом либо лишенный отдельных фрагментов данных, то сеть способна восстановить полный и очищенный от шумов исходный вектор путем генерации соответствующего ему выходного вектора.

Различные способы объединения нейронов между собой и организации их взаимодействия привели к созданию сетей разных типов. Каждый тип сети, в свою очередь, тесно связан с соответствующим методом подбора весов межнейронных связей (т.е. обучения). Среди множества существующих видов сетей в качестве важнейших можно выделить многослойный персептрон, радиальные сети RBF, сети с самоорганизацией в результате конкуренции нейронов, сети с самоорганизацией корреляционного типа, а также рекуррентные

сети, в которых имеются сигналы обратной связи. Особую разновидность составляют нечеткие нейронные сети, функционирование которых основано на принципах нечеткой логики.

Интересным представляется объединение различных видов нейронных сетей между собой, особенно сетей с самоорганизацией и обучаемых с учителем. Такие комбинации получили название "гибридные сети". Первый компонент – это сеть с самоорганизацией на основе конкуренции, функционирующая на множестве входных сигналов и группирующая их в кластеры по признакам совпадения свойств. Она играет роль препроцессора данных. Второй компонент – в виде сети, обучаемой с учителем (например, персептронной), сопоставляет входным сигналам, отнесенным к конкретным кластерам, соответствующие им заданные значения (*постпроцессинг*). Подобная сетевая структура позволяет разделить фазу обучения на две части: вначале тренируется компонент с самоорганизацией, а потом – сеть с учителем. Дополнительное достоинство такого подхода заключается в снижении вычислительной сложности процесса обучения, а также в лучшей интерпретации получаемых результатов.

В предлагаемой вниманию русскоязычного читателя книге уделяется внимание важнейшим перечисленным выше типам искусственных нейронных сетей, методам их обучения и практического использования при решении конкретных задач обработки информации. По сравнению с польским изданием она дополнена описанием ряда новейших алгоритмов, в частности, в разделе о нечетких нейронных сетях.

Автор выражает благодарность издательству "Финансы и статистика" за издание книги в России, И.Д. Рудинскому за перевод ее на русский язык, а также издательству Варшавского политехнического университета "Oficyna wydawnicza" за предоставленные иллюстративные материалы. Хотелось бы выразить надежду, что изложенные в книге материалы будут полезными для лучшего понимания проблематики искусственных нейронных сетей и их применения для обработки информации.

С. Осовский
Варшава,
июль, 2001г.

Предисловие

Книга "Нейронные сети для обработки информации" – это оригинальное изложение новейших достижений в области искусственных нейронных сетей и их практических приложений. Оно представляет собой расширенную и в значительной степени модифицированную версию более ранних изданий "Нейронные сети" ("Sieci neuronowe" Oficyna Wydawnicza PW, 1994) и "Алгоритмическое описание нейронных сетей" ("Sieci neuronowe w ujeciu algorytmicznym" WNT, Warszawa, 1996). Нейронные сети получили широкую популярность в научной среде благодаря способности относительно легко адаптироваться к различным отраслям знаний. Они обладают свойствами, необходимыми для различных практических приложений: предоставляют универсальный механизм аппроксимации, адекватный многомерным массивам данных, способны обучаться и адаптироваться к изменяющимся условиям окружающей среды, могут обобщать полученные знания и на этой основе считаются системами искусственного интеллекта. Базис функционирования таких сетей составляют алгоритмы обучения, позволяющие оптимизировать весовые коэффициенты.

В книге представлены важнейшие разделы теории искусственных нейронных сетей. Основное внимание уделяется алгоритмам обучения и их применению для обработки измерительной информации. Приводятся детальный обзор и описание важнейших методов обучения сетей различной структуры, иллюстрируемый численными экспериментами с практически подтвержденными результатами.

Книга предназначена для студентов старших курсов, аспирантов и научных работников, интересующихся методами теории искусственного интеллекта. С учетом междисциплинарного характера обсуждаемой тематики она может оказаться полезной как для специалистов в области информатики, статистики, физики и технических дисциплин, так и для биомедицинских отраслей знаний. Благодаря обсуждению и базовых и производных понятий искусственных нейронных сетей, книга будет одинаково полезной как для начинающих, так и для профессионалов в этой предметной области.

Автор выражает признательность Издательству WNT за разрешение использовать фрагменты книги "Sieci neuronowe w ujeciu algorytmicznym" при подготовке настоящей монографии.

Автор

Раздел 1

ВВЕДЕНИЕ

1.1. Биологические основы функционирования нейрона

Тематика искусственных нейронных сетей относится к междисциплинарной сфере знаний, связанных с биокибернетикой, электроникой, прикладной математикой, статистикой, автоматикой и даже с медициной [16, 46, 51, 77, 113, 152, 182]. Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ. Они представляют собой попытку использования процессов, происходящих в нервных системах, для выработки новых технологических решений.

Нервная клетка, сокращенно называемая *нейроном*, является основным элементом нервной системы. Изучение механизмов функционирования отдельных нейронов и их взаимодействия принципиально важно для познания протекающих в нервной системе процессов поиска, передачи и обработки информации. С этой точки зрения представляется необходимым построить и изучить модель биологического нейрона.

Как и у любой другой клетки, у нейрона имеется тело со стандартным набором органелл, называемое *сомой*, внутри которого располагается ядро [152]. Из сомы нейрона выходят многочисленные отростки, играющие ключевую роль в его взаимодействии с другими нервными клетками. Можно выделить два типа отростков: многочисленные тонкие, густо ветвящиеся *дендриты* и более толстый, расщепляющийся на конце *аксон* (рис. 1.1).

Входные сигналы поступают в клетку через *синапсы*, тогда как выходной сигнал отводится аксоном через его многочисленные нервные окончания, называемые *колатералами*. Колатералы контактируют с сомой и дендритами других нейронов, образуя очередные синапсы. Очевидно, что синапсы, подключающие к клетке выходы других нейронов, могут находиться как иа дендритах, так и непосредственно на теле клетки.

Передача сигналов внутри нервной системы – это очень сложный электрохимический процесс. С большим упрощением можно считать, что передача нервного импульса между двумя клетками основана на выделении особых химических субстанций, называемых *нейромедиаторами*, которые формируются под

влиянием поступающих от синапсов раздражителей. Эти субстанции воздействуют на клеточную мембрану, вызывая изменение ее энергетического потенциала, причем величина этого изменения пропорциональна количеству нейромедиатора, попадающего на мембрану.

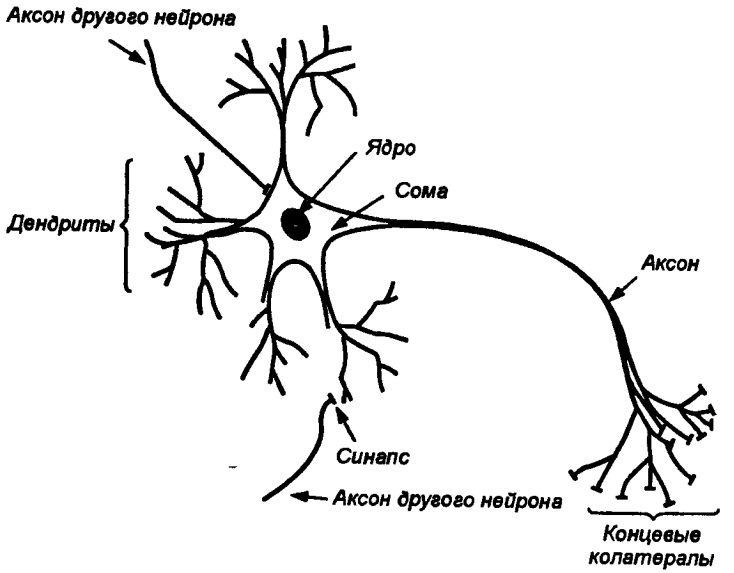

Рис. 1.1. Упрощенная структура биологической нервной клетки

Синапсы отличаются друг от друга размерами и возможностями концентрации нейромедиатора вблизи своей оболочки. По этой причине импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут возбуждать ее в разной степени. Мерой возбуждения клетки считается уровень поляризации ее мембраны, зависящий от суммарного количества нейромедиатора, выделенного на всех синапсах.

Из сказанного следует, что каждому входу клетки можно сопоставить численные коэффициенты (веса), пропорциональные количеству нейромедиатора, однократно выделяемого на соответствующем синапсе. В математической модели нейрона входные сигналы должны умножаться на эти коэффициенты для того, чтобы корректно учитывать влияние каждого сигнала на состояние нервной клетки. Синапсические веса должны быть натуральными числами, принимающими как положительные, так и отрицательные значения. В первом случае синапс оказывает возбуждающее, а во втором – тормозящее действие, препятствующее возбуждению клетки другими сигналами. Таким образом, действие возбуждающего синапса может моделироваться положительным значением синапсического веса, а действие тормозящего синапса – отрицательным значением.

В результате поступления входных импульсов на конкретные синапсы и высвобождения соответствующих количеств нейромедиатора происходит определенное электрическое возбуждение нервной клетки. Если отклонение от

состояния электрического равновесия невелико либо если баланс возбуждений и торможений является отрицательным, клетка самостоятельно возвращается в исходное состояние, и на ее выходе какие-либо изменения не регистрируются. В этом случае считается, что уровень возбуждения клетки был ниже порога ее срабатывания. Если же сумма возбуждений и торможений превысила порог активации клетки, значение выходного сигнала начинает лавинообразно нарастать, принимая характерный вид нервного импульса (рис. 1.2), пересылаемого аксоном на другие нейроны, подключенные к данной клетке. Величина этого сигнала не зависит от степени превышения порога. Клетка действует по

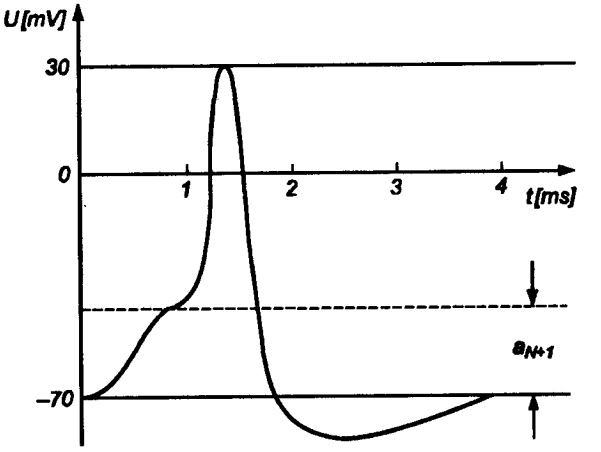

Рис. 1.2. Типичная форма нервного импульса

принципу "все или ничего". После выполнения своей функции нейромедиатор удаляется. Механизм удаления заключается либо во всасывании этой субстанции клеткой, либо в ее разложении, либо в удалении за пределы синапса.

Одновременно с генерацией нервного импульса в клетке запускается процесс рефракции. Он проявляется как стремительное возрастание порога активации клетки до значения "плюс бесконечность", в результате чего сразу после генерации импульса нейрон теряет способность вырабатывать очередной сигнал даже при сильном возбуждении. Такое состояние сохраняется в течение времени Δt_r, называемого периодом абсолютной рефракции. По окончании этого срока наступает период относительной рефракции Δt_w, за который порог срабатывания возвращается к первоначальному значению. В это время клетку можно активировать, но только с приложением более сильных возбуждений. В естественных процессах, как правило, выполняется отношение $\Delta t_w >> \Delta t_r$.

Количество взаимодействующих друг с другом нервных клеток чрезвычайно велико. Считается, что человеческий мозг содержит около 10^{11} нейронов [152], каждый из которых выполняет относительно примитивные функции суммирования весовых коэффициентов входных сигналов и сравнения полученной суммы с пороговым значением. Каждый нейрон имеет свои веса и свое пороговое значение. Они определяются местонахождением нейрона и решаемой им задачей и

могут интерпретироваться аналогично содержимому локальной памяти процессора.

Громадное количество нейронов и межнейронных связей (до 1000 входов в каждый нейрон) приводит к тому, что ошибка в срабатывании отдельного нейрона остается незаметной в общей массе взаимодействующих клеток. Нейронная сеть проявляет высокую устойчивость к помехам – это "стабильная" сеть, в которой отдельные сбои не оказывают существенного влияния на результаты ее функционирования. Таково главное отличие нейронных систем от обычных электронных систем, созданных человеком. Следует подчеркнуть, что ни одна современная технология не позволяет построить искусственную нейронную сеть, близкую по масштабам к нейронной сети мозга. Однако изучение и копирование биологических нервных систем позволяют надеяться на создание нового поколения электронных устройств, имеющих аналогичные характеристики.

Другая важная особенность нервных систем – высокая скорость их функционирования, несмотря на относительно длительный цикл срабатывания каждой отдельной клетки, измеряемый в миллисекундах и показанный на рис. 1.2. Она достигается благодаря параллельной обработке информации в мозге огромным количеством нейронов, соединенных многочисленными межнейронными связями. Такие операции, как распознавание образов и звуков либо принятие решений, выполняются человеческим мозгом за промежутки времени, измеряемые миллисекундами. Достижение такого результата при использовании полупроводниковой технологии VLSI все еще выходит за границы современных технических возможностей, хотя цикл срабатывания отдельных исполнительных элементов СБИС является достаточно коротким и имеет порядок 10^{-8} с. Если удастся, взяв за образец нервную систему, создать устройство с высокой степенью параллельности выполнения независимых операций, то скорость его функционирования может быть существенно увеличена и приближена к уровню, наблюдаемому в процессах обработки информации биологическими объектами.

1.2. Первые модели нейронной сети

Из приведенных выше рассуждений следует, что каждый нейрон можно считать своеобразным процессором: он суммирует с соответствующими весами сигналы, приходящие от других нейронов, выполняет нелинейную (например, пороговую) решающую функцию и передает результирующее значение связанным с ним нейронам. В соответствии с действующим правилом "все или ничего" в простейших моделях нейронов выходной сигнал принимает двоичные значения: 0 или 1. Значение 1 соответствует превышению порога возбуждения нейрона, а значение 0 – возбуждению ниже порогового уровня.

В одной из первых моделей нейрона, называемой моделью МакКаллока–Питса (предложенной в 1943 г.), нейрон считается бинарным элементом [98, 135]. Структурная схема этой модели представлена на рис. 1.3. Входные сигналы

x_j (j = 1, 2, ..., N) суммируются с учетом соответствующих весов w_{ij} (сигнал поступает в направлении от узла i к узлу j) в сумматоре, после чего результат сравнивается с пороговым значением w_{i0}. Выходной сигнал нейрона y_i определяется при этом зависимостью

$$y_i = f\left(\sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0} \right). \qquad (1.1)$$

Аргументом функции выступает суммарный сигнал $u_i = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0}$. Функция $f(u_i)$ называется функцией активации. В модели МакКаллока-Питса это пороговая функция вида

$$f(u) = \begin{cases} 1 & \text{для } u > 0 \\ 0 & \text{для } u \le 0 \end{cases}. \qquad (1.2)$$

Коэффициенты w_{ij}, присутствующие в формуле (1.1), представляют веса синаптических связей. Положительное значение w_{ij} соответствует возбуждающим синапсам, отрицательное значение w_{ij} – тормозящим синапсам, тогда как $w_{ij} = 0$ свидетельствует об отсутствии связи между i-м и j-м нейронами. Модель МакКаллока–Питса – это дискретная модель, в которой состояние нейрона в момент $(t + 1)$ рассчитывается по значениям его входных сигналов в предыдущий момент t. Построение дискретной модели обосновывается проявлением рефракции у биологических нейронов, приводящей к тому, что нейрон может изменять свое состояние с конечной частотой, причем длительность периодов бездействия зависит от частоты его срабатывания.

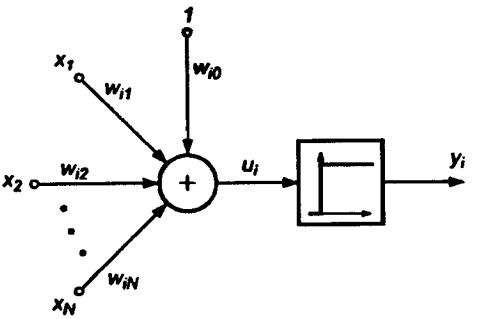

Рис. 1.3. Модель нервной клетки по МакКаллоку–Питсу

Через несколько лет Д. Хебб в процессе исследования ассоциативной памяти предложил теорию обучения (подбора весов w_{ij}) нейронов. При этом он использовал наблюдение, что веса межнейронных соединений при активации нейронов могут возрастать. В модели Хебба приращение веса Δw_{ij} в процессе обучения пропорционально произведению выходных сигналов y_i и y_j нейронов, связанных весом w_{ij}:

$$w_{ij}(k + 1) = w_{ij}(k) + \eta y_i(k) y_j(k), \qquad (1.3)$$

где k означает номер цикла, а η - это коэффициент обучения.

В начале 60-х годов Б. Видроу [167] предложил теоретическое обоснование и сформулировал принципы практической реализации адаптивных устройств обработки сигналов, что стало существенным вкладом в развитие нейронных сетей, функционирующих в режимах "онлайн" и "оффлайн".

В 1962 г. была опубликована книга Ф. Розенблатта [135], в которой представлена теория динамических нейронных систем для моделирования мозговой деятельности, основанная на персептронной модели нервной клетки. В этой теории использовалось представление нейрона моделью МакКаллока-Питса, в которой функция активации принимала двоичные значения 0 и 1.

Ограниченные возможности одиночного персептрона и составляемых из таких элементов одноуровневых сетей подверглись критике в книге М. Минского и С. Пейперта [101], что вызвало резкое снижение финансирования этой сферы научных исследований и привело в результате к замедлению развития искусственных нейронных сетей. Только отдельные научные группы, сконцентрированные вокруг таких ученых, как Гроссберг, Видроу, фон дер Мальсбург, Амари, Фукушима и Кохонен, продолжали работу в этой области. И только бурное развитие в 80-х годах технологии производства полупроводниковых устройств сверхвысокой степени интеграции (VLSI) привело к резкому возрастанию интереса к средствам параллельной обработки информации, которыми считаются и искусственные нейронные сети. Начиная с опубликованных в 1982 г. работ Дж. Хопфилда [53], теория нейронных сетей развивается в стремительном темпе, а количество научных центров, занимающихся этой междисциплинарной сферой знаний, непрерывно увеличивается. Доработка или, точнее, повторное открытие принципа обратного распространения [51] в применении к обучению многослойных сетей сняли те ограничения, которые стали главным объектом критики в книге М. Минского и С. Пейперта. Масштабное увеличение финансирования этой научной отрасли предопределило существенный прогресс как в теории, так и в практических приложениях. С учетом взрывного развития вычислительных систем это создало базу для реализации новых технологических решений в сфере технического распознавания образов, восприятия и объяснения, в управлении сложными системами, для обработки речевых сообщений и т.п. В настоящее время искусственные нейронные сети представляют собой высокоразвитую (особенно в теоретическом аспекте) отрасль знаний.

1.3. Прикладные возможности нейронных сетей

Любая нейронная сеть используется в качестве самостоятельной системы представления знаний, которая в практических приложениях выступает, как правило, в качестве одного из компонентов системы управления либо модуля принятия решений, передающих результирующий сигнал на другие элементы, не связанные непосредственно с искусственной нейронной сетью. Выполняемые сетью функции можно распределить на несколько основных групп: аппроксимации и интерполяции; распознавания и классификации образов; сжатия данных; прогнозирования; идентификации; управления; ассоциации.

В каждом из названных приложений нейронная сеть играет роль универсального аппроксиматора функции от нескольких переменных [1, 56],

реализуя нелинейную функцию

$$y = f(x), \tag{1.4}$$

где x – это входной вектор, а y – реализация векторной функции нескольких переменных. Постановки значительного количества задач моделирования, идентификации и обработки сигналов могут быть сведены именно к аппроксимационному представлению.

Для классификации и распознавания образов сеть обучается важнейшим их признакам, таким, как геометрическое отображение точечной структуры изображения, относительное расположение важнейших элементов образа, компоненты преобразования Фурье и другие подобные факторы. В процессе обучения выделяются признаки, отличающие образы друг от друга, которые и составляют базу для принятия решений об отнесении образов к соответствующим классам.

При решении задач прогнозирования роль нейронной сети состоит в предсказании будущей реакции системы по ее предшествующему поведению. Обладая информацией о значениях переменной x в моменты, предшествующие прогнозированию $x(k–1)$, $x(k–2)$, ..., $x(k–N)$, сеть вырабатывает решение, каким будет наиболее вероятное значение последовательности $\hat{x}(k)$ в текущий момент k. Для адаптации весовых коэффициентов сети используются фактическая погрешность прогнозирования $\varepsilon = x(k) - \hat{x}(k)$ и значения этой погрешности в предшествующие моменты времени.

При решении задач идентификации и управления динамическими процессами нейросеть, как правило, выполняет несколько функций. Она представляет собой нелинейную модель этого процесса, обеспечивающую выработку соответствующего управляющего воздействия. Сеть также выступает в роли следящей системы, адаптирующейся к изменяющимся условиям окружающей среды. Очень большое значение, особенно при управлении роботами, имеет функция классификации, реализуемая при выработке решения о дальнейшем развитии процесса.

В задачах ассоциации нейронная сеть играет роль ассоциативного запоминающего устройства (ЗУ). Можно выделить ЗУ автоассоциативного типа, с помощью которых определяется корреляция между отдельными компонентами одного и того же входного вектора, и ЗУ гетероассоциативного типа, средствами которых устанавливается корреляция между двумя различными векторами. Если на вход сети подается неструктурированный вектор (например, содержащий искаженные шумом компоненты или вообще не содержащий отдельные компоненты), нейронная сеть сможет восстановить оригинальный и очищенный от шумов вектор и сгенерировать при этом полную версию ассоциированного с ним вектора.

Важнейшее свойство нейронных сетей, свидетельствующее об их огромном потенциале и широких прикладных возможностях, состоит в параллельной обработке информации одновременно всеми нейронами. Благодаря этой способности при большом количестве межнейронных связей достигается

значительное ускорение процесса обработки информации. Во многих ситуациях становится возможной обработка сигналов в реальном масштабе времени.

Очень большое количество межнейронных соединений приводит к тому, что сеть становится нечувствительной к ошибкам, возникающим в отдельных контактах. Функции поврежденных соединений принимают на себя другие элементы, в результате в деятельности сети не наблюдаются заметные нарушения. Это свойство используется, в частности, при поиске оптимальной архитектуры нейронной сети путем разрыва отдельных связей. Алгоритм такого поиска, названный "*Optimal Brain Damage*" [84], является прекрасной иллюстрацией этого свойства нейронной сети.

Другое не менее важное свойство нейронной сети состоит в способности к обучению и к обобщению полученных знаний. Сеть обладает чертами так называемого искусственного интеллекта. Натренированная на ограниченном множестве обучающих выборок, она обобщает накопленную информацию и вырабатывает ожидаемую реакцию применительно к данным, не обрабатывавшимся в процессе обучения. Несмотря на значительное количество уже известных практических приложений искусственных нейронных сетей, возможности их дальнейшего использования для обработки сигналов не изучены окончательно, и можно высказать предположение, что нейронные сети еще в течение многих лет будут средством развития информационной техники.

Раздел 2

МОДЕЛИ НЕЙРОНОВ И МЕТОДЫ ИХ ОБУЧЕНИЯ

В соответствии с принципами функционирования биологических нейронов созданы различные математические модели, которыми в большей или меньшей степени реализуются свойства природной нервной клетки. Обобщенная схема, составляющая основу большинства таких моделей, восходит к представленной на рис. 1.3 модели МакКаллока–Питса, содержащей сумматор взвешенных входных сигналов и нелинейный блок выработки выходного сигнала нейрона, функционально зависящего от выходного сигнала сумматора. Свойства нелинейной функции, особенно ее непрерывность, оказывают определяющее влияние на выбор способа обучения нейрона (подбор весовых коэффициентов). Другим важным фактором становится выбор стратегии обучения. Можно выделить два подхода: *обучение с учителем*[1] (англ.: *supervised learning*) и *обучение без учителя* (англ.: *unsupervised learning*).

При *обучении с учителем* предполагается, что, помимо входных сигналов, составляющих вектор x, известны также и ожидаемые выходные сигналы нейрона d_i, составляющие вектор d (от англ. *destination*). В подобной ситуации подбор весовых коэффициентов должен быть организован так, чтобы фактические выходные сигналы нейрона y_i принимали бы значения, как можно более близкие к ожидаемым значениям d_i. Ключевым элементом процесса обучения с учителем является знание ожидаемых значений d_i выходного сигнала нейрона.

Если такой подход невозможен, остается выбрать стратегию обучения без учителя. Подбор весовых коэффициентов в этом случае проводится на основании либо конкуренции нейронов между собой (стратегии "*Winner Takes All – WTA*" (Победитель получает все) или "*Winner Takes Most – WTM*" (Победитель получает больше), либо с учетом корреляции обучающих и выходных сигналов (обучение по Хеббу). При *обучении без учителя* на этапе адаптации нейрона мы не можем прогнозировать его выходные сигналы, тогда как при обучении с учителем результат обучения предопределен заранее благодаря априори заданным обучающим выборкам. В этом разделе книги обсуждаются наиболее репрезентативные модели, реализующие каждый из указанных подходов.

---

[1] *Обучение с учителем* также называют *обучением под надзором*.

2.1. Персептрон

Простой персептрон – это обычная модель МакКаллока–Питса с соответствующей стратегией обучения [51]. Структурная схема и обозначения элементов i-го персептрона представлены на рис. 1.3. Весовые коэффициенты входов сумматора, на которые поступают входные сигналы x_j, обозначаются w_{ij}, а пороговое значение, поступающее с так называемого поляризатора, – w_{i0}. Нелинейная функция активации персептрона представляет собой дискретную функцию ступенчатого типа, вследствие чего выходной сигнал нейрона может принимать только два значения – 0 или 1 в соответствии с правилом

$$y_i(u_i) = \begin{cases} 1 & \text{для } u \geq 0 \\ 0 & \text{для } u < 0 \end{cases}, \tag{2.1}$$

где u_i обозначен выходной сигнал сумматора

$$u_i = \sum_{j=0}^{N} w_{ij} x_j . \tag{2.2}$$

В приведенной формуле подразумевается, что имеющий длину N вектор $\boldsymbol{x}$ дополнен нулевым членом $x_0 = 1$, формирующим сигнал поляризации, $\boldsymbol{x} = [x_0, x_1, \ldots, x_N]$. Обучение персептрона требует наличия учителя и состоит в таком подборе весов w_{ij}, чтобы выходной сигнал y_i был наиболее близок к заданному значению d_i. Это обучение гетероассоциативного типа, при котором каждой обучающей выборке, представляемой вектором $\boldsymbol{x}$, априори поставлено в соответствие ожидаемое значение d_i на выходе i-го нейрона.

Наиболее популярный метод обучения персептрона состоит в применении правила персептрона [1, 51, 114, 135], в соответствии с которым подбор весов осуществляется по следующему алгоритму:

- При первоначально выбранных (как правило, случайным образом) значениях весов w_{ij} на вход нейрона подается обучающий вектор $\boldsymbol{x}$ и рассчитывается значение выходного сигнала y_i. По результатам сравнения фактически полученного значения y_i с заданным значением d_i уточняются значения весов.
- Если значение y_i совпадает с ожидаемым значением d_i, то весовые коэффициенты w_{ij} не изменяются.
- Если $y_i = 0$, а соответствующее заданное значение $d_i = 1$, то значения весов уточняются в соответствии с формулой $w_{ij}(t + 1) = w_{ij}(t) + x_j$, где t обозначает номер предыдущего цикла, а $(t + 1)$ – номер текущего цикла.
- Если $y_i = 1$, а соответствующее заданное значение $d_i = 0$, то значения весов уточняются в соответствии с формулой $w_{ij}(t + 1) = w_{ij}(t) - x_j$, где t обозначает номер предыдущего цикла, а $(t + 1)$ – номер текущего цикла.

По завершении уточнения весовых коэффициентов представляются очередной обучающий вектор $\boldsymbol{x}$ и связанное с ним ожидаемое значение d_i, и значения весов уточняются заново. Этот процесс многократно повторяется на всех обучающих выборках, пока не будут минимизированы различия между всеми

значениями y_i и соответствующими им ожидаемыми значениями d_i.

Следует отметить, что правило персептрона представляет собой частный случай предложенного гораздо позже правила Видроу–Хоффа [114, 166]. В соответствии с этим правилом подбор весовых коэффициентов нейрона (необязательно персептронного типа) проводится по формулам:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}, \tag{2.3}$$

$$\Delta w_{ij} = x_j(d_i - y_i). \tag{2.4}$$

Аналогичные соотношения используются при подборе веса поляризатора w_{i0}, для которого входной сигнал всегда равен 1, в связи с чем

$$\Delta w_{i0} = (d_i - y_i). \tag{2.5}$$

Легко заметить, что если сигналы y_i и d_i принимают только двоичные значения 0 и 1, то правило Видроу–Хоффа превращается в правило персептрона.

Характерная особенность как правила персептрона, так и обобщенного правила Видроу–Хоффа состоит в использовании для обучения информации только о текущем и ожидаемом значениях выходного сигнала. В связи с разрывностью нелинейной функции активации персептрона невозможно учитывать информацию об изменении значения y_i (т.е. ее производную). Минимизация различий между фактическими реакциями нейрона y_i и ожидаемыми значениями d_i может быть представлена как минимизация конкретной функции погрешности (целевой функции) E, чаще всего определяемой как

$$E = \sum_{k=1}^{p} (y_i^{(k)} - d_i^{(k)})^2 , \tag{2.6}$$

где p означает количество предъявляемых обучающих выборок. Такая минимизация при использовании правила персептрона проводится по методу безградиентной оптимизации [51]. Эффективность метода при большом количестве обучающих выборок относительно невелика, а количество циклов обучения и его длительность возрастают очень быстро, причем без всякой гарантии достижения минимума целевой функции. Устранить эти недостатки можно только в случае применения непрерывной функции активации, при которой целевая функция E также становится непрерывной, что дает возможность использовать в процессе обучения информацию о величине градиента.

2.2. Сигмоидальный нейрон

Нейрон сигмоидального типа (рис. 2.1) имеет структуру, подобную модели МакКаллока–Питса, с той разницей, что функция активации является непрерывной и может быть выражена в виде сигмоидальной униполярной или биполярной функции [46, 114]. Униполярная функция, как правило, представляется формулой

$$f(x) = \frac{1}{1+e^{-\beta x}}, \tag{2.7}$$

тогда как биполярная функция задается в виде

$$f(x) = \tanh(\beta x). \tag{2.8}$$

В этих формулах параметр β подбирается пользователем. Его значение влияет на форму функции активации. На рис. 2.2 представлены графики сигмоидальной функции от переменной x для различных значений β, причем на рис. 2.2*а* показана униполярная, а на рис. 2.2*б* – биполярная функция. Графики обеих функций сильно зависят от значения β. При малых величинах β график функции достаточно пологий, но по мере роста значения β крутизна графика увеличивается. При $\beta \to \infty$ сигмоидальная функция превращается в функцию ступенчатого типа, идентичную функции активации персептрона. На практике чаще всего для упрощения используется значение $\beta = 1$.

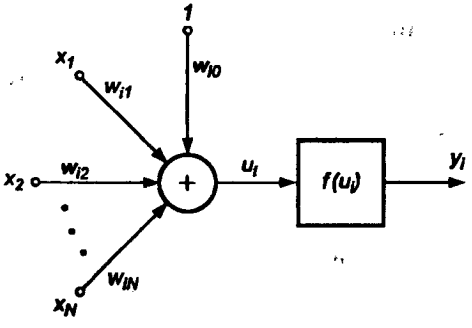

Рис. 2.1. Модель сигмоидального нейрона

Важным свойством сигмоидальной функции является ее дифференцируемость. Для униполярной функции имеем

$$\frac{df(x)}{dx} = \beta f(x)(1-(x)), \tag{2.9}$$

тогда как для биполярной функции

$$\frac{df(x)}{dx} = \beta(1-f^2(x)). \tag{2.10}$$

И в первом, и во втором случае график изменения производной относительно переменной x имеет колоколообразную форму, а его максимум соответствует значению $x = 0$ (рис. 2.3).

Сигмоидальный нейрон, как правило, обучается с учителем по принципу минимизации целевой функции, которая для единичного обучающего кортежа $<\boldsymbol{x}, \boldsymbol{d}>$ i-го нейрона определяется в виде

$$E = \frac{1}{2}(y_i - d_i)^2, \tag{2.11}$$

где

$$y_i = f(u_i) = f\left(\sum_{j=0}^{N} w_{ij} x_j\right). \tag{2.12}$$

Функция $f(u_i)$ является сигмоидальной, $\boldsymbol{x}$ – это входной вектор, $\boldsymbol{x} = [x_0, x_1, \ldots, x_N]^T$ со значением $x_0 = 1$ при наличии поляризации и $x_0 = 0$ при ее отсутствии, а d_i – соответствующее ему ожидаемое значение на выходе i-го нейрона. Применение непрерывной функции активации позволяет использовать при обучении градиентные методы. Проще всего реализовать

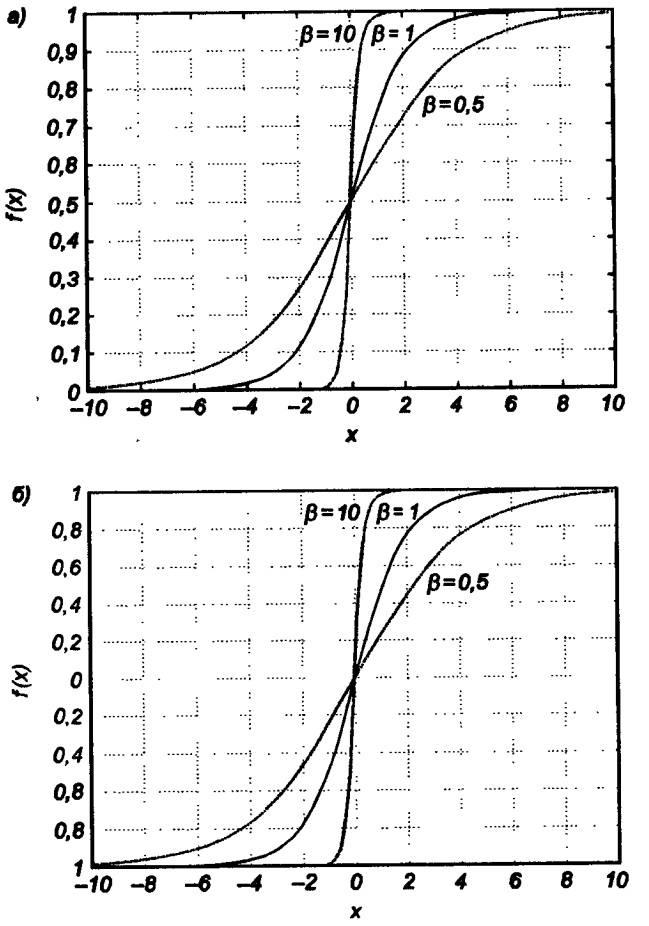

Рис. 2.2. График сигмоидальной функции:
а) униполярной; *б*) биполярной при различных значениях коэффициента β

метод наискорейшего спуска, в соответствии с которым уточнение вектора весов $w = [w_{i0}, w_{i1}, \ldots, w_{iN}]^T$ проводится в направлении отрицательного градиента целевой функции. Если эта функция определена выражением (2.11), j-я составляющая градиента имеет вид:

$$\nabla_j E = \frac{dE}{dw_{ij}} = e_i x_j \frac{df(u_i)}{du_i}, \qquad (2.13)$$

где $e_i = (y_i - d_i)$ означает разницу между фактическим и ожидаемым значением выходного сигнала нейрона. Если ввести обозначение $\delta_i = e_i \frac{df(u_i)}{du_i}$, то можно получить выражение, определяющее j-ю составляющую градиента в виде

$$\nabla_j E = \delta_i x_j . \qquad (2.14)$$

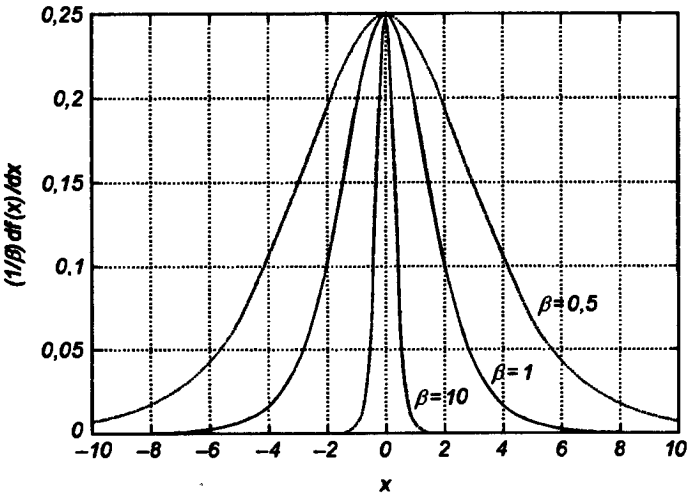

Рис. 2.3. График производной от сигмоидальной функции при различных значениях коэффициента β

Значения весовых коэффициентов также могут уточняться дискретным способом:

$$w_{ij}(t+1) = w_{ij}(t) - \eta\delta_i x_j \ , \qquad (2.15)$$

где η – это коэффициент обучения, значение которого, как правило, выбирают либо эмпирически из интервала (0,1), либо решением разностного уравнения

$$\frac{dw_{ij}}{dt} = -\mu\delta_i x_j \ , \qquad (2.16)$$

в котором константа μ выступает в роли, аналогичной значению η в уравнении (2.15). Два последних уравнения определяют алгоритм обучения нейрона. На эффективность обучения оказывает сильное влияние подбор коэффициента обучения. В существующих приложениях его величина может задаваться константой либо быть переменной величиной, значение которой изменяется в процессе обучения адаптивным способом либо подбирается на каждом шаге по принципу направленной минимизации. Наиболее эффективным, но одновременно и наиболее трудоемким считается метод направленной минимизации, по которому коэффициент обучения подбирается на каждом шаге путем минимизации целевой функции от одной переменной в направлении наискорейшего уменьшения значений этой целевой функции.

Необходимо подчеркнуть, что применение градиентного метода для обучения нейрона гарантирует достижение только локального минимума. В случае полимодальной целевой функции найденный локальный минимум может быть достаточно далек от глобального минимума. Выход из окрестности локального минимума при использовании простого алгоритма наискорейшего спуска невозможен. Результативным может оказаться обучение с *моментом* или *разбросом* [51, 114]. В этом методе процесс уточнения весов определяется не

только информацией о градиенте функции, но также и фактическим трендом изменений весов. Подобный способ обучения может быть задан следующим математическим выражением, определяющим приращение значений весов:

$$\Delta w_{ij}(t+1) = -\eta \delta_i x_j + \alpha \Delta w_{ij}(t) \ , \qquad (2.17)$$

в котором первый член соответствует обычному методу наискорейшего спуска, тогда как второй член, называемый *моментом*, отражает последнее изменение весов и не зависит от фактического значения градиента. Значение коэффициента момента α, как правило, выбирается из интервала $0 < \alpha < 1$. Следует обратить внимание, что влияние момента на подбор весов увеличивается с ростом значения α. Такое влияние существенным образом усиливается при непосредственной близости локального минимума, где значение градиента стремится к нулю. В этом случае возможны такие изменения весов, которые приводят к возрастанию значения целевой функции и выходу за пределы области локального минимума. Такая ситуация применительно к аппроксимирующей сети (выполняющей аппроксимацию входных данных) иллюстрируется на рис. 2.4. Отмеченные на графике точки соответствуют значениям целевой функции, получаемым на каждом шаге обучения. Локальный минимум P_1 был покинут благодаря действию момента. Это позволило найти в точке P_2 новый минимум с меньшим значением целевой функции, который оказался более подходящим с позиций приближения фактического значения y_i к ожидаемому значению d_i.

Следует отметить, что показатель момента не должен доминировать в процессе обучения, так как это приведет к нестабильности (расходимости) алгоритма. Как правило, в процессе обучения отслеживается значение погрешности e_i с тем, чтобы не допустить его возрастания сверх некоторого допустимого предела, например 5%. В подобном случае, если

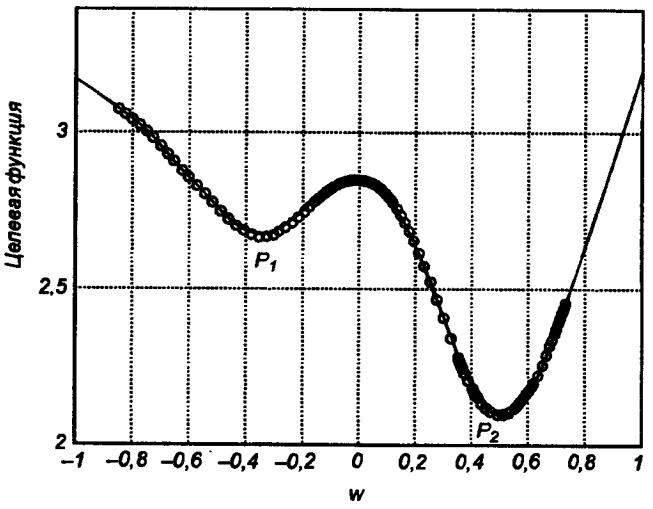

Рис. 2.4. Иллюстрация влияния момента на процесс обучения нейронной сети

$e_i(t+1) < 1{,}05\,e_i(t)$, очередной шаг считается целесообразным, и уточнение весов проводится. Если же $e_i(t+1) = 1{,}05e_i(t)$, изменения игнорируются, принимается $\Delta w_{ij}(t) = 0$, и в выражении (2.17) градиентная составляющая оказывается доминирующей над составляющей момента.

2.3. Нейрон типа "адалайн"

Модель нейрона типа "адалайн" (англ.: *ADAptive LInear NEuron* – адаптивный линейный нейрон) была предложена Б. Видроу [167]. Ее структурная схема, демонстрирующая адаптивный способ подбора весовых коэффициентов, изображена на рис. 2.5. По методу весового суммирования сигналов нейрон типа "адалайн" аналогичен представленным ранее моделям нейронов. Функция активации имеет тип signum, т.е.

$$y_i(u_i) = \begin{cases} 1 & \text{для } u_i > 0 \\ -1 & \text{для } u_i \le 0 \end{cases}. \qquad (2.18)$$

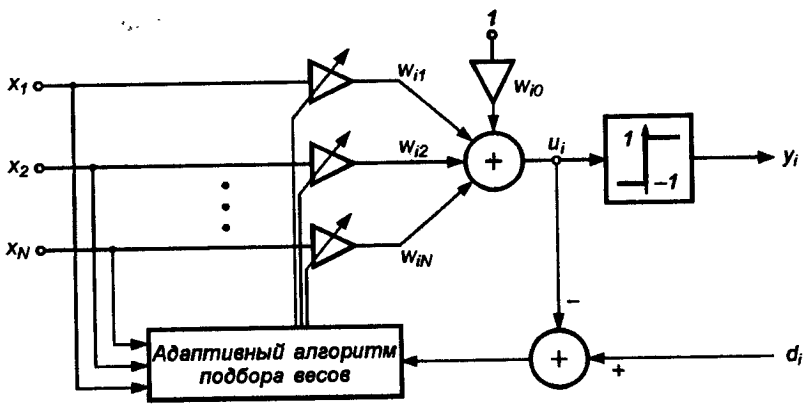

Рис. 2.5. Структурная схема нейрона типа "адалайн"

Адаптивный подбор весовых коэффициентов осуществляется в процессе минимизации квадратичной ошибки, определяемой как

$$E(w) = \frac{1}{2}e_i^2 = \frac{1}{2}\left[d_i - \left(\sum_{j=0}^{N} w_{ij}x_j\right)\right]^2. \qquad (2.19)$$

Следует обратить внимание, что, несмотря на нелинейный характер модели, в целевой функции присутствуют только линейные члены, представляющие собой сумму взвешенных входных сигналов. В связи с выполнением условия непрерывности целевой функции стало возможным применение алгоритма градиентного обучения. Как и в ситуации с сигмоидальным нейроном,

алгоритме Видроу для минимизации целевой функции применяется метод наискорейшего спуска. Значения весовых коэффициентов могут уточняться либо дискретным способом

$$w_{ij}(t+1) = w_{ij}(t) + \eta e_i x_j, \tag{2.20}$$

либо аналоговым способом – путем решения разностных уравнений вида

$$\frac{dw_{ij}}{dt} = \mu e_i x_j, \tag{2.21}$$

в которых в соответствии с зависимостью (2.19) $e_i = \left(d_i - \sum_{j=0}^{N} w_{ij} x_j\right)$. Несмотря на то, что адалайн имеет на выходе нелинейный блок типа signum, он все же считается линейным элементом, поскольку в определении целевой функции нелинейности отсутствуют, а подбор весов происходит так, как будто никакой нелинейности не существует.

Нейрон типа "адалайн" имеет относительно простую практическую реализацию [15, 113, 167] как в случае аналогового подхода на основе уравнения (2.21), так и в дискретном варианте на базе выражения (2.20). Основные компоненты модели в первом случае – это вычислительные элементы (интеграторы и сумматоры), тогда как во втором случае – это элементы задержки, описываемые оператором запаздывания z^{-1}, и также интеграторы и сумматоры. Обе адалайн-модели могут служить базой для компьютерного моделирования нейрона этого типа.

Подчеркнем, что в практических приложениях нейроны типа "адалайн" всегда используются группами, образуя слои, называемые *мадалайн* (англ.: *Many adaline* – много адалайн). Каждый входящий в слой нейрон обучается по правилу адалайн. Выходные сигналы отдельных нейронов такого слоя могут формироваться различными способами. Б. Видроу [167] предложил три базовых типа межнейронных соединений: OR, AND и мажоритарное. На рис. 2.6 *а*, *б* и *в*

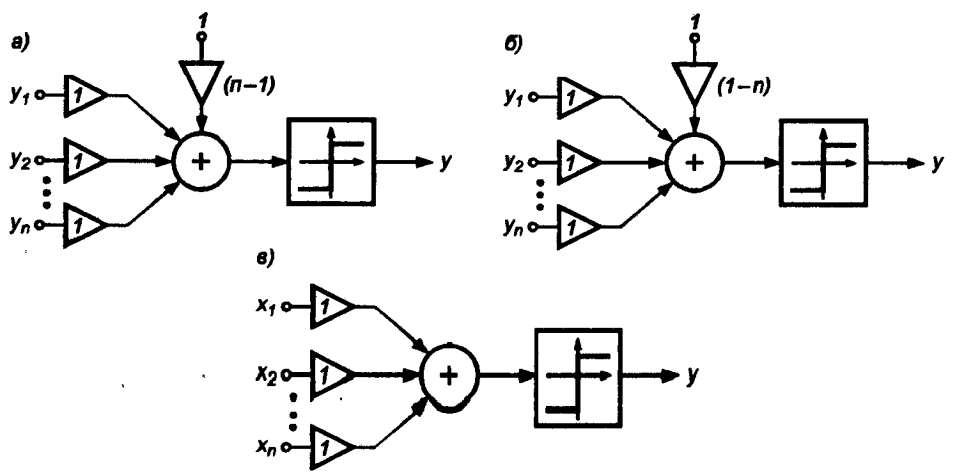

Рис. 2.6. Сеть мадалайн с выходами типа: *а*) OR; *б*) AND; *в*) мажоритарный

показаны схемы таких соединений. Конкретные сигналы y_i суммируются с учетом порогового значения, устанавливаемого раздельно для каждого типа связи. Для схемы OR порог имеет значение $(n-1)$, для схемы AND – значение $(1-n)$, а для мажоритарной схемы – нулевое значение. Благодаря применению функции активации типа signum выходной сигнал y принимает значение +1, когда хотя бы один из входных сигналов имеет значение +1 (OR), когда все входные сигналы y_i имеют значения +1 (AND) либо когда большинство сигналов y_i имеет значение +1 (мажоритарное соединение).

2.4. Инстар и оутстар Гроссберга

Нейроны типа инстар и оутстар – это взаимодополняющие элементы. Инстар адаптирует веса сигналов, поступающих на сумматор нейрона, к своим входным сигналам, а оутстар согласовывает веса выходящих из нейрона связей с узлами, в которых формируются значения выходных сигналов. Нейрон типа инстар был определен С. Гроссбергом. На рис. 2.7 представлена структурная схема инстара.

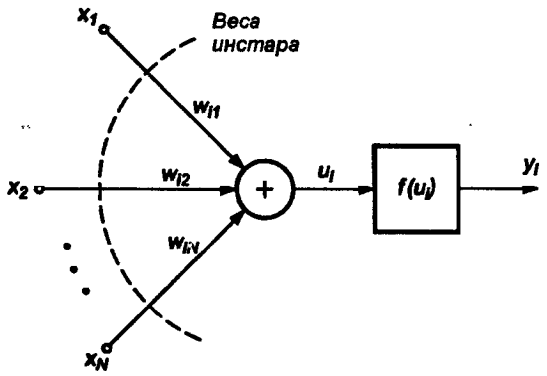

Рис. 2.7. Структурная схема инстара

Сигналы x_j, подаваемые с весовыми коэффициентами w_{ij} на вход i-го инстара, суммируются в соответствии с выражением

$$u_i = \sum_{j=1}^{N} w_{ij} x_j . \qquad (2.22)$$

В соответствии с функцией активации на выходе нейрона вырабатывается выходной сигнал $y_i = f(u_i)$. Часто в инстаре применяется линейная форма функции активации, и тогда $y_i = u_i$. Обучение инстара (подбор весов w_{ij}) производится по правилу Гроссберга, в соответствии с которым

$$w_{ij}(t+1) = w_{ij}(t) + \eta y_i [x_j - w_{ij}(t)] . \qquad (2.23)$$

где η – это коэффициент обучения, значение которого, как правило, выбирается из интервала (0,1). Входные данные, представляемые в виде вектора x, выражены чаще всего в нормализованной форме, в которой $\|x\| = 1$. Нормализация компонентов вектора x выполняется по формуле

$$x_j \leftarrow \frac{x_j}{\sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}} . \qquad (2.24)$$

Результаты обучения по методу Гроссберга в значительной степени зависят от коэффициента обучения η. При выборе $\eta = 1$ веса w_{ij} становятся равными значениям x_j уже после первой итерации. Ввод очередного входного вектора x вызовет адаптацию весов к новому вектору и абсолютное "забывание" предыдущих значений. Выбор $\eta < 1$ приводит к тому, что в результате обучения весовые коэффициенты w_{ij} принимают усредненные значения обучающих векторов x.

Допустим, что i-й инстар был обучен на некотором нормализованном относительно своих компонентов входном векторе x_1. В этом случае на векторе весов инстара выполняется отношение: $w = [w_{i1}, w_{i2}, \ldots, w_{iN}]^T = x_1$. В режиме классификации при вводе очередного входного вектора x_2 инстар вырабатывает сигнал u_i вида

$$u_i = w^T x_2 = x_1^T x_2 = \|x_1\| \, \|x_2\| \cos\varphi_{12} . \qquad (2.25)$$

Вследствие нормализации амплитуд входных векторов получаем:

$$u_i = \cos\varphi_{12} . \qquad (2.26)$$

При выполнении условия $x_2 = x_1$ реакция инстара будет равна $u_i = 1$. В случае, когда входные векторы отличаются друг от друга, реакция инстара будет пропорциональна косинусу угла между этими векторами. Для ортогональных векторов $u_i = 0$.

В итоге натренированный инстар функционирует как векторный классификатор, сопоставляющий очередной поданный на его вход вектор с вектором, сформированным в процессе обучения. В случае максимального совпадения этих векторов реакция инстара будет максимальной (наиболее близкой к единице). Если инстар обучался на группе достаточно похожих векторов с коэффициентом обучения $\eta < 1$, то его весовые коэффициенты примут значения, усредненные по этим векторам, и в режиме классификации он будет лучше всего реагировать на входные векторы, параметры которых наиболее близки к средним значениям векторов, входивших в обучающую группу.

Необходимо подчеркнуть, что инстар может обучаться как с учителем, так и без него. Во втором случае в правиле Гроссберга в качестве значения y_i принимается фактическое значение выходного сигнала инстара. При обучении с учителем значение y_i заменяется ожидаемым значением d_i, т.е. $y_i = d_i$.

Для примера рассмотрим обучение четырехвходового инстара с одноступенчатой функцией активации. Инстар тренируется с учителем, а обучающие векторы x и значения d имеют вид:

$$x_1 = \begin{bmatrix} 0{,}2630 \\ 0{,}3482 \\ 0{,}5000 \\ 0{,}7481 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0{,}5800 \\ 0{,}1000 \\ 0{,}7400 \\ 0{,}3256 \end{bmatrix}, \quad d_1 = 1, \quad d_2 = 0 .$$

Значение весового коэффициента поляризации принято равным – 0, 95. Это означает, что выходной сигнал нейрона будет равен 1 при u_i = 0,95, т.е. при значении u_i, достаточно близком к единице. При нулевых начальных значениях весов и коэффициенте обучения η = 0,4 при обучении с учителем стабилизация значений весов была достигнута уже после десяти циклов обучения. Численные результаты обучения имеют вид

$$w = \begin{bmatrix} 0{,}2614 \\ 0{,}3461 \\ 0{,}4970 \\ 0{,}7436 \end{bmatrix},$$

достаточно близкий к реализации первого входного вектора x_1. При выборе коэффициента обучения η = 0,75 нейрон оказался натренированным уже после четырех циклов обучения. Второй вектор x_2 в процессе обучения с учителем не оказывал никакого влияния на результаты обучения вследствие того, что $d_2 = 0$.

В процессе функционирования при подаче на вход вектора x_1 вырабатывается значение u_1 = 0,9940, при котором нейрон формирует выходной сигнал, равный 1. При подаче на вход вектора x_2 вырабатывается значение $u_2 = 0{,}7961 < 0{,}95$., при котором нейрон формирует нулевой выходной сигнал.

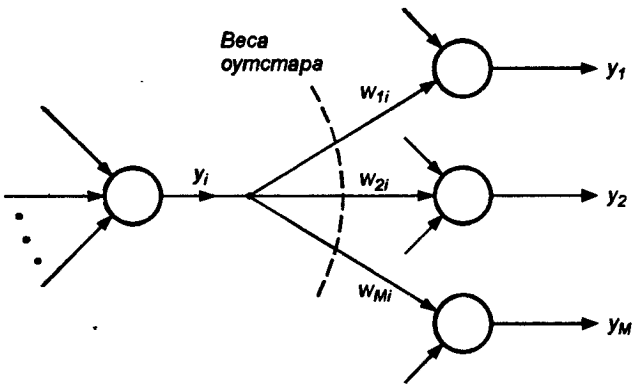

Рис. 2.8. Структурная схема оутстара

Нейрон типа оутстар Гроссберга представляет собой комплементарное дополнение инстара. Если инстар обучается с целью распознавать вектор, подаваемый на его вход, то оутстар должен генерировать вектор, необходимый связанным с ним нейронам. Структурная схема оутстара представлена на рис. 2.8. i-й нейрон-источник высылает свой выходной сигнал y_i взаимодействующим с ним нейронам, выходные сигналы которых обозначены $y_j (j = 1, 2, \ldots, M)$. Оутстар, как правило, является линейным нейроном. Обучение состоит в таком подборе его весов w_{ij}, чтобы выходные сигналы оутстара были равны ожидаемым значениям y_j взаимодействующих с ним нейронов. Обучение оутстара согласно правилу Гроссберга проводится в соответствии с выражением

$$w_{ji}(t+1) = w_{ji}(t) + \eta y_i (y_j - w_{ji}(t)), \qquad (2.27)$$

в котором η – это коэффициент обучения, а y_i – выходной сигнал i-го нейрона, выступающего в роли источника. Зависимость (2.27) для оутстара аналогична выражению (2.23), по которому обучается инстар. В режиме распознавания в момент активизации нейрона-источника оутстар будет генерировать сигналы, соответствующие ожидаемым значениям y_j.

Нейроны типа инстар и оутстар существенным образом отличаются от нейронов трех типов, определенных ранее в этом разделе. Основу обучения персептрона, сигмоидального нейрона и адалайна составляет пара обучающих векторов (x, d). Они могут обучаться только с учителем. При обучении инстара и оутстара весовые коэффициенты подстраиваются под входные или выходные векторы. Обучение может проводиться как с учителем, так и без него.

2.5. Нейроны типа WTA

Нейроны типа WTA (англ.: *Winner Takes All* – Победитель получает все) [46, 73, 114] имеют входной модуль в виде стандартного сумматора, рассчитывающего сумму входных сигналов с соответствующими весами w_{ij}. Выходной сигнал i-го сумматора определяется согласно формуле

$$u_i = \sum_{j=0}^{N} w_{ij} x_j . \qquad (2.28)$$

Группа конкурирующих между собой нейронов (рис. 2.9) получает одни и те же входные сигналы x_j. В зависимости от фактических значений весовых коэффициентов суммарные сигналы u_i отдельных нейронов могут различаться. По результатам сравнения этих сигналов победителем признается нейрон, значение u_i у которого оказалось наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные (проигравшие) нейроны переходят в состояние 0.

Для обучения нейронов типа WTA не требуется учитель, оно протекает аналогично обучению инстара, с использованием нормализованных входных векторов x. На начальном этапе случайным образом выбираются весовые коэффициенты каждого нейрона, нормализуемые относительно 1. После подачи

первого входного вектора x определяется победитель этапа. Победивший в этом соревновании нейрон переходит в состояние 1, что позволяет ему провести уточнение весов его входных линий w_{ij} (по правилу Гроссберга).

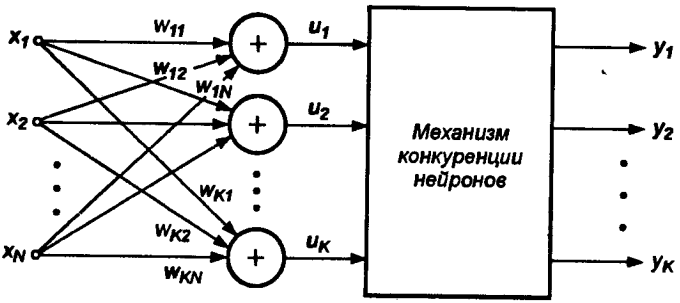

Рис. 2.9. Схема соединения нейронов типа WTA

Проигравшие нейроны формируют на своих выходах состояние 0, что блокирует процесс уточнения их весовых коэффициентов. Вследствие бинарности значений выходных сигналов конкурирующих нейронов (0 или 1) правило Гроссберга может быть несколько упрощено:

$$w_{ij}(t+1) = w_{ij}(t) + \eta[x_j - w_{ij}(t)]. \qquad (2.29)$$

На функционирование нейронов типа WTA оказывает существенное влияние нормализация входных векторов и весовых коэффициентов. Выходной сигнал u_i i-го нейрона в соответствии с формулой (2.25) может быть описан векторным отношением

$$u_i = w^T x = \|w\| \, \|x\| \cos\varphi_i . \qquad (2.30)$$

Поскольку $\|w\| = \|x\| = 1$, значение u_i определяется углом между векторами x и w, $u_i = \cos\varphi_i$. Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучающему вектору x. В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям текущего обучающего вектора x. Если на вход сети будет подаваться множество близких по значениям векторов, побеждать будет один и тот же нейрон. Поэтому его веса станут равными усредненным значениям тех входных векторов, благодаря которым данный нейрон оказался победителем. Проигравшие нейроны не изменяют свои веса. Только победа при очередном представлении входного вектора позволит им произвести уточнение весовых коэффициентов и продолжить процесс обучения в случае еще одной победы.

Следствием такой конкуренции становится самоорганизация процесса обучения. Нейроны уточняют свои веса таким образом, что при предъявлении группы близких по значениям входных векторов победителем всегда оказывается один и тот же нейрон. В процессе функционирования именно этот нейрон благодаря соперничеству распознает свою категорию входных данных. Системы такого типа чаще всего применяются для классификации векторов.

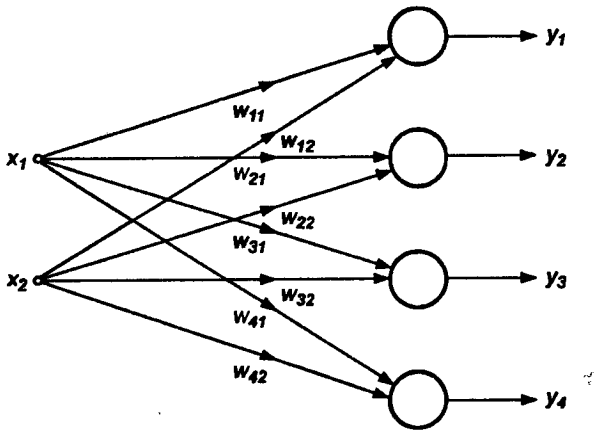

Рис. 2.10. Нейронная сеть типа WTA

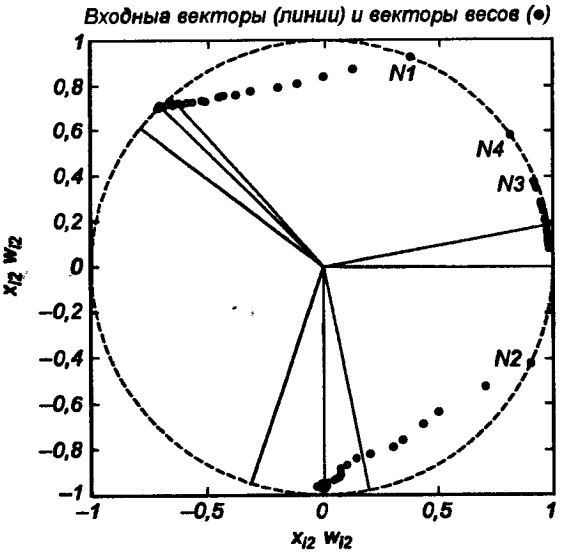

Рис. 2.11. Процесс обучения изображенной на рис. 2.10 нейронной сети типа WTA

В качестве примера рассмотрим нейронную сеть, состоящую из четырех нейронов типа WTA и предназначенную для классификации входных двухкомпонентных векторов (рис. 2.10). Входные обучающие векторы x представлены в нормализованной форме:

$$x_1 = \begin{bmatrix} 0{,}97 \\ 0{,}20 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1{,}00 \\ 0{,}00 \end{bmatrix}, \quad x_3 = \begin{bmatrix} -0{,}72 \\ 0{,}70 \end{bmatrix}, \quad x_4 = \begin{bmatrix} -0{,}67 \\ 0{,}74 \end{bmatrix},$$

$$x_5 = \begin{bmatrix} -0{,}80 \\ 0{,}60 \end{bmatrix}, \quad x_6 = \begin{bmatrix} 0{,}00 \\ -1{,}00 \end{bmatrix}, \quad x_7 = \begin{bmatrix} 0{,}20 \\ -0{,}97 \end{bmatrix}, \quad x_8 = \begin{bmatrix} -0{,}30 \\ -0{,}95 \end{bmatrix}.$$

Процесс обучения сети представлен на рис. 2.11. Окружностями обозначены позиции очередных векторов весов тех нейронов, которые побеждали в соревновании. Можно отметить, что в процессе обучения побеждали только три нейрона. Четвертый нейрон остался мертвым (он не победил ни разу) и не настроился ни на одну категорию векторов.

При значении коэффициента обучения $\eta = 0{,}05$ после 320 обучающих циклов были получены следующие веса трех первых нейронов:

$$w_1 = \begin{bmatrix} -0{,}7314 \\ 0{,}6786 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 0{,}0276 \\ -0{,}9790 \end{bmatrix}, \quad w_3 = \begin{bmatrix} 0{,}9904 \\ -0{,}0656 \end{bmatrix}.$$

Они отражают три категории входных векторов, на которые было самостоятельно разделено множество исходных данных.

Серьезной проблемой при обучении WTA остается проблема мертвых нейронов, которые после инициализации ни одного раза не победили в конкурентной борьбе и остались в состоянии, сформированном в начальный момент времени. Каждый мертвый нейрон уменьшает эффективное количество элементов, прошедших обучение, и соответственно увеличивает общую погрешность распознавания данных. Для разрешения этой проблемы применяется модифицированное обучение, основанное на учете прошлых побед каждого нейрона и штрафовании (временной дисквалификации) тех из них, которые побеждали чаще всего. Дисквалификация слишком активных нейронов может осуществляться либо назначением порогового числа побед, по достижении которого наступает обязательная пауза, либо уменьшением фактического значения u_i при нарастании количества побед i-го нейрона.

2.6. Модель нейрона Хебба

Д. Хебб в процессе исследования нервных клеток [49, 51] заметил, что связь между двумя клетками усиливается, если обе клетки пробуждаются (становятся активными) в один и тот же момент времени. Если j-я клетка с выходным сигналом y_j связана с i-й клеткой, имеющей выходной сигнал y_i, связью с весом w_{ij}, то на силу связи этих клеток влияют значения выходных сигналов y_i и y_j.

Д. Хебб предложил формальное правило, в котором отразились результаты его наблюдений. В соответствии с правилом Хебба [49], вес w_{ij} нейрона изменяется пропорционально произведению его входного и выходного сигналов[1]

$$\Delta w_{ij} = \eta y_j y_i, \tag{2.31}$$

[1] Известно также антихеббовское правило (англ.: *antihebbian learning*), в соответствии с которым $\Delta w_{ij} = -\eta\, y_j y_i$.

где η – это коэффициент обучения, значение которого выбирается в интервале (0,1). Правило Хебба может применяться для нейронных сетей различных типов с разнообразными функциями активации моделей отдельных нейронов.

Структурная схема нейрона Хебба, представленная на рис. 2.12, соответствует стандартной форме модели нейрона. Связь с весом w_{ij}, способ подбора значения которого задается отношением (2.31), соединяет входной сигнал y_j с сумматором i-го нейроиа, вырабатывающего выходной сигнал y_i. Обучение нейрона по правилу Хебба может проводиться как с учителем, так и без него. Во втором случае в правиле Хебба используется фактическое значение y_i выходного сигнала нейрона. При обучении с учителем вместо значения выходного сигнала y_i используется ожидаемая от этого нейрона реакция d_i. В этом случае правило Хебба записывается в виде

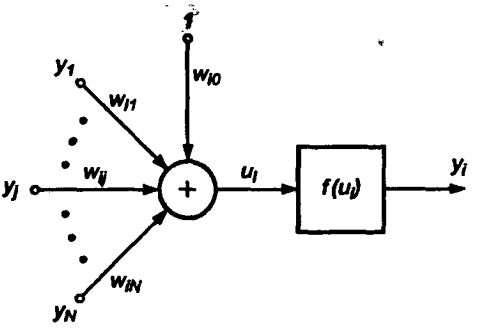

Рис. 2.12. Структурная схема нейрона Хебба

$$\Delta w_{ij} = \eta\, y_j\, d_i\,. \tag{2.32}$$

Правило Хебба характеризуется тем, что в результате его применения веса могут принимать произвольно большие значения, поскольку в каждом цикле обучения происходит суммирование текущего значения веса и его приращения Δw_{ij}:

$$w_{ij}\,(t+1) = w_{ij}\,(t) + \Delta w_{ij}\,. \tag{2.33}$$

Один из способов стабилизации процесса обучения по правилу Хебба состоит в учете для уточнения веса последнего значения w_{ij}, уменьшенного на коэффициент забывания γ[51]. При этом правило Хебба представляется в виде

$$w_{ij}\,(t+1) = w_{ij}\,(t)\,(1-\gamma) + \Delta w_{ij}\,. \tag{2.34}$$

Значение коэффициента забывания γ выбирается, как правило, из интервала (0, 1) и чаще всего составляет некоторый процент от коэффициента обучения η. Применение больших значений γ приводит к тому, что нейрон забывает значительную часть того, чему он обучился в прошлом. Рекомендуемые значения коэффициента забывания – $\gamma < 0{,}1$, при которых нейрон сохраняет большую часть информации, накопленной в процессе обучения, и получает возможность стабилизировать значения весов на определенном уровне.

В качестве примера рассмотрим обучение без учителя с забыванием сети, состоящей из четырех нейронов с одноступенчатой нелинейностью, причем на входы каждого нейрона подаются все четыре компонента вектора $\boldsymbol{x}$ (рис. 2.13). Примем, что веса поляризации w_{i0} для i = 1, 2, 3, 4 постоянны и равны –0,5.

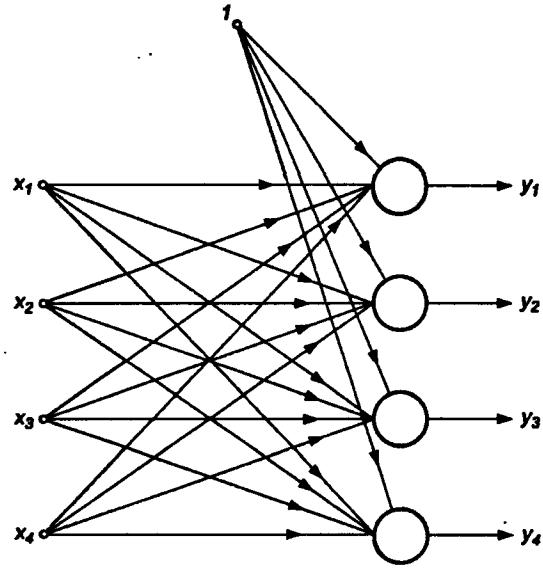

Рис. 2.13. Структура сети Хебба

Обучающие выборки x представлены в следующем виде:

$$x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad x_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Начальные значения весовых коэффициентов w_{ij} заданы единичной матрицей **w**, в которой каждый i-й столбец соответствует весам i-го нейрона:

$$\mathbf{w} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

После нескольких циклов обучения при $\eta = 0{,}1$ и $\gamma = \frac{\eta}{3}$ матрица весов приняла вид:

$$\mathbf{w} = \begin{bmatrix} 0{,}971 & 0 & 0 & 0 \\ 0 & 1{,}029 & 0 & 0{,}899 \\ 0 & 0 & 0{,}999 & 0 \\ 0 & 0 & 0 & 1{,}029 \end{bmatrix}.$$

В результате обучения связи между четвертым нейроном и вторым входом, а также между вторым нейроном и четвертым входом были усилены. После проведенного тренинга оба нейрона (второй и четвертый) одинаково реагируют на единичный сигнал, поступающий как на второй, так и на четвертый вход. Веса первого и третьего нейронов подверглись минимальным изменениям, соответствующим принятым коэффициентам обучения η и забывания γ. Сеть самостоятельно (обучение проводилось без учителя) обрела способность распознавать определенные зависимости между вторым входом и четвертым нейроном, а также между четвертым входом и вторым нейроном. По этой причине обучение по Хеббу считается обучением ассоциативного типа.

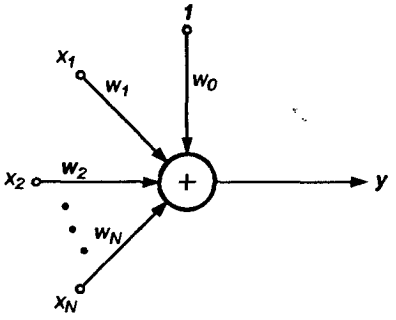

Рис. 2.14. Модель линейного нейрона Хебба

При обучении линейного нейрона по правилу Хебба стабилизация не происходит даже при вводе коэффициента забывания. Выходной сигнал нейрона, структурная схема которого приведена на рис. 2.14, определяется выражением

$$y = \sum_j w_j x_j = \boldsymbol{w}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{w} \ . \tag{2.35}$$

Если согласно правилу Хебба

$$\Delta \boldsymbol{w} = \eta \boldsymbol{x} y \tag{2.36}$$

подставить выражение (2.35) в формулу (2.36) и выбрать для упрощения $\eta = 1$, то получим приращение вектора весов $\Delta \boldsymbol{w}$ в виде

$$\Delta \boldsymbol{w} = \mathbf{C} \boldsymbol{w}, \tag{2.37}$$

где $\mathbf{C} = \boldsymbol{x}\boldsymbol{x}^T$ – это матрица корреляции, которая по определению является симметричной и положительно полуопределенной и, следовательно, имеет собственные натуральные и неотрицательные значения. При выполнении операций, описываемых зависимостью (2.37) и повторяемых на положительно полуопределенной матрице $\mathbf{C}$, процесс становится расходящимся, а значения компонентов вектора $\boldsymbol{w}$ стремятся к бесконечности.

Нестабильность правила Хебба в процессе обучения можно устранить ограничением вектора весов за счет операции ренормализации, т.е. таким подбором пропорционального коэффициента α на каждом шаге обучения, чтобы $\boldsymbol{w}' = \alpha \boldsymbol{w}$ при $\|\boldsymbol{w}'\| = 1$. Этот метод достаточно сложен и требует дополнительных трудозатрат на этапе обучения.

Е. Ойя [112] модифицировал правило Хебба таким образом, что и без ренормализации процесса обучения вектор весов самостоятельно стремится к $\|\boldsymbol{w}\| = 1$. В соответствии с правилом Ойи уточнение весов производится согласно выражению

$$\Delta w = \eta y (x_i - y w_i). \tag{2.38}$$

Это правило напоминает обратное распространение, поскольку сигнал x_i модифицируется обратным сигналом, связанным с выходным сигналом y нейрона. Для каждого отдельно взятого нейрона правило Ойя может считаться локальным, так как в процессе модификации x_i принимается во внимание только тот весовой коэффициент, значение которого подбирается в текущий момент времени.

Доказательство ограниченности весов, уточняемых по правилу Ойя, можно получить, заменяя скалярное выражение (2.38) векторной формой, которая с учетом упрощения $\eta = 1$ и в соответствии с (2.38) приобретает вид:

$$\Delta \boldsymbol{w} = \mathbf{C}\boldsymbol{w} - (\boldsymbol{w}^T \mathbf{C}\boldsymbol{w})\,\boldsymbol{w}\,. \tag{2.39}$$

Стабильность процесса обучения достигается, когда при достаточно длительном обучении обеспечивается $\|\Delta \boldsymbol{w}\| = 0$, т.е.

$$\mathbf{C}\boldsymbol{w} = (\boldsymbol{w}^T \mathbf{C}\boldsymbol{w})\,\boldsymbol{w}\,. \tag{2.40}$$

Если собственное значение корреляционной матрицы $\mathbf{C}$ обозначить λ, а вектор $\boldsymbol{w}$ подбирать как связанный с ней собственный вектор, то по определению собственного значения имеем $\mathbf{C}\boldsymbol{w} = \lambda\,\boldsymbol{w}$. Подставляя это выражение в формулу (2.39), получаем:

$$\lambda = \boldsymbol{w}^T \mathbf{C}\boldsymbol{w} = \boldsymbol{w}^T \lambda\;\boldsymbol{w} = \lambda\,|\boldsymbol{w}|^2\,. \tag{2.41}$$

Из (2.41) следует, что применение для обучения модифицированного правила Хебба приводит к ограничению модуля вектора $\boldsymbol{w}$ единицей $|\boldsymbol{w}| = 1$, обеспечивающему ограниченность значений весовых коэффициентов.

2.7. Стохастическая модель нейрона

В отличие от всех детерминированных моделей, определенных ранее в этом разделе, в стохастической модели [51] выходное состояние нейрона зависит не только от взвешенной суммы входных сигналов, но и от некоторой случайной переменной, значения которой выбираются при каждой реализации из интервала (0,1).

В стохастической модели нейрона выходной сигнал y_i принимает значения ± 1 с вероятностью $\mathrm{Prob}(y_i = \pm 1) = 1/(1 + \exp(\mp 2\beta u_i))$, где u_i обозначена взвешенная сумма входных сигналов i-го нейрона, а β – это положительная константа, чаще всего равная 1. Процесс обучения нейрона в стохастической модели состоит из следующих этапов:

- Расчет взвешенной суммы $u_i = \sum_{j=0}^{N} w_{ij} x_j$ для каждого нейрона сети.
- Расчет вероятности того, что y_i принимает значение ± 1 в соответствии с формулой

$$\mathrm{Prob}(y_i = \pm 1) = \frac{1}{1 + \exp(\mp 2\beta u_i)}\,. \tag{2.42}$$

- Генерация значения случайной переменной $R \in (0,1)$ и формирование выходного сигнала $y_i = \pm 1$, если $R < \text{Prob}(y_i = \pm 1)$ или $y_i = \mp 1$, в противном случае.
- Определенный таким образом процесс осуществляется на случайно выбранной группе нейронов, вследствие чего их состояние модифицируется в соответствии с предложенным правилом.
- После фиксации состояния отобранных нейронов их весовые коэффициенты модифицируются по применяемому правилу уточнения весов. Например, при обучении с учителем по правилу Видроу–Хоффа адаптация весов проводится по формуле

$$\Delta w_{ij} = \eta x_j (d_i - y_i) . \qquad (2.43)$$

Доказано [51], что такой способ подбора весов приводит в результате к минимизации целевой функции, определенной как среднеквадратичная погрешность

$$E = \frac{1}{2} \sum_{k=1}^{p} \sum_{i=1}^{n} \left(d_i^{(k)} - y_i^{(k)} \right)^2,$$

рассчитываемая по всем n нейронам и p обучающим выборкам.

Раздел 3

ОДНОНАПРАВЛЕННЫЕ МНОГОСЛОЙНЫЕ СЕТИ СИГМОИДАЛЬНОГО ТИПА

Объединенные между собой нейроны образуют систему, которая в дальнейшем будет называться искусственной нейронной сетью (сокращенно – ИНС). В зависимости от способа объединения нейронов они могут быть сетями *однонаправленными* либо *рекуррентными* (с обратной связью).

Среди различных известных видов ИНС наибольший интерес вызывает однонаправленная многослойная сеть MLP, состоящая из нейронов сигмоидального типа, наиболее корректно называемая многослойным персептроном (MLP – MultiLayerPerceptron)[1] [46, 135]. Передача сигналов в таких сетях происходит только в одном направлении от входа к выходу. Их математическое описание относительно просто и прозрачно, а результат может быть выражен в виде точной функциональной зависимости алгебраического типа. Методы обучения подобных сетей также достаточно просты и имеют несложную практическую реализацию. Обучение многослойного персептрона проводится, как правило, с учителем, а основная идея обучения состоит в подборе кортежей $<\boldsymbol{x}, \boldsymbol{d}>$, в которых $\boldsymbol{x}$ – входной вектор, а $\boldsymbol{d}$ – соответствующий ему ожидаемый выходной вектор сети. Если векторы $\boldsymbol{x}$ и $\boldsymbol{d}$ не равны между собой, сеть называется *гетероассоциативной*. В случае, когда $\boldsymbol{x} = \boldsymbol{d}$, сеть называется *автоассоциативной*. В сетях подобного типа используются персептронные модели нейронов либо их обобщенная форма в виде сигмоидальной модели.

В настоящем разделе представляются базовые математические зависимости, определяющие многослойные сигмоидальные сети. Мы обсудим основные методы обучения подобных сетей, в том числе алгоритм обратного распространения ошибок, методы минимизации целевой функции, а также различные методы подбора начальных значений весовых коэффициентов сети, ускоряющие процесс обучения и позволяющие избежать прекращения этого процесса в точках локальных минимумов.

С исторической точки зрения первыми были созданы однослойные сети и методы их обучения, и только через много лет (с конца семидесятых годов двадцатого века) была предложена эффективная методика обучения многослойной сети.

---

[1] Также встречается название "слоистая сеть". – *Примеч. перев.*

3.1. Однослойная сеть

Однослойную сеть образуют нейроны, расположенные в одной плоскости (рис. 3.1). Каждый i-й нейрон имеет поляризацию (связь с весом w_{i0}, по которой поступает единичный сигнал), а также множество связей с весами w_{ij}, по которым поступают входные сигналы x_j. Значения весов подбираются в процессе обучения сети, состоящем в приближении выходных сигналов y_i к ожидаемым значениям d_i. Мерой близости считается значение целевой функции, также называемой стоимостной функцией. При использовании p обучающих векторов $<x, d>$ для обучения сети, включающей M выходных нейронов, целевую функцию можно определить эвклидовой метрикой вида

$$E = \frac{1}{2}\sum_{k=1}^{p} \| y^{(k)} - d^{(k)} \|^2 = \frac{1}{2}\sum_{k=1}^{p}\sum_{i=1}^{M} (y_i^{(k)} - d_i^{(k)})^2. \qquad (3.1)$$

Выходные сигналы нейрона y_i являются функциями весов сети w_{ij}, значения которых уточняются в процессе обучения по критерию минимизации целевой функции.

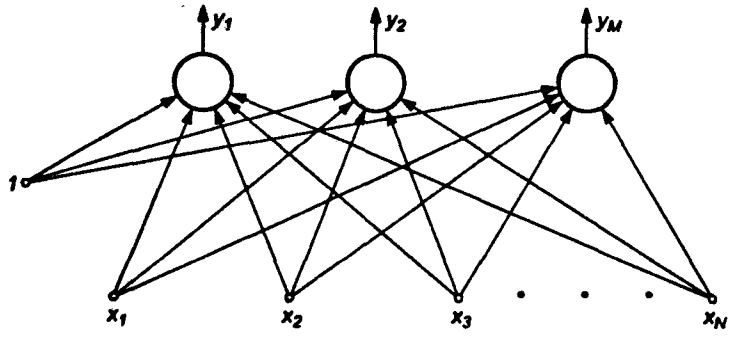

Рис. 3.1. Структура однослойной сигмоидальной нейронной сети

Расположенные на одном уровне нейроны функционируют независимо друг от друга, поэтому возможности такой сети ограничиваются свойствами отдельных нейронов. Веса нейронов образуют определенное пространство решений. Следует учитывать, что каждый нейрон реализует функциональное отображение $y_i = f(\sum_{j=0}^{N} w_{ij} x_j)$. Принимая во внимание, что сигмоидальная функция $f(*)$ представляет собой непрерывный аналог одноступенчатой пороговой функции, можно заметить, что выходной сигнал нейрона (значение 1 или 0) будет зависеть от знака выражения $\sum_{j=0}^{N} w_{ij} x_j$. Это уравнение линейно относительно весов w_{ij}. Выходной сигнал y_i при фиксированных значениях весов зависит от расположения входного вектора x, который определяет гиперплоскость, разделяющую многомерное пространство на два подпространства. Поэтому задача классификации (приписывания значения 0 или 1 выходному сигналу нейрона) может быть решена с помощью единственного нейрона, если

она относится к классу задач линейной сепарации (например, с применением логических функций AND или OR).

Продемонстрируем ограниченность возможностей однослойных сетей на примере реализации двухвходовой логической функции XOR [101]. Для упрощения будем использовать функцию активации в виде одноступенчатого порога. Множество данных для обучения логической функции XOR представлено в табл. 3.1.

Таблица 3.1

Множество данных для обучения функции XOR

x_1	0	0	1	1
x_2	0	1	0	1
d	0	1	1	0

Легко показать, что в этом случае невозможно провести единственную линию, разделяющую пространство данных на два подпространства, из которых одно соответствовало бы входному сигналу 1, а другое – 0 (на рис. 3.2 заштрихованная область относится к одному классу, а незаштрихованная – ко второму). Внутри заштрихованной области выходной сигнал нейроиа должен быть равен 1, а за ее пределами – 0. Такое условие не может быть выполнено при использовании для разделения пространства единственной прямой (одного нейрона) независимо от значений параметров этой прямой (весов w_{10}, w_{11}, w_{12}). Таким образом, однослойный персептрон не в состоянии реализовать даже такую несложную функцию, как XOR. Эту проблему легко разрешить путем расширения искусственной нейронной сети. С этой целью добавим в слой еще один нейрон и подберем веса обоих нейронов таким образом, чтобы они разделяли пространство на две части в зависимости от входного вектора $x = [x_1, x_2]^T$: $u_1 = w_{11} x_1 + w_{12} x_2 + w_{10} > 0$ и $u_1 = w_{11} x_1 + w_{12} x_2 + w_{10} < 0$ (первый нейрон) и $u_2 = w_{21} x_1 + w_{22} x_2 + w_{20} > 0$ и $u_2 = w_{21} x_2 + w_{22} x_2 + w_{20} < 0$ (второй нейрон). Подбор весов должен обеспечить разделение пространства, показанное на рис. 3.3. Общая часть подмножеств, соответствующая условиям $u_1 > 0$, $u_2 > 0$, определила область, отделенную от остального пространства, соответствующего условиям $u_1 < 0$, $u_2 < 0$.

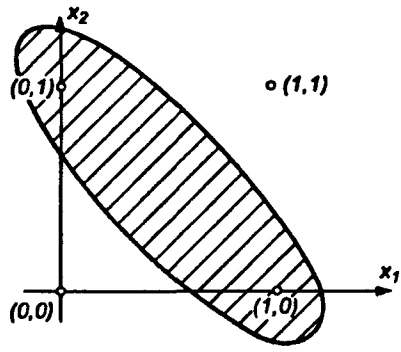

Рис. 3.2. Иллюстрация иевозможности линейного разделения обучающих данных, соответствующих логической функции XOR

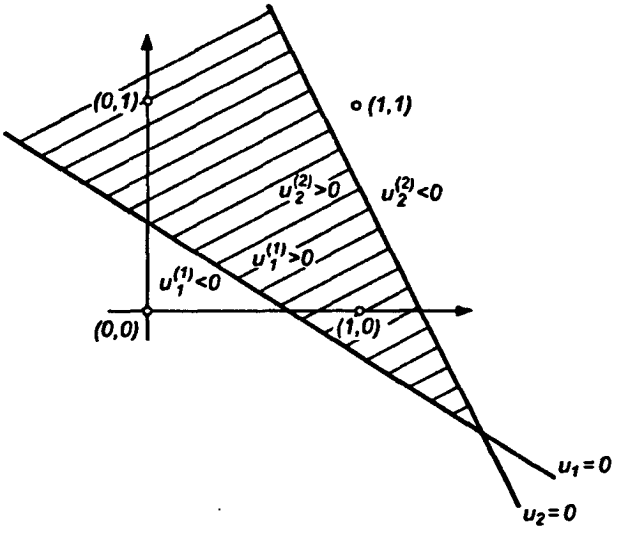

Рис. 3.3. Решение проблемы нелинейного разделения путем применения двух линейных разделителей

Добавлением на выходе сети еще одного слоя, состоящего из единственного нейрона, можно реализовать функцию логического суммирования, выделяющую общую часть подмножеств $u_1 > 0$, $u_2 > 0$. Окончательная структура ИНС, выполняющей функцию XOR, представлена на рис. 3.4. Следует отметить, что добавление в сеть дополнительного слоя позволило разрешить проблему невозможности линейного разделения данных. Каждый нейрон скрытого слоя осуществляет дополнительное линейное разделение плоскости, причем граница такого раздела на области $u_i > 0$ и $u_i < 0$ зависит от значений весов нейрона. Выходной слой выполняет соответствующую линейную комбинацию (например, логическую сумму) подобластей, на которые множество входных данных было разделено нейронами скрытого слоя.

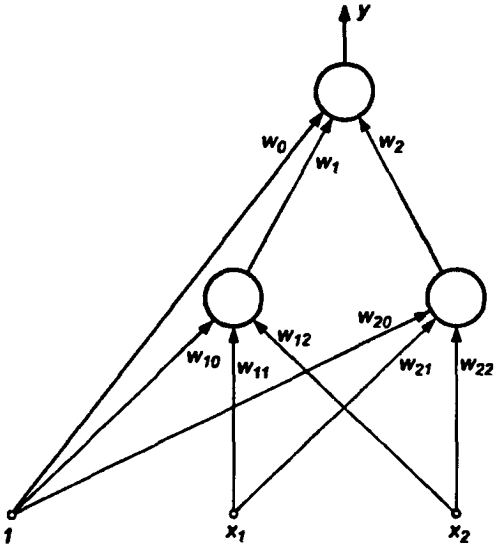

Рис. 3.4. Структура ИНС, выполняющей функцию XOR

Несмотря на то, что однослойная сеть имеет небольшое практическое значение, ее продолжают использовать там, где для решения поставленной задачи достаточно и одного слоя нейронов.

Выбор архитектуры такой сети весьма прост. Количество входных нейронов определяется размерностью входного вектора x, а количество выходных нейронов определяется размерностью вектора d. Обучение сети производится, как правило, с учителем и является точной копией обучения одиночного нейрона.

3.2. Многослойный персептрон

3.2.1. Структура персептронной сети

Многослойная сеть состоит из нейронов, расположенных на разных уровнях, причем, помимо входного и выходного слоев, имеется еще, как минимум, один внутренний, т.е. скрытый, слой. Как уже отмечалось в литературе, посвященной проблематике нейронных сетей, такая нейронная система называется *многослойным персептроном* [46, 135, 136].

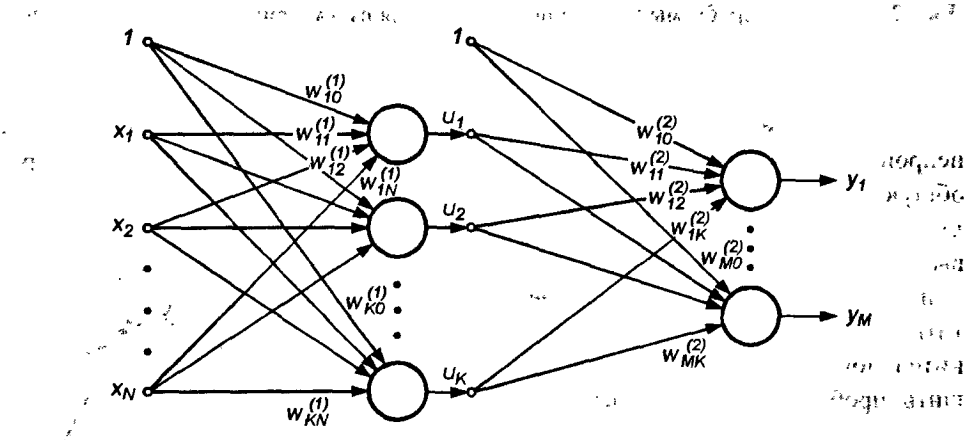

Рис. 3.5. Обобщенная структура двухслойной сигмоидальной нейронной сети (с одним скрытым слоем)

На рис. 3.5 представлена сеть с одним скрытым слоем. Все последующие рассуждения относятся к сетям именно такого типа. Обозначения сигналов и весов также будут соответствовать этому рисунку. Веса нейронов скрытого слоя пометим верхним индексом (1), а выходного слоя – верхним индексом (2). Выходные сигналы нейронов скрытого слоя обозначим v_j (j = 1, 2, ..., K), а выходного слоя – y_j (j = 1, 2, ..., M). Примем, что функция активации нейронов задана в сигмоидальной униполярной или биполярной форме. Для упрощения описания будем использовать расширенное обозначение входного вектора сети в виде $x = [x_0, x_1, ..., x_N]^T$, где $x_0 = 1$ соответствует единичному сигналу поляризации. С вектором x связаны два входных вектора сети: вектор

фактических выходных сигналов $\boldsymbol{y} = [y_0, y_1, ..., y_M]^T$ и вектор ожидаемых выходных сигналов $\boldsymbol{d} = [d_0, d_1, ..., d_M]^T$.

Цель обучения состоит в подборе таких значений весов $w_{ij}^{(1)}$ и $w_{ij}^{(2)}$ для всех слоев сети, чтобы при заданном входном векторе $\boldsymbol{x}$ получить на выходе значения сигналов y_i, которые с требуемой точностью будут совпадать с ожидаемыми значениями d_i для $i = 1, 2, ..., M$. Если рассматривать единичный поляризационный сигнал как один из компонентов входного вектора $\boldsymbol{x}$, то веса поляризации можно добавить в векторы весов соответствующих нейронов обоих слоев. При таком подходе выходной сигнал i-го нейрона скрытого слоя удается описать функцией

$$v_j = f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right), \tag{3.2}$$

в которой индекс 0 соответствует сигналу и весам поляризации, причем $v_0 \equiv 1$, $x_0 \equiv 1$. В выходном слое k-й нейрон вырабатывает выходной сигнал, определяемый как

$$y_k = f\left(\sum_{i=0}^{K} w_{ki}^{(2)} v_i\right) = f\left(\sum_{i=0}^{K} w_{ki}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right)\right). \tag{3.3}$$

Из формулы (3.3) следует, что на значение выходного сигнала влияют веса обоих слоев, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

3.2.2. Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки определяет стратегию подбора весов многослойной сети с применением градиентных методов оптимизации. "Изобретенный заново" несколько раз [46], он в настоящее время считается одним из наиболее эффективных алгоритмов обучения многослойной сети. Его основу составляет целевая функция, формулируемая, как правило, в виде квадратичной суммы разностей между фактическими и ожидаемыми значениями выходных сигналов. В случае единичной обучающей выборки ($\boldsymbol{x}$, $\boldsymbol{d}$) целевая функция определяется в виде

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{M}(y_k - d_k)^2 . \tag{3.4}$$

При большем количестве обучающих выборок j ($j = 1, 2, ..., p$) целевая функция превращается в сумму по всем выборкам

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{M}(y_k^{(j)} - d_k^{(j)})^2 . \tag{3.5}$$

Уточнение весов может проводиться после предъявления каждой обучающей выборки (так называемый режим "онлайн") либо однократно после предъяв-

ления всех выборок, составляющих цикл обучения (режим "оффлайн"). В последующем изложении используется целевая функция вида (3.4), которая соответствует актуализации весов после предъявления каждой выборки.

Для упрощения можно считать, что цель обучения состоит в таком определении значений весов нейронов каждого слоя сети, чтобы при заданном входном векторе получить на выходе значения сигналов y_i, совпадающие с требуемой точностью с ожидаемыми значениями d_i при $i = 1, 2, ..., M$.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в несколько этапов. На первом из них предъявляется обучающая выборка $\boldsymbol{x}$ и рассчитываются значения сигналов соответствующих нейронов сети. При заданном векторе $\boldsymbol{x}$ определяются вначале значения выходных сигналов v_i скрытого слоя, а затем значения y_i нейронов выходного слоя. Для расчета применяются формулы (3.2) и (3.3). После получения значений выходных сигналов y_i становится возможным рассчитать фактическое значение целевой функции $E(\boldsymbol{w})$, заданной выражением (3.4). На втором этапе минимизируется значение этой функции.

Если принять, что целевая функция непрерывна, то наиболее эффективными способами обучения оказываются градиентные методы оптимизации, согласно которым уточнение вектора весов (обучение) производится по формуле

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \Delta \boldsymbol{w}, \tag{3.6}$$

где

$$\Delta \boldsymbol{w} = \eta \boldsymbol{p}(\boldsymbol{w}), \tag{3.7}$$

η – коэффициент обучения, а $\boldsymbol{p}(\boldsymbol{w})$ – направление в многомерном пространстве $\boldsymbol{w}$.

Обучение многослойной сети с применением градиентных методов требует определения вектора градиента относительно весов всех слоев сети, что необходимо для правильного выбора направления $\boldsymbol{p}(\boldsymbol{w})$. Эта задача имеет очевидное решение только для весов выходного слоя. Для других слоев создана специальная стратегия, которая в теории искусственных нейронных сетей называется *алгоритмом обратного распространения ошибки* (англ.: *error backpropagation*) [46, 51], отождествляемым, как правило, с процедурой обучения сети. В соответствии с этим алгоритмом в каждом цикле обучения выделяются следующие этапы [46].

1. Анализ нейронной сети в прямом направлении передачи информации при генерации входных сигналов, составляющих очередной вектор $\boldsymbol{x}$. В результате такого анализа рассчитываются значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие

производные $\frac{df(u_i^{(1)})}{du_i^{(1)}}, \frac{df(u_i^{(2)})}{du_i^{(2)}}, \ldots, \frac{df(u_i^{(m)})}{du_i^{(m)}}$ функций активации каждого слоя (m – количество слоев сети).

2. Создание сети обратного распространения ошибок путем изменения направлений передачи сигналов, замена функций активации их производными и подача на бывший выход (а в настоящий момент – вход) сети возбуждения в виде разности между фактическим и ожидаемым значением. Для определенной таким образом сети необходимо рассчитать значения требуемых обратных разностей.
3. Уточнение весов (обучение сети) производится по предложенным выше формулам на основе результатов, полученных в п. 1 и 2, для оригинальной сети и для сети обратного распространения ошибки.
4. Описанный в п. 1, 2 и 3 процесс следует повторить для всех обучающих выборок, продолжая его вплоть до выполнения условия остановки алгоритма. Действие алгоритма завершается в момент, когда норма градиента упадет ниже априори заданного значения ε, характеризующего точность процесса обучения.

Базовые формулы и их модификации для конкретных типов нейронных сетей считаются классическими для теории нейронных сетей. По этой причине мы рассмотрим только условия, относящиеся к сети с одним скрытым слоем. Используемые обозначения представлены на рис. 3.5.

Как и ранее, количество входных узлов обозначим буквой N, количество нейронов в скрытом слое K, а количество нейронов в выходном слое M. Будем использовать сигмоидальную функцию активации этих нейронов. Основу алгоритма составляет расчет значения целевой функции как квадратичной суммы разностей между фактическими и ожидаемыми значениями выходных сигналов сети. В случае единичной обучающей выборки ($\boldsymbol{x}, \boldsymbol{d}$) целевая функция задается формулой (3.4), а для множества обучающих выборок j ($j = 1, 2, \ldots, p$) – формулой (3.5). Для упрощения излагаемого материала будем использовать целевую функцию вида (3.4), которая позволяет уточнять веса после предъявления каждой обучающей выборки.

С учетом обозначений, введенных на рис. 3.5, эта функция определяется выражением

$$E = \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{ki}^{(2)} v_i\right) - d_k\right]^2 = \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{ki}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_i\right)\right) - d_k\right]^2. \quad (3.8)$$

Конкретные компоненты градиента рассчитываются дифференцированием зависимости (3.8). В первую очередь подбираются веса нейронов выходного слоя. Для выходных весов получаем:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}} v_j, \quad (3.9)$$

где $u_i^{(2)} = \sum_{j=0}^{K} w_{ij}^{(2)} v_j$. Если ввести обозначение $\delta_i^{(2)} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}}$, то соответствующий компонент градиента относительно весов нейронов выходного слоя можно представить в виде

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \partial_i^{(2)} v_j . \tag{3.10}$$

Компоненты градиента относительно нейронов скрытого слоя определяются по тому же принципу, однако они описываются другой, более сложной зависимостью, следующей из существования функции, заданной в виде

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k)\frac{dy_k}{dv_i}\frac{dv_i}{dw_{ij}^{(1)}} . \tag{3.11}$$

После конкретизации отдельных составляющих этого выражения получаем:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k)\frac{df(u_k^{(2)})}{du_k^{(2)}} w_{ki}^{(2)} \frac{df(u_i^{(1)})}{du_i^{(1)}} x_j . \tag{3.12}$$

Если ввести обозначение

$$\partial_i^{(1)} = \sum_{k=1}^{M} (y_k - d_k)\frac{df(u_k^{(2)})}{du_k^{(2)}} w_{ki}^{(2)} \frac{df(u_i^{(1)})}{du_i^{(1)}} , \tag{3.13}$$

то получим выражение, определяющее компоненты градиента относительно весов нейронов скрытого слоя в виде

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \partial_i^{(1)} x_j . \tag{3.14}$$

В обоих случаях (формулы (3.10) и (3.14)) описание градиента имеет аналогичную структуру и представляется произведением двух сигналов: первый соответствует начальному узлу данной взвешенной связи, а второй – величине погрешности, перенесенной на узел, с которым эта связь установлена. Определение вектора градиента очень важно для последующего процесса уточнения весов. В классическом алгоритме обратного распространения ошибки фактор $\boldsymbol{p}(\boldsymbol{w})$, учитываемый в выражении (3.6), задает направление отрицательного градиента, поэтому

$$\Delta \boldsymbol{w} = -\eta \nabla E(\boldsymbol{w}) . \tag{3.15}$$

В следующем разделе представляются другие, более эффективные методы выбора направления $\boldsymbol{p}(\boldsymbol{w})$.

3.3. Потоковые графы и их применение для генерации градиента

Знакомство с формулами для расчета градиента показывает, что они довольно сложны и неудобны для практического применения, особенно если сеть содержит более одного скрытого слоя. Поэтому представляется интересным, что на основе метода потоковых графов удается построить очень простые правила формирования компонентов градиента, которые имеют постоянную структуру, не зависящую от сложности сети. При этом базу таких правил составляют соотношения, полученные в результате анализа чувствительности сети методом сопряженных элементов. В теории систем [114] под полной чувствительностью объекта понимается производная любого циркулирующего в нем сигнала относительно значений весов, которая может быть рассчитана на основании знаний о сигналах, распространяющихся по обычному графу (обозначаемому G) и сопряженному с ним графу (обозначаемому $\hat{G}$). Граф $\hat{G}$ определяется как исходный граф G, в котором направленность всех дуг изменена на противоположную. Линейная дуга графа G и соответствующая ей дуга сопряженного графа $\hat{G}$ имеют идентичные описания. В случае нелинейной связи $f(x, k)$, где x – входной сигнал, а k – параметр, соответствующая ей дуга графа $\hat{G}$ линеаризуется с коэффициентом $\beta = \frac{\partial f(x,k)}{\partial x}$, рассчитанным для фактического входного сигнала x графа G.

Как показано в работах [113, 114, 126], метод расчета чувствительности нейронной сети с использованием потоковых графов основан на анализе исходного графа G и сопряженного с ним графа $\hat{G}$ при возбуждении последнего единичным сигналом, подаваемым на вход $\hat{G}$ (соответствующий выходу G). Чувствительность графа G относительно параметров дуг этого графа к произвольному входному сигналу v_0 можно выразить следующим образом:

- для линейной дуги w_{ij} графа G

$$\frac{dv_0}{dw_{ij}} = v_j \hat{v}_i , \tag{3.16}$$

  где w_{ij} – это коэффициент усиления линейной дуги, направленной от j-го узла к i-му; v_j обозначает сигнал j-го узла графа G, а $\hat{v}_i$ – сигнал i-го узла сопряженного графа $\hat{G}$, для которого в качестве входного сигнала задается значение $\hat{v}_0 = 1$;
- для нелинейной дуги графа G, объединяющей l-й и k-й узлы и описываемой функцией $v_k = f_{kl}(v_l, K)$, чувствительность относительно параметра K определяется выражением

$$\frac{dv_0}{dK} = \hat{v}_k \frac{\partial f_{kl}(v_l, K)}{\partial K}, \tag{3.17}$$

где $\frac{\partial f_{kl}(v_l, K)}{\partial K}$ рассчитывается для сигнала v_l l-го узла графа G.

Обозначим w вектор оптимизированных параметров (весов w_i) системы, представленной графом G, $w = [w_1, w_2, \ldots, w_n]^T$, а $E(w)$ – целевую функцию. Тогда градиент $\nabla E(w)$, сокращенно обозначаемый как $g(w) = \nabla E(w)$, можно определить в виде

$$g(w) = \left[\frac{\partial E(w)}{\partial w_1}, \frac{\partial E(w)}{\partial w_2}, \ldots, \frac{\partial E(w)}{\partial w_n}\right]^T \tag{3.18}$$

(в этом выражении для обозначения элементов вектора w использован один индекс $i = 1, 2, \ldots, n$). Если представить целевую функцию в форме, учитывающей только одну обучающую выборку

$$E(w) = \frac{1}{2}\sum_{i=1}^{M}(y_i - d_i)^2, \tag{3.19}$$

где d_i обозначено ожидаемое значение i-го выходного нейрона, $i = 1, 2, \ldots, M$, то градиент целевой функции принимает вид:

$$g(w) = [g_1(w), g_2(w), \ldots, g_n(w)]^T, \tag{3.20}$$

в которой

$$g_k(w) = \sum_{i=1}^{M}(y_i - d_i)\frac{\partial y_i}{\partial w_k}. \tag{3.21}$$

Для задания вектора градиента также необходимы производные выходных сигналов y_i графа относительно весов w_k (показатели их чувствительности), умноженные на величину погрешности $(y_i - d_i)$. Благодаря использованию методов теории графов все эти операции (в том числе и суммирование) можно выполнить за один шаг с помощью исходного графа G и сопряженного с ним графа $\hat{G}$ при соблюдении соответствующих условий возбуждения графа $\hat{G}$. Как показано в [126], вследствие линейности сопряженного графа все эти операции могут быть реализованы автоматически в случае, когда в сопряженном графе вместо единичных возбуждений генерируются сигналы в виде разностей между фактическими y_i и ожидаемыми d_i значениями выходных сигналов. Способ формирования сопряженного графа $\hat{G}$ и методика его возбуждения для автоматического расчета вектора градиента на основе анализа только двух графов G и $\hat{G}$ представлены на рис. 3.6. При замене всех единичных возбуждений в $\hat{G}$ на $(y_i - d_i)$ любой компонент вектора градиента $g_k(w)$ может быть рассчитан по соответствующим сигналам исходного графа G и сопряженного с ним графа $\hat{G}$ точно так же, как и при определении обычной

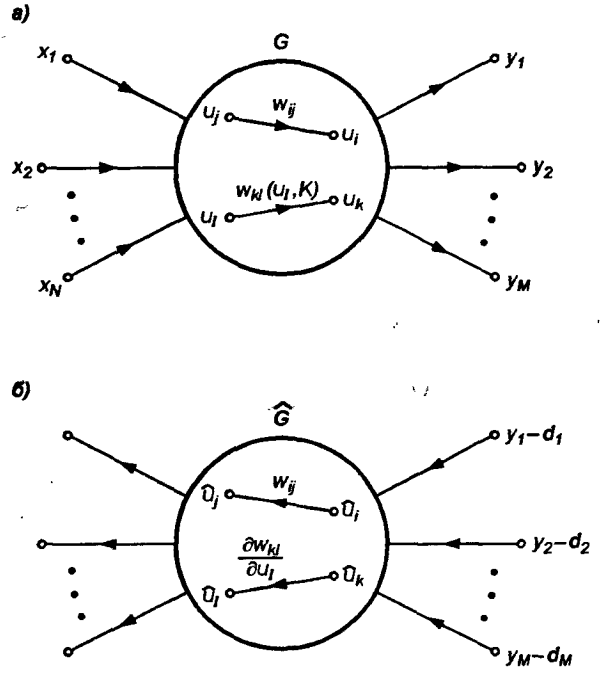

Рис. 3.6. Иллюстрация применения способа формирования и возбуждения сопряженного графа: *а*) исходный граф G; *б*) сопряженный граф $\hat{G}$

чувствительности. Для линейной дуги графа G, описываемой весом w_{ij}, формула имеет вид:

$$\frac{\partial E(w)}{\partial w_{ij}} = v_j \hat{v}_i \ . \tag{3.22}$$

Для нелинейной дуги графа G, описываемой функцией $w_{kl}(v_l, K)$, получаем:

$$\frac{\partial E(w)}{\partial K} = \hat{v}_k \frac{\partial w_{kl}(v_l, K)}{\partial K} \ . \tag{3.23}$$

Представленные выражения применимы для любых систем (линейных, нелинейных, рекуррентных и т.п.). Они практически применяются для анализа однонаправленных многослойных нейронных сетей, описываемых потоковым графом прохождения сигналов.

Рассмотрим изображенную на рис. 3.7*а* типовую многослойную сеть (состоящую из m слоев) с произвольной непрерывной функцией активации нейронов. Количество нейронов в каждом слое будем обозначать K_j ($j = 1, 2, ..., m$), причем последний слой является выходным слоем сети, содержащим $K_m = M$ нейронов. Выходные сигналы нейронов в конкретных слоях обозначим $v_j^{(k)}$, причем для последнего слоя $v_j^{(m)} = y_j$.

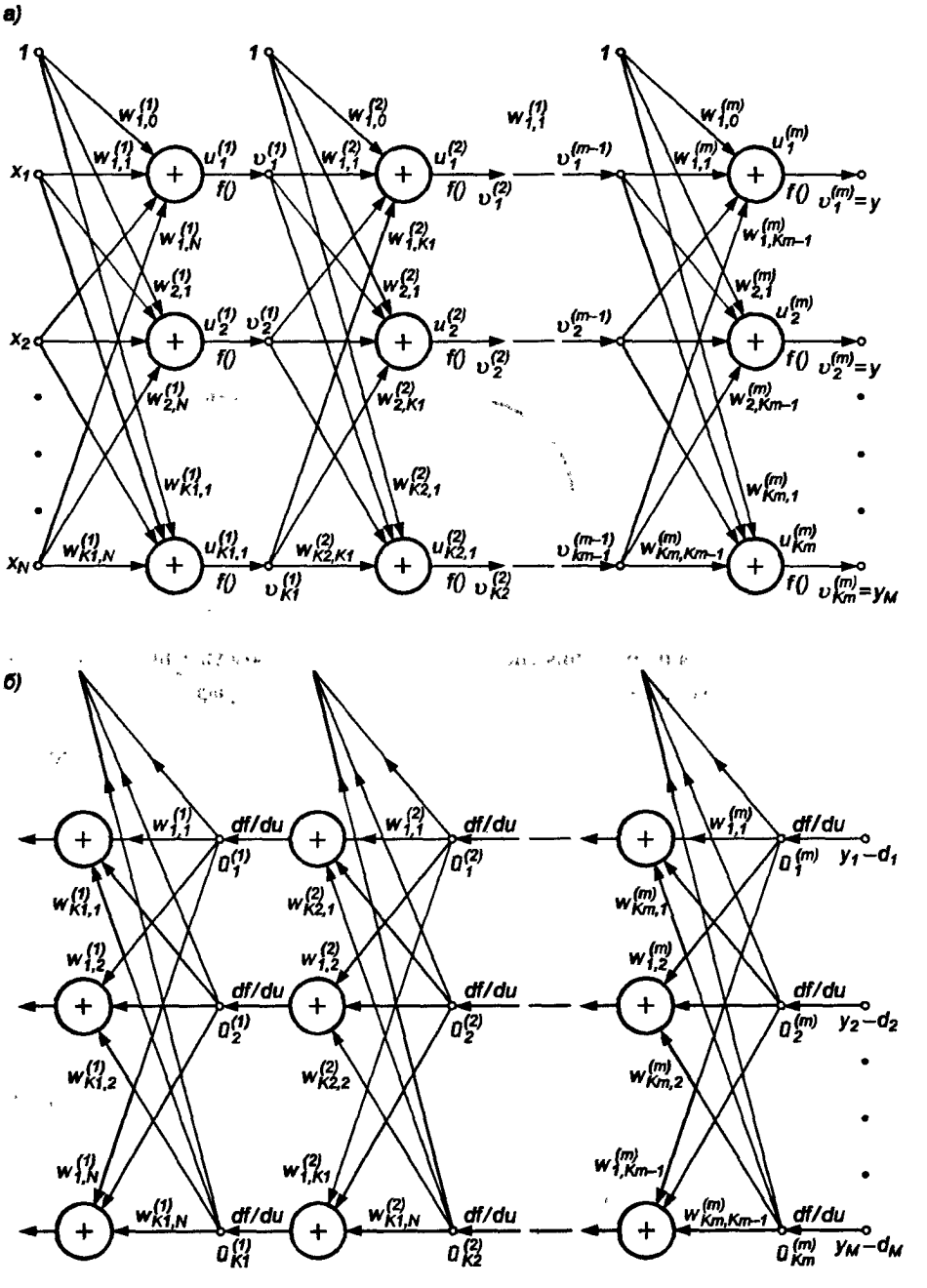

Рис. 3.7. Иллюстрация применения метода сопряженных графов для генерации вектора градиента однонаправленной многослойной сети:
а) выходной граф сети; *б*) сопряженный граф

Для определения компонентов градиента относительно весов конкретных слоев сети будем применять формулировки, относящиеся к сопряженному графу. На рис 3.7*б* представлен сопряженный граф сети, изображенной на рис. 3.7*а*, причем для унификации описания все сигналы в сопряженном графе обозначены символами с "крыжиком" ($\hat{u}$). Сопряженный граф возбуждается разностями между фактическими y_i и ожидаемыми d_i значениями выходных сигналов. Нелинейные дуги графа G заменяются в сопряженном графе $\hat{G}$ производными $\frac{df(x)}{dx}$, значения которых рассчитываются раздельно для каждого слоя в точках $x = u_i$. Если, например, функция активации нейронов имеет сигмоидальную униполярную форму $f(x) = \frac{1}{1+\exp(-x)}$, то $\frac{df(x)}{dx} = f(x)(1-f(x))$ рассчитывается непосредственно на основе известного значения сигмоидальной функции в точке x и не требует никаких дополнительных вычислений.

Опираясь на предложенный алгоритм определения градиента методами теории графов, можно рассчитать конкретные компоненты вектора градиента $\boldsymbol{g}(\boldsymbol{w})$ для любого слоя нейронов:

- для выходного слоя

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{ij}^{(m)}} = v_j^{(m-1)}\,\hat{u}_i^{(m)}; \tag{3.24}$$

- для k-го скрытого слоя

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{ij}^{(k)}} = v_j^{(k-1)}\,\hat{u}_i^{(k)}; \tag{3.25}$$

- для первого скрытого слоя

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{ij}^{(1)}} = x_j u_i^{(1)}. \tag{3.26}$$

Из приведенных формул видно, что их структуры (при использовании соответствующих обозначений сигналов) абсолютно идентичны независимо от того, в каком слое нейронов находится учитываемый вес. Сформулированное правило является чрезвычайно простым с прикладной точки зрения, поскольку для расчета любого компонента градиента необходимо знать только два сигнала: от узла, из которого исходит взвешенная дуга в оригинальном графе, и от узла, из которого исходит взвешенная дуга в сопряженном графе. В этом смысле правило расчета может считаться локальным.

Еще одним важным достоинством графического метода, помимо значительного упрощения вычислительных процедур, считается возможность учета равенства значений различных весов сети [114]. Если, например, вес со

значением w относится к дуге w_{ij}, соединяющей i-й и j-й узлы (в направлении от j-го к i-му), и к дуге w_{kl}, соединяющей k-й и l-й узлы (в направлении от l-го к k-му), то легко заметить, что вес w будет присутствовать в двух различных позициях выражения, определяющего целевую функцию. Согласно правилу дифференцирования составной функции ее производная представляется в виде суммы производных относительно w_{ij} и w_{lk}. Следовательно,

$$\frac{\partial E(\boldsymbol{w})}{\partial w} = \frac{\partial E}{\partial w_{ij}} + \frac{\partial E}{\partial w_{kl}}. \qquad (3.27)$$

С учетом рассуждений относительно сопряженного графа при вводе унифицированных обозначений сигналов для любых узлов в виде v ($\hat{v}$) с соответствующими индексами получаем окончательное выражение

$$\frac{\partial E(\boldsymbol{w})}{\partial w} = v_j \hat{u}_i + v_l \hat{u}_k. \qquad (3.28)$$

Как следует из формулы (3.28), учет равенства отдельных весов ($w_{ij} = w_{lk}$) не только не усложняет общую расчетную формулу, но, напротив, упрощает ее за счет уменьшения количества переменных. Необходимо отметить, что совпадающие веса (англ.: *shared weights*) могут лежать как в одном и том же, так и в совершенно разных слоях. Сущность формулы (3.28) при этом совершенно не меняется. В этом заключается важнейшее отличие метода генерации градиента, основанного на потоковых графах распространения сигналов, от классического подхода, широко представленного в мировой литературе [46, 51].

3.4. Градиентные алгоритмы обучения сети

3.4.1. Основные положения

Задачу обучения нейронной сети будем рассматривать на данном этапе как требование минимизировать априори определенную целевую функцию $E(\boldsymbol{w})$. При таком подходе можно применять для обучения алгоритмы, которые в теории оптимизации считаются наиболее эффективными. К ним, без сомнения, относятся градиентные методы, чью основу составляет выявление градиента целевой функции. Они связаны с разложением целевой функции $E(\boldsymbol{w})$ в ряд Тейлора в ближайшей окрестности точки имеющегося решения $\boldsymbol{w}$. В случае целевой функции от многих переменных ($\boldsymbol{w} = [w_1, w_2, ..., w_n]^T$) такое представление связывается с окрестностью ранее определенной точки (в частности, при старте алгоритма это исходная точка $\boldsymbol{w}_0$) в направлении $\boldsymbol{p}$. Подобное разложение описы-

вается универсальной формулой вида [39, 170]

$$E(\boldsymbol{w}+\boldsymbol{p}) = E(\boldsymbol{w}) + [\boldsymbol{g}(\boldsymbol{w})]^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \boldsymbol{H}(\boldsymbol{w})\boldsymbol{p} + \dots , \qquad (3.29)$$

где $\boldsymbol{g}(\boldsymbol{w}) = \nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_n}\right]^T$ – это вектор градиента, а симметричная квадратная матрица

$$\mathbf{H}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \dots & \dots & \dots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}$$

является матрицей производных второго порядка, называемой *гессианом*.

В выражении (3.29) $\boldsymbol{p}$ играет роль направляющего вектора, зависящего от фактических значений вектора $\boldsymbol{w}$. На практике чаще всего рассчитываются три первых члена ряда (3.29), а последующие просто игнорируются. При этом зависимость (3.29) может считаться квадратичным приближением целевой функции $E(\boldsymbol{w})$ в ближайшей окрестности найденной точки $\boldsymbol{w}$ с точностью, равной локальной погрешности отсеченной части $O(h^3)$, где $h = \| \boldsymbol{p} \|$. Для упрощения описания значения переменных, полученные в k-м цикле, будем записывать с нижним индексом k. Точкой решения $\boldsymbol{w} = \boldsymbol{w}_k$ будем счнтать точку, в которой достигается минимум целевой функции $E(\boldsymbol{w})$ и $\boldsymbol{g}(\boldsymbol{w}_k) = 0$, а гессиан $\mathbf{H}(\boldsymbol{w}_k)$ является положительно определенным [39, 42]. При выполнении этих условий функция в любой точке, лежащей в окрестности $\boldsymbol{w}_k$, имеет большее значение, чем в точке $\boldsymbol{w}_k$, поэтому точка $\boldsymbol{w}_k$ является решением, соответствующим критерию минимизации целевой функции.

В процессе поиска минимального значения целевой функции направление поиска $\boldsymbol{p}$ и шаг h подбираются таким образом, чтобы для каждой очередной точки $\boldsymbol{w}_k+1 = \boldsymbol{w}_k + \eta_k \boldsymbol{p}_k$ выполнялось условие $E(\boldsymbol{w}_{k+1}) < E(\boldsymbol{w}_k)$. Поиск минимума продолжается, пока норма градиента не упадет ниже априори заданного значения допустимой погрешности либо пока не будет превышено максимальное время вычислений (количество итераций).

Универсальный оптимизационный алгоритм обучения нейронной сети можно представить в следующем виде (будем считать, что начальное значение оптимизируемого вектора известно и составляет $\boldsymbol{w}_k = \boldsymbol{w}_0$):

1. Проверка сходимости и оптимальности текущего решения $\boldsymbol{w}_k$. Если точка $\boldsymbol{w}_k$ отвечает градиентным условиям остановки процесса – завершение вычислений. В противном случае перейти к п.2.
2. Определение вектора направления оптимизации $\boldsymbol{p}_k$ для точки $\boldsymbol{w}_k$.
3. Выбор величины шага η_k в направлении $\boldsymbol{p}_k$, при котором выполняется условие $E(\boldsymbol{w}_k + \eta_k \boldsymbol{p}_k) < E(\boldsymbol{w}_k)$.

4. Определение нового решения $w_{k+1} = w_k + \eta_k p_k$, а также соответствующих ему значений $E(w_k)$ и $g(w_k)$, а если требуется – то и $\mathbf{H}(w_k)$, и возврат к п.1.

3.4.2. Алгоритм наискорейшего спуска

Если при разложении целевой функции E(w) в ряд Тейлора ограничиться ее линейным приближением, то мы получим алгоритм наискорейшего спуска. Для выполнения соотношения $E(w_{k+1}) < E(w_k)$ достаточно подобрать $g(w_k)^T p < 0$. Условию уменьшения значения целевой функции отвечает выбор вектора направления

$$p_k = -g(w_k). \tag{3.30}$$

Именно выражением (3.30) определяется вектор направления p в методе наискорейшего спуска.

Ограничение слагаемым первого порядка при разложении функции в ряд Тейлора не позволяет использовать информацию о ее кривизне. Это обусловливает медленную сходимость метода (она остается линейной). Указанный недостаток, а также резкое замедление минимизации в ближайшей окрестности точки оптимального решения, когда градиент принимает очень малые значения, делают алгоритм наискорейшего спуска низкоэффективным. Тем не менее с учетом его простоты, невысоких требований к объему памяти и относительно небольшой вычислительной сложности именно этот метод в течение многих лет был и остается в настоящее время основным способом обучения многослойных сетей. Повысить его эффективность удается путем модификации (как правило, эвристической) выражения, определяющего направление. Хорошие результаты приносит применение метода обучения с так называемым моментом. При этом подходе уточнение весов сети ($w_{k+1} = w_k + \Delta w_k$) производится с учетом модифицированной формулы определения значения Δw_k

$$\Delta w_k = \eta_k p_k + \alpha(w_k - w_{k-1}), \tag{3.31}$$

где α – это коэффициент момента, принимающий значения в интервале [0, 1]. Первое слагаемое этого выражения соответствует обычному обучению по методу наискорейшего спуска, тогда как второе учитывает последнее изменение весов и не зависит от фактического значения градиента. Чем больше значение коэффициента α, тем большее значение оказывает показатель момента на подбор весов. Это влияние существенно возрастает на плоских участках целевой функции, а также вблизи локального минимума, где значение градиента близко к нулю.

На плоских участках целевой функции приращение весов (при постоянном значении коэффициента обучения $\eta_k = \eta$) остается приблизительно одним и

тем же. Это означает, что $\Delta w_k = \eta p_k + \alpha \Delta w_k$, поэтому эффективное приращение значений весов можно описать отношением

$$\Delta w_k = \frac{\eta}{1-\alpha} p_k . \qquad (3.32)$$

При значении $\alpha = 0{,}9$ это соответствует 10-кратному увеличению эффективного значения коэффициента обучения и, следовательно, также 10-кратному ускорению процесса обучения.

Вблизи локального минимума показатель момента, не связанный с градиентом, может вызвать слишком большое изменение весов, приводящее к увеличению значения целевой функции и к выходу из "зоны притяжения" этого минимума. При малых значениях градиента показатель момента начинает доминировать в выражении (3.31), что приводит к такому приращению весов Δw_k, которое соответствует увеличению значения целевой функции, позволяющему выйти из зоны локального минимума. Однако показатель момента не должен полностью доминировать на протяжении всего процесса обучения, поскольку это привело бы к нестабильности алгоритма. Для предотвращения такого избыточного доминирования значение целевой функции E контролируется так, чтобы допускать его увеличение только в ограниченных пределах, например не более 4%. При таком подходе, если на очередных (k-м и (k+1)-м) шагах итерации выполняется условие $E(k+1) < 1{,}04\, E(k)$, то изменения игнорируются и считается, что $(w_k - w_{k-1}) = 0$. При этом показатель градиента начинает доминировать над показателем момента и процесс развивается в направлении минимизации, заданном вектором градиента. Следует подчеркнуть, что подбор величины коэффициента момента является непростым делом и требует проведения большого количества экспериментов, имеющих целью выбрать такое значение, которое наилучшим образом отражало бы специфику решаемой проблемы.

3.4.3. Алгоритм переменной метрики

В методе переменной метрики используется квадратичное приближение функции $E(w)$ в окрестности полученного решения w_k. Если в формуле (3.29) ограничиться тремя первыми слагаемыми, то получим:

$$E(w_k + p_k) \cong E(w_k) + g(w_k)^T p_k + \frac{1}{2} p_k^T \mathbf{H}(w_k) p_k + 0(h^3) . \qquad (3.33)$$

Для достижения минимума функции (3.33) требуется, чтобы $\frac{dE(w_k + p_k)}{dp_k} = 0$. При выполнении соответствующего дифференцирования можно получить условие оптимальности в виде

$$g(w_k) + \mathbf{H}(w_k) p_k = 0 .$$

Элементарное преобразование этого выражения дает очевидное решение:

$$p_k = -[\mathbf{H}(w_k)]^{-1} g(w_k). \qquad (3.34)$$

Формула (3.34) однозначно указывает направление p_k, которое гарантирует достижение минимального для данного шага значения целевой функции. Из него следует, что для определения этого направления необходимо в каждом цикле вычислять значение градиента g и гессиана $\mathbf{H}$ в точке известного (последнего) решения w_k.

Формула (3.34), представляющая собой основу ньютоновского алгоритма оптимизации, является чисто теоретическим выражением, поскольку ее применение требует положительной определенности гессиана на каждом шаге, что в общем случае практически неосуществимо. По этой причине в имеющихся реализациях алгоритма, как правило, вместо точно определенного гессиана $\mathbf{H}(w_k)$ используется его приближение $\mathbf{G}(w_k)$. Одним из наиболее популярных считается метод переменной метрики [39, 170]. В соответствии с этим методом на каждом шаге гессиан или обратная ему величина, полученная на предыдущем шаге, модифицируется на величину некоторой поправки. Если прирост вектора w_k и градиента g на двух последовательных шагах итерации обозначить соответственно s_k и r_k, т.е. $s_k = w_k - w_{k-1}$ и $r_k = g(w_k) - g(w_{k-1})$, а матрицу, обратную приближению гессиана $\mathbf{V}_k = [\mathbf{G}(w_k)]^{-1}$, $\mathbf{V}_{k-1} = [\mathbf{G}(w_{k-1})]^{-1}$, обозначить $\mathbf{V}$, то в соответствии с очень эффективной формулой Бройдена–Флетчера–Гольдфарба–Шенно (англ.: *Broyden–Fletcher–Goldfarb–Shanno – BFGS*) процесс уточнения значения матрицы $\mathbf{V}$ можно описать рекуррентной зависимостью [39]:

$$\mathbf{V}_k = \mathbf{V}_{k-1} + \left[1 + \frac{r_k^T \mathbf{V}_{k-1} r_k}{s_k^T r_k}\right] \frac{s_k s_k^T}{s_k^T r_k} - \frac{s_k r_k^T \mathbf{V}_{k-1} + \mathbf{V}_{k-1} r_k s_k^T}{s_k^T r_k}. \qquad (3.35)$$

В другом известном алгоритме Девидона–Флетчера–Пауэлла (англ.: *Davidon–Fletcher–Powell – DFP*) значение гессиана уточняется согласно выражению [39]

$$\mathbf{V}_k = \mathbf{V}_{k-1} + \frac{s_k s_k^T}{s_k^T r_k} - \frac{\mathbf{V}_{k-1} r_k r_k^T \mathbf{V}_{k-1}}{r_k^T \mathbf{V}_{k-1} r_k}. \qquad (3.36)$$

В качестве начального значения обычно принимается $\mathbf{V}_0 = 1$, а первая итерация проводится в соответствии с алгоритмом наискорейшего спуска. Как показано в [39, 170], при начальном значении $\mathbf{V}_0 = 1$ и при использовании направленной минимизации на каждом шаге оптимизации можно обеспечить положительную определенность аппроксимированной матрицы гессиана. Направленная минимизация необходима при реализации как стратегии BFGS, так и DFP, причем в соответствии с проведенными тестами метод BFGS менее чувствителен к различным погрешностям вычислительного процесса. По этой причине, несмотря на несколько большую вычислительную сложность, метод BFGS применяется чаще, чем DFP.

Метод переменной метрики характеризуется более быстрой сходимостью, чем метод наискорейшего спуска. Кроме того, факт положительной определенности гессиана на каждом шаге итерации придает уверенность в том, что выполнение условия $g(w_k) = 0$ действительно гарантирует решение проблемы оптимизации. Именно этот метод считается в настоящее время одним из наиболее эффективных способов оптимизации функции нескольких переменных. Его недостаток состоит в относительно большой вычислительной сложности (связанной с необходимостью расчета в каждом цикле n^2 элементов гессиана), а также в использовании значительных объемов памяти для хранения элементов гессиана, что в случае оптимизации функции с большим количеством переменных может стать серьезной проблемой. По этой причине метод переменной метрики применяется для не очень больших сетей. В частности, с использованием персонального компьютера была доказана его эффективность для сети, содержащей не более тысячи взвешенных связей.

3.4.4. Алгоритм Левенберга–Марквардта

Другим приложением ньютоновской стратегии оптимизации является алгоритм Левенберга–Марквардта [39]. При его использовании точное значение гессиана $\mathbf{H}(w)$ в формуле (3.34) заменяется аппроксимированным значением $G(w)$, которое рассчитывается на основе содержащейся в градиенте информации с учетом некоторого регуляризационного фактора.

Для описания этого метода представим целевую функцию в виде, отвечающем существованию единственной обучающей выборки,

$$E(w) = \frac{1}{2}\sum_{i=1}^{M}[e_i(w)]^2, \tag{3.37}$$

где $e_i = [y_i(w) - d_i]$. При использовании обозначений

$$\mathbf{e}(w) = \begin{bmatrix} e_1(w) \\ e_2(w) \\ \cdots \\ e_M(w) \end{bmatrix}, \quad \mathbf{J}(w) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_n} \\ \cdots & \cdots & \cdots & \\ \frac{\partial e_M}{\partial w_1} & \frac{\partial e_M}{\partial w_2} & \cdots & \frac{\partial e_M}{\partial w_n} \end{bmatrix}, \tag{3.38}$$

вектор градиента и аппроксимированная матрица гессиана, соответствующие целевой функции (3.37), определяются в виде

$$g(w) = [\mathbf{J}(w)]^T \mathbf{e}(w), \tag{3.39}$$

$$\mathbf{G}(w) = [\mathbf{J}(w)]^T \mathbf{J}(w) + \mathbf{R}(w), \tag{3.40}$$

где $\mathbf{R}(w)$ обозначены компоненты гессиана $\mathbf{H}(w)$, содержащие высшие

производные относительно w. Сущность подхода Левенберга–Марквардта состоит в аппроксимации $\mathbf{R}(w)$ с помощью регуляризационного фактора $v\mathbf{1}$, в котором переменная v, называемая параметром Левенберга–Марквардта, является скалярной величиной, изменяющейся в процессе оптимизации. Таким образом, аппроксимированная матрица гессиана на k-м шаге алгоритма приобретает вид:

$$\mathbf{G}(w_k) = [\mathbf{J}(w_k)]^T \mathbf{J}(w_k) + v_k \mathbf{1} . \tag{3.41}$$

В начале процесса обучения, когда фактическое значение w_k еще далеко от искомого решения (велико значение вектора погрешности e), используется значение параметра v_k, намного превышающее собственное значение матрицы $[\mathbf{J}(w_k)]^T\mathbf{J}(w_k)$. В таком случае гессиан фактически подменяется регуляризационным фактором:

$$\mathbf{G}(w_k) \cong v_k \mathbf{1}, \tag{3.42}$$

а направление минимизации выбирается по методу наискорейшего спуска:

$$p_k = -\frac{g(w_k)}{v_k} . \tag{3.43}$$

По мере уменьшения погрешности и приближения к искомому решению величина параметра v_k понижается и первое слагаемое в формуле (3.40) начинает играть все более важную роль.

На эффективность алгоритма влияет грамотный подбор величины v_k. Слишком большое начальное значение v_k по мере прогресса оптимизации должно уменьшаться вплоть до нуля при достижении фактического решения, близкого к искомому. Известны различные способы подбора этого значения, но мы ограничимся описанием только одной оригинальной методики, предложенной Д. Марквардтом [95]. Пусть значения целевой функции на k-м и $(k–1)$-м шагах итерации обозначаются соответственно E_k и E_{k-1}, а значения параметра v на этих же шагах – v_k и v_{k-1}. Коэффициент уменьшения значения v обозначим r, причем $r>1$. В соответствии с классическим алгоритмом Левенберга–Марквардта значение v изменяется по следующей схеме:

- если $E\left(\frac{v_{k-1}}{r}\right) \le E_k$, то принять $v_k = \frac{v_k - 1}{r}$;
- если $E\left(\frac{v_{k-1}}{r}\right) > E_k$ и $E(v_{k-1}) < E_k$, то принять $v_k = v_{k-1}$;
- если $E\left(\frac{v_{k-1}}{r}\right) > E_k$ и $E(v_{k-1}) > E_k$, то увеличить последовательно m раз значение v до достижения $E(v_{k-1} r^m) \le E_k$, одновременно принимая $v_k = v_{k-1} r^m$.

Такая процедура изменения значения v выполняется до момента, в котором так называемый коэффициент верности отображения q, рассчитываемый по формуле

$$q = \frac{E_k - E_{k-1}}{[\Delta w_k]^T g_k + 0{,}5[\Delta w_k]^T G_k \Delta w_k}, \tag{3.44}$$

достигнет значения, близкого к единице. При этом квадратичная аппроксимация целевой функции имеет высокую степень совпадения с истинными значениями, что свидетельствует о близости оптимального решения. В такой ситуации регуляризационный фактор $v_k\mathbf{1}$ в формуле (3.41) может быть опущен ($v_k = 0$), процесс определения гессиана сводится к непосредственной аппроксимации первого порядка, а алгоритм Левенберга–Марквардта превращается в алгоритм Гаусса–Ньютона, характеризующийся квадратичной сходимостью к оптимальному решению.

3.4.5. Алгоритм сопряженных градиентов

В этом методе при выборе направления минимизации не используется информация о гессиане. Направление поиска $\boldsymbol{p}_k$ выбирается таким образом, чтобы оно было ортогональным и сопряженным ко всем предыдущим направлениям $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, ..., $\boldsymbol{p}_{k-1}$. Множество векторов $\boldsymbol{p}_i$, $i = 0, 1, ..., k$, будет взаимно сопряженным относительно матрицы $\mathbf{G}$, если

$$\boldsymbol{p}_i^T \mathbf{G} \boldsymbol{p}_j = 0, \quad i \neq j \ . \tag{3.45}$$

Как показано в [39, 167], вектор $\boldsymbol{p}_k$, удовлетворяющий заданным выше условиям, имеет вид:

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_{k-1} \boldsymbol{p}_{k-1} \ , \tag{3.46}$$

где $\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{w}_k)$ обозначает фактическое значение вектора градиента.

Из формулы (3.46) следует, что новое направление минимизации зависит только от значения градиента в точке решения $\boldsymbol{w}_k$ и от предыдущего направления поиска $\boldsymbol{p}_{k-1}$, умноженного на коэффициент сопряжения β_{k-1}. Этот коэффициент играет очень важную роль, аккумулируя в себе информацию о предыдущих направлениях поиска. Существуют различные правила расчета его значения. Наиболее известны среди них [39, 170]

$$\beta_{k-1} = \frac{\boldsymbol{g}_k^T (\boldsymbol{g}_k - \boldsymbol{g}_{k-1})}{\boldsymbol{g}_{k-1}^T \boldsymbol{g}_{k-1}} \tag{3.47}$$

и

$$\beta_{k-1} = \frac{\boldsymbol{g}_k^T (\boldsymbol{g}_k - \boldsymbol{g}_{k-1})}{-\boldsymbol{p}_{k-1} \boldsymbol{g}_{k-1}} . \tag{3.48}$$

Ввиду накопления погрешностей округления в последовательных циклах вычислений практическое применение метода сопряженных градиентов связано с постепенной утратой свойства ортогональности между векторами направлений минимизации. По этой причине после выполнения n итераций (значение n рассчитывается как функция от количества переменных, подлежащих оптимизации) производится рестарт процедуры, на первом шаге которой направление минимизации из точки полученного решения выбирается по алгоритму наискорейшего спуска. Метод сопряженных градиентов имеет

сходимость, близкую к линейной, и он менее эффективен, чем метод переменной метрики, однако заметно быстрее метода наискорейшего спуска. Он широко применяется как единственно эффективный алгоритм оптимизации при весьма значительном количестве переменных, которое может достигать нескольких десятков тысяч. Благодаря невысоким требованиям к памяти и относительно низкой вычислительной сложности метод сопряженных градиентов позволяет успешно решать очень серьезные оптимизационные задачи.

3.5. Подбор коэффициента обучения

Алгоритмы, представленные в предыдущем подразделе, позволяют определить только направление, в котором уменьшается целевая функция, но не говорят ничего о величине шага, при котором эта функция может получить минимальное значение. После выбора правильного направления $\boldsymbol{p}_k$ следует определить на нем новую точку решения $\boldsymbol{w}_{k+1}$, в которой будет выполняться условие $E(\boldsymbol{w}_{k+1}) < E(\boldsymbol{w}_k)$. Необходимо подобрать такое значение η_k, чтобы новое решение $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta_k \boldsymbol{p}_k$ лежало как можно ближе к минимуму функции $E(\boldsymbol{w})$ в направлении $\boldsymbol{p}_k$. Грамотный подбор коэффициента η_k оказывает огромное влияние на сходимость алгоритма оптимизации к минимуму целевой функции. Чем сильнее величина η_k отличается от значения, при котором $E(\boldsymbol{w})$ достигает минимума в выбранном направлении $\boldsymbol{p}_k$, тем большее количество итераций потребуется для поиска оптимального решения. Слишком малое значение η не позволяет минимизировать целевую функцию за один шаг и вызывает необходимость повторно двигаться в том же направлении. Слишком большой шаг приводит к "перепрыгиванию" через минимум функции и фактически заставляет возвращаться к нему.

Существуют различные способы подбора значения η, называемого в теории нейронных сетей *коэффициентом обучения*. Простейший из них (относительно редко применяемый в настоящее время, главным образом для обучения в режиме "онлайн") основан на фиксации постоянного значения η на весь период оптимизации. Этот способ практически используется только совместно с методом наискорейшего спуска. Он имеет низкую эффективность, поскольку значение коэффициента обучения никак не зависит от вектора фактического градиента и, следовательно, от направления $\boldsymbol{p}$ на данной итерации. Величина η подбирается, как правило, раздельно для каждого слоя сети с использованием различных эмпирических зависимостей. Один из подходов состоит в определении минимального значения коэффициента η для каждого слоя по формуле [72]

$$\eta \le \min\left(\frac{1}{n_i}\right), \tag{3.49}$$

где n_i обозначает количество входов i-го нейрона в слое.

Другой более эффективный метод основан на адаптивном подборе коэффициента η с учетом фактической динамики величины целевой функции в результате обучения. В соответствии с этим методом стратегия изменения

значения η определяется путем сравнения суммарной погрешности ε на i-й итерации с ее предыдущим значением, причем ε рассчитывается по формуле

$$\varepsilon = \sqrt{\sum_{j=1}^{M}(y_j - d_j)^2}. \qquad (3.50)$$

Для ускорения процесса обучения следует стремиться к непрерывному увеличению η при одновременном контроле прироста погрешности ε по сравнению с ее значением на предыдущем шаге. Незначительный рост этой погрешности считается допустимым.

Если погрешности на (i–1) и i-й итерациях обозначить соответственно ε_{i-1} и ε_i, а коэффициенты обучения на этих же итерациях – η_{i-1} и η_i, то в случае $\varepsilon_i > k_w\varepsilon_{i-1}$ (k_w – коэффициент допустимого прироста погрешности) значение η должно уменьшаться в соответствии с формулой

$$\eta_{i+1} = \eta_i \rho_d, \qquad (3.51)$$

где ρ_d – коэффициент уменьшения η. В противном случае, когда $\varepsilon_i \le k_w\varepsilon_{i-1}$, принимается

$$\eta_{i+1} = \eta_i \rho_i, \qquad (3.52)$$

где ρ_i – коэффициент увеличения η. Несмотря на некоторое возрастание объема вычислений (необходимых для дополнительного расчета значений ε), возможно существенное ускорение процесса обучения. Например, реализация представленной стратегии в программе *MATLAB* [27] со значениями k_w = 1,41, ρ_d = 0,7, ρ_i = 1,05 позволила в несколько раз ускорить обучение при решении проблемы аппроксимации нелинейных функций.

Интересно проследнть характер изменения коэффициента η в процессе обучения. Как правило, на начальных этапах доминирует тенденция к его увеличению, однако при достижении некоторого квазистационарного состояния величина η постепенно уменьшается, но не монотонно, а циклически возрастая и понижаясь в следующих друг за другом циклах.

Однако необходимо подчеркнуть, что адаптивный метод подбора η сильно зависит от вида целевой функции и значений коэффициентов k_w, ρ_d и ρ_i. Значения, оптимальные для функции одного вида, могут замедлять процесс обучения при использовании другой функции. Поэтому при практической реализации этого метода следует обращать внимание на механизмы контроля и управления значениями коэффициентов, подбирая их в соответствии со спецификой решаемой задачи.

Наиболее эффективный, хотя и наиболее сложный, метод подбора коэффициента обучения связан с направленной минимизацией целевой функции в выбранном заранее направлении $\boldsymbol{p}_k$. Необходимо так подобрать скалярное значение η_k, чтобы новое решение $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta_k\boldsymbol{p}_k$ соответствовало минимуму целевой функции в данном направлении $\boldsymbol{p}_k$. В действительности получаемое решение $\boldsymbol{w}_{k+1}$ только с определенным приближением может считаться настоящим

минимумом. Это результат компромисса между объемом вычислений и влиянием величины η_k на сходимость алгоритма.

Среди наиболее популярных способов направленной минимизации можно выделить *безградиентные* и *градиентные методы*. В *безградиентных методах* используется только информация о значениях целевой функции, а ее минимум достигается в процессе последовательного уменьшения диапазона значений вектора w. Примерами могут служить методы деления пополам, золотого сечения либо метод Фибоначчи [39, 170], различающиеся способом декомпозиции получаемых поддиапазонов.

Заслуживает внимания метод аппроксимации целевой функции $E(w)$ в предварительно выбранном направлении p_k с последующим расчетом минимума, получаемого таким образом, функции одной переменной η. Выберем для аппроксимации многочлен второго порядка вида

$$E(w) \to P_2(\eta) = a_2\eta^2 + a_1\eta + a_0, \qquad (3.53)$$

где a_2, a_1 и a_0 обозначены коэффициенты, определяемые в каждом цикле оптимизации. Выражение (3.53) – это многочлен P_2 одной скалярной переменной η. Если для расчета входящих в P_2 коэффициентов используются три произвольные точки w_1, w_2 и w_3, лежащие в направлении p_k, т.е. $w_1 = w + \eta_1 p_k$, $w_2 = w + \eta_2 p_k$, $w_3 = w + \eta_3 p_k$ (в этом выражении w обозначено предыдущее решение), а соответствующие этим точкам значения целевой функции $E(w)$ обозначены $E_1 = E(w_1)$, $E_2 = E(w_2)$, $E_3 = E(w_3)$, то

$$P_2(\eta_1) = E_1, P_2(\eta_2) = E_2, P_2(\eta_3) = E_3 . \qquad (3.54)$$

Коэффициенты a_2, a_1 и a_0 многочлена P_2 рассчитываются в соответствии с системой линейных уравнений, описываемых в (3.54). Для определения минимума этого многочлена его производная $\frac{dP_2}{d\eta} = 2a_2\eta + a_1$ приравнивается к нулю, что позволяет получить значение η в виде $\eta_{\min} = \frac{-a_1}{2a_2}$. После подстановки выражений для E_1, E_2 и E_3 в формулу расчета $\eta_{\min}$ получаем:

$$\eta_{\min} = \eta_2 - \frac{1}{2}\frac{(\eta_2 - \eta_1)^2(E_2 - E_3) - (\eta_2 - \eta_3)^2(E_2 - E_1)}{(\eta_2 - \eta_1)(E_2 - E_3) - (\eta_2 - \eta_3)(E_2 - E_1)} . \qquad (3.55)$$

Однако лучшим решением считается применение градиентных методов, в которых, кроме значения функции, учитывается также и ее производная вдоль направляющего вектора p_k. Они позволяют значительно ускорить достижение минимума, поскольку используют информацию о направлении уменьшения величины целевой функции. В такой ситуации применяется, как правило, аппроксимирующий многочлен третьего порядка

$$P_3(\eta) = a_3\eta^3 + a_2\eta^2 + a_1\eta + a_0 . \qquad (3.56)$$

Значения четырех коэффициентов a_i этого многочлена можно получить исходя из информации о величине функции и ее производной всего лишь в двух

точках. Если приравнять к нулю производную многочлена относительно η, то можно получить формулу для расчета $\eta_{\min}$ в виде

$$\eta_{\min} = \frac{-a_2 + \sqrt{a_2^2 - 3a_2 a_1}}{3a_3}. \qquad (3.57)$$

Более подробное описание этого подхода можно найти в первоисточниках, посвященных теории оптимизации [39,170].

Алгоритм обучения многослойного персептрона реализован в Институте теоретической электротехники и электроизмерений Варшавского политехнического университета на языке *C++* в виде программы *Netteach* [144]. Для выбора направления минимизации в нем применяется метод переменной метрики или классический метод сопряженных градиентов, а для расчета оптимального значения коэффициента обучения – аппроксимация многочленом третьего порядка. Программа позволяет работать с произвольным количеством слоев, причем функция активации нейронов каждого слоя задается индивидуально. На выбор предлагаются функции: сигмоидальная униполярная (*Sigm*), сигмоидальная биполярная (*Bip*), а также линейная (*Lin*). Градиент рассчитывается с применением метода потоковых графов.

3.6. Эвристические методы обучения сети

Помимо алгоритмов обучения, реализующих апробированные методы оптимизации нелинейной целевой функции (такие, как методы переменной метрики, Левенберга–Марквардта либо сопряженных градиентов), создано огромное количество алгоритмов эвристического типа, представляющих собой в основном модификацию методов наискорейшего спуска или сопряженных градиентов. Подобные модификации широко известных алгоритмов связаны с внесением в них некоторых изменений, ускоряющих (по мнению авторов) процесс обучения. Как правило, такие методы не имеют серьезного теоретического обоснования, особенно это относится к процедуре подбора управляющих параметров. Однако в таких алгоритмах реализуется личный опыт работы авторов с нейронными сетями. К наиболее известным эвристическим алгоритмам относится *Quickprop* С. Фальмана [33] (использованный среди прочих и в программе *Cascor* [34]), а также RPROP М. Ридмиллера и Х. Брауна [133], реализованный в программе *SNNS* [178].

3.6.1. Алгоритм Quickprop

Quickprop содержит элементы, предотвращающие зацикливание в точке неглубокого локального минимума, возникающего в результате работы нейрона на фазе насыщения сигмоидальной кривой, где из-за близости к нулю производной функции активации процесс обучения практически прекращается.

Вес w_{ij} на k-м шаге алгоритма изменяется согласно правилу

$$\Delta w_{ij}(k) = -\eta_k \left[\frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \right] + \alpha_{ij}^{(k)} \Delta w_{ij}(k-1) . \qquad (3.58)$$

Первое слагаемое $\frac{\partial E}{\partial w_{ij}}$ соответствует оригинальному алгоритму наискорейшего спуска, последнее слагаемое, – фактору момента, а средний член γw_{ij} предназначен для минимизации абсолютных значений весов. Коэффициент γ, имеющий обычно малую величину (типовое значение $\gamma = 10-4$), – это фактор, приводящий к уменьшению весов вплоть до возможного разрыва соответствующих взвешенных связей. Константа η_k – это коэффициент обучения, который в данном алгоритме может иметь ненулевое значение η_0 (как правило, $0{,}01 \le \eta_0 \le 0{,}6$) на старте процесса обучения, когда $\Delta w_{ij}(k-1) = 0$ либо когда $\left[\frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k)\right] \Delta w_{ij} > 0$, или нулевое значение – в противном случае.

Важную роль в алгоритме Quickprop играет фактор момента, который адаптируется к текущим результатам процесса обучения. В соответствии с алгоритмом Фальмана коэффициент момента α_{ij} подбирается индивидуально для каждого веса по правилу

$$\alpha_{ij}^{(k)} = \begin{cases} \alpha_{\max}, \text{ где } \beta_{ij}(k) > \alpha_{\max}, \text{ где } S_{ij}(k)\Delta w_{ij}(k-1)\ \beta_{ij}(k) < 0 \\ \beta_{ij}(k) \end{cases}, \qquad (3.59)$$

причем

$$S_{ij}(k) = \frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) , \qquad (3.60)$$

$$\beta_{ij}^{(k)} = \frac{S_{ij}(k)}{S_{ij}(k-1) - S_{ij}(k)} . \qquad (3.61)$$

Константа $\alpha_{\max}$ – это максимальное значение коэффициента момента, которая по предложению Фальмана принимается равной $\alpha_{\max} = 1{,}75$.

Также известна упрощенная версия алгоритма Quickprop, в которой значения весов изменяются в соответствии с правилом

$$\Delta w_{ij}(k) = \begin{cases} \alpha_{ij}(k)\Delta w_{ij}(k-1) & \text{для } \Delta w_{ij}(k-1) \neq 0 \\ \eta_0 \frac{\partial E}{\partial w_{ij}} \end{cases} ; \qquad (3.62)$$

$$\alpha_{ij}(k) = \min\left\{ \frac{S_{ij}(k)}{S_{ij}(k-1) - S_{ij}(k)}, \alpha_{\max} \right\}, \qquad (3.63)$$

где $S_{ij}(k) = \frac{\partial E(w(k))}{\partial w_{ij}}$. В нем уменьшено количество управляющих параметров и упрощена сама формула уточнения значений весов. Согласно представленным в [159] результатам эффективность модифицированного алгоритма сравнима с оригинальным алгоритмом Фальмана.

3.6.2. Алгоритм RPROP

Другой простой эвристический алгоритм, демонстрирующий высокую эффективность обучения, – это алгоритм М. Ридмиллера и Х. Брауна, называемый RPROP (англ.: *Resilient back PROPagation*) [133, 178]. В этом алгоритме при уточнении весов учитывается только знак градиентной составляющей, а ее значение игнорируется:

$$\Delta w_{ij}(k) = -\eta_{ij}(k)\,\mathrm{sgn}\left(\frac{\partial E(w(k))}{\partial w_{ij}}\right). \qquad (3.64)$$

Коэффициент обучения подбирается индивидуально для каждого веса w_{ij} с учетом изменения значения градиента:

$$\begin{cases} \min\left(a\eta_{ij}(k-1),\ \eta_{\max}\right) & \text{для } S_{ij}(k)\,S_{ij}(k-1) > 0 \\ \max\left(b\eta_{ij}(k-1),\ \eta_{\min}\right) & \text{для } S_{ij}(k)\,S_{ij}(k-1) < 0 \\ \eta_{ij}(k-1) & \text{в остальных случаях} \end{cases}, \qquad (3.65)$$

где $S_{ij}(k) = \frac{\partial E(w(k))}{\partial w_{ij}}$, a и b – константы: $a=1{,}2$; $b=0{,}5$. Минимальное и максимальное значения коэффициента обучения обозначены соответственно $\eta_{\min}$ и $\eta_{\max}$; для алгоритма RPROP они составляют $\eta_{\min} = 10^{-6}$ и $\eta_{\max} = 50$ [178]. Функция sgn( ) принимает значение, равное знаку градиента.

Алгоритм RPROP, в котором игнорируется информация о значении градиента, позволяет значительно ускорить процесс обучения в тех случаях, когда угол наклона целевой функции невелик. В соответствии со стратегией подбора весов, если на двух последовательных шагах знак градиента не изменяется, предусматривается увеличение коэффициента обучения. Если же знак градиента изменяется, то коэффициент обучения уменьшается.

3.7. Сравнение эффективности алгоритмов обучения

Эффективность алгоритмов обучения проверяется на определенных тестах, соответствующих принятым мировым стандартам. Такими стандартными тестами, в частности, считаются задача логистики, задача парности, кодирование и декодирование двоичных данных, аппроксимация определенного вида нелинейной функции, задача двух спиралей и многие другие [155]. Различные алгоритмы сравниваются по количеству циклов обучения, количеству расчетов значения целевой функции, количеству знакопеременных произведений, чувствительности к локальным минимумам и т.п.

Например, решение задачи логистики состоит в предсказании очередного значения x_{n+1} случайной цифровой последовательности по предыдущему значению x_n. Этап обучения сети, имеющей, к примеру, структуру 1-5-1 (1

входной узел, 5 скрытых нейронов, 1 выходной нейрон), имеет целью сформировать такие значения весов, чтобы реализовать логистическое отображение

$$x_{n+1} = r\,x_n(1 - x_n)$$

для $0 \le x_n \le 1$, которое при значении $r = 4$ будет демонстрировать свойства случайной последовательности.

В свою очередь, тестовая задача кодирования двоичных векторов заключается в таком подборе весов сети, чтобы при размерности N входного вектора закодировать его с помощью q нейронов скрытого слоя, с последующим декодированием в выходном слое к исходному виду. Обучающие векторы состоят из одной единицы и $(N-1)$ нулей. Каждому сформированному таким образом входному вектору сопоставляется идентичный выходной вектор.

По результатам многих имитационных экспериментов можно утверждать, что наименее эффективным является алгоритм наискорейшего спуска, особенно при постоянном шаге обучения. Стратегия выбора этого шага имеет ключевое значение для эффективности алгоритма. Чем ближе минимальное значение целевой функции в направлении $\boldsymbol{p}$, тем лучше результаты обучения на отдельных итерациях и тем выше конечный результат. С этой точки зрения наибольший эффект обеспечивает метод направленной минимизации, применяемый в каждом оптимизационном цикле для выбора оптимального размера шага. Однако при сравнении эффективности различных методов следует принимать во внимание объем дополнительных вычислений, требуемых для расчета оптимальной величины η.

Эффективность различных алгоритмов сравнивается либо путем измерения среднего времени, требуемого для решения конкретной задачи, либо по количеству циклов обучения, либо по количеству знакопеременных операций (по вычислительной сложности алгоритма). Эти характеристики могут существенно отличаться в зависимости от характера тестовой задачи, объема обучающих данных, размерности нейронной сети, используемого вычислительного оборудования, а также деталей реализации отдельных этапов алгоритма. Поэтому невозможно дать однозначный ответ на вопрос: какой алгоритм считается абсолютно лучшим?

В табл. 3.2 представлены результаты, полученные на компьютере Macintosh Powerbook 1400 при использовании прикладного пакета "Neural Networks" программы Matlab [27], позволяющие сравнить длительность, количество циклов обучения и вычислительную сложность различных алгоритмов. В ходе экспериментов обучался многослойный персептрон со структурой 1-10-1, предназначенный для аппроксимации 41 пары обучающих одномерных данных. Все алгоритмы были реализованы в инструментальной среде одной и той же программы Matlab, что создало основу для получения объективных оценок.

Таблица 3.2

Сравнение эффективности алгоритмов обучения

Алгоритм	Время, (с)	Количество циклов	Количество операций, $\times 10^6$
Наискорейшего спуска с адаптируемым шагом	57,71	980	2,50
Сопряженных градиентов	19,16	89	0,75
Переменной метрики BFGS	10,86	44	1,02
Левенберга–Марквардта	1,87	6	0,46
RPROP	12,96	185	0,56

Получены усредненные результаты по 20 процессам обучения. На малой сети, использованной в ходе тестирования, наибольшую эффективность продемонстрировал алгоритм Левенберга–Марквардта (наименьшее время обучения, наименьшее количество циклов обучения, наименьшая вычислительная сложность). Следующими по количеству циклов и времени обучения идут алгоритмы переменной метрики BFGS и сопряженных градиентов. Самую низкую эффективность в ходе тестирования показал алгоритм наискорейшего спуска (все показатели имеют наихудшие значения). Эвристический алгоритм RPROP в этом соревновании выглядел совсем неплохо – он занял второе место по вычислительной сложности.

По результатам многочисленных и различных тестов сделан общий вывод, что ньютоновские алгоритмы, в том числе методы переменной метрики и Левенберга–Марквардта, по эффективности доминируют как над методами наискорейшего спуска, так и над методом сопряженных градиентов. Однако это очевидное превосходство исчезает при значительном увеличении размеров сети. Уже при 1000 взвешенных связей наиболее эффективным становится, как правило, метод сопряженных градиентов.

3.8. Элементы глобальной оптимизации

При обучении нейронных сигмоидальных сетей, основанном на минимизации значения целевой функции, даже при решении относительно простых технических задач необходимо учитывать возможность появления большого количества локальных минимумов. Проблемы обучения таких сетей хорошо иллюстрирует следующий пример. Рассмотрим сеть, состоящую из одного нейрона, связанного с входным узлом дугой с весом w_1 и с единичным поляризатором дугой с весом w_0. Нейрон выполняет функцию классификатора данных, относящихся к двум классам. Имеются обучающие данные в виде (–4, 1), (–3, 1), (–2, 1), (–1, 1), (1, –1), (3, –1), (4, –1). При использовании линейной функции активации нейрона график зависимости целевой функции от весов w_0

и w_1 принимает вид выпуклой кривой (рис. 3.8 *а*), единственный минимум которой можно легко рассчитать при любых начальных условиях обучения. Переход к сигмоидальной функции активации принципиально меняет форму целевой функции. Эта ситуация демонстрируется на рис. 3.8 *б*, причем сигмоидальная функция активации задана в виде гиперболического тангенса. На графике видны многочисленные плоские участки и множество локальных минимумов, которые осложняют процесс обучения и представляют собой ловушки на пути к глобальному минимуму, в котором целевая функция принимает наименьшее значение.

Хотя графики целевой функции, представленные на рис. 3.8, относятся к простейшей однонейронной сети, они хорошо иллюстрируют проблемы, создаваемые нелинейностью функции активации. Увеличение размеров сети создает еще большие сложности, поскольку количество локальных минимумов также возрастает.

Все представленные ранее методы обучения нейронных сетей являются локальными. Они ведут к одному из локальных минимумов целевой функции, лежащему в окрестности точки начала обучения. Только в ситуации, когда значение глобального минимума известно, удается оценить, находится ли найденный локальный минимум в достаточной близости от искомого решения. Если локальное решение признается неудовлетворительным, следует повторить процесс обучения при других начальных значениях весов и с другими управляющими параметрами. Можно либо проигнорировать полученное решение и начать обучение "с чистого листа" при новых (как правило, случайных) значениях весов, либо изменить случайным образом найденное локальное решение и продолжить обучение сети. Последняя методика, имеющая английское название "*jog of weights*" (встряхивание весов), представляется вполне разумной, поскольку ее применение позволяет использовать полученные ранее результаты обучения [72].

Случайное приращение весов соответствует переходу из точки локального минимума в иную точку пространства целевой функции. Вследствие случайного характера таких приращений переход в новую точку связан с определенной вероятностью того, что возобновление процесса обучения выведет поиск из "сферы притяжения" локального минимума. Случайный выбор значений весов, применяемый как в начале обучения, так и для вывода решения из зоны локального минимума, играет роль стохастического алгоритма, взаимодействующего с детерминированным алгоритмом обучения сети. Однако возмущение весов, вызванное добавлением случайных поправок к ранее найденному решению, не вызывает длительной потери предыдущих результатов обучения. Сеть проявляет интересную способность "запоминания" наилучших результатов и после кратковременной амнезии быстро восстанавливается, а затем и (чаще всего) улучшает предыдущие показатели.

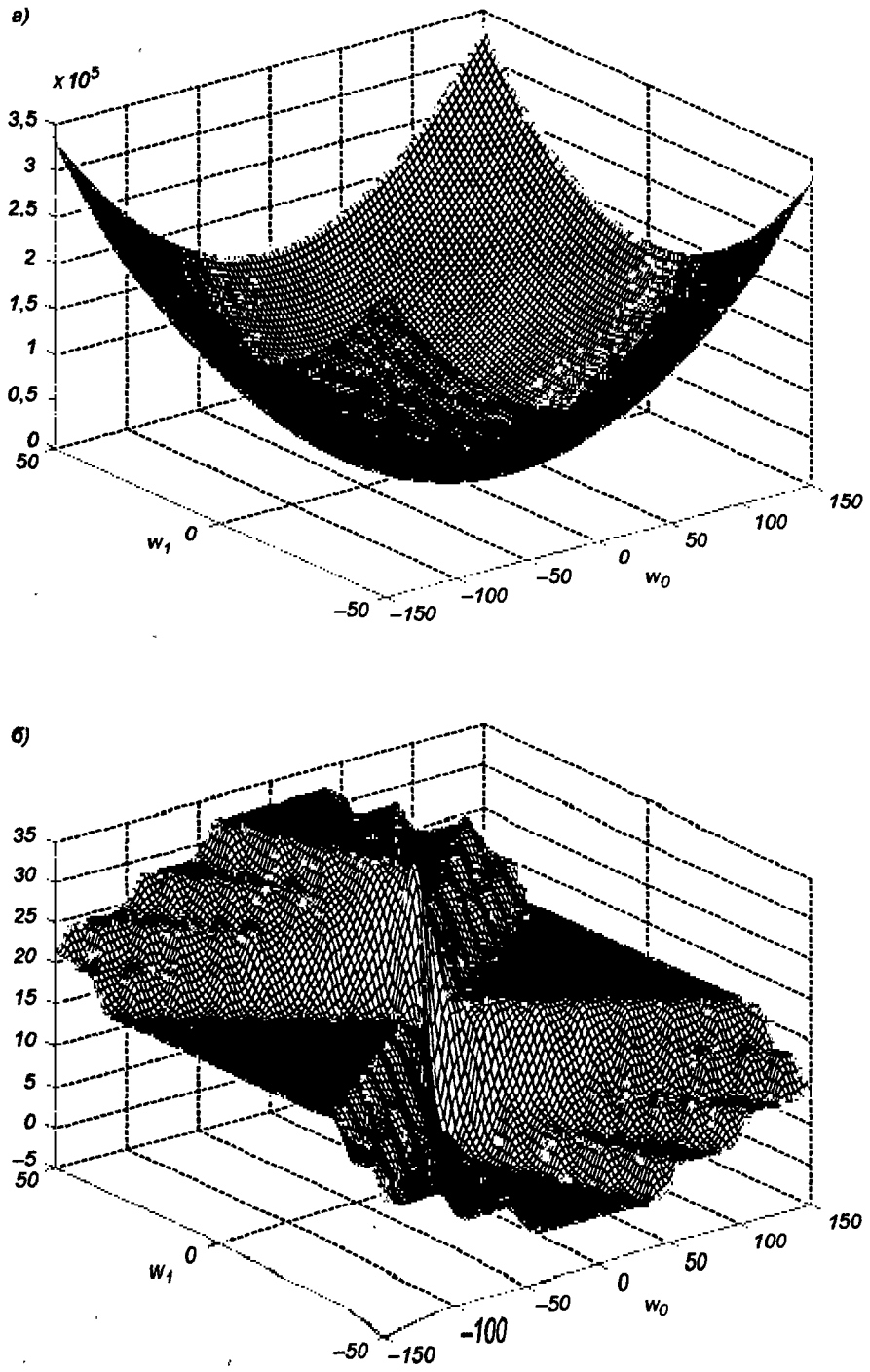

Рис. 3.8. Пример графика целевой функции нейронной сети:
а) линейная функция активации; *б*) сигмоидальная функция активации

При решении реальных как технических, так и экономических задач в общем случае даже приблизительная оценка глобального минимума оказывается неизвестной. По этой причине возникает необходимость применения методов глобальной оптимизации. Из множества разработанных в этой области подходов выберем и подробно рассмотрим два: *метод имитации отжига*[1] и *генетические алгоритмы* [41, 149].

3.8.1. Алгоритм имитации отжига

Метод имитации отжига основан на идее, заимствованной из статической механики. Он отражает поведение материального тела при отвердевании с применением процедуры отжига (управляемого охлаждения. – *Примеч. ред.*) при температуре, последовательно понижаемой до нуля. Как показали исследования, при отвердевании расплавленного материала его температура должна уменьшаться постепенно, вплоть до момента полной кристаллизации. Если процесс остывания протекает слишком быстро, образуются значительные нерегулярности структуры материала, которые вызывают внутренние напряжения. В результате общее энергетическое состояние тела, зависящее от его внутренней напряженности, остается на гораздо более высоком уровне, чем при медленном охлаждении. Быстрая фиксация энергетического состояния тела на уровне выше нормального аналогична сходимости оптимизационного алгоритма к точке локального минимума. Энергия состояния тела соответствует целевой функции, а абсолютный минимум этой энергии – глобальному минимуму. В процессе медленного управляемого охлаждения, называемого отжигом, кристаллизация тела сопровождается глобальным уменьшением его энергии, однако допускаются ситуации, в которых она может на какое-то время возрастать (в частности, при подогреве тела для предотвращения слишком быстрого его остывания – *Примеч. ред.*). Благодаря допустимости кратковременного повышения энергетического уровня возможен выход из ловушек локальных минимумов, которые возникают при реализации процесса. Только понижение температуры тела до абсолютного нуля делает невозможным какое-либо самостоятельное повышение его энергетического уровня. В этом случае любые внутренние изменения ведут только к уменьшению общей энергии тела.

В реальных процессах кристаллизации твердых тел температура понижается ступенчатым образом. На каждом уровне она какое-то время поддерживается постоянной, что необходимо для обеспечения термического равиовесия. На протяжении всего периода, когда температура остается выше абсолютного нуля, она может как понижаться, так и повышаться. За счет удержания температуры процесса поблизости от значения, соответствующего непрерывно снижающемуся уровню термического рав-

[1] Оригинальное английское название *simulated annealing*.

новесия, удается обходить ловушки локальных минимумов, что при достижении нулевой температуры позволяет получить и минимальный энергетический уровень.

Метод имитации отжига представляет собой алгоритмический аналог физического процесса управляемого охлаждения. Предложенный Н. Метрополисом в 1953 г. [61, 71] и доработанный многочисленными последователями, он в настоящее время считается одним из немногих алгоритмов, позволяющих практически находить глобальный минимум функции нескольких переменных.

Классический алгоритм имитации отжига можно описать следующим образом [61].

1. Запустить процесс из начальной точки w при заданной начальной температуре $T = T_{max}$.
2. Пока $T > 0$, повторить L раз следующие действия:
   - выбрать новое решение w' из окрестности w;
   - рассчитать изменение целевой функции $\Delta = E(w') - E(w)$;
   - если $\Delta \le 0$, принять $w = w'$; в противном случае (при $\Delta > 0$) принять, что $w = w'$ с вероятностью $\exp(-\Delta / T)$ путем генерации случайного числа R из интервала (0, 1) с последующим сравнением его со значением $\exp(-\Delta / T)$; если $\exp(-\Delta / T) > R$, принять новое решение $w = w'$; в противном случае проигнорировать его.
3. Уменьшить температуру ($T \leftarrow rT$) с использованием коэффициента уменьшения r, выбираемого из интервала (0 , 1), и вернуться к п. 2.
4. После снижения температуры до нулевого значения провести обучение сети любым из представленных выше детерминированных методов, вплоть до достижения минимума целевой функции.

В описании алгоритма в качестве названия параметра, влияющего на вероятность увеличения значения целевой функции, используется выбранный его автором Н. Метрополисом термин "температура", хотя с формальной точки зрения приведенная модель оптимизации является только математической аналогией процесса отжига. Алгоритм имитации отжига выглядит концептуально несложным и логически обоснованным. В действительности приходится решать много фундаментальных проблем, которые влияют на его практическую применимость. Первой следует назвать проблему *длительности имитации*. Для повышения вероятности достижения глобального минимума длительность отжига (представляемая количеством циклов L, повторяемых при одном и том же значении температуры) должна быть достаточно большой, а коэффициент уменьшения температуры r – низким. Это увеличивает продолжительность процесса моделирования, что может дискредитировать его с позиций практической целесообразности.

Возникает также и проблема *конкурентоспособности метода* по сравнению, например, с методами локальной оптимизации в связи с возможностью многократного возобновления процесса из различных точек в пространстве

параметров. При таком подходе грамотная статистическая обработка позволяет с высокой вероятностью и достаточно быстро локализовать зону глобального минимума и достичь его с применением технологии детерминированной оптимизации.

Огромное влияние на эффективность метода имитации отжига оказывает *выбор* таких *параметров*, как начальная температура T_{max}, коэффициент уменьшения температуры r и количество циклов L, выполняемых на каждом температурном уровне.

Максимальная температура подбирается по результатам многочисленных предварительных имитационных экспериментов. На их основе строится распределение вероятности стохастических изменений текущего решения при конкретных значениях температуры (зависимость $A = f(T)$). В последующем, задаваясь процентным значением допустимости изменений в качестве порогового уровня, из сформированного распределения можно найти искомую начальную температуру. Главной проблемой остается определение порогового уровня, оптимального для каждой реализации процесса имитации отжига. Для отдельных практических задач этот уровень может иметь различные значения, однако общий диапазон остается неизменным. Как правило, начальная температура подбирается так, чтобы обеспечить реализацию порядка 50% последующих случайных изменений решения. Поэтому знание предварительного распределения вероятностей таких изменений позволяет получить приблизительную оценку начальной температуры.

Методики выбора как максимального количества циклов L для конкретных температурных уровней, так и определение значения коэффициента уменьшения температуры r не столь однозначны. При подборе этих параметров приходится учитывать динамику изменения величины целевой функции в зависимости от количества выполненных циклов обучения.

Большая часть вычислительных ресурсов расходуется на начальной стадии процесса, когда средняя скорость изменения целевой функции невелика и прогресс оптимизации минимален. Это "высокотемпературная" стадия имитационного процесса. Быстрее всего величина целевой функции уменьшается на средней стадии процесса при относительно небольшом количестве приходящихся на нее итераций. Завершающая стадия процесса имеет стабилизационный характер. На ней независимо от количества итераций прогресс оптимизации становится практически незаметным. Такое наблюдение позволяет существенно редуцировать начальную стадию отжига без снижения качества конечного результата. Модификации обычно подвергается количество циклов, выполняемых при высоких температурах, – оно сокращается в случае, когда оказался выполненным весь запланированный объем изменений текущего решения. Такой подход позволяет сэкономить до 20% времени.

Исключение последней, плоской части характеристической кривой целевой функции также возможно. В соответствии с обычным критерием остановки

алгоритма, если при нескольких последовательных снижениях температуры (типовое значение 5) не регистрируется уменьшение величины целевой функции, то процесс останавливается, а наилучшее достигнутое решение считается глобальным минимумом. Дальнейшее уменьшение критерия остановки не рекомендуется, поскольку оно ведет к снижению вероятности достижения глобального минимума. В то же время заметное влияние на конечную стадию процесса оказывают коэффициент понижения температуры r и количество циклов L. Ее длительность удается сократить более частым изменением температуры при уменьшении количества циклов, но при сохранении неизменным общего объема итераций.

Еще одна проблема связана с определением длительности моделирования процесса отжига, пропорциональной суммарному количеству итераций. Поскольку отводимое для оптимизации время всегда ограничено, все его можно потратить либо на одну реализацию процесса с соответствующим удлинением циклов, либо сократить длительность всех циклов, а за счет этого выполнить несколько реализаций и принять в качестве результата наилучшее решение. В ходе различных компьютерных экспериментов установлено, что при малом лимите времени лучшие результаты дает единичная реализация. Если же моделирование может быть более длительным, статистически лучшие результаты достигаются при многократной реализации процесса имитации отжига, при больших (близких к 1) значениях коэффициента r.

Однако наибольшее ускорение процесса имитации отжига можно достичь путем замены случайных начальных значений весов w тщательно подобранными значениями с использованием любых доступных способов предварительной семантической обработки исходных данных. В такой ситуации в зависимости от количества оптимизируемых весов и степени оптимальности начальных значений удается добиться даже многократного сокращения времени моделирования.

Таким образом, метод имитации отжига оказывается особенно удачным для полимодальных комбинаторных проблем с весьма большим количеством возможных решений, например, для машины Больцмана, в которой каждое состояние системы считается допустимым. При решении наиболее распространенных задач обучения многослойных нейронных сетей наилучшие результаты в общем случае достигаются применением стохастически управляемого метода повторных рестартов совместно с детерминированными алгоритмами, приведенными в предыдущем подразделе.

3.8.2. Генетические алгоритмы

Идея генетических алгоритмов была предложена Дж. Холландом в 70-х годах XX в. [41], а их интенсивное развитие и практическая реализация для численных оптимизационных расчетов были инициированы Д. Гольдбергом [41]. Эти алгоритмы имитируют процессы наследования свойств живыми организмами и генерируют последовательности новых векторов w, содержащие оптимизи-

рованные переменные: $\boldsymbol{w} = [w_1, w_2,\ldots,w_n]^T$. При этом выполняются операции трех видов: селекция, скрещивание и мутация.

Отдельные компоненты вектора $\boldsymbol{w}$ могут кодироваться в двоичной системе либо натуральными числами [41]. При двоичном кодировании применяется обычный код, или код Грея. После соответствующего масштабирования отдельные биты представляют конкретные значения параметров, подлежащих оптимизации. Каждому вектору $\boldsymbol{w}$ сопоставляется определенное значение целевой функции. Генетические операции (селекция, скрещивание и мутация) выполняются для подбора таких значений переменных w_i вектора $\boldsymbol{w}$, при которых максимизируется величина так называемой функции приспособленности (англ.: *fitness function*). Функция приспособленности $F(\boldsymbol{w})$ определяется через целевую функцию $E(\boldsymbol{w})$ путем ее инвертирования (целевая функция минимизируется, а функция приспособленности максимизируется). Например, она может иметь вид: $F(\boldsymbol{w}) = -E(\boldsymbol{w})$.

На начальной стадии выполнения генетического алгоритма инициализируется определенная популяция хромосом (векторов $\boldsymbol{w}$). Она формируется случайным образом, хотя применяются также и самонаводящиеся способы (если их можно определить заранее). Размер популяции, как правило, пропорционален количеству оптимизируемых параметров. Слишком малая популяция хромосом приводит к замыканию в неглубоких локальных минимумах. Слишком большое их количество чрезмерно удлиняет вычислительную процедуру и также может не привести к точке глобального минимума. При случайном выборе векторов $\boldsymbol{w}$, составляющих исходную популяцию хромосом, они статистически независимы и представляют собой начальное погружение в пространство параметров. Одна часть этих хромосом лучше "приспособлена к существованию" (у них большие значения функции соответствия и меньшие – целевой функции), а другая часть хуже. Упорядочение хромосом в популяции, как правило, производится в направлении от лучших к худшим. Хромосомы отбираются (подвергаются селекции) для формирования очередного поколения по значениям функции соответствия.

Селекция хромосом для спаривания (необходимого для создания нового поколения) может основываться на различных принципах. Одним из наиболее распространенных считается принцип элитарности, в соответствии с которым наиболее приспособленные хромосомы сохраняются, а наихудшие отбраковываются и заменяются вновь созданным потомством, полученным в результате скрещивания пар родителей. На этапе скрещивания подбираются такие пары хромосом, потомство которых может быть включено в популяцию путем селекции. Существует огромное количество методов спаривания, от полностью случайного (как правило, среди наиболее приспособленных хромосом), через взвешенно-случайное спаривание и вплоть до так называемой турнирной системы. В последнем случае случайным образом отбирается несколько хромосом, среди которых определяются наиболее приспособленные (с

наименьшим значением целевой функции). Из победителей последовательно проведенных турниров формируются пары для скрешивания.

При взвешенно-случайной системе в процессе отбора учитывается информация о текущем значении функции приспособленности. Отбор может происходить по принципу "рулетки", при этом площадь сегмента колеса рулетки, сопоставленного конкретной хромосоме, пропорциональна величине ее относительной функции приспособленности. Ситуация, типичная для такого отбора, иллюстрируется на рис. 3.9. Полная площадь круга соответствует сумме

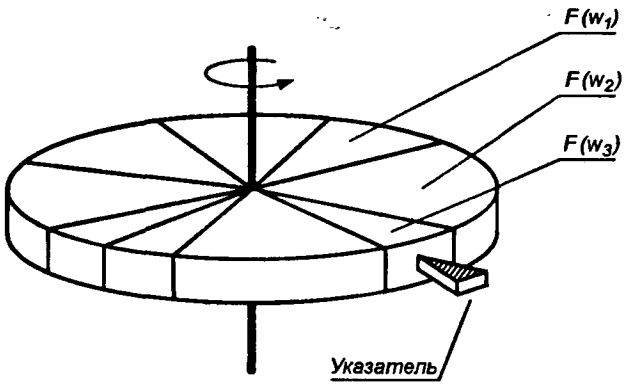

Рис. 3.9. Схема "рулетки", используемая при выборе родителей для будущего поколения. Площадь отдельных сегментов пропорциональна значениям соответствующих функций приспособленности

значений функций приспособленности всех хромосом данной популяции. Каждый выделенный сегмент отвечает конкретной хромосоме. Наиболее приспособленным особям отводятся большие сегменты колеса рулетки, увеличивающие их шансы попадания в переходную популяцию.

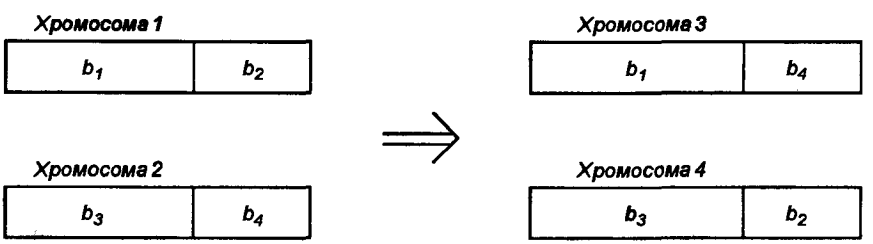

Рис. 3.10. Иллюстрация операции скрещивания, применяемой в генетическом алгоритме

Процесс скрещивания основан на рассечении пары хромосом на две части (рис. 3.10) с последующим обменом этих частей в хромосомах родителей[1]. Место рассечения также выбирается случайным образом. В ситуации, показанной на

[1] Также может применяться разделение родителей на несколько одинаковых частей с последующим обменом комплементарными компонентами.

рис. 3.10, после скрещивания хромосомы 1 (фрагменты b_1 и b_2) с хромосомой 2 (фрагменты b_3 и b_4) образовалась пара новых хромосом: хромосома 3 (фрагменты b_1 и b_4) и хромосома 4 (фрагменты b_3 и b_2). Количество новых потомков равно количеству отбракованных в результате селекции. После их добавления к оставшимся хромосомам размер популяции остается неизменным. Как правило, признается допустимым перенос в очередное поколение некоторых случайно выбранных хромосом вообще без скрешивания. Это соответствует ситуации, когда скрещивание достигает успеха с определенной вероятностью, обычно на уровне 0,6 – 1.

Последняя генетическая операция – это мутация, состоящая в замене значений отдельных битов (при двоичном кодировании) на противоположные. При кодировании натуральными десятичными цифрами мутация заключается в замене значения какого-либо элемента вектора другим, случайно выбранным допустимым значением. Мутация обеспечивает защиту как от слишком раннего завершения алгоритма (в случае выравнивания значений всех хромосом и целевой функции), так и от представления в какой-либо конкретной позиции всех хромосом одного и того же значения. Однако необходимо иметь в виду, что случайные мутации приводят к повреждению уже частично приспособленных векторов. Обычно мутации подвергается не более 1–5% бит всей популяции хромосом. Как и при выполнении большинства генетических операций, элемент, подвергаемый мутации, отбирается случайным образом.

На рис. 3.11 представлены схема формирования переходной популяции и проводимые после него операции скрещивания и мутации, приводящие к образованию популяции потомков. Отметим, что в этих операциях не обязательно участвуют все хромосомы, входящие в переходную популяцию.

Исследованиями доказано [41], что каждое последующее поколение, сформированное после выполнения селекции, скрещивания и мутации, имеет статистически лучшие средние показатели приспособленности (меньшие значения целевой функции). Типичная динамика изменения среднего по популяции и минимального (т.е. соответствующего наиболее приспособленной хромосоме) значения целевой функции для последовательных поколений генетического процесса представлена на рис. 3.12.

В качестве окончательного решения принимается наиболее приспособленная хромосома, имеющая минимальное значение целевой функции. Генетический процесс завершается либо в момент генерации удовлетворяющего нас решения, либо при выполнении максимально допустимого количества итераций. При реализации генетического процесса отслеживается, как правило, не только минимальное значение целевой функции, но и среднее значение по всей популяции хромосом, а также их вариации. Решение об остановке алгоритма может приниматься и в случае отсутствия прогресса минимизации, определяемого по изменениям названных характеристик.

Хорошие результаты обучения приносит объединение алгоритмов глобальной оптимизации с детерминированными методами. На первом этапе обучения сети

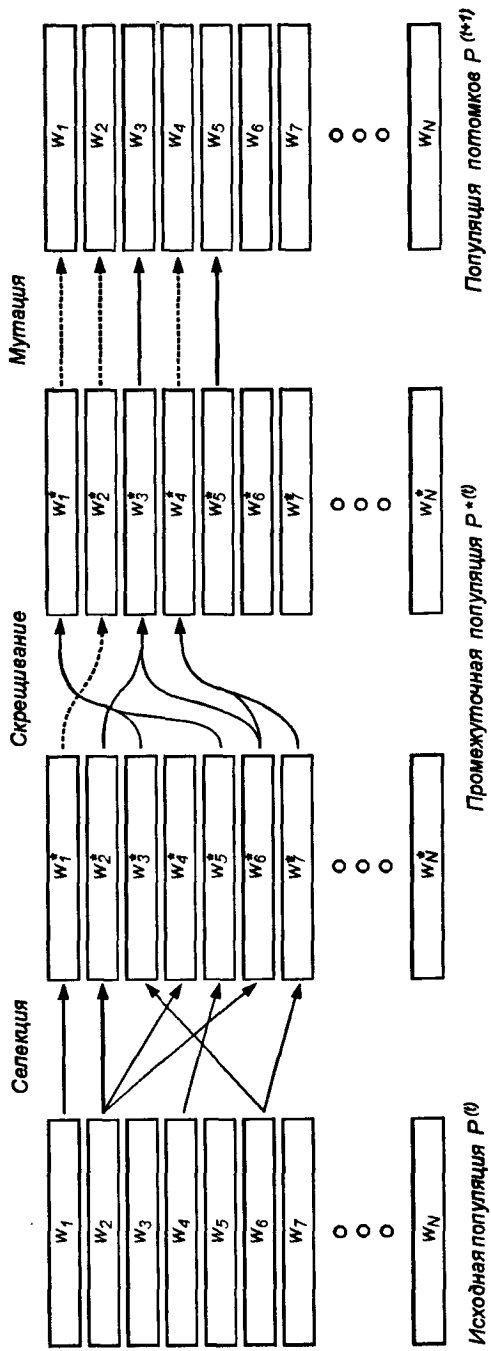

Рис. 3.11. Упрощенная схема генетического алгоритма, иллюстрирующая операции селекции, скрещивания и мутации

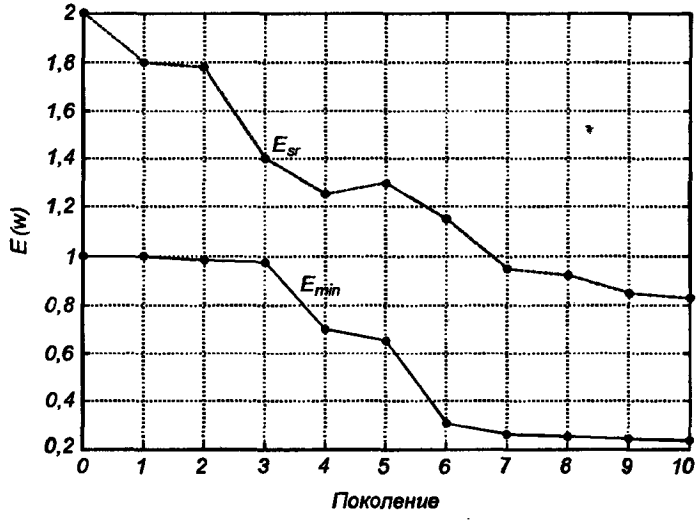

Рис. 3.12. Типичная динамика среднего (E_s) и минимального ($E_{\min}$) значения целевой функции в последовательных поколениях генетического алгоритма

применяется выбранный алгоритм глобальной оптимизации, а после достижения целевой функцией определенного уровня включается детерминированная оптимизация с использованием какого-либо локального алгоритма (наискорейшего спуска, переменной метрики, Левенберга–Марквардта или сопряженных градиентов).

3.9. Методы инициализации весов

Обучение нейронных сетей, даже при использовании самых эффективных алгоритмов, представляет собой трудоемкий процесс, далеко не всегда дающий ожидаемые результаты. Проблемы возникают из-за нелинейных функций активации, образующих многочисленные локальные минимумы, к которым может сводиться процесс обучения. Конечно, применение продуманной стратегии поведения (например, имитации отжига, метода мультистара, генетических алгоритмов) уменьшает вероятность остановки процесса в точке локального минимума, однако платой за это становится резкое увеличение трудоемкости и длительности обучения. Кроме того, для применения названных методов необходим большой опыт в области решения сложных проблем глобальной оптимизации, особенно для правильного подбора управляющих параметров.

На результаты обучения огромное влияние оказывает подбор начальных значений весов сети. Идеальными считаются начальные значения, достаточно близкие к оптимальным. При этом удается не только устранить задержки в точках локальных минимумов, но и значительно ускорить процесс обучения. К сожалению, не существует универсального метода подбора весов, который бы

гарантировал нахождение наилучшей начальной точки для любой решаемой задачи. По этой причине в большинстве практических реализаций чаще всего применяется случайный подбор весов с равномерным распределением значений в заданном интервале.

Неправильный выбор диапазона случайных значений весов может вызвать слишком раннее насыщение нейронов, в результате которого, несмотря на продолжающееся обучение, среднеквадратичная погрешность будет оставаться практически постоянной. Явление этого типа не означает попадания в точку локального минимума, а свидетельствует о достижении седловой зоны целевой функции вследствие слишком больших начальных значений весов. При определенных обучающих сигналах в узлах суммирующих нейронов генерируются сигналы $u_i = \sum_j w_{ij} x_j$ со значениями, соответствующими глубокому насыщению сигмоидальной функции активации. При этом поляризация насыщения обратна ожидаемой (выходной сигнал нейрона равен +1 при ожидаемой величине −1 и обратно). Значение возвратного сигнала, генерируемое в методе обратного распространения, пропорционально величине производной от функции активации $\frac{\partial f}{\partial x}$, в точке насыщения близко нулю. Поэтому изменения значений весов, выводящие нейрон из состояния насыщения, происходят очень медленно. Процесс обучения надолго застревает в седловой зоне. Следует обратить внимание, что в состоянии насыщения может находиться одна часть нейронов, тогда как другая часть остается в линейном диапазоне, и для них обратный обучающий сигнал принимает нормальный вид. Это означает, что связанные с такими нейронами веса уточняются нормальным образом, и процесс их обучения ведет к быстрому уменьшению погрешности. Как следствие, нейрон, остающийся в состоянии насыщения, не участвует в отображении данных, сокращая таким образом эффективное количество нейронов в сети. В итоге процесс обучения чрезвычайно замедляется, поэтому состояние насыщения отдельных нейронов может длиться практически непрерывно вплоть до исчерпания лимита итераций.

Случайная инициализация, считающаяся единственным универсальным способом приписывания начальных значений весам сети, должна обеспечить такую стартовую точку активации нейронов, которая лежала бы достаточно далеко от зоны насыщения. Это достигается путем ограничения диапазона допустимых разыгрываемых значений. Оценки нижней и верхней границ такого диапазона, предлагаемые различными исследователями на основании многочисленных компьютерных экспериментов, отличаются в деталях, однако практически все лежат в пределах (0 , 1).

В работе [155] предложено равномерное распределение весов, нормализованное для каждого нейрона по амплитуде $\frac{2}{\sqrt{n_{in}}}$, где n_{in} означает количество входов нейрона. Значения весов поляризации для нейронов скрытых слоев должны принимать случайные значения из интервала $\left[-\frac{\sqrt{n_{in}}}{2}, \frac{\sqrt{n_{in}}}{2}\right]$, а для выходных нейронов – нулевые значения.

Д. Нгуен и Б. Видроу в своих рассуждениях на тему оптимальных значений начальных весов используют кусочно-линейную аппроксимацию сигмоидальной функции активации. На этой основе они определили оптимальную длину случайного вектора весов нейронов скрытых слоев равной $\sqrt[n_{in}]{N_h}$, где N_h означает количество нейронов в скрытом слое, а n_{in} – количество входов данного нейрона. Оптимальный диапазон весов поляризации для нейронов скрытого слоя определен в пределах $\left[-\sqrt[n_{in}]{N_h}, \sqrt[n_{in}]{N_h}\right]$. Начальные веса выходных нейронов, по мнению упомянутых авторов, не должны зависеть от топологии области допустимых значений и могут выбираться случайным образом из интервала [– 0,5, 0,5].

Решение представленных проблем случайной инициализации весов сети опирается либо на интуицию исследователя, либо на результаты большого количества численных экспериментов. Более детальный анализ событий, происходящих в процессе обучения, позволит точнее выявить причины замедления обучения персептронной сети, задержек в седловых зонах, а также слишком раннего завершения обучения в точках локальных минимумов, далеких от оптимального решения. Результатом такого анализа должны стать меры предупреждения этих нежелательных явлений за счет применения соответствующих процедур предварительной обработки обучающих данных для необходимой инициации как структуры сети, так и значений весов. Эти процедуры базируются либо на анализе данных с использованием конкуренции [28], подобно тому, как это происходит в сетях, самоорганизующихся на основе конкуренции, либо на использовании информации о корреляционных зависимостях обучающих данных [65].

Раздел 4

ПРОБЛЕМЫ ПРАКТИЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

4.1. Предварительный подбор архитектуры сети

Для решения какой-либо задачи с применением искусственной нейронной сети следует прежде всего спроектировать структуру сети, адекватную поставленной задаче. Это предполагает выбор количества слоев сети и нейронов в каждом слое, а также определение необходимых связей между слоями.

Подбор количества нейронов во входном слое обусловлен размерностью входного вектора x. Подобная ситуация и с выходным слоем, в котором количество нейронов принимается равным размерности ожидаемого вектора d. Серьезной проблемой остается подбор количества скрытых (внутренних) слоев и числа нейронов в каждом из них. Теоретическое решение этой задачи в смысле условия достаточности было предложено математиками, занимающимися аппроксимацией функции нескольких переменных. Следует отметить, что ИНС выступает в роли универсального аппроксиматора обучающих данных (x, d) [46, 56]. В процессе обучения подбираются его функциональные коэффициенты (векторы весов отдельных нейронов). На этапе функционирования при зафиксированных значениях весов производится простой расчет значения аппроксимирующей функции при заданном входном векторе.

Определение минимального количества скрытых слоев сети основано на использовании свойств аппроксимирующих функций. Каждая заданная функция может быть выражена линейной комбинацией локальных импульсов, которые имеют ненулевое значение только в ближайшей окрестности текущего значения x. Импульсная функция определенной структуры может быть сформирована как суперпозиция двух функций, сдвинутых относительно друг друга [38, 113]. На рис. 4.1 демонстрируется способ формирования импульса для одномерной сети (имеющей единственный вход). Две сдвинутые относительно друг друга идентичные сигмоиды y_1 и y_2 создают в результате вычитания импульс с длительностью, пропорциональной разности смещений этих сигмоидальных функций. Соответствующим подбором функциональных параметров можно добиться формирования такого импульса, который будет возникать в необходимом для нас месте, будет иметь требуемую ширину и крутизну нарастания.

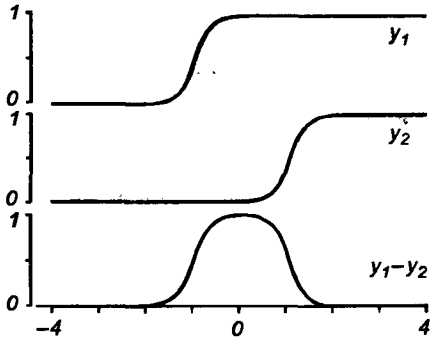

Рис. 4.1. Иллюстрация способа формирования локального одномерного импульса из двух сигмоидальных функций

В случае двухмерной сети можно аналогичным способом сформировать импульс на плоскости [38]. Разность двух определенных на плоскости и сдвинутых относительно друг друга сигмоидальных функций образует гребень бесконечной длины, показанный на рис. 4.2*а*. Добавляя следующую пару сдвинутых относительно друг друга сигмоидальных функций и вычисляя их разность, можно получить второй гребень бесконечной длины. При подборе параметров обеих сигмоидальных пар таким образом, чтобы их гребни располагались под определенным углом (например, 90°), можно получить в результате суммирования этих гребней структуру двухмерного горба, представленного на рис. 4.2 *б*. В месте пересечения гребней образуется импульс двухмерной формы, ограниченный с четырех сторон ответвлениями бесконечной длительности. Эти ответвления можно ликвидировать передачей всего импульса на следующую сигмоидальную функцию (дополнительный слой нейронов) с соответственно подобранным порогом. Как следствие, выполняется фильтрация значения суммы на определенном уровне, подобранном так, что импульс, сформированный сложением двух гребней, пропускается, тогда как ответвления гребней отсекаются. Структура сгенерированного таким образом импульса показана на рис. 4.2*в*.

Созданная этим способом двухвходовая ИНС содержит скрытый слой, состоящий из четырех нейронов, и выходной слой, на котором расположен один нейрон сигмоидального типа. При построении сигмоидальной функции активации с соответствующим порогом он выполняет суммирование сигналов от предыдущих четырех нейронов и отсечение ответвлений.

Возможность обобщения приведенных рассуждений на случай многовходовой сети следует из теории Колмогорова [46, 50, 76, 114]. Если ограничиться непрерывной функцией, трансформирующей N-мерное множество входных данных $\boldsymbol{x}$ в M-мерный выходной вектор $\boldsymbol{d}$, то можно доказать, что аппроксимация такого типа осуществима при использовании сети с одним скрытым слоем. При N входных нейронах будет достаточно использовать для реализации этой функции

скрытый слой с $(2N + 1)$ нейронами. Архитектура ИНС, удовлетворяющая теореме Колмогорова, изображена на рис. 4.3. В предложенном Колмогоровым доказательстве теоремы принято, что выходные сигналы отдельных слоев описываются зависимостями вида

$$z_k = \sum_{j=1}^{N} A_j \psi(x_j + b_j) + A_{0k} \tag{4.1}$$

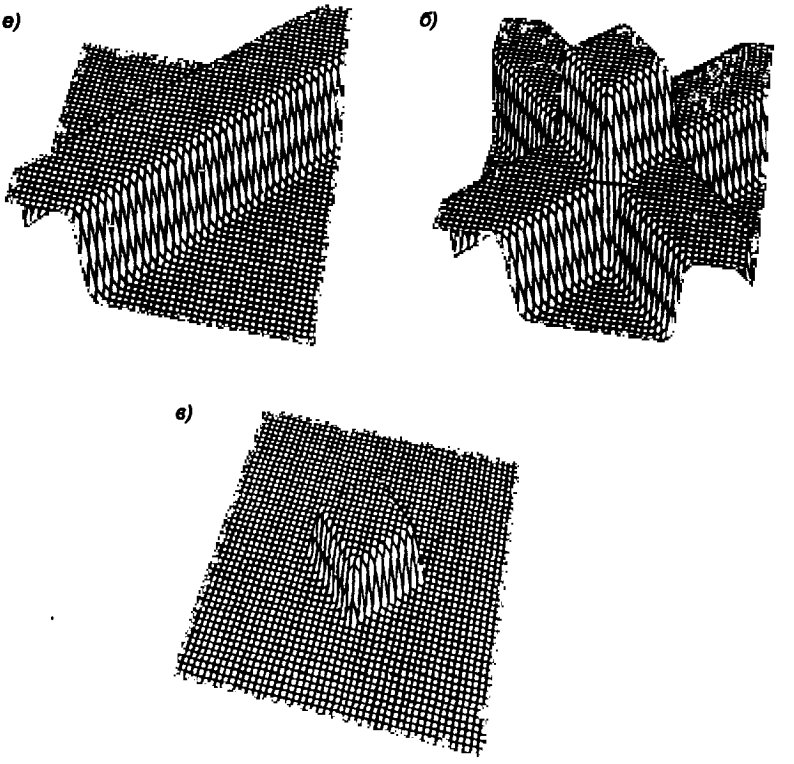

Рис. 4.2. Иллюстрация способа формирования импульса двухмерной сетью: *а*) разность пары двухмерных сигмоидальных функций; *б*) структура, сформированная в результате суммирования разностей двух пар двухмерных сигмоидальных функций; *в*) форма импульса после обработки его пороговой сигмоидальной функцией

для нейронов скрытого слоя при $k = 1, 2, ..., 2N+1$ либо

$$y_i = \sum_{k=1}^{2N+1} C_k g(z_k + d_k) + C_{0i} \tag{4.2}$$

для нейронов выходного слоя. Символами $y(*)$ и $g(*)$ обозначены некоторые точно не определенные непрерывные функции, а все используемые в этих формулах коэффициенты подбираются в процессе обучения.

В случае дискретного преобразования $x \rightarrow y$ одного скрытого слоя уже недостаточно и необходимо создание еще одного слоя нейронов [46]. Это

означает, что независимо от вида многовходовой аппроксимирующей функции максимальное количество скрытых слоев, достаточных для аппроксимации заданного преобразования, не превышает двух.

Результат, полученный благодаря применению теоремы Колмогорова, носит теоретический характер. Он определяет максимальное количество слоев и число нейронов в отдельных слоях, достаточных для аппроксимации заданного преобразования. Теорема не уточняет ни вид нелинейных функций, ни

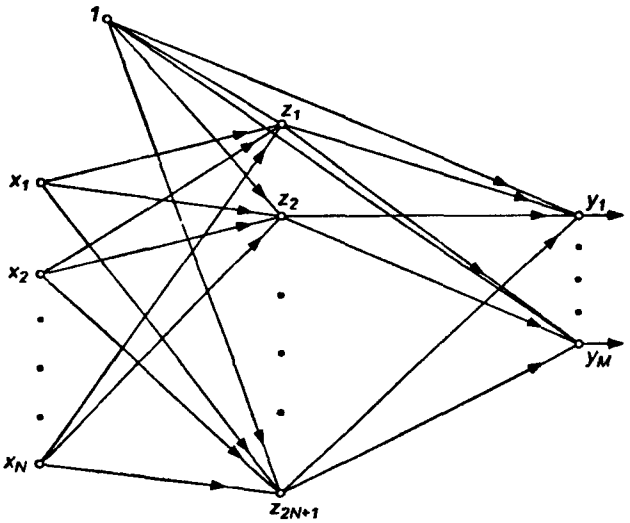

Рис. 4.3. Архитектура сети, удовлетворяющей теореме Колмогорова

методы обучения сети, создаваемой для реализации данного преобразования. Однако она представляет собой фактор, важный для минимизации структуры ИНС. В практических реализациях сетей как количество слоев, так и число нейронов в каждом из них может отличаться от предлагаемых теоремой Колмогорова. Помимо немногочисленных исключений (например, неокогнитрон [36]), чаще всего используются сети, имеющие один скрытый слой (максимум – два), причем количество нейронов в слое может различаться (как правило, от N до $3N$).

4.2. Подбор оптимальной архитектуры сети

4.2.1. Способность к обобщению

Одно из важнейших свойств нейронной сети – это способность к обобщению полученных знаний. Сеть, натренированная на некотором множестве обучающих выборок, генерирует ожидаемые результаты при подаче на ее вход данных

относящихся к тому же множеству, но не участвовавших непосредственно в процессе обучения. Разделение данных на обучающее и тестовое подмножества представлено на рис. 4.4. Множество данных, на котором считается истинным некоторое правило R, разбито на подмножества L и G, при этом в составе L, в свою очередь, можно выделить определенное подмножество контрольных данных V, используемых для верификации степени обучения сети. Обучение проводится на данных, составляющих подмножество L. Способность отображения сетью элементов L может считаться показателем степени накопления обучающих данных, тогда как способность распознавания данных, входящих во множество G и не использованных для обучения, характеризует ее возможности обобщения (генерализации) знаний. Данные, входящие и в L, и в G, должны быть типичными элементами множества R. В обучающем подмножестве не должно быть уникальных данных, свойства которых отличаются от ожидаемых типичных) значений.

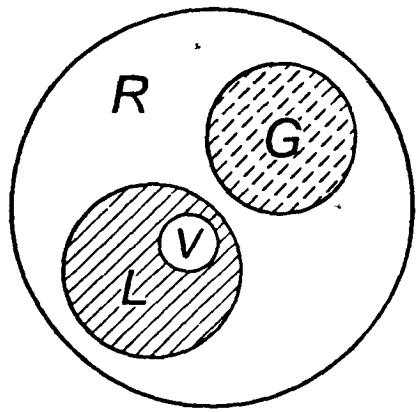

Рис. 4.4. Иллюстрация разделения данных, подчиняющихся правилу R, на обучающее подмножество L, тестовое подмножество G и контрольное подмножество V

Феномен обобщения возникает вследствие большого количества комбинаций входных данных, которые могут кодироваться в сети с N входами. Если в качестве простого примера рассмотреть однослойную сеть с одним выходным нейроном, то для нее может быть составлено 2^N входных выборок. Каждой выборке может соответствовать единичное или нулевое состояние выходного нейрона. Таким образом, общее количество различаемых сигналов составит 2^N. Если для обучения сети используются p из общего числа 2^N входных выборок, то оставшиеся незадействованными $(2^N - p)$ допустимых комбинаций характеризуют потенциально возможный уровень обобщения знаний.

Подбор весов сети в процессе обучения имеет целью найти такую комбинацию их значений, которая наилучшим образом воспроизводила бы последовательность ожидаемых обучающих пар $(\boldsymbol{x}_i, \boldsymbol{d}_i)$. При этом наблюдается тесная связь между количеством весов сети (числом степеней свободы) и количеством обучающих выборок. Если бы целью обучения было только запоминание обучающих выборок, их количество могло быть равным числу весов. В таком случае каждый вес соответствовал бы единственной обучающей паре. К сожалению, такая сеть не будет обладать свойством обобщения и сможет только восстанавливать данные. Для обретения способности обобщать информацию сеть должна тренироваться на избыточном множестве данных, поскольку тогда веса будут адаптироваться не к уникальным выборкам, а к их

статистически усредненным совокупностям. Следовательно, для усиления способности к обобщению необходимо не только оптимизировать структуру сети в направлении ее минимизации, но и оперировать достаточно большим объемом обучающих данных.

Обратим внимание на определенную непоследовательность процесса обучения сети. Собственно обучение ведется путем минимизации целевой функции $E(w)$, определяемой только на обучающем подмножестве L, при этом $E(w) = \sum_{k=1}^{p} E(y_k(w), d_k)$, где p обозначено количество обучающих пар (x_k, d_k), а y_k – вектор реакции сети на возбуждение x_k. Минимизация этой функции обеспечивает достаточное соответствие выходных сигналов сети ожидаемым значениям из обучающих выборок.

Истинная цель обучения состоит в таком подборе архитектуры и параметров сети, которые обеспечат минимальную погрешность распознавания тестового подмножества данных, не участвовавших в обучении. Эту погрешность будем называть погрешностью обобщения $E_G(w)$. Со статистической точки зрения погрешность обобщения зависит от уровня погрешности обучения $E_L(w)$ и от доверительного интервала ε. Она характеризуется отношением [46]

$$E_G(w) = \leq E_L(w) + \varepsilon\left(\frac{p}{\eta}, E_L\right). \qquad (4.3)$$

В работе [46] показано, что значение ε функционально зависит от уровня погрешности обучения $E_L(w)$ и от отношения количества обучающих выборок p к фактическому значению η параметра, называемого мерой Вапника–Червоненкиса и обозначаемого $VC\dim$. Мера $VC\dim$ отражает уровень сложности нейронной сети и тесно связана с количеством содержащихся в ней весов. Значение ε уменьшается по мере возрастания отношения количества обучающих выборок к уровню сложности сети.

По этой причине обязательным условием выработки хороших способностей к обобщению считается грамотное определение меры Вапника–Червоненкиса для сети заданной структуры. Метод точного определения этой меры не известен, о нем можно лишь сказать, что ее значение функционально зависит от количества сииаптических весов, связывающих нейроны между собой. Чем больше количество различных весов, тем больше сложность сети и соответственно значение меры $VC\dim$. В [158] предложено определять верхнюю и нижнюю границы этой меры в виде

$$2\left[\frac{K}{2}\right]N \leq VC\dim \leq 2N_w(1+\lg N_n), \qquad (4.4)$$

где [ ] обозначена целая часть числа, N – размерность входного вектора, K – количество нейронов скрытого слоя, N_w – общее количество весов сети, а N_n – общее количество нейронов сети.

Из выражения (4.4) следует, что нижняя граница диапазона приблизительно равна количеству весов, связывающих входной и скрытый слои, тогда как верх-

няя граница превышает двукратное суммарное количество всех весов сети. В связи с невозможностью точного определения меры $VC\dim$ в качестве ее приближенного значения используется общее количество весов нейронной сети.

Таким образом, на погрешность обобщения оказывает влияние отношение количества обучающих выборок к количеству весов сети. Небольшой объем обучающего подмножества при фиксированном количестве весов вызывает хорошую адаптацию сети к его элементам, однако не усиливает способности к обобщению, так как в процессе обучения наблюдается относительное превышение числа подбираемых параметров (весов) над количеством пар фактических и ожидаемых выходных сигналов сети. Эти параметры адаптируются с чрезмерной (а вследствие превышения числа параметров над объемом обучающего множества – и неконтролируемой) точностью к значениям конкретных выборок, а не к диапазонам, которые эти выборки должны представлять. Фактически задача аппроксимации подменяется в этом случае задачей приближенной интерполяции. В результате всякого рода нерегулярности обучающих данных и измерительные шумы могут восприниматься как существенные свойства процесса. Функция, воспроизводимая в точках обучения, будет хорошо восстанавливаться только при соответствующих этим точкам значениях. Даже минимальное отклонение от этих точек вызовет значительное увеличение погрешности, что будет восприниматься как ошибочное обобщение. По результатам разнообразных численных экспериментов установлено, что высокие показатели обобщения достигаются в случае, когда количество обучающих выборок в несколько раз превышает меру $VC\dim$ [57].

На рис. 4.5*а* представлена графическая иллюстрация эффекта гиперразмерности сети (слишком большого количества нейронов и весов). Аппроксимирующая сеть, скрытый слой которой состоит из 80 нейронов, на основе интерполяции в 21-й точке адаптировала свои выходные сигналы с нулевой погрешностью обучения. Минимизация этой погрешности на слишком малом (относительно количества весов) количестве обучающих выборок спровоцировала случайный характер значений многих весов, что при переходе от обучающих выборок к тестовым стало причиной значительных отклонений фактических значений y от ожидаемых значений d. Уменьшение количества скрытых нейронов до 5 при неизменном объеме обучающего множества позволило обеспечить и малую погрешность обучения, и высокий уровень обобщения (рис. 4.5*б*). Дальнейшее уменьшение количества скрытых нейронов может привести к потере сетью способности восстанавливать обучающие данные (т.е. к слишком большой погрешности обучения $E_L(w)$). Подобная ситуация иллюстрируется на рис. 4.5*в*, где задействован только один скрытый нейрон. Сеть оказалась не в состоянии корректно воспроизвести обучающие данные, поскольку количество ее степеней свободы слишком мало по сравнению с необходимым для такого воспроизведения. Очевидно, что в этом случае невозможно достичь требуемого уровня обобщения, поскольку он явно зависит от погрешности обучения $E_L(w)$. На практике подбор количества скрытых нейронов (и связанный с ним подбор количества весов) может, в частности, выполняться путем тренинга нескольких

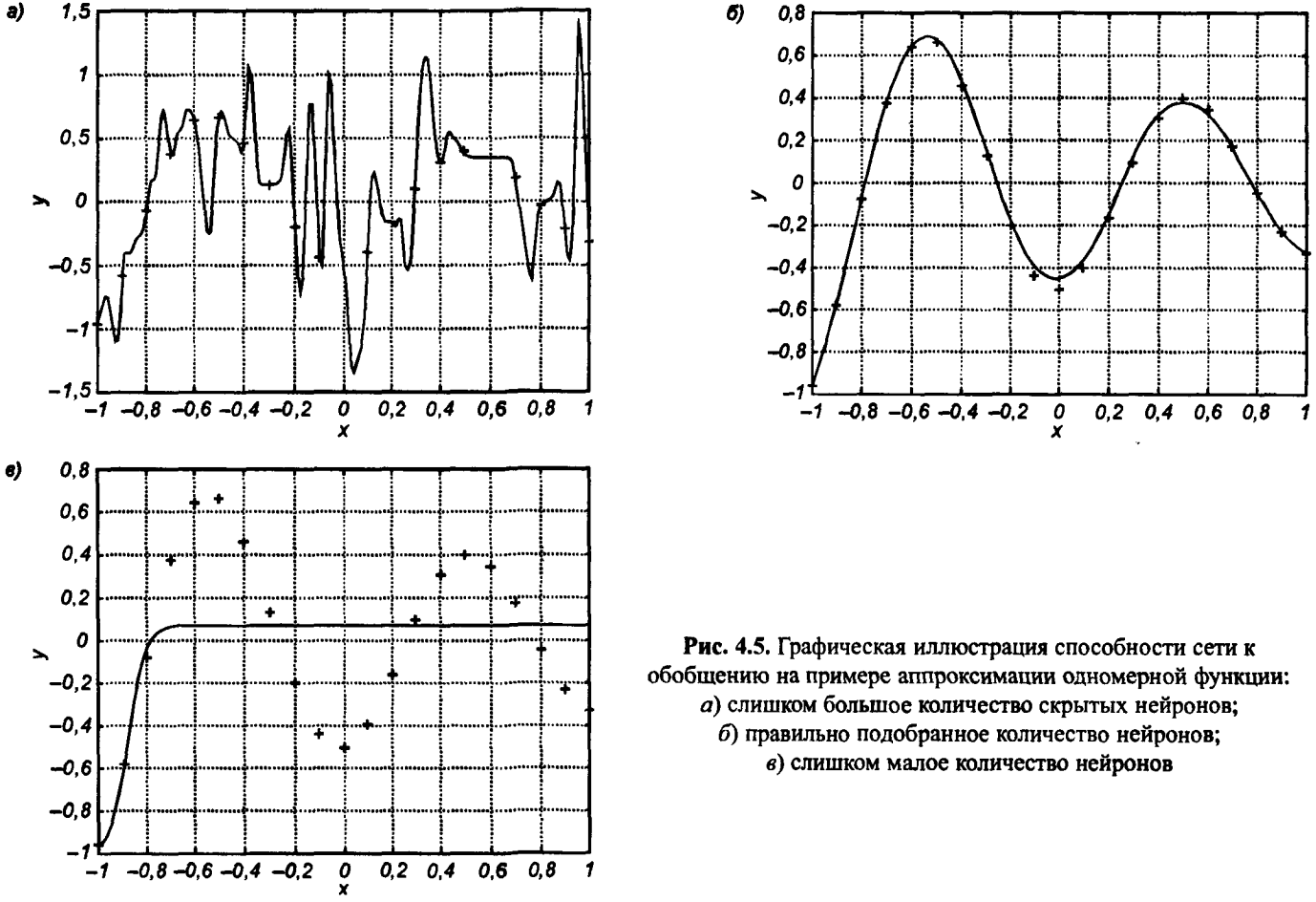

Рис. 4.5. Графическая иллюстрация способности сети к обобщению на примере аппроксимации одномерной функции:
а) слишком большое количество скрытых нейронов;
б) правильно подобранное количество нейронов;
в) слишком малое количество нейронов

сетей с последующим выбором той из них, которая содержит наименьшее количество скрытых нейронов при допустимой погрешности обучения.

Решение по выбору окончательной схемы сети может быть принято только после полноценного обучения (с уменьшением погрешности до уровня, признаваемого удовлетворительным) различных вариантов ее структуры. Однако нет никакой уверенности в том, что этот выбор будет оптимальным, поскольку тренируемые сети могут отличаться различной чувствительностью к подбору начальных значений весов и параметров обучения. По этой причине базу для редукции сети (англ. *pruning*) составляют алгоритмы отсечения взвешенных связей либо исключения нейронов в процессе обучения или после его завершения.

Как правило, методы непосредственного отсечения связей, основанные на временном присвоении им нулевых значений, с принятием решения о возобновлении их обучения по результатам наблюдаемых изменений величины целевой функции (если это изменение слишком велико, следует восстановить отсеченную связь), оказываются неприменимыми из-за слишком высокой вычислительной сложности. Большинство применяемых в настоящее время алгоритмов редукции сети можно разбить на две категории. Методы первой группы исследуют чувствительность целевой функции к удалению веса или нейрона. С их помощью устраняются веса с наименее заметным влиянием, оказывающие минимальное воздействие на величину целевой функции, и процесс обучения продолжается уже на редуцированной сети.

Методы второй группы связаны с модификацией целевой функции, в которую вводятся компоненты, штрафующие за неэффективную структуру сети. Чаще всего это бывают элементы, усиливающие малые значения амплитуды весов. Такой способ менее эффективен по сравнению с методами первой группы, поскольку малые значения весов не обязательно ослабляют их влияние на функционирование сети.

Принципиально иной подход состоит в начале обучения при минимальном (обычно нулевом) количестве скрытых нейронов и последовательном их добавлении вплоть до достижения требуемого уровня натренированности сети на исходном множестве обучающих выборок. Добавление нейронов, как правило, производится по результатам оценивания способности сети к обобщению после определенного количества циклов обучения. В частности, именно такой прием реализован в алгоритме каскадной корреляции Фальмана.

При обсуждении способности сети к обобщению невозможно обойти вниманием влияние на ее уровень длительности обучения. Численные эксперименты показали, что погрешность обучения при увеличении количества итераций монотонно уменьшается, тогда как погрешность обобщения снижается только до определенного момента, после чего начинает расти. Типичная динамика этих показателей представлена на рис. 4.6, где погрешность обучения E_L обозначена сплошной, а погрешность обобщения E_G – пунктирной линией. Приведенный график однозначно свидетельствует, что слишком долгое обучение может привести к "переобучению" сети, которое выражается в слишком детальной адаптации весов к несущественным флуктуациям обучающих данных.

Такая ситуация имеет место при использовании сети с чрезмерным (по сравнению с необходимым) количеством весов, и она тем более заметна, чем больше "лишних" весов содержит сеть. Излишние веса адаптируются к любым нерегулярностям обучающих данных, воспринимая их в качестве важных характеристик. Как следствие, на этапе тестирования они становятся причиной возникновения значительных погрешностей воспроизведения.

Для предупреждения переобучения в обучающем множестве выделяется область контрольных данных (подмножество V на рис. 4.4), которые в процессе обучения применяются для оперативной проверки фактически набранного уровня обобщения.

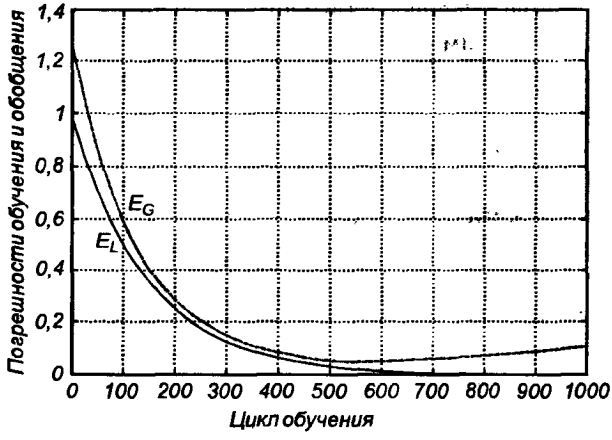

Рис. 4.6. Иллюстрация влияния длительности обучения на погрешность обучения E_L и на погрешность тестирования (обобщения) E_G

Обучение прекращается, когда погрешность обобщения на этом подмножестве достигнет минимального значения (или начнет возрастать).

4.2.2. Методы редукции сети с учетом чувствительности

Редукция сети производится для уменьшения количества скрытых нейронов межнейронных связей. Поскольку каждый скрытый нейрон представляет гиперплоскость, разделяющую множество данных на кластеры, редукция сети упрощает такое разделение и усиливает способность к обобщению.

Простейшим критерием редукции считается учет величины весов. Веса, которые значительно меньше средних, оказывают незначительное влияние на общий уровень выходного сигнала связанного с ними нейрона. Поэтому их можно отсечь без существенного вреда для его функционирования.

Однако в некоторых случаях малые значения весов не обязательно оказывают наименьшее воздействие на поведение нейрона. В таких ситуациях их отсечение может привести к серьезным изменениям в работе сети. Поэтому лучшим критерием следует признать учет чувствительности сети к вариациям весов. Без

серьезных последствий для сети из нее могут быть исключены только те веса, чувствительность к изменениям которых оказывается минимальной.

Такой подход к проблеме отсечения весов может быть обоснован разложением целевой функции в ряд Тейлора. В соответствии с ним изменение величины целевой функции, вызванное вариацией весов, можно выразить формулой

$$\Delta E = \sum_i g_i \Delta w_i + \frac{1}{2}\left[\sum_i h_{ii}[\Delta w_{ii}]^2 + \sum_{i \neq j} h_{ij}\Delta w_i \Delta w_j\right] + O(\|\Delta w\|^2), \qquad (4.5)$$

в которой Δw_i означает вариацию i-го веса, g_i – i-ю составляющую вектора градиента относительно этого веса, $g_{ij} = \frac{\partial E}{\partial w_i}$, а h_{ij} – это элементы гессиана, $h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$.

Не рекомендуется отсекать веса в процессе обучения, поскольку низкая чувствительность сети к конкретному весу может быть связана с его текущим значением либо с неудачно выбранной начальной точкой (например, при застревании нейрона в зоне глубокого насыщения). Рекомендуется отсекать веса (проводить регуляризацию сети) только по завершении процесса обучения, когда все нейроны обретут свои постоянные характеристики. Это исключает применение градиента в качестве показателя чувствительности, поскольку минимум целевой функции характеризуется нулевым значением градиента. Поэтому в качестве показателя важности конкретных весов приходится использовать вторые производные целевой функции (элементы гессиана).

Одним из лучших способов регуляризации сети считается метод, предложенный ЛеКуном [84]. Он называется OBD (англ.: *Optimal Brain Damage*). Исходная позиция этого метода – разложение целевой функции в ряд Тейлора в окрестности текущего решения. Для упрощения задачи ЛеКун при использовании метода OBD исходит из того, что вследствие положительной определенности гессиана матрица **H** является диагонально доминирующей. Поэтому можно учитывать только диагональные элементы h_{kk} и игнорировать все остальные. В качестве меры значимости веса w_{ij} в методе OBD используется показатель S_{ij}, называемый *коэффициентом асимметрии* (англ.: *saliency*), который определяется в виде [84]

$$S_{ij} = \frac{1}{2}\frac{\partial^2 E}{\partial w_{ij}^2} w_{ij}^2 . \qquad (4.6)$$

Отсечение весов с наименьшими значениями показателя S_{ij} не вызовет существенных изменений в процессе функционирования сети. Процедуру OBD редукции сети можно описать в следующем виде:

1. Полное предварительное обучение сети выбранной структуры с использованием любого алгоритма.
2. Определение диагональных элементов гессиана ($h_{kk} = \frac{\partial^2 E}{\partial w_{ij}^2}$), соответствующих каждому весу, и расчет значений параметра $S_{ij} = \frac{1}{2} h_{kk} w_{ij}^2$, характеризующего значимость каждой синаптической связи для сети в целом.

3. Сортировка весов в порядке убывания приписанных им параметров S_{ij} и отсечение тех из них, которые имеют наименьшие значения.
4. Возврат к п. 1 для обучения сети с редуцированной структурой и повторение процесса отсечения вплоть до исключения всех весов, оказывающих наименьшее влияние на величину целевой функции.

Метод OBD считается одним из лучших способов редукции сети среди методов учета чувствительности. Его применение обеспечивает достижение сетью высокого уровня обобщения, лишь незначительно отличающегося от уровня погрешности обучения. Особенно хорошие результаты дает повторное обучение сети после отсечения наименее значимых весов.

В качестве примера рассмотрим реализацию этого метода для регуляризации персептронной сети, использованной авторами [123] для прогнозирования перегрузок в Польской энергетической системе. На рис. 4.7*а* представлена исходная структура сети, а на рис. 4.7*б* – структура сети после регуляризации по методу OBD. В результате отсечения весов из состава сети были исключены три скрытых нейрона и ряд взвешенных связей, подходивших к оставшимся нейронам. Из 201 веса оригинальной сети (рис. 4.7*а*) была исключена почти треть (62 веса). Решения об отсечении принимались по результатам анализа коэффициентов асимметрии S_{ij}, рассчитанных для всех весов сети. На рис. 4.8 приведен график распределения значений этих коэффициентов, упорядоченных в порядке их возрастания. Процесс отсечения весов состоял из трех фаз. На первой фазе было исключено 38 весов, на второй – 16 и на третьей – 8. После каждой фазы отсечения обучение сети повторялось. Применение процедуры OBD позволило уменьшить погрешность обобщения на 7 %.

Дальнейшим развитием метода OBD считается метод OBS (англ.: *Optimal Brain Surgeon*), предложенный Б. Хассиби и Д. Шторком тремя годами позднее [45]. Отправная точка этого метода (так же как и в OBD) – разложение целевой функции в ряд Тейлора и игнорирование членов первого порядка. В этом методе учитываются все компоненты гессиана, а коэффициент асимметрии веса определяется в виде (для избавления от четверных индексов вес w_{kl} обозначается одиночным индексом как w_i)

$$S_i = \frac{1}{2}\frac{w_i^2}{[\mathbf{H}^{-1}]_{ii}}. \qquad (4.7)$$

Отсечению подвергается вес с наименьшим значением S_i. Дополнительный результат такого подхода заключается в несложной формуле коррекции оставшихся весов, позволяющей вернуть сеть в состояние, соответствующее минимуму целевой функции, несмотря на отсечение веса. Уточнение значений оставшихся (неотсеченных) весов выполняется согласно выражению [45]

$$\Delta \boldsymbol{w} = \frac{w_i}{[\mathbf{H}^{-1}]_{ii}}\mathbf{H}^{-1}\boldsymbol{e}_i, \qquad (4.8)$$

где $\boldsymbol{e}_i$ означает единичный вектор с единицей в i-й позиции, т.е. $\boldsymbol{e}_i = [0, ..., 0, 1, ..., 0]^T$. Коррекция выполняется после отсечения каждого

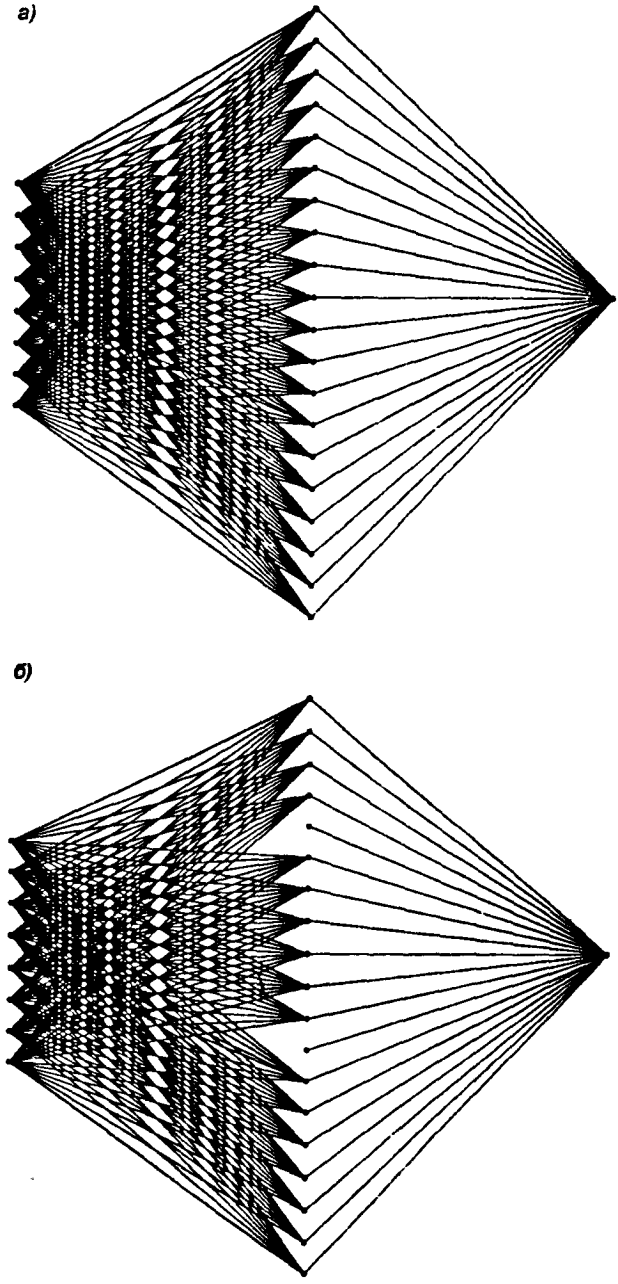

Рис. 4.7. Иллюстрация влияния отсечения весов (методом OBD) на структуру сети, использованной для прогнозирования перегрузок Польской энергетической системы: *а*) исходная структура сети; *б*) сеть после отсечения наименее значимых весов

очередного веса и заменяет повторное обучение сети, необходимое при использовании метода OBD. Процедуру OBS регуляризации сети можно описать в следующем виде [45]:

1. Обучение нейронной сети предварительно отобранной структуры вплоть до отыскания минимума целевой функции.
2. Расчет обратной гессиану матрицы $\mathbf{H}^{-1}$ и выбор веса w_i, имеющего наименьшее значение показателя $S_i = \frac{1}{2}\frac{w_i^2}{[\mathbf{H}^{-1}]_{ii}}$. Если изменение величины целевой функции в результате отсечения этого веса намного меньше значения E, вес w_i отсекается и осуществляется переход к п. 3, в противном случае отсечение завершается.
3. Коррекция значений весов, оставшихся в сети после отсечения i-го веса, в соответствии с формулой (4.8) с последующим возвратом к п. 2. Процесс продолжается вплоть до отсечения всех мало значащих весов.

Основное отличие метода OBS от OBD, помимо другого определения коэффициента асимметрии, состоит в коррекции весов после отсечения наименее важного веса без повторного обучения сети. В методе OBS всякий раз отсекается только один вес, тогда как при использовании OBD можно на каждом шаге отсекать произвольное количество весов. Вычислительная сложность метода OBS гораздо выше. Расчет диагональных элементов гессиана в нем заменяется расчетом полной матрицы и обратной ей формы. На практике этот этап можно значительно упростить при использовании аппроксимированной формы матрицы, обратной гессиану, определяемой, например, методом переменной метрики. Однако такое упрощение вызывает снижение точности расчетов и несколько ухудшает качество искомого решения.

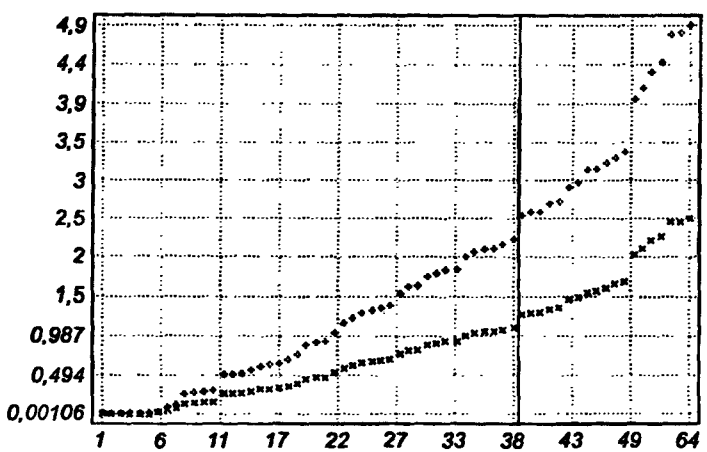

Рис. 4.8. Графики изменения значений коэффициента асимметрии весов (кривая ×) и функции погрешности (кривая +) для различного количества весов нейронной сети, упорядоченные по возрастанию значений весов. Вертикальная прямая указывает предлагаемое количество отсекаемых весов

4.2.3. Методы редукции сети с использованием штрафной функции

Другой метод редукции весов основан на такой организации процесса обучения, которая провоцирует самостоятельное уменьшение значений весов и в результате позволяет исключить те из них, величина которых опускается ниже установленного порога. В отличие от методов учета чувствительности в обсуждаемых методах сама целевая функция модифицируется таким образом, чтобы в процессе обучения значения весов минимизировались автоматически вплоть до достижения определенного порога, при пересечении которого значения соответствующих весов приравниваются к нулю.

Простейший метод модификации целевой функции предусматривает добавление в нее слагаемого, штрафующего за большие значения весов:

$$E(w) = E^{(0)}(w) + \gamma \sum_{ij} w_{ij}^2 . \qquad (4.9)$$

В этой формуле $E^{(0)}(w)$ означает стандартно определенную целевую функцию, заданную, например, в виде эвклидовой нормы, а γ – коэффициент штрафа за достижение весами больших значений. При этом каждый цикл обучения складывается из двух этапов: минимизации величины функции $E^{(0)}(w)$ стандартным методом обратного распространения и коррекции значений весов, обусловленной модифицирующим фактором. Если значение веса w_{ij} после первого этапа обозначить $w_{ij}^{(0)}$, то в результате коррекции этот вес будет модифицирован по градиентному методу наискорейшего спуска согласно формуле

$$w_{ij} = w_{ij}^{(0)}(1 - \eta\gamma), \qquad (4.10)$$

где η обозначает константу обучения. Определенная таким образом штрафная функция вызывает уменьшение значений всех весов даже тогда, когда с учетом специфики решаемой задачи отдельные веса должны иметь большие значения. Уровень значений, при котором вес может быть отсечен, должен подбираться с особой тщательностью на основе многочисленных экспериментов, указывающих, при каком пороге отсечения процесс обучения сети подвергается наименьшим возмущениям.

Более приемлемые результаты, не вызывающие уменьшения значений всех весов, можно получить модификацией представления целевой функции в форме

$$E(w) = E^{(0)}(w) + \frac{1}{2}\gamma \sum_{i,j} \frac{w_{ij}^2}{(1 + \sum_k w_{ik}^2)} . \qquad (4.11)$$

Минимизация этой функции вызывает не только редукцию межнейронных связей, но может также привести к исключению тех нейронов, для которых величина $\sum_k |w_{ik}|$ близка к нулю. Легко доказать, что правило коррекции весов в этом случае может быть задано выражением

$$w_{ij} = w_{ij}^{(0)}\left[1-\eta\gamma\frac{1+2\sum_{k\neq j}(w_{ij}^{(0)})^2}{\left[1+\sum_{k}(w_{ik}^{(0)})^2\right]^2}\right]. \qquad (4.12)$$

При малых значениях весов w_{ik}, подходящих к i-му нейрону, происходит дальнейшее их уменьшение. Это ведет к ослаблению выходного сигнала до нуля и в итоге к исключению его из сети. При больших значениях весов, ведущих к i-му нейрону, их коррекционная составляющая исчезающе мала и очень слабо влияет на процесс редукции сети.

Другой способ минимизации сети основан на такой модификации целевой функции, которая позволяет исключать скрытые нейроны, в наименьшей степени изменяющие свою активность в процессе обучения. При этом учитывается, что если выходной сигнал какого-либо нейрона при любых обучающих выборках остается неизменным (на его выходе постоянно вырабатывается 1 или 0), то его присутствие в сети излишне. И напротив, при высокой активности нейрона считается, что его функционирование дает важную информацию. Й. Шовен в [7] предложил следующую модификацию целевой функции:

$$E(w) = E^{(0)}(w) + \mu\sum_{i=1}^{K}\sum_{j=1}^{p} e(\Delta_{ij}^2). \qquad (4.13)$$

В этом выражении Δ_{ij} означает изменение значения выходного сигнала i-го нейрона для j-й обучающей выборки, а $e(\Delta_{ij}^2)$ – это корректирующий фактор целевой функции, зависящий от активности всех K скрытых нейронов для всех j (j = 1, 2, ..., p) обучающих выборок. Коэффициент m определяет степень относительного влияния корректирующего фактора на значение целевой функции. Вид корректирующей функции подбирается так, чтобы изменение целевой функции зависело от активности скрытого нейрона, причем при высокой его активности (т.е. частых изменениях значения выходного сигнала) величина ΔE должна быть малой, а при низкой активности – большой. Это достигается применением функции e, удовлетворяющей отношению

$$e' = \frac{\partial e(\Delta_i^2)}{\partial \Delta_i^2} = \frac{1}{(1+\Delta_i^2)^n}. \qquad (4.14)$$

Индекс n позволяет управлять процессом штрафования за низкую активность. При $n=2$ функция e принимает вид: $e = \frac{1}{1+\Delta_i^2}$. Малая активность нейронов карается сильнее, чем высокая, что в результате может привести к полному исключению пассивных нейронов из сети.

Оба подхода к редукции сети, основанные как на учете чувствительности, так и на модификациях целевой функции, ведут к минимизации количества весов и нейронов сети, уменьшая таким образом уровень ее сложности и улучшая

соотношение между количеством обучающих выборок и мерой $VC\dim$. В итоге возрастает способность сети к обобщению.

4.3. Методы наращивания сети

В алгоритмах редукции в качестве исходной точки используется избыточная архитектура сети, которая в процессе обучения либо по его завершении упрощается путем исключения наименее значимых весов.

Противоположный подход заключается в первоначальном включении в сеть небольшого количества скрытых нейронов (часто они вообще отсутствуют), но по мере развития процесса обучения их число постепенно увеличивается. Среди многих существующих методов расширения нейронной сети можно выделить: алгоритм Мезарда–Надала [51], алгоритм Мерчанда [51] и метод Ли–Тафтса [86], в которых все обучающие выборки ортогонально проецируются в одномерное пространство с последующим выбором такой гиперплоскости, которая отделила бы данные требуемого класса от остальных. Многократно повторяя эту процедуру на оставшемся множестве выборок, в конечном счете можно обеспечить полное разделение данных. Минимизация количества гиперплоскостей (скрытых нейронов) в методе [86] достигается применением булевой алгебры, в частности карты Карно. Перечисленные методы имеют относительно низкую эффективность при большой размерности входного вектора и не являются серьезной альтернативой методам редукции сети.

Одним из наиболее известных методов расширения сети считается алгоритм каскадной корреляции С. Фальмана [34], который будет подробно изложен в разделе 6.

4.4. Подбор обучающих выборок

С точки зрения цели функционирования нейронная сеть может рассматриваться как векторный классификатор, определяющий принадлежность конкретного входного вектора x к определенной группе. Каждый слой нейронов при этом выполняет в составе сети собственную функцию [99]. Нейроны первого скрытого слоя образуют гиперплоскости, разделяющие N-мерное пространство данных (где N – количество входов сети) на области, содержащие данные, принадлежащие к одному и тому же классу (англ.: *cluster*). Нейроны выходного (либо второго скрытого) слоя представляют множество данных, составляющих конкретный кластер. При ограниченном выборе обучающих данных из универсального множества их размещение относительно конкретных гиперплоскостей становится очень важным. Наилучшие результаты достигаются в случае, когда они располагаются с разных сторон границ гиперплоскостей, разделяющих пространство данных. На рис. 4.9 представлены два различных способа выбора

обучающих данных (обведенных окружностями). Выбор, иллюстрируемый рис. 4.9*а*, позволил определить две гиперплоскости (два нейрона), однако он не решает проблему разделения двух классов данных (*x* и *s*). При таком выборе потребуется еще одна гиперплоскость (т.е. еще один нейрон) для разделения областей *B* и *D*. При выборе обучающих данных, лежащих на границах этих областей (рис. 4.9*б*), получено полное разделение обоих классов. Кроме того, области B и D не содержат обучающих данных (это пустая область пространства), что свидетельствует о возможности удаления одной гиперплоскости (сокращение скрытого слоя до одного нейрона).

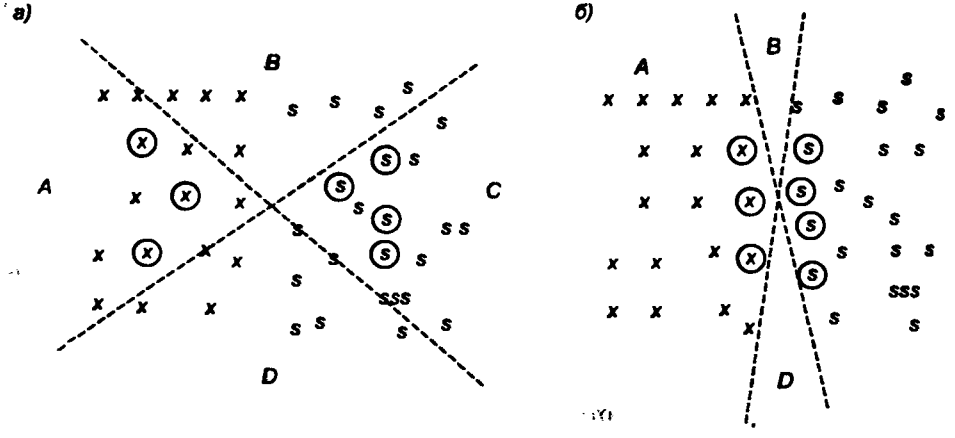

Рис. 4.9. Примеры выбора обучающих данных (обведены окружностями) из универсального множества: *а*) некорректный выбор; *б*) корректный выбор

При подборе обучающих данных очень важна предварительная информация о количестве областей, по которым распределены эти данные. Пространственные границы областей задаются сегментами гиперплоскостей (при проекции на плоскость такие сегменты отображаются отрезками прямых). На рис. 4.10 представлены сегменты трех гиперплоскостей и области, образованные в двухмерном пространстве в результате их пересечения. Области обозначены латинскими буквами от *a* до *g*, а сегменты гиперплоскостей – цифрами от 1 до 9. В [99] доказано, что если обозначить $R(n, N)$ максимальное количество областей, на которые N-мерное пространство разделяется n гиперплоскостями (n нейронами), то

$$R(n,N)=\sum_{i=0}^{N} C_i^n \ , \qquad (4.15$$

где

$$C_i^n=\begin{cases}\dfrac{n!}{i!(n-i)!} & \text{для } n \geq i \\ 0 & \text{для } n < i\end{cases}. \qquad (4.16$$

Если решаемая задача содержит m классов данных, то подбор минимального количества нейронов должен выполняться таким образом, чтобы одновременно выполнялись условия $R(n,N) \geq m$ и $R(n-1,N) < m$. Выбор количества нейронов в слое (количества гиперплоскостей) позволяет определить не только

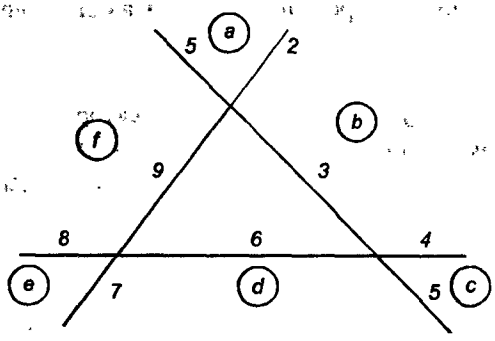

Рис. 4.10. Иллюстрация способа образования нейронной сетью гиперплоскостей и областей данных

число областей, но и количество сегментов гиперплоскостей, ограничивающих эти области. Если обозначить количество этих сегментов $A(n, N)$, то в соответствии с [99]

$$\frac{min(n,N)}{2} \leq \frac{A(n,N)}{R(n,N)} \leq \frac{min(n,2N)}{2}. \qquad (4.17)$$

Оценка количества сегментов гиперплоскостей очень важна для определения объема множества обучающих выборок. Принимая во внимание, что оптимально выбранные обучающие векторы должны располагаться вблизи конкретных сегментов гиперплоскостей, можно сделать вывод, что количество обучающих выборок должно быть пропорционально либо $A(n, N)$, либо $\min(n, N) \cdot R(n,N)$.

4.5. Добавление шума в обучающие выборки

Представленные в предыдущих подразделах процедуры формирования сети позволяют улучшить ее способности к обобщению за счет воздействия на архитектуру сети. Это основной метод, обеспечивающий достижение требуемого уровня обобщения. Однако и после формирования стабильной и минимальной архитектуры сети возможно дальнейшее улучшение ее способностей за счет специальной подготовки множества обучающих выборок. Для хорошо натренированной сети становится актуальной задача выработки у выходных сигналов нечувствительности к вариациям входных величин при условии, что эти вариации находятся в определенных допустимых границах, а сеть реализует монотонное отображение. Другими словами,

аналогичные входные сигналы должны вызывать аналогичные реакции даже в случае, если они не входили в состав обучающего множества.

Для математического обоснования такого требования рассмотрим многослойную сеть с большим количеством входов и выходов. При обозначении вектора всех весов сети w, а векторов входных и выходных сигналов соответственно x и y можно определить вектор y в общем виде как

$$y = f(w, x) \tag{4.18}$$

либо сокращенно как $y = f(x)$, где f обозначен вектор, составленный из сигмоидальных функций активации выходных нейронов. Аргументом функции активации каждого нейрона является сумма весов, определяемая обычным способом, представленным в разделе 2.

Для последующих рассуждений введем различные обозначения обучающего и тестирующего входного вектора. Пусть $\hat{x}_k$ обозначает k-й обучающий, а x – тестирующий вектор. Решение задачи обучения, критерий которого определяется как минимизация целевой функции

$$E = \frac{1}{2}\sum_{k=1}^{p} \|d_k - f(\hat{x}_k)\|^2, \tag{4.19}$$

позволяет оптимизировать значения весов с учетом множества только обучающих, но не тестирующих выборок.

Минимизация этой функции не может гарантировать правильную реакцию сети на возбуждение вектором x, который не был элементом множества обучающих данных. Для исследования чувствительности сети к небольшим вариациям обучающего вектора $\hat{x}_k$ предположим, что тестирующий вектор x_k незначительно отличается от $\hat{x}_k$. Представим это отличие в виде

$$x_k = \hat{x}_k + s, \tag{4.20}$$

где $s = [s_1, s_2, \ldots, s_N]^T$ обозначает вектор шума, составленный из случайных переменных с малой амплитудой. Можно считать, что в тестирующем векторе x_k, близком к соответствующему обучающему вектору $\hat{x}_k$, содержится шум, который вызывает вариации выходного сигнала y_k, определяемые выражением

$$\Delta y_k = f(\hat{x}_k + s) - f(\hat{x}_k) \approx \frac{\partial f(\hat{x}_k)}{\partial x} s\ , \tag{4.21}$$

где $\frac{\partial f}{\partial x}$ обозначен якобиан векторной функции $f(x)$.

Для дальнейших рассуждений предположим, что вектор шума s имеет математическое ожидание $< s >$, равное нулю, и среднеквадратичное отклонение $< ss^T > = \sigma^2 \mathbf{E}$, где $\mathbf{E}$ обозначена единичная матрица размерностью N, а $<\bullet>$ – ожидаемое статистическое значение. Символом R будем обозначать относительную чувствительность сети

$$R(w) = \frac{\sum_{k=1}^{p} \left\langle \| \Delta y_k \|^2 \right\rangle}{\left\langle \| s \|^2 \right\rangle}, \qquad (4.22)$$

отражающую степень изменения значений выходных нейронов (вектор Δy_k), вызванного наличием шума (вектор s) в тестирующих выборках. Принимая во внимание зависимости (4.20) и (4.21), функцию чувствительности можно представить в виде

$$R(w) = \frac{1}{\left\langle s^T s \right\rangle} \sum_{k=1}^{p} \left\langle s^T \left(\frac{\partial f(\hat{x}_k)}{\partial x} \right)^T \left(\frac{\partial f(\hat{x}_k)}{\partial x} \right) s \right\rangle. \qquad (4.23)$$

С учетом принятых допущений относительно величин математического ожидания и среднеквадратичного отклонения шума [97] упростим выражение (4.23) и приведем его к виду

$$R(w) = \sum_{k=1}^{p} \frac{1}{N} \left\| \frac{\partial f(\hat{x}_k)}{\partial x} \right\|^2, \qquad (4.24)$$

где $\|A\|$ означает норму Фробениуса матрицы, $\|A\| = \text{trace}\,(AA^T) = \sum a_{ij}^2$.

Очевидно, что чем меньше чувствительность R, тем слабее реагирует сеть на "возмущения" входного вектора x по отношению к соответствующему обучающему вектору $\hat{x}$, поэтому способность сети к обобщению усиливается. Фактор чувствительности может учитываться на стадии обучения сети. Для этого целевая функция должна быть модифицирована. Если определить ее в форме взвешенной суммы

$$L(w) = E(w) + \alpha R(w), \qquad (4.25)$$

где $\alpha > 0$ – весовой коэффициент, то получим

$$L(w) = \sum_{k=1}^{p} \left\{ \| d_k - f(w, \hat{x}_k) \|^2 + \frac{\alpha}{N} \left\| \frac{\partial f(\hat{x}_k)}{\partial x} \right\|^2 \right\}. \qquad (4.26)$$

Вместо минимизации модифицированной целевой функции $L(w)$ можно принять, что отношение $\frac{\alpha}{N}$ определяет среднеквадратичное отклонение некоторого шума, образующего вектор $n = [n_1, n_2, ..., n_N]^T$ с нулевым ожидаемым значением $\langle n \rangle = 0$ или $\langle nn^T \rangle = \varepsilon 1$. В этом случае целевую функцию $L(w)$ удается преобразовать к виду [97]

$$L(w) = \sum_{k=1}^{p} \left\langle \left\| d_k - f(w, \hat{x}_k) - \frac{\partial f(\hat{x}_k)}{\partial x} n \right\|^2 \right\rangle \cong \left\langle \sum_{k=1}^{p} \| d_k - f(\hat{x}_k + n) \|^2 \right\rangle. \qquad (4.27)$$

Выражение, которым определяется модифицированная целевая функция, имеет форму, идентичную стандартному представлению (4.19), с той разницей, что вместо входного вектора $\hat{x}$ используется зашумленный вектор $\hat{x} + n$. В итоге

при минимизации этой функции учитывается не только слагаемое (4.19), но также и фактор чувствительности $R(w)$, определяемый выражением (4.24). Следовательно, в процессе обучения должны приниматься во внимание характерные для тестовых последовательностей выборки, по которым и подбираются оптимальные значения весов. Это подтверждает вывод, что при зафиксированной архитектуре сети ее способности к обобщению можно дополнительно улучшить.

Подбор среднеквадратичного отклонения шума, при котором действительно можно повысить качество обобщения, представляет собой самостоятельную задачу. Ее теоретическое решение весьма сложно, однако относительно просто получить экспериментальную оценку. По результатам многочисленных тестов можно утверждать, что среднеквадратичное отклонение шума должно коррелировать с фактическим распределением разности между обучающими (незашумленными) выборками и тестовыми данными и составлять небольшой процент от нее.

4.6. Примеры использования персептронной сети

Однонаправленные нейронные сети с сигмоидальной функцией активации широко применяются на практике, составляя важное звено процесса выработки решений. В настоящем подразделе мы ограничимся обсуждением нескольких приложений, позволяющим подчеркнуть универсальность и разнородность функций, которые они могут выполнять.

4.6.1. Распознавание и классификация образов

Распознаванием и *классификацией образа* будем называть его идентификацию и отнесение к соответствующему классу данных. При решении этой задачи нейронная сеть может выполнять функцию как экстрактора (определителя) свойств, так и классификатора, приписывающего образ конкретному классу. Однако чаще всего экстракция свойств производится на отдельном этапе предварительного преобразования измерительных сигналов. Для определения свойств применяются различные методы, в том числе: метод статистических моментов [81], метод преобразования Фурье [30, 151], волновое преобразование [4, 23, 93], преобразование PCA [82], преобразование Карьюнена–Лёве [70] и т.п.

В качестве примера рассмотрим, как персептронная нейронная сеть используется для распознавания и классификации двухмерных образов по их внешним описаниям. На этапе предварительной обработки сигналов будет применяться преобразование Фурье. Описание самого образа должно приводиться к виду, обеспечивающему его независимость от возможного перемещения, ротации и масштабирования. В результате такого преобразования

формируются значения свойств образа, подаваемые для распознавания на вход нейронной сети. Важным достоинством преобразования Фурье считается стабильность трансформации образа, которая в значительной степени обеспечивает независимость распознавания от уровня шумов в исходном сигнале, а также простой и быстрый в реализации алгоритм преобразования.

Начальная обработка данных на основе быстрого преобразования Фурье (FFT)

При распознавании образов, заданных некоторой структурой, подлежащий распознаванию элемент определяется множеством координат (x, y) его контура. Координатное описание контура представляется комплексным числом

$$z(n) = x(n) + jy(n), \tag{4.28}$$

где n – номер очередной пары измерительных данных, описывающих образ. Для их обработки будем использовать дискретное преобразование Фурье (DFT) в виде [64]

$$F_k = F(k) = \sum_{n=0}^{M-1} z(n) \exp\left(-j\frac{2\pi}{M}kn\right) \tag{4.29}$$

для $k = 1, 2, ..., M-1$, где M означает количество точек описания структуры, а $z(n)$ – комплексное число, определенное выражением (4.28). Отдельные компоненты преобразования Фурье образуют вектор $\boldsymbol{F}$.

$$\boldsymbol{F} = [F_0, F_1, \ldots, F_{M-1}]. \tag{4.30}$$

Этот вектор также определяет структуру образа, но в совершенно другом пространстве параметров. Компоненты этого описания позволяют легко преобразовывать данные независимо от их положения, масштаба, угла поворота, а также выбранной начальной точки и общего их количества. Следует подчеркнуть, что знания составляющих вектора $\boldsymbol{F}$ достаточны для полного восстановления формы кривой с помощью обратного преобразования Фурье (IDFT).

Нулевой компонент $\boldsymbol{F}_0$ преобразования Фурье представляет собой среднее значение (центр тяжести) измерительных выборок (x_i, y_i), поскольку

$$F_0 = \frac{1}{M}\sum_{n=0}^{M-1} z(n).$$

Приравниванием этого выражения к нулю образ, представленный вектором $\boldsymbol{F}$, перемещается на стандартную позицию относительно системы координат, не зависящую от фактического первоначального расположения в пространстве данных. По этой причине вектор $\boldsymbol{F}$ после такого преобразования имеет вид: $F_{xy} = [0, F_1, F_2, ..., F_{M-1}]$, инвариантный относительно смещения.

Использование в преобразовании Фурье различного количества оригинальных выборок (x_i, y_i) отражается на размерности формируемых векторов $\boldsymbol{F}$. Для унификации процесса обработки данных количество наиболее значимых компонентов этого преобразования устанавливается априорно. Согласно теории преобразования Фурье [64] наиболее значимыми для отображения структуры компонентами считаются пары координат F_1 и F_{M-1}, следующими – F_2 и F_{M-2} и т.д. При определении K таких пар формируется редуцированное представление вектора F_k = [0, F_1, F_2, ..., F_K, F_{M-K}, ..., F_{M-2}, F_{M-1}], которое независимо от количества измерительных выборок, использованных в преобразовании Фурье, имеет одну и ту же априорно установленную размерность $(2K+1)$.

Инвариантность относительно масштаба образа можно обеспечить нормированием всех высших компонентов разложения Фурье, амплитудой компонента, соответствующего паре F_1 и F_{M-1}. Если обозначить коэффициент масштабирования K_s, то его можно определить выражением [30, 151]

$$K_s = \sqrt{|F_1|^2 + |F_{M-1}|^2}. \tag{4.31}$$

В этом случае нормализация компонентов F_k вектора $\boldsymbol{F}$ выполняется согласно формуле

$$F_{ks} = \frac{F_k}{K_s}. \tag{4.32}$$

При таком преобразовании данных полученная форма вектора $\boldsymbol{F}$ не зависит от размера образа.

Преобразование Фурье состоит из компонентов, допускающих оригинальную качественную интерпретацию. Пары компонентов (F_1 , F_{M-1}), (F_2 , F_{M-2}) и т.д. имеют свой эквивалент в обратном преобразовании IDFT, которое, в частности, для только одной (первой) пары можно представить в виде

$$\begin{aligned} z_1 &= \frac{1}{M}\left(F_1 \exp\left(-j\frac{2\pi k}{M}\right) + F_{M-1}\exp\left(-j\frac{2p(M-1)k}{M}\right)\right) = \\ &= \frac{1}{M}\left(F_1 \exp\left(-j\frac{2pk}{M}\right) + F_{M-1}\exp\left(j\frac{2\pi k}{M}\right)\right). \end{aligned} \tag{4.33}$$

Уравнение (4.33) описывает эллипс. Первая пара (F_1 , F_{M-1}) задает главный эллипс с наиболее длинной осью, вторая пара (F_2 , F_{M-2}) – следующий по величине и т.д. Поворот кривой относительно начальной позиции вызывает поворот главной оси эллипса. Поэтому для обеспечения неизменности измерительных данных относительно угла их поворота следует нормализовать положение этой оси. Коэффициент нормализации угла поворота может быть определен выражением [30, 151]

$$K_r = \exp\left(-j\frac{w_{F_1} + w_{F_{M-1}}}{2}\right), \tag{4.34}$$

где Ψ_{F_1} и $\Psi_{F_{M-1}}$ - это углы степенного представления комплексных чисел F_1 и F_{M-1} соответственно. Нормализация данных, обеспечивающая их инвариантность

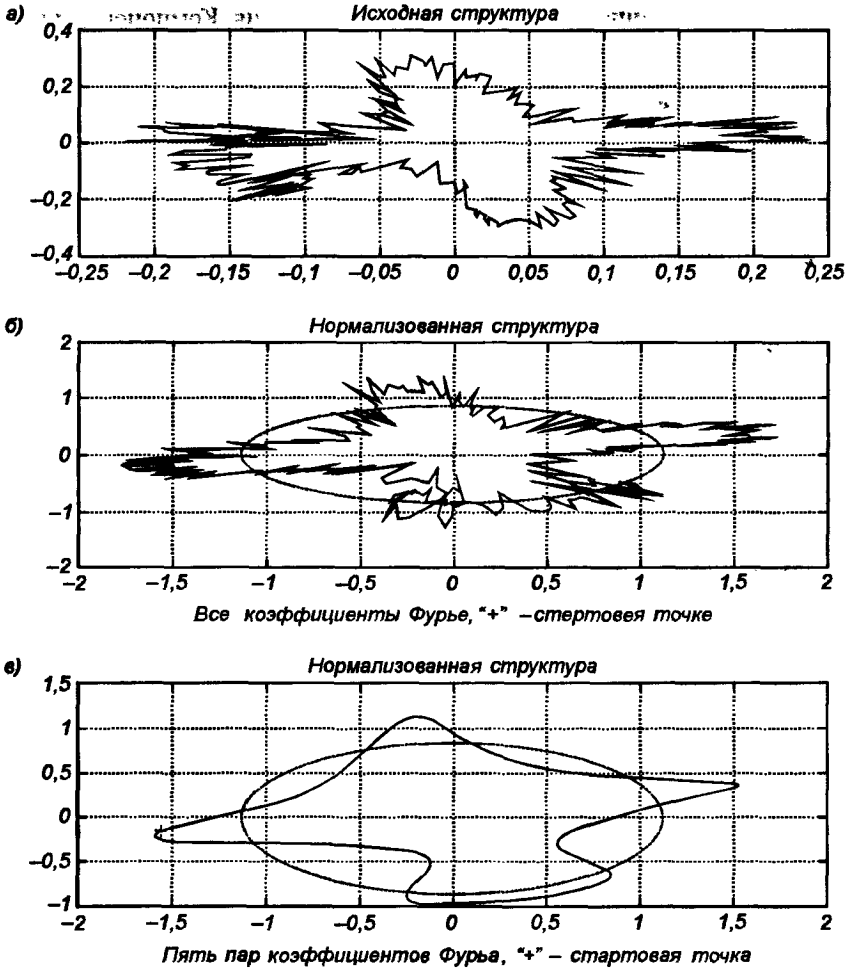

Рис. 4.11. Влияние нормализации и учета ограниченного количества дескрипторов Фурье на представление кривой с большим содержанием шума:
а) форма исходной кривой; *б*) нормализованная форма кривой, содержащая все дескрипторы Фурье; *в*) нормализованная форма кривой, содержащая только пять пар дескрипторов Фурье

относительно угла поворота, основана на умножении каждого компонента вектора преобразования Фурье F_k на коэффициент K_r:

$$F_{kr} = K_r F_k . \qquad (4.35)$$

После такого преобразования вектор характеристик образа, подаваемый на вход нейронной сети, не будет зависеть от угла поворота этого образа. Аналогично можно унифицировать выбор точки начального описания образа [151]. Последовательное выполнение описанных преобразований применительно к исходным компонентам вектора F обеспечивает полную инвариантность

относительно перемещения, поворота и масштабирования. Компоненты преобразованного таким образом вектора называются дескрипторами образов.

Важным следствием применения преобразования Фурье в качестве препроцессора считается уменьшение зависимости результатов распознавания от шума, возмущающего измерения. Помехи, как правило, имеют характер высокочастотного шума. В преобразовании Фурье это соответствует полосе разложения в высокочастотном диапазоне (компоненты высшего порядка вектора F). Отсечение этих компонентов вызывает автоматическое уменьшение уровня шума в образе сигнала после его воспроизведения. На рис. 4.11 иллюстрируется влияние конечного количества дескрипторов Фурье на форму воспроизведенных образов [30]: оригинальный зашумленный образ (рис. *а*); образ, воспроизведенный с использованием всех 64 дескрипторов Фурье (рис. *б*), образ, воспроизведенный с использованием пяти наиболее значимых нормализованных дескрипторов Фурье (рис. *в*). Из рисунка видно, что уменьшение количества дескрипторов Фурье автоматически повышает качество воспроизведенного образа. Коррекция качества заметна также и при анализе численных значений дескрипторов Фурье. Из анализа зашумленных данных следует [31], что даже при значительном присутствии шума в измерительных сигналах амплитудные характеристики дескрипторов изменяются очень незначительно. Это очень полезное качество для распознавания образов, скрытых сильными помехами.

Нейронный классификатор

Выходные сигналы препроцессора в виде последовательности компонентов дескрипторов Фурье после преобразования, обеспечивающего инвариантность к перемещению, повороту и масштабированию, становятся входными сигналами для многослойной нейронной сети, играющей роль системы распознавания образов и одновременно выполняющей их классификацию (отнесение каждого образа к соответствующему эталонному классу). Количество входных узлов сети равно количеству дескрипторов Фурье, учитываемых при классификации. Если допустить, что каждый выходной нейрон представляет единственный класс, то их количество также будет постоянной величиной, равной числу классов. Поэтому в соответствии с методикой, предложенной в начале настоящего раздела, подбираться может только количество скрытых слоев и число нейронов в каждом слое.

Классификатор тренируется методом обратного распространения с использованием одного из обучающих алгоритмов на множестве обучающих данных, последовательно представляющих все классы образов, подлежащих распознаванию. В режиме воспроизведения классифицируемый образ, прошедший через все фазы препроцессора, подается на вход сети, возбуждая тот выходной нейрон, который соответствует требуемому классу.

Из-за зашумленности образов на этапе их распознавания выходные сигналы нейронов сети могут принимать непрерывные значения из интервала [0, 1] вместо

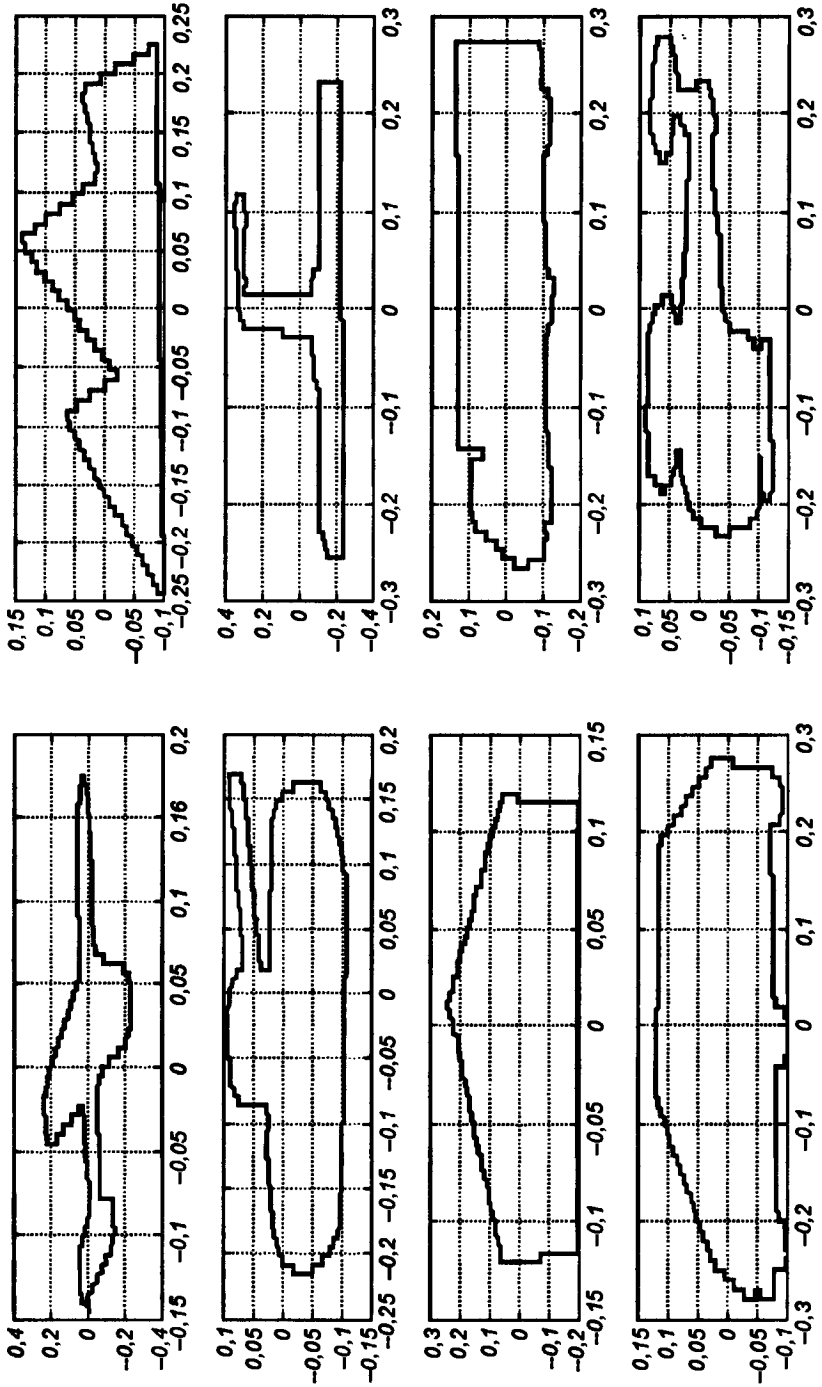

Рис. 4.12. Примеры образов, использованных в численном эксперименте, связанном с классификацией

ожидаемых бинарных нуль-единичных значений (с единицей, обозначающей распознанный класс).

Один из способов решения этой проблемы заключается в том, что в качестве представителя распознанного класса признается наиболее активный нейрон (выработавший самый сильный выходной сигнал). Однако такой подход не позволяет сравнивать активность различных нейронов и приводит к ситуации, в которой решение о победе конкретного нейрона принимается даже тогда, когда активность всех нейронов близка к нулю. Это может приводить к ошибочной классификации.

Наилучшим подходом представляется двухуровневая интерпретация. Вначале проверяется, насколько максимальный сигнал превышает следующий за ним. Если эта разница достаточно велика, победителем признается наиболее активный нейрон. В противном случае, а также если уровни активации всех нейронов не превышают определенного порога, интерпретатор при объявлении результата сообщает, что классификация считается неполной и тем самым предостерегает пользователя от возможной ошибки.

Подобная реализация нейросетевого классификатора была апробирована при распознавании и классификации многих разнообразных образов, в том числе букв и цифр, предметов, объектов и т.п. На рис. 4.12 представлено тестовое множество объектов различных классов, которые распознавались с использованием персептронной сети. После предварительной обработки этих данных с помощью преобразования FFT были сформированы 18-элементные векторные дескрипторы (пять пар наиболее значимых коэффициентов Фурье для амплитуды и для фазы, при этом имеющие нулевые значения фазовые компоненты F_1 и F_{M-1} не использовались). В ходе многочисленных экспериментов количество скрытых нейронов выбрано равным 8. Применялась простейшая интерпретация результатов. Выходной сигнал в диапазоне 0–0,5 рассматривался как нулевой, а свыше 0,5 – как единица. После нормализации компонентов преобразования Фурье эффективность распознавания незашумленных сигналов составила 100 %. Только значительная зашумленность измерительных сигналов (исходные данные зрительно почти не распознавались) с уровнем шума порядка 70 % уменьшила эффективность распознавания до 90 %.

4.6.2. Нейронная сеть для сжатия данных

Задача сжатия (компрессии) данных состоит в уменьшении количества хранимой или передаваемой информации с возможностью ее полного восстановления (декомпрессии). Применение нейронной сети позволяет получить новые решения для сжатия с потерей (с допустимой утратой определенной части информации) при хороших обобщающих способностях и относительно высоком коэффициенте компрессии.

Для иллюстрации будем использовать линейную сеть с одним скрытым слоем, изображенную на рис. 4.13. Количество нейронов выходного слоя равно числу узлов входного слоя. Скрытый слой содержит q нейронов, причем $q << n$. Входной и скрытый слои выполняют собственно компрессию данных, тогда как скрытый и выходной слои осуществляют декомпрессию. Сеть является автоассоциативной, поэтому ее обучающий вектор $\boldsymbol{d}$ совпадает с входным вектором $\boldsymbol{x}$, а выходные сигналы сети соответствуют входным сигналам x_i.

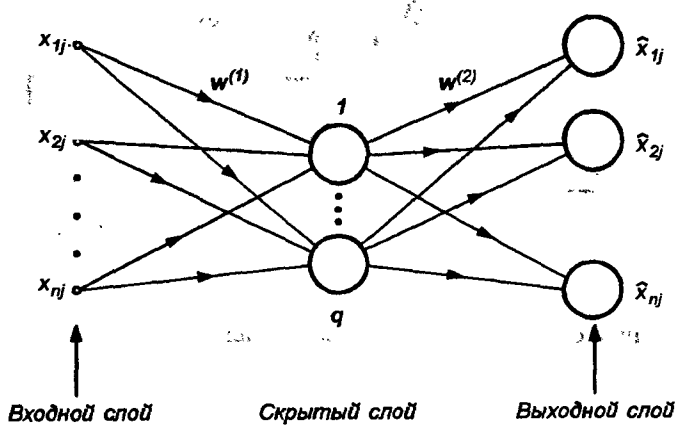

Рис. 4.13. Структура нейронной сети для сжатия данных

Компрессии подвергаются данные, разделенные на кадры, представляемые последовательностью n-элементных векторов (n – количество входных узлов). Кадры имеют форму прямоугольника с размерами h пикселов по горизонтали и v – по вертикали. Градации интенсивности пикселов, входящих в кадр, задаются значениями компонентов вектора $\boldsymbol{x}$. Пример разделения изображения на кадры с последующим соотнесением повторяющихся пикселов изображения вектору $\boldsymbol{x}$ представлен на рис. 4.14.

Поскольку $q << n$, в скрытом слое может храниться меньше информации, чем во входном слое, однако она будет репрезентативной для множества данных и достаточной для реконструкции с заранее заданной точностью оригинальных входных данных. Сигналы скрытого слоя образуют главные компоненты преобразования PCA (англ.: *Principal Component Analysis*), *из которых и образуется информационное ядро [29, 82]. Количество этих компонентов равно числу нейронов q скрытого слоя. Большее значение q* соответствует увеличению объема информации, хранящейся в нейронах скрытого слоя, что в свою очередь обеспечивает лучшее восстановление входной информации в процессе декомпрессии. В примере используется полностью линейная сеть. Веса скрытого слоя в матричной форме обозначаются $\mathbf{W}^{(1)}$, а выходного слоя – $\mathbf{W}^{(2)}$.

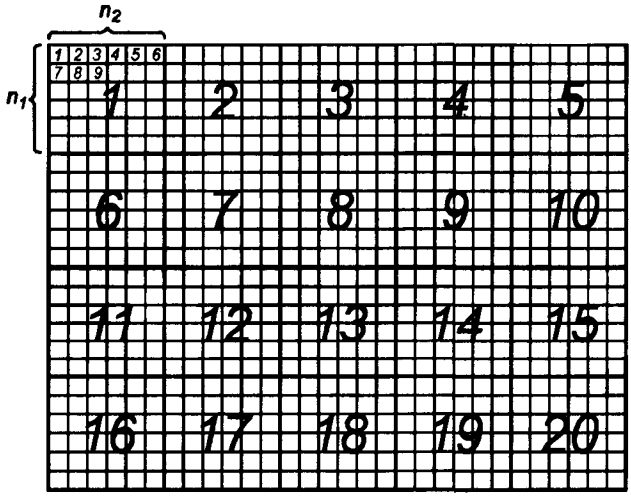

Рис. 4.14. Иллюстрация способа разделения образа на прямоугольные кадры

С учетом однонаправленного распространения сигналов можно получить:

- вектор сигналов скрытого слоя (сжатые сигналы):

$$\boldsymbol{h} = \mathbf{W}^{(1)}\boldsymbol{x}\,; \tag{4.36}$$

- вектор выходных сигналов (сигналы, восстановленные в результате декомпрессии):

$$\hat{\boldsymbol{x}} = \mathbf{W}^{(2)}\boldsymbol{h} = \boldsymbol{h} = \mathbf{W}^{(2)}\mathbf{W}^{(1)}\boldsymbol{x}\,. \tag{4.37}$$

Обучение сети, состоящее в оптимальном подборе весов, образующих матрицы $\mathbf{W}^{(1)}$ и $\mathbf{W}^{(2)}$, направлено на то, чтобы разность между $x_j^{(i)}$ и $\hat{x}_j^{(i)}$ для всех N составляющих вектора $\boldsymbol{x}^{(i)}$ при $i = 1, 2, ..., p$ (где p обозначено количество векторов) была минимальной. Целевая функция, удовлетворяющая этому условию, может быть определена в виде

$$E = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{N}(x_j^{(i)} - \hat{x}_j^{(i)})^2. \tag{4.38}$$

Вследствие прямоугольности обеих матриц $\mathbf{W}^{(1)}$ и $\mathbf{W}^{(2)}$ аналитического решения этой задачи не существует, а результат процесса минимизации целевой функции (4.38) неоднозначен по отношению к решению, получаемому путем преобразования Карьюнена–Лёве, потому что любые матрицы, представляющие собой линейные трансформации матриц $\mathbf{W}^{(1)}$ и $\mathbf{W}^{(2)}$, будут одинаково хорошо отвечать уравнению (4.37).

Поскольку количество нейронов скрытого слоя ограничено, данные, восстановленные в результате декомпрессии (и обозначаемые $\hat{\boldsymbol{x}}$), будут иметь определенную погрешность. Меру этой погрешности определим в форме MSE как

$$\mathrm{MSE} = \frac{1}{Np}\sum_{k=1}^{p}\sum_{i=1}^{h}\sum_{j=1}^{v}(x_{ij}^{(k)} - \hat{x}_{ij}^{(k)})^2, \tag{4.39}$$

где p – это количество кадров, h и v – размер кадра соответственно по осям x и y, а N – размерность вектора данных, составляющих каждый кадр, причем $N = hv$.

Важным параметром, характеризующим соотношение количества информации, содержащейся в образе до его компрессии, к количеству информации, описывающей сжатый образ, считается коэффициент компрессии, отражающий отношение исходного и сжатого количества информации и определяемый в виде

$$K_r = \frac{p \times N \times T}{p \times q \times T + q \times N \times t}, \qquad (4.40)$$

где T и t обозначают количество кодируемых битов для данных и весов соответственно. При большом количестве кадров ($p >> N$) в знаменателе доминирует первый фактор, поэтому формулу расчета коэффициента компрессии можно упростить и представить как отношение количеств входных нейронов N и скрытых q, т.е. $K_r \approx \frac{N}{q}$. Чем больше значение K_r, тем больший эффект достигается при хранении или передаче информации. Вместе с тем обучение сети становится все более сложным, и, как правило, в восстановленном образе появляется все больше искажений.

Уровень декомпрессионного искажения чаще всего оценивается коэффициентом PSNR (англ.: *Peak Signal-to-Noise Ratio*), измеряемым в децибелах и определяемым в виде

$$\text{PSNR}=10 \lg \frac{(2^k - 1)^2}{\text{MSE}}, \qquad (4.41)$$

где k – количество битов, используемых для кодирования градаций интенсивности изображения. При 8-битовом представлении коэффициент PSNR рассчитывается по формуле

$$\text{PSNR}=10 \lg \frac{255^2}{\text{MSE}}. \qquad (4.42)$$

Большее значение коэффициента PSNR соответствует лучшему качеству восстановленного изображения. Для достижения наилучших результатов обучения сети, предназначенной для сжатия данных, необходимо в качестве обучающих выборок использовать как можно большее количество различных образов, хотя вполне удовлетворительные показатели дает и обучение на всего лишь одном изображении. После фиксации подобранных значений весов сеть может использоваться в качестве системы кодирования (скрытый слой) либо декодирования (выходной слой) произвольных образов.

На рис. 4.15*а* представлен исходный образ "Бабуин", который подвергался вначале кодированию, а затем декодированию с помощью нейронной сети, имеющей по 64 входа и выхода. Приведенное на рис. 4.15*б* восстановленное изображение получено благодаря пяти скрытым нейронам (коэффициент компрессии около 12). Исходное изображение имело размер 512 x 512 пикселов. Сеть была предварительно обучена на другом образе "Лес" [114], имеющем

такие же размеры. Качество восстановленного изображения можно признать удовлетворительным. Значение коэффициента PSNR для восстановленного образа составило 22,83 дБ. Поскольку сравнительный визуальный анализ исходного и реконструированного образов недостаточно объективен, на рис. 4.15*в* приведен так называемый дифференциальный образ, подчер-

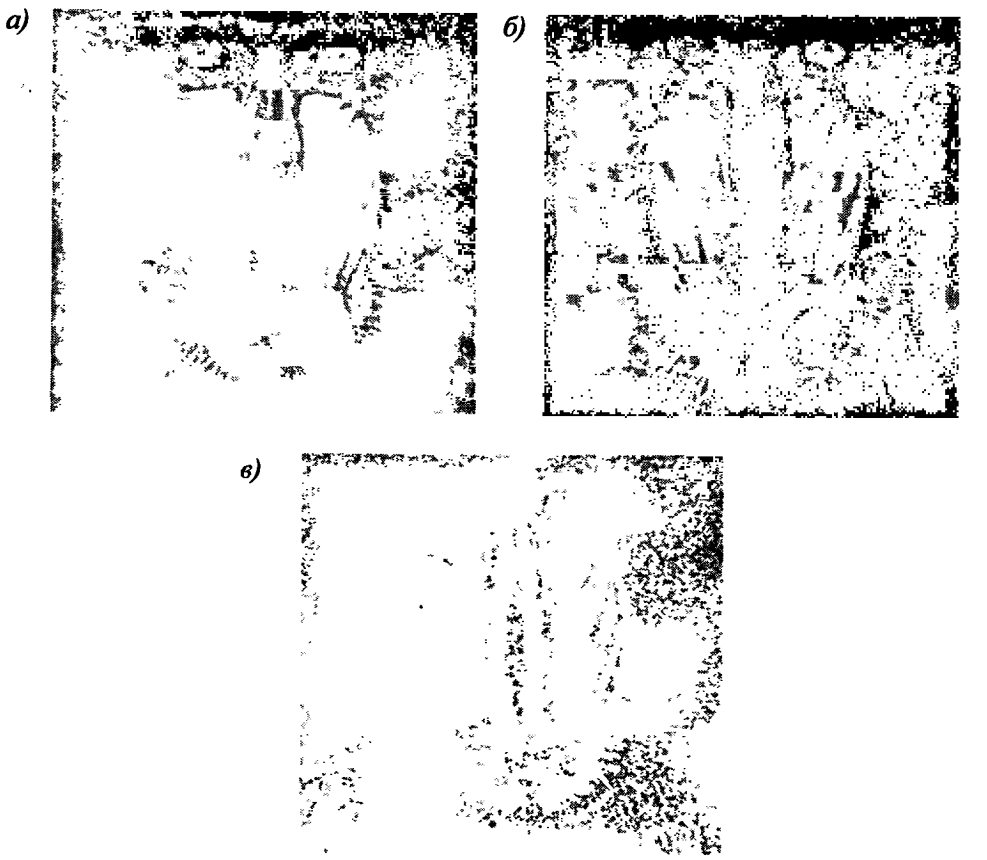

Рис. 4.15. Образ "Бабуин", подвергнутый сжатию и декомпрессии с помощью нейронной сети:
а) исходный образ; *б*) реконструированный образ; *в*) дифференциальный образ

кивающий разницу между ними. Он демонстрирует фактические погрешности, допущенные нейронной сетью при восстановлении данных.

4.6.3. Идентификация динамических объектов

В динамических системах подлежащий распознаванию объект зависит от мгновенных значений обучающих пар, представляющих собой функцию времени. Если в отличие от предыдущих обозначений принять $\boldsymbol{x}$ в качестве вектора

состояния $x \in R^n$, u – входного вектора $u \in R^N$, а y – выходного вектора $y \in R^M$, то общее описание нелинейной системы, функционирующей в дискретном времени, может быть представлено в виде

$$x(k+1) = \phi[x(k), u(k)]; \tag{4.43}$$

$$y(k) = \psi[x(k)], \tag{4.44}$$

где $x(k)$, $u(k)$, $y(k)$ обозначают векторы мгновенных значений соответствующих переменных, ϕ и ψ – знаки векторных статистических нелинейных функций, $\phi \in R^n$, $\psi \in R^M$, определяющих инвариантный во времени конкретный нелинейный объект.

В отличие от линейных уравнений связи, определяемые нелинейными зависимостями, более сложны, и до настоящего времени не существует универсального метода их аналитического решения. В качестве его заменителя применяются приближенные математические модели, уточняющиеся в процессе обучения.

Таким образом, проблема идентификации объекта сводится к построению такой его параметрической модели, чтобы отклики объекта $y(k)$ и модели $\hat{y}(k)$ на одно и то же возбуждение $u(k)$ совпадали в пределах допустимой погрешности ε, т.е.

$$\| \hat{y} - y \| \le \varepsilon . \tag{4.45}$$

Среди многих возможных подходов к реализации такой нелинейной системы выберем способ, основанный на применении нейронной сигмоидальной сети, в общем случае многослойной. На рис. 4.16 представлена универсальная схема подключения нейронной сети в качестве нелинейной модели динамической системы.

Если ограничиться одним входом и выходом, а также представить векторы возбуждения u и отклика объекта y состоящими из элементов запаздывания, т.е. $u(k) = [u(k), u(k-1), ..., u(k-p)]^T$, $y(k) = [y(k), y(k-1), ..., y(k-q)]^T$, то общее описание нелинейной динамической модели можно выразить без вектора состояния x в форме

$$\hat{y}(k+1) = f(y(k), u(k)). \tag{4.46}$$

В этом уравнении $y(k+1)$ обозначает отклик нелинейного объекта в момент $k+1$, а $\hat{y}(k+1)$ – отклик нейронной модели этого объекта в тот же момент времени. Разностный сигнал $e(k+1) = y(k+1) - \hat{y}(k+1)$ управляет процессом адаптации параметров модели. Ряд элементов запаздывания на входе системы образует линию задержки с ответвлениями (англ.: *Tapped Delay Line – TDL*).

В случае применения для идентификации объектов нейронная сеть, как правило, подключается порядково-параллельным способом и использует для предсказания задерживаемые отклики объекта так, как это показано на рис. 4.16. Достоинства такого подключения – это, во-первых, гарантированная ограниченность входных сигналов модели, представляющих собой прошедшие через элементы задержки отклики объекта (он априорно считается устойчивым), во-

вторых — упрощение формулы генерации градиента. Следует отметить, что такое подключение нейронной сети обеспечивает однонаправленное распространение сигналов, поскольку выходной сигнал объекта является сигналом изначально известным (в отличие от выходного сигнала модели),

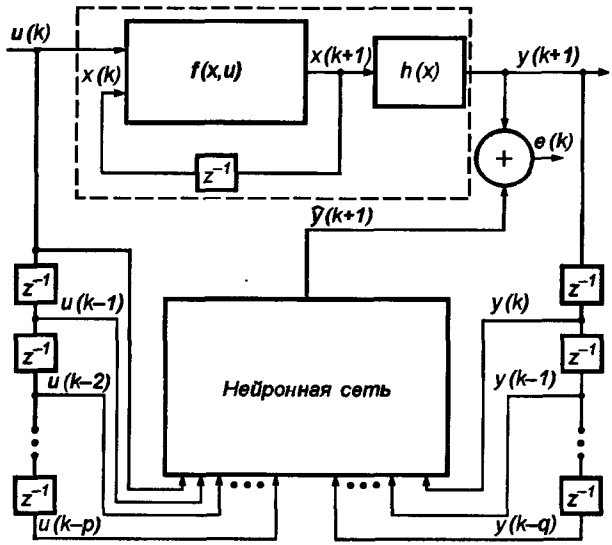

Рис. 4.16. Способ подключения нейронной сети для идентификации динамического объекта

поэтому сеть не должна быть рекуррентной. Поэтому вектор градиента формируется в соответствии со стандартным для многослойной сети методом обратного распространения, описанным в разделе 3.

При таком подключении отклик $y(k)$ сети зависит от вектора $\boldsymbol{u}(k)$, представляющего собой ряд прошедших через элементы задержки реализаций возбуждающего сигнала, а также от вектора $\boldsymbol{d}(k) = \boldsymbol{y}(k)$, представляющего собой ряд прошедших через элементы задержки реализаций заданного сигнала, составляющих ожидаемый выходной вектор сети. В этой ситуации нейронная сеть выполняет функции классической многослойной статической сети.

Для примера рассмотрим идентификацию нелинейного динамического объекта Винера, состоящего из каскадно подключенных линейного фильтра Баттерворта шестого порядка и нелинейного элемента в форме полиномиальной функции x^3. В нейронной модели этого объекта использована сеть с одним скрытым слоем, содержащим 25 нейронов. Входной слой состоит из 24 узлов, а выходной вектор составлен из 12 прошедших через элементы задержки реализаций входного вектора $\boldsymbol{x}$ и 12 реализаций вектора $\boldsymbol{d}$, сформированного из откликов объекта.

В качестве входных сигналов $u(k)$ использовались случайные значения. Обучение проводилось с применением программы *Netteach*. После подбора

значений весов тестировалась способность сети к обобщению, для чего на ее вход подавались детерминированные сигналы фиксированной структуры. Демонстрируемые результаты относятся только к возбуждению в форме синусоидального сигнала. На рис. 4.17*а* показаны графики изменения сигнала, сгенерированного нелинейным объектом (пунктирная линия), и сигнала, полученного на выходе нейронной модели (непрерывная линия) при синусоидальном возбуждающем сигнале. Разность (рис. 4.17*б*) между значениями заданными и фактически сгенерированными моделью системы, подвергнутой идентификации, относительно мала и свидетельствует о высоком качестве полученного решения.

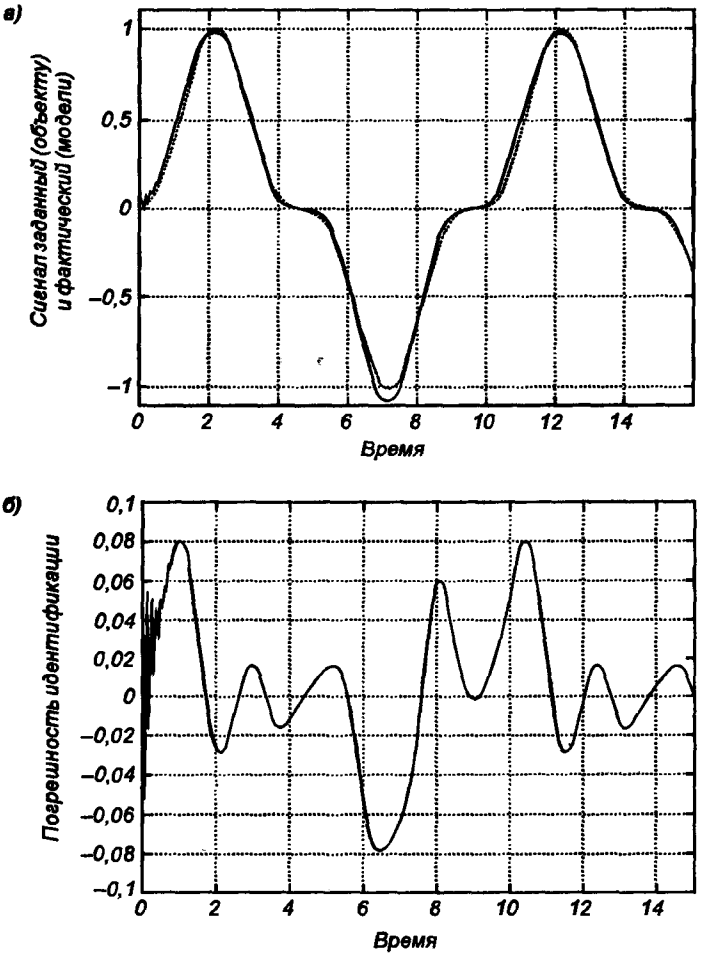

Рис. 4.17. Результаты тестирования обученной сети из примера обработки входных синусоидальных данных:
а) выходной заданный сигнал (пунктирная линия) и выходной сигнал нейронной сети (непрерывная линия); *б*) график погрешности идентификации

4.6.4. Прогнозирование нагрузок энергетической системы

Другим важным свойством нейронных сетей считается способность прогнозировать временные ряды. В настоящем подразделе в качестве примера мы рассмотрим решение задачи предсказания 24-часовых нагрузок Польской электроэнергетической системы (PSE).

Так же как и при классификации образов, предсказание базируется на учете свойств прогнозируемого процесса. Главная особенность часовых нагрузок энергетической системы – это определенная повторяемость характеризующих их выборок в зависимости от дня недели и месяца. Выделяются либо четыре основных вида нагрузок, соответствующих субботе, воскресенью, понедельнику и остальным четырем рабочим дням, либо только два вида, соответствующие праздничным (т.е. нерабочим) и рабочим дням. В ходе проведенных авторами [124] статистических исследований установлено, что распределение по четырем типам дней хотя и снижает погрешность обучения, однако увеличивает погрешность обобщения (ухудшает результаты собственно прогнозирования). Поэтому для предсказания 24-часовых нагрузок использовалось распределение на два типа дней, что потребовало введения одного дополнительного входного узла с двоичным кодированием: 0 – праздничный день, 1 – рабочий день.

Следующий фактор, который учитывался в прогнозе, – это деление суток на четыре периода: равномерный ночной, пиковый утренний, равномерный дневной и пиковый вечерний. Принятое разделение суток предусматривало смещение выделенных периодов соответственно различным временам года. Для двоичного кодирования этих четырех периодов в сеть были введены еще два входных узла.

И все же важнейшим фактором стал учет зависимости прогноза от значений нагрузки в предыдущие часы и дни (динамические зависимости). Необходимо учитывать как текущий день, так и несколько дней, предшествующих прогнозируемому. При прогнозировании нагрузки $P(D, h)$ на h-й час в D-й день во входном векторе сети учитываются следующие величины: $P(D, h–1)$, $P(D, h–2)$, ..., $P(D, h–g)$, $P(D–1, h)$, $P(D–1, h–1)$, ..., $P(D–1, h–g)$, ..., $P(D–d, h)$, $P(D–d, h–1)$, ..., $P(D–d, h–g)$. Число d указывает, сколько предшествующих дней, а число g – сколько предшествующих часов принимается во внимание при прогнозировании. Проведенные исследования показали, что удовлетворительные результаты достигаются при $d = 3$ и $g = 4$. С учетом двух типов дня при разделении суток на четыре периода размерность входного вектора равна 22.

Последняя задача подготовки данных состояла в их разделении на обучающее и тестовое подмножества. Принимая во внимание огромную базу данных PSE, было решено ограничиться избранными днями, представляющими все времена года за последние несколько лет.

Для прогнозирования нагрузок использовалась сигмоидальная сеть с одним скрытым слоем. Объем входного слоя выбран равным размерности входного вектора x. Количество выходных нейронов определяется количеством прогнозируемых периодов. Соответственно для 24-часового прогнозирования выходной слой должен состоять из 24 линейных нейронов. Самая трудная задача – подбор количества нейронов скрытого слоя. Если их слишком мало, то погрешность обучения невозможно уменьшить до требуемого уровня. Слишком большое их количество приводит к росту погрешности обобщения. Такая сеть с практической точки зрения не будет иметь никакой ценности. Как правило, количество скрытых нейронов можно либо подобрать экспериментально так, чтобы уменьшить до минимума погрешность обобщения, либо применить один из методов построения оптимальной структуры сети, представленных ранее в настоящем разделе. Процесс прогнозирования нагрузок состоит из следующих этапов.

- Подбор архитектуры нейронной сети.
- Выбор обучающих данных и структуры входных векторов.
- Тренинг нейронной сети.
- Тестирование сети на контрольном множестве данных и при необходимости ее дообучение.
- Использование сети в качестве средства прогнозирования почасовой нагрузки (этап фактического использования по назначению).
- Возможное дообучение сети по истечении определенного времени, например одного года эксплуатации.

Качество прогнозирования оценивается показателем процентной погрешности MAPE (англ.: *Mean Absolute Percentage Error*), определяемым в виде

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|P_t - \hat{P}_t\right|}{P_t}\cdot 100\% \quad , \tag{4.47}$$

где $\hat{P}$ – прямо спрогнозированное значение, а P – фактическая нагрузка системы, тогда как n обозначено число часов, на которые составлялся прогноз.

Рассмотрим результаты 24-часового прогноза для PSE, полученные при помощи персептронной сети с одним скрытым слоем, состоящим из 25 нейронов. Структура сети имела вид: 22-25-24. Сеть обучалась с использованием данных за 1993 и 1994 гг. (выбрана четверть дней различных времен года). Тестирование проводилось на данных 1991–1995 гг. На рис. 4.18 представлено распределение погрешности MAPE для этих лет по 24-часовому прогнозу. Минимальная погрешность, полученная для 1995 г., составила 3,4%. Несколько лучшие результаты получены при прогнозировании нагрузок только по рабочим дням. На рис. 4.19 приведены погреш-

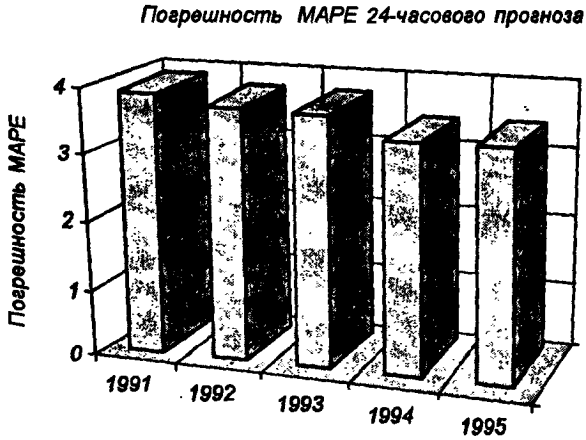

Рис. 4.18. Распределение погрешности MAPE прогноза 24-часовой нагрузки для PSE, рассчитанного персептронной сетью по всем дням 1991–1995 гг.

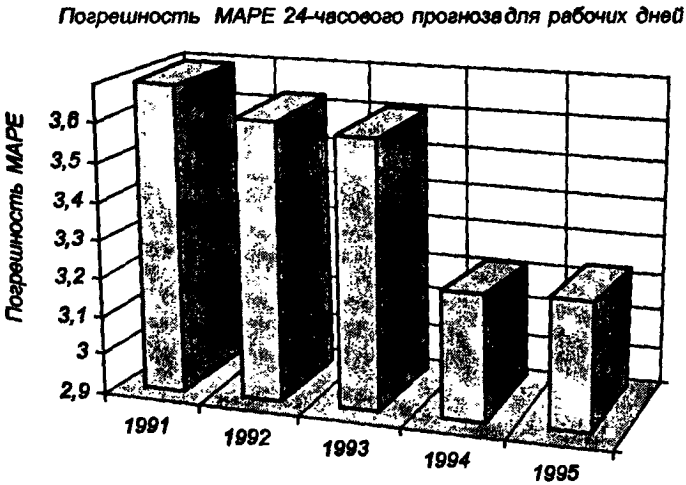

Рис. 4.19. Распределение погрешности MAPE прогноза 24-часовой нагрузки для PSE, рассчитанного персептронной сетью по всем рабочим дням 1991–1995 гг.

ности MAPE 24-часового прогноза, рассчитанные для рабочих дней. Минимальная погрешность MAPE для 1995 г. составила в этом случае 3,24 %. Аиализ ежедневного распределения погрешности показал возрастание точности прогнозирования для дней со стабильно высокой нагрузкой (например, рабочие и зимние дни) и снижение точности для летних месяцев, в частности в период отпусков. Стабильно высокой оказалась точность прогноза на часы, в которые нагрузка изменяется незначительно (например, на 4-00, 10-00, 14-00, 22-00, 24-00). В то же время фиксировался рост средней

погрешности для часов, в которые ожидались значительные колебания нагрузок системы, связанные с организацией повседневной жизни (например, 7-00, 16-00, 19-00). Можно сделать общий вывод, что использование многослойной персептронной сети не позволяет существенно снизить погрешность 24-часового прогноза. Лучших результатов, как это будет показано в разделе 9, можно достичь при использовании самоорганизующихся сетей.

Однако многослойный персептрон является очень хорошим средством для предсказания среднесуточных нагрузок энергетической системы. Значения таких нагрузок требуются, например, при прогнозировании с применением самоорганизации. В [124] авторы представили структуру персептронной сети с одним скрытым слоем, позволяющую весьма точно предсказывать среднесуточные нагрузки по тем же самым значениям, взятым из прошлых периодов. Прогнозирующая модель содержит девять входных узлов, представляющих среднесуточные нагрузки данного дня за последние годы, время года и тип дня. Тип дня кодировался одним двоичным узлом (0 – праздник, 1 – рабочий день). Кодирование времени года требует двух узлов. Применялись следующие коды: 11 – зима, 01 – весна, 00 – лето и 10 – осень. На этапе обучения сети в качестве ожидаемых значений выступали известные среднесуточные нагрузки энергетической системы за прошедшие годы.

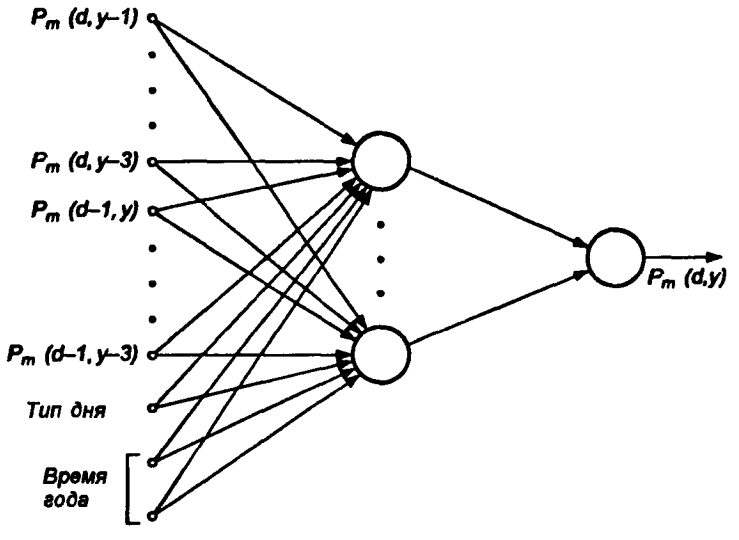

Рис. 4.20. Структура персептронной сети для предсказания среднесуточных нагрузок

Структура нейронной сети, применявшейся для предсказания нагрузок, изображена на рис. 4.20. Значение $P_m(d, y)$ соответствует нормализованной среднесуточной нагрузке в день d года y. Для улучшения способностей к обобщению количество скрытых нейронов было подобрано экспериментально (в рассматриваемом примере оно было принято равным 5). Сеть была

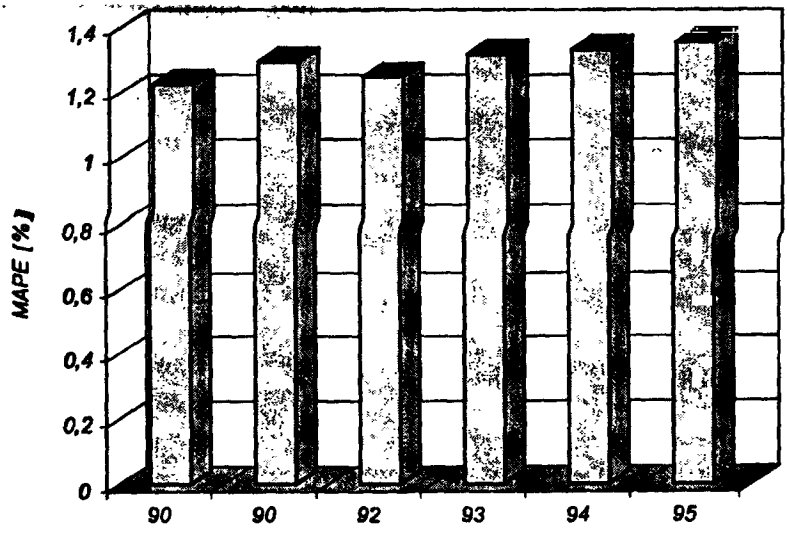

Рис. 4.21. Распределение погрешности MAPE прогноза среднесуточной нагрузки для PSE, рассчитанного персептронной сетью MAPE

обучена по данным PSE за 1990–1995 гг. На рис. 4.21 представлено распределение погрешности MAPE прогноза среднесуточных значений на период 1990–1995 гг. Наибольшая погрешность MAPE в течение года не превышала 1,3 %.

Раздел 5

РАДИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ

Многослойные нейронные сети, представленные в предыдущих разделах, с точки зрения математики выполняют аппроксимацию стохастической функции нескольких переменных путем преобразования множества входных переменных $x \in R^N$ во множество выходных переменных $y \in R^M$ [46, 56]. Вследствие характера сигмоидальной функции активации осуществляется аппроксимация глобального типа. В результате ее нейрон, который был однажды включен (после превышения суммарным сигналом u_i определенного порогового значения), остается в этом состоянии при любом значении u_i, превышающем этот порог. Поэтому всякий раз преобразование значения функции в произвольной точке пространства выполняется объединенными усилиями многих нейронов, что и объясняет название *глобальная аппроксимация.*

Другой способ отображения входного множества в выходное заключается в преобразовании путем адаптации нескольких одиночных аппроксимирующих функций к ожидаемым значениям, причем эта адаптация проводится только в ограниченной области многомерного пространства. При таком подходе отображение всего множества данных представляет собой сумму локальных преобразований. С учетом роли, которую играют скрытые нейроны, они составляют множество базисных функций локального типа. Выполнение одиночных функций (при ненулевых значениях) регистрируется только в ограниченной области пространства данных – отсюда и название *локальная аппроксимация.*

Особое семейство образуют сети с радиальной базисной функцией, в которых скрытые нейроны реализуют функции, радиально изменяющиеся вокруг выбранного центра $\vec{n}$ и принимающие ненулевые значения только в окрестности этого центра. Подобные функции, определяемые в виде $\varphi(x) = \varphi(\|x - c\|)$, будем называть *радиальными базисными функциями.* В таких сетях роль скрытого нейрона заключается в отображении радиального пространства вокруг одиночной заданной точки либо вокруг группы таких точек, образующих кластер. Суперпозиция сигналов, поступающих от всех скрытых нейронов, которая выполняется выходным нейроном, позволяет получить отображение всего многомерного пространства.

Сети радиального типа представляют собой естественное дополнение сигмоидальных сетей. Сигмоидальный нейрон представляется в многомерном пространстве гиперплоскостью, которая разделяет это пространство на две

категории (два класса), в которых выполняется одно из двух условий: либо $\sum_j w_{ij} x_j > 0$, либо $\sum_j w_{ij} x_j < 0$. Такой подход продемонстрирован на рис. 5.1*а*. В свою очередь радиальный нейрон представляет собой гиперсферу, которая осуществляет шаровое разделение пространства вокруг центральной точки (рис. 5.1*б*). Именно с этой точки зрения он является естественным дополнением сигмоидального нейрона, поскольку в случае круговой симметрии данных

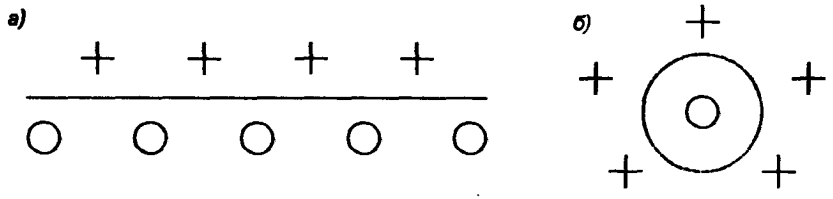

Рис. 5.1. Иллюстрация способов разделения пространства данных: *а*) сигмоидальным нейроном; *б*) радиальным нейроном

позволяет заметно уменьшить количество нейронов, необходимых для разделения различных классов. Поскольку нейроны могут выполнять различные функции, в радиальных сетях отсутствует необходимость использования большого количества скрытых слоев. Структура типичной радиальной сети включает входной слой, на который подаются сигналы, описываемые входным вектором $\boldsymbol{x}$, скрытый слой с нейронами радиального типа и выходной слой, состоящий, как правило, из одного или нескольких линейных нейронов. Функция выходного нейрона сводится исключительно к взвешенному суммированию сигналов, генерируемых скрытыми нейронами.

5.1. Математические основы

Математическую основу функционирования радиальных сетей составляет теорема Т. Ковера [20] о распознаваемости образов, в соответствии с которой нелинейные проекции образов в некоторое многомерное пространство могут быть линейно разделены с большей вероятностью, чем при их проекции в пространство с меньшей размерностью.

Если вектор радиальных функций $\varphi(x) = [\varphi_1(x), \varphi_2(x), ..., \varphi_K(x)]^T$ в N-мерном входном пространстве обозначить $\varphi(x)$, то это пространство является нелинейно φ-разделяемым на два пространственных класса X^+ и X^- тогда, когда существует такой вектор весов w, что

$$w^T \varphi(x) > 0 \quad x \in X^+ , \tag{5.1}$$

$$w^T \varphi(x) < 0 \quad x \in X^- . \tag{5.2}$$

Граница между этими классами определяется уравнением $w^T \varphi(x) = 0$.

В [20] доказано, что каждое множество образов, случайным образом размещенных в многомерном пространстве, является φ-разделяемым с вероятностью 1 при условии соответственно большой размерности K этого пространства. На практике это означает, что применение достаточно большого количества скрытых нейронов, реализующих радиальные функции $\varphi_i(x)$, гарантирует решение задачи классификации при построении всего лишь двухслойной сети: скрытый слой должен реализовать вектор $\varphi(x)$, а выходной слой может состоять из единственного линейного нейрона, выполняющего суммирование выходных сигналов от скрытых нейронов с весовыми коэффициентами, заданными вектором w.

Простейшая нейронная сеть радиального типа функционирует по принципу многомерной интерполяции, состоящей в отображении p различных входных векторов x_i ($i = 1, 2, \ldots p$) из входного N-мерного пространства во множество из p рациональных чисел d_i ($i = 1, 2, \ldots p$). Для реализации этого процесса необходимо использовать p скрытых нейронов радиального типа и задать такую функцию отображения $F(x)$, для которой выполняется условие интерполяции

$$F(x_i) = d_i. \tag{5.3}$$

Использование p скрытых нейронов, соединяемых связями с весами w_i с выходными линейными нейронами, означает формирование выходных сигналов сети путем суммирования взвешенных значений соответствующих базисных функций. Рассмотрим радиальную сеть с одним выходом и p обучающими парами (x_i, d_i). Примем, что координаты каждого из p центров узлов сети определяются одним из векторов x_i, т.е. $c_i = x_i$. В этом случае взаимосвязь между входными и выходными сигналами сети может быть определена системой уравнений, линейных относительно весов w_i, которая в матричной форме имеет вид:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1p} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{p1} & \varphi_{p2} & \cdots & \varphi_{pp} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_p \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_p \end{bmatrix}, \tag{5.4}$$

где $\varphi_{ji} = (\| x_j - x_i \|)$ определяет радиальную функцию с центром в точке x_i с вынужденным вектором x_j. Если обозначить матрицу из элементов φ_{ji} как Φ и ввести обозначения векторов $w = [w_1, w_2, \ldots, w_p]^T$, $d = [d_1, d_2, \ldots, d_p]^T$, система уравнений (5.4) может быть представлена в редуцированной матричной форме

$$\Phi w = d. \tag{5.5}$$

В [20] доказано, что для ряда радиальных функций в случае $x_1 \neq x_2 \neq \ldots \neq x_p$ квадратная интерполяционная матрица Φ является несобственной и при этом неотрицательно определенной. Поэтому существует решение уравнения (5.5) в виде

$$w = \Phi^{-1} d, \tag{5.6}$$

что позволяет получить вектор весов выходного нейрона сети.

Теоретическое решение проблемы, представленное выражением (5.6), не может считаться абсолютно истинным по причине серьезного ограничения общих свойств сети, вытекающих из сделанных вначале допущений. При очень большом количестве обучающих выборок и равном ему количестве радиальных функций проблема с математической точки зрения становится бесконечной (плохо структурированной), поскольку количество уравнений начинает превышать число степеней свободы физического процесса, моделируемого уравнением (5.4). Это означает, что результатом такого чрезмерного количества весовых коэффициентов станет адаптация модели к разного рода шумам или нерегулярностям, сопровождающим обучающие выборки. Как следствие, интерполирующая эти данные гиперплоскость не будет гладкой, а обобщающие возможности останутся очень слабыми.

Чтобы их усилить, следует уменьшить количество радиальных функций и получить из избыточного объема данных дополнительную информацию для регуляризации задачи и улучшения ее обусловленности.

5.2. Радиальная нейронная сеть

Использование в разложении p базисных функций, где p – это количество обучающих выборок, недопустимо также и с практической точки зрения, поскольку обычно количество этих выборок очень велико, и в результате вычислительная сложность обучающего алгоритма становится чрезмерной. Решение системы уравнений (5.4) размерностью (p Ч p) при больших значениях p становится затруднительным, так как очень большие матрицы (за исключением ортогональных), как правило, имеют порядковый характер, а коэффициент порядка может достигать величины даже 1020. Поэтому так же как и для многослойных сетей, необходимо редуцировать количество весов, что в этом случае сводится к уменьшению количества базисных функций. Поэтому ищется субоптимальное решение в пространстве меньшей размерности, которое с достаточной точностью аппроксимирует точное решение. Если ограничиться K базисными функциями, то аппроксимирующее решение можно представить в виде

$$F(x) = \sum_{i=1}^{K} w_i \varphi(\|x - c_i\|) , \qquad (5.7)$$

где $K < p$, а c_i (i = 1, 2, ..., K) – множество центров, которые необходимо определить. В особом случае, если принять $K = p$, то можно получить точное решение $c_i = x_i$.

Задача аппроксимации состоит в подборе соответствующего количества радиальных функций $\varphi(\|x - c_i\|)$ и их параметров, а также в таком подборе весов w_i (i = 1, 2, ..., K), чтобы решение уравнения (5.7) было наиболее близким точному. Поэтому проблему подбора параметров радиальных функций и значений весов w_i сети можно свести к минимизации целевой функции, которая пр

использовании метрики Эвклида записывается в форме

$$E=\sum_{i=1}^{p}\left[\sum_{j=1}^{K} w_j \varphi(\| x_i - c_i \|) - d_i\right]^2 . \qquad (5.8)$$

В этом уравнении K представляет количество радиальных нейронов, а p – количество обучающих пар (x_i, d_i), где x_i – это входной вектор, а d_i – соответствующая ему ожидаемая величина. Обозначим $\boldsymbol{d} = [d_1, d_2, ..., d_p]^T$ вектор ожидаемых значений, $\boldsymbol{w} = [w_1, w_2, ..., w_K]^T$ – вектор весов сети, а $\mathbf{G}$ – радиальную матрицу, называемую матрицей Грина [46].

$$\mathbf{G}=\begin{bmatrix} \varphi(\| x_1 - c_1 \|) & \varphi(\| x_1 - c_2 \|) & \cdots & \varphi(\| x_1 - c_K \|) \\ \varphi(\| x_2 - c_1 \|) & \varphi(\| x_2 - c_2 \|) & \cdots & \varphi(\| x_2 - c_K \|) \\ \cdots & \cdots & \cdots & \cdots \\ \varphi(\| x_p - c_1 \|) & \varphi(\| x_p - c_2 \|) & \cdots & \varphi(\| x_p - c_K \|) \end{bmatrix}.$$

При ограничении K базисными функциями матрица $\mathbf{G}$ становится прямоугольной с количеством строк, как правило, значительно большим, чем число столбцов ($p >> K$).

Если допустить, что параметры радиальных функций известны, то оптимизационная задача (5.8) сводится к решению системы уравнений, линейных относительно весов $\boldsymbol{w}$ [46]

$$\mathbf{G}(\boldsymbol{w}) = \boldsymbol{d} . \qquad (5.9)$$

Вследствие прямоугольности матрицы $\mathbf{G}$ можно определить вектор весов $\boldsymbol{w}$ с использованием операции псевдоинверсии матрицы $\mathbf{G}$, т.е.

$$\boldsymbol{w} = \mathbf{G}^{+}\boldsymbol{d} , \qquad (5.10)$$

где $\mathbf{G}^{+}=(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$ обозначает псевдоинверсию прямоугольной матрицы $\mathbf{G}$. В вычислительной практике псевдоинверсия рассчитывается с применением декомпозиции SVD [42].

В обсуждаемом до этого момента решении использовалось представление базисных функций матрицей Грина, зависящее от эвклидовой нормы вектора $\| x - t_i \|$. Если принять во внимание, что многомерная функция может иметь различный масштаб по каждой оси, с практической точки зрения оказывается полезным уточнить норму масштабирования путем ввода в определение эвклидовой метрики весовых коэффициентов в виде матрицы $\mathbf{Q}$:

$$\| x \|_Q^2 = (\mathbf{Q}x)^T(\mathbf{Q}x) = x^T\mathbf{Q}^T\mathbf{Q}x . \qquad (5.11)$$

Масштабирующая матрица при N-мерном векторе x имеет вид:

$$\mathbf{Q}=\begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1N} \\ Q_{21} & Q_{21} & \cdots & Q_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ Q_{N1} & Q_{N2} & \cdots & Q_{NN} \end{bmatrix}.$$

При обозначении произведения матриц $\mathbf{Q}^T\mathbf{Q}$ матрицей корреляции $\mathbf{C}$ в общем случае получим:

$$\| x \|_Q^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} x_i x_j \ . \tag{5.12}$$

Если масштабирующая матрица $\mathbf{Q}$ имеет диагональный вид, то получаем $\| x \|_Q^2 = \sum_{i=1}^{N} C_{ij} x_i^2$. Это означает, что норма масштабирования вектора x рассчитывается согласно стандартной формуле Эвклида, с использованием индивидуальной шкалы для каждой переменной x_i. При $\mathbf{Q} = \mathbf{1}$ взвешенная метрика Эвклида сводится к классической (немасштабируемой) метрике $\| x \|_Q^2 = \| x \|^2$.

Чаще всего в качестве радиальной функции применяется функция Гаусса. При размещении ее центра в точке c_i она может быть определена в сокращенной форме как

$$\varphi(x) = \varphi(\| x - c_i \|) = \exp\left(-\frac{\| x - c_i \|^2}{2\sigma_i^2}\right). \tag{5.13}$$

В этом выражении σ_i – параметр, от значения которого зависит ширина функции. В случае гауссовской формы радиальной функции с центром в точке c_i и масштабирующей взвешенной матрицы $\mathbf{Q}_i$, связанной с i-й базисной функцией, получаем обобщенную форму функции Гаусса

$$\varphi(x) = \varphi(\| x - c_i \|_{Q_i}) = \exp\left[-(x - c_i)^T \mathbf{Q}_i^T \mathbf{Q}_i (x - c_i)\right] =$$
$$= \exp\left[\frac{1}{2}(x - c_i)^T \mathbf{C}_i (x - c_i)\right], \tag{5.14}$$

где матрица $\frac{1}{2}\mathbf{C}_i = \mathbf{Q}_i^T\mathbf{Q}_i$ играет роль скалярного коэффициента $\frac{1}{2\sigma_i^2}$ стандартной многомерной функции Гаусса, заданной выражением (5.13).

Полученное решение, представляющее аппроксимирующую функцию в многомерном пространстве в виде взвешенной суммы локальных базисных радиальных функций (выражение (5.7)), может быть интерпретировано радиальной нейронной сетью, представленной на рис. 5.2 (для упрощения эта сеть имеет только один выход), в которой φ_i определяется зависимостью (5.13) либо (5.14). Это сеть с двухслойной структурой, в которой только скрытый слой выполняет нелинейное отображение, реализуемое нейронами с базисными радиальными функциями. Выходной нейрон, как правило, линеен, а его роль сводится к взвешенному суммированию сигналов, поступающих от нейронов скрытого слоя. Вес w_0, как и при использовании сигмоидальных функций, представляет поляризацию, вводящую показатель постоянного смещения функции.

Полученная архитектура радиальных сетей имеет структуру, аналогичную многослойной структуре сигмоидальных сетей с одним скрытым слоем. Роль

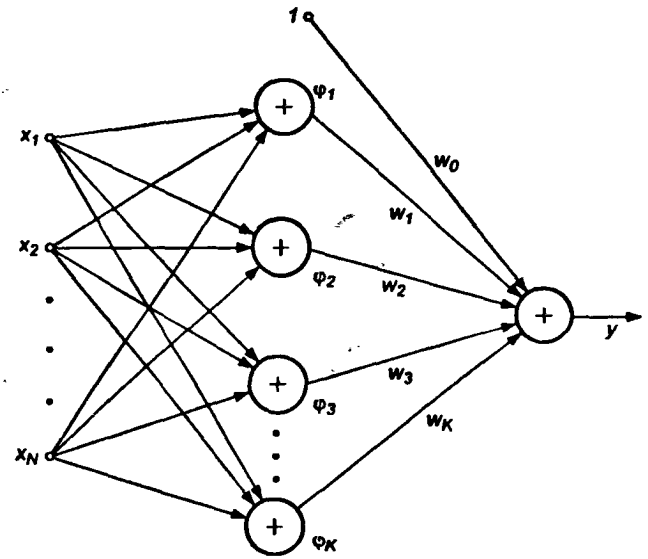

Рис. 5.2. Обобщенная структура радиальной сети RBF

скрытых нейронов в ней играют базисные радиальные функции, отличающиеся своей формой от сигмоидальных функций. Несмотря на отмеченное сходство, сети этих типов принципиально отличаются друг от друга. Радиальная сеть имеет фиксированную структуру с одним скрытым слоем и линейными выходными нейронами, тогда как сигмоидальная сеть может содержать различное количество слоев, а выходные нейроны бывают как линейными, так и нелинейными. Используемые радиальные функции могут иметь весьма разнообразную структуру [46, 60, 160]. Нелинейная радиальная функция каждого скрытого нейрона имеет свои значения параметров c_i и s_i, тогда как в сигмоидальной сети применяются, как правило, стандартные функции активации с одним и тем же для всех нейронов параметром β. Аргументом радиальной функции является эвклидово расстояние образца x от центра c_i, а в сигмоидальной сети это скалярное произведение векторов $w^T x$.

Еще большие отличия между этими сетями можно заметить при детальном сравнении их структур. Сигмоидальная сеть имеет многослойную структуру, в которой способ упорядочения нейронов повторяется от слоя к слою. Каждый нейрон в ней выполняет суммирование сигналов с последующей активацией. Структура радиальной сети несколько иная. На рис. 5.3 изображена подробная схема сети RBF с радиальной функцией вида (5.13) при классическом понимании эвклидовой метрики. Из рисунка видно, что первый слой составляют нелинейные радиальные функции, параметры которых (центры c_i и коэффициенты s_i) уточняются в процессе обучения. Первый слой не содержит линейных весов в понимании, характерном для сигмоидальной сети.

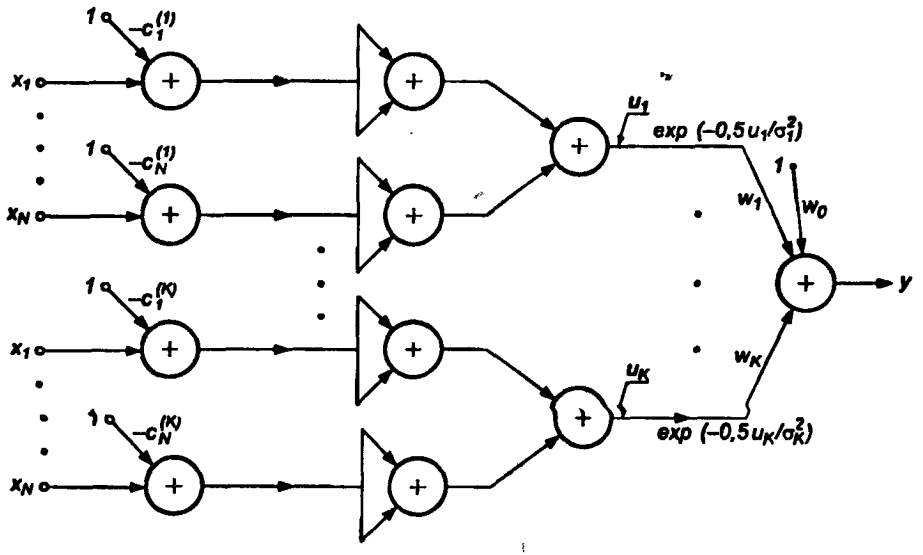

Рис. 5.3. Детальная схема структуры радиальной сети RBF

Еще более сложной оказывается детальная структура сети, реализующей масштабированную радиальную функцию в виде, определенном выражением (5.14). Такая сеть, представленная на рис. 5.4, называется HRBF (англ.: *Hyper Radial Basis Function*). Радиальный нейрон в ней имеет особенно сложную структуру, содержащую и сумматоры сигналов, аналогичные применяемым в сигмоидальной сети, и показательные

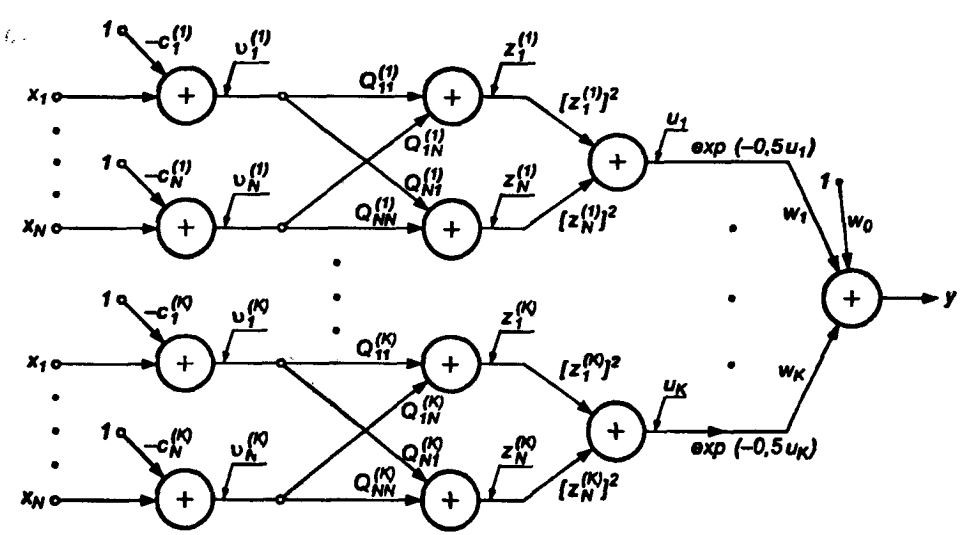

Рис. 5.4. Детальная схема структуры радиальной сети HRBF с масштабирующей матрицей Q произвольного вида

функции активации с параметрами, подлежащими уточнению в процессе обучения. Веса $Q_{ij}^{(k)}$ k-го радиального нейрона скрытого слоя – это элементы матрицы $\mathbf{Q}(k)$, играющие роль масштабирующей системы. Они вводят дополнительные степени свободы сети, что позволяет лучше приблизить выходной сигнал сети $y = f(x)$ к ожидаемой функции $d(x)$.

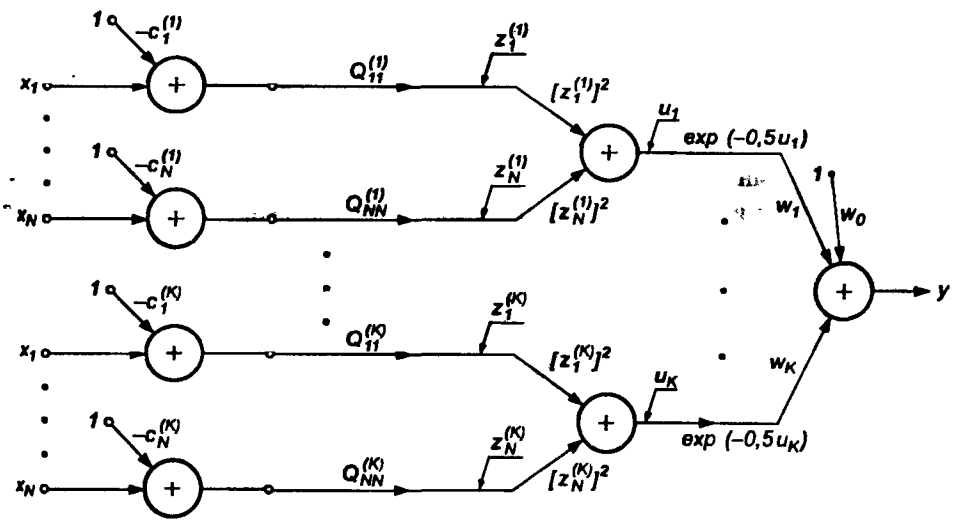

Рис. 5.5. Детальная схема структуры радиальной сети HRBF с диагональной масштабирующей матрицей **Q**

Во многих практических приложениях масштабирующая матрица $\mathbf{Q}(k)$ имеет диагональную форму, в которой только элементы $Q_{ii}^{(k)}$ принимают ненулевые значения. В такой системе отсутствует круговое перемешивание сигналов, соответствующих различным компонентам вектора x, а элемент $Q_{ii}^{(k)}$ играет роль индивидуального масштабирующего коэффициента для i-го компонента вектора x k-го нейрона. На рис. 5.5 представлена структура упрощенной сети HRBF с диагональными матрицами $\mathbf{Q}(k)$. Следует отметить, что в сетях HRBF роль коэффициентов σ_i^2 выполняют элементы матрицы **Q**, которые уточняются в процессе обучения.

5.3. Методы обучения радиальных нейронных сетей

Введенные в предыдущем подразделе методы подбора весов w_i выходного слоя радиальной сети RBF были основаны на предположении, что параметры самих базисных функций известны, в связи с чем матрицы Грина считаются определенными, и, следовательно, задача сводится к решению избыточной системы линейных уравнений вида (5.9). Практически такой подход

возможен только в абсолютно нереальном случае при $K = p$, при котором центры $c_i = x_i$ известны заранее, а значение параметра σ_i можно легко подобрать экспериментальным путем при соблюдении определенного компромисса между монотонностью и точностью отображения. В действительности всегда $K << p$, поэтому процесс обучения сети RBF с учетом выбранного типа радиальной базисной функции сводится:

- к подбору центров c_i и параметров σ_i формы базисных функций;
- к подбору весов нейронов выходного слоя.

При этом проблема уточнения весов нейронов выходного слоя значительно упрощается. В соответствии с формулой (5.10) вектор весов w может быть определен за один шаг псевдоинверсией матрицы **G**, $w = \mathbf{G}^{+}d$. Матрица **G**, имеющая p строк и K столбцов, представляет реакции нейронов скрытого слоя на очередные возбуждения векторами x_i ($i = 1, 2, ..., p$). Практически псевдоинверсия матрицы **G** рассчитывается с использованием разложения по собственным значениям, в соответствии с которым

$$\mathbf{G} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{5.15}$$

Матрицы **U** и **V** ортогональны и имеют размерности (pЧp) и (KЧK) соответственно, тогда как **S** – это псевдодиагональная матрица с размерностью (pЧK). При этом $K < p$, а диагональные элементы $s_1 \geq s_2 \ldots \geq s_K \geq 0$. Допустим, что только r первых элементов s_i имеют значимую величину, а остальными можно пренебречь. Тогда количество столбцов ортогональных матриц **U** и **V** может быть уменьшено до r. Полученные таким образом редуцированные матрицы $\mathbf{U}_r$ и $\mathbf{V}_r$ имеют вид:

$$\mathbf{U}_r = [\boldsymbol{u}_1 \boldsymbol{u}_2 \cdots \boldsymbol{u}_r],$$

$$\mathbf{V}_r = [\boldsymbol{v}_1 \boldsymbol{v}_2 \cdots \boldsymbol{v}_r],$$

а матрица $\mathbf{S}_r = \mathrm{diag}[s_1, s_2, ..., s_r]$ становится полностью диагональной (квадратной). Эту матрицу описывает зависимость (5.15) в форме

$$\mathbf{G} \cong \mathbf{U}_r \mathbf{S}_r \mathbf{V}_r^{T} . \tag{5.16}$$

Псевдообратная к **G** матрица определяется в этом случае выражением

$$\mathbf{G}^{+} = \mathbf{V}_r \mathbf{S}_r^{-1} \mathbf{U}_r^{T} , \tag{5.17}$$

в котором $\mathbf{S}_r^{-1}[1/s_1, 1/s_2, ..., 1/s_r]$, а вектор весов сети, подвергающейся обучению, задается формулой

$$w = \mathbf{V}_r \mathbf{S}^{-1} \mathbf{U}_r^{T} d . \tag{5.18}$$

Достоинство формулы (5.18) – ее простота. Выходные веса сети подбираются за один шаг простым перемножением соответствующих матриц, при этом некоторые из них ($\mathbf{U}_r, \mathbf{V}_r$) ортогональные и по своей природе хорошо упорядочены (коэффициент порядка равен 1).

Принимая во внимание решение (5.18), определяющее значения весов выходного слоя, главной проблемой обучения радиальных сетей остается подбор параметров нелинейных радиальных функций, особенно центров c_i.

Одним из простейших, хотя и не самым эффективным, способом определения параметров базисных функций, считается случайный выбор. В этом решении центры c_i базисных функций выбираются случайным образом на основе равномерного распределения. Такой подход допустим применительно к классическим радиальным сетям при условии, что равномерное распределение обучающих данных хорошо соответствует специфике задачи. При выборе гауссовской формы радиальной функции задается значение стандартного отклонения s_i, зависящее от разброса выбранных случайным образом центров c_i:

$$\varphi(\| x - c_i \|^2) = \exp\left(- \frac{\| x - c_i \|^2}{\frac{d^2}{K}} \right) \qquad (5.19)$$

для $i = 1, 2, ..., K$, где d обозначает максимальное расстояние между центрами c_i. Из выражения (5.19) следует, что стандартное отклонение гауссовской функции, характеризующее ширину кривой, устанавливается при случайном выборе равным $\sigma = \frac{d}{\sqrt{2K}}$ и постоянно для всех базисных функций. Ширина функции пропорциональна максимальному разбросу центров и уменьшается с ростом их количества.

Среди многих специализированных методов подбора центров рассмотрим несколько наиболее важных: самоорганизующийся процесс разделения на кластеры, гибридный алгоритм и обучение с учителем.

5.3.1. Применение процесса самоорганизации для уточнения параметров радиальных функций

Неплохие результаты уточнения параметров радиальных функций можно получить при использовании алгоритма самоорганизации. Процесс самоорганизации обучающих данных автоматически разделяет пространство на так называемые области Вороного, определяющие различающиеся группы данных. Пример такого разделения двухмерного пространства показан на рис. 5.6. Данные, сгруппированные внутри кластера, представляются центральной точкой, определяющей среднее значение всех его элементов. Центр кластера в дальнейшем будем отождествлять с центром соответствующей радиальной функции. По этой причине количество таких функций равно количеству кластеров и может корректироваться алгоритмом самоорганизации.

Разделение данных на кластеры можно выполнить с использованием одной из версий алгоритма Линде–Бузо–Грея [89], называемого также алгоритмом *K*-усреднений (англ.: *K-means*). В прямой (онлайн) версии этого алгоритма уточнение

центров производится после предъявления каждого очередного вектора x из множества обучающих данных. В накопительной версии (оффлайн) центры уточняются одновременно после предъявления всех элементов множества. В обоих случаях предварительный выбор центров выполняется чаще всего случайным образом с использованием равномерного распределения.

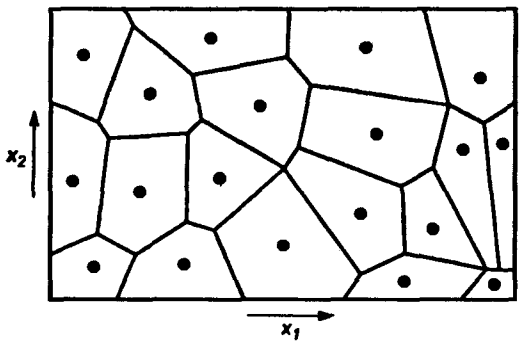

Рис. 5.6. Иллюстрация способа разделения пространства данных на сферы влияния отдельных радиальных функций

Если обучающие данные представляют непрерывную функцию, начальные значения центров в первую очередь размещают в точках, соответствующих всем максимальным и минимальным значениям функции. Данные об этих центрах и их ближайшем окружении впоследствии удаляются из обучающего множества, а оставшиеся центры равномерно распределяются в сфере, образованной оставшимися элементами этого множества.

В прямой версии после предъявления k-го вектора x_k, принадлежащего обучающему множеству, выбирается центр, ближайший к x_k, относительно применяемой метрики. Этот центр подвергается уточнению в соответствии с алгоритмом WTA

$$c_i(k+1) = c_i(k) + \eta\,[x_k - c_i(k)], \tag{5.20}$$

где η – коэффициент обучения, имеющий малое значение (обычно $\eta << 1$), причем уменьшающееся во времени. Остальные центры не изменяются. Все обучающие векторы x предъявляются по несколько раз, как правило, в случайной последовательности вплоть до стабилизации значений центров.

Также применяется разновидность алгоритма, в соответствии с которой значение центра-победителя уточняется в соответствии с формулой (5.20), а один или несколько ближайших к нему центров отодвигаются в противоположном направлении [83], и этот процесс реализуется согласно выражению

$$c_i(k+1) = c_i(k) - \eta\,[x_k - c_i(k)]. \tag{5.21}$$

Такая модификация алгоритма позволяет отдалить центры, расположенные близко друг к другу, что обеспечивает лучшее обследование всего пространства данных ($\eta_1 < \eta$).

В накопительной версии предъявляются все обучающие векторы x, и каждый из них сопоставляется какому-либо центру. Множество векторов, приписанных одному и тому же центру, образует кластер, новый центр которого определяется как среднее соответствующих векторов:

$$c_i(k+1)=\frac{1}{N_i}\sum_{j=1}^{N_i}x_j(k)\ . \qquad (5.22)$$

В этом выражении N – количество векторов $x(k)$, приписанных в k-м цикле к i-му центру. Значения всех центров уточняются параллельно. Процесс предъявления множества векторов x и уточнения значений центров повторяется многократно вплоть до стабилизации значений центров. На практике чаще всего применяется прямой алгоритм, имеющий несколько лучшую сходимость. Однако ни один алгоритм не гарантирует абсолютную сходимость к оптимальному решению в глобальном смысле, а обеспечивает только локальную оптимизацию, зависящую от начальных условий и параметров процесса обучения. При неудачно выбранных начальных условиях некоторые центры могут застрять в области, где количество обучающих данных ничтожно мало либо они вообще отсутствуют, поэтому процесс модификации центров затормозится или остановится. Способом разрешения этой проблемы считается одновременная корректировка размещения большого количества центров с фиксацией значения η для каждого из них. Центр, наиболее близкий к текущему вектору x, модифицируется сильнее всего, а остальные – обратно пропорционально их расстоянию до этого текущего вектора.

Другой подход состоит в использовании взвешенной меры расстояния от каждого конкретного центра до предъявляемого вектора x. Весовая норма делает "фаворитами" те центры, которые реже всего становились победителями. Оба подхода не гарантируют 100%-ную оптимальность решения, поскольку представляют собой фактически процедуры возмущения предопределенного процесса локальной оптимизации [11]. Трудность состоит также в подборе коэффициента обучения η. При использовании постоянного значения η он должен быть очень малым для гарантированной сходимости алгоритма, что непомерно увеличивает время обучения. Адаптивные методы подбора η позволяют сделать его значение зависимым от времени, т.е. уменьшать по мере роста номера итерации k. Наиболее известным представителем этой группы считается алгоритм Даркена–Муди [11], согласно которому

$$\eta(k)=\frac{\eta_0}{1+\frac{k}{T}}\ . \qquad (5.23)$$

Коэффициент T обозначает постоянную времени, подбираемую индивидуально для каждой задачи. При $k < T$ значение η практически неизменно, но при $k > T$ оно постепенно уменьшается до нуля. Несмотря на то, что адаптивные методы подбора η более прогрессивны по сравнению с постоянным значением, они тоже не могут считаться наилучшим решением, особенно при моделировании динамических процессов.

После фиксации местоположения центров проводится подбор значений параметров σ_j, соответствующих конкретным базисным функциям. Параметр σ_j радиальной функции влияет на форму функции и величину области ее охвата, в

которой значение этой функции не равно нулю (точнее, превышает определенное пороговое значение ε). Подбор σ_j должен проводиться таким образом, чтобы области охвата всех радиальных функций накрывали все пространство входных данных, причем любые две зоны могут перекрываться только в незначительной степени. При такой организации подбора значения σ_j реализуемое радиальной сетью отображение функции будет относительно монотонным.

Проще всего в качестве значения σ_j j-й радиальной функции принять эвклидово расстояние между j-м центром $\boldsymbol{c}_j$ и его ближайшим соседом [154]. В другом алгоритме, учитывающем более широкое соседство, на значение σ_j влияет расстояние между j-м центром $\boldsymbol{c}_j$ и его P ближайшими соседями. В этом случае значение σ_j определяется по формуле

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{k=1}^{P} \| \boldsymbol{c}_j - \boldsymbol{c}_k \|^2} \ . \qquad (5.24)$$

На практике значение P обычно лежит в интервале [3 – 5].

При решении любой задачи ключевая проблема, определяющая качество отображения, состоит в предварительном подборе количества радиальных функций (скрытых нейронов). Как правило, при этом руководствуются общим принципом: чем больше размерность вектора $\boldsymbol{x}$, тем большее количество радиальных функций необходимо для получения удовлетворительного решения. Детальное описание процесса подбора количества радиальных функций будет представлено в последующих подразделах.

5.3.2. Вероятностный алгоритм подбора параметров радиальных функций

Требования к количеству скрытых нейронов можно смягчить применением сети типа HRBF, реализующей радиальное отображение с использованием взвешенной эвклидовой метрики. Коэффициенты $Q_{jk}^{(i)}$ масштабирующей матрицы $\mathbf{Q}_i$, связанные с соответствующими компонентами вектора $\boldsymbol{x}$, определяющими i-ю радиальную функцию (см. рис. 5.4), представляют собой еще одну группу параметров, подлежащих подбору и облегчающих аппроксимацию обучающих данных радиальной сетью. За счет увеличения количества подбираемых параметров требуемая точность может быть достигнута сетью HRBF при меньшем числе нейронов. На рис. 5.7 представлена примерная зависимость (в процентах) величины погрешности классификации 10-мерных обучающих данных, представляющих 3 класса, от количества скрытых нейронов [154] для радиальной сети RBF (кривая, обозначенная r) и для сети HRBF с диагональной матрицей $\mathbf{Q}$ (нижняя кривая на рисунке). Кроме заметного снижения уровня погрешности классификации, в используемой в данном примере сети HRBF количество скрытых нейронов было снижено со 160 (сеть RBF) до 110 (сеть HRBF).

Оптимальные значения центров и коэффициентов Q_{ij} для каждой базисной функции могут быть подобраны с помощью модифицированного алгоритма, изменяющего одновременно и характеристики

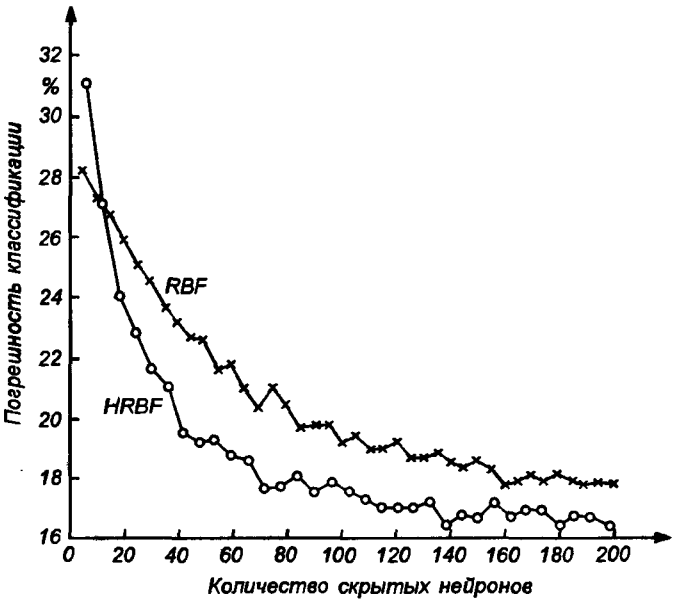

Рис. 5.7. Иллюстрация влияния архитектуры сетей RBF и HRBF на эффективность классификации при различном количестве радиальных нейронов

центров, и матрицу Q. Одним из таких алгоритмов, разработанных для сети HRBF с диагональной масштабирующей матрицей Q, является вероятностный алгоритм, предложенный в работе [154]. При равномерном распределении обучающих данных x и при использовании диагональной масштабирующей матрицы Q процесс адаптации центров и элементов матрицы Q_i описывается рекуррентными соотношениями

$$c_i(k+1)=\frac{c_i(k)+\alpha_k[\varphi_i(x_k)x_k-c_i(k)]}{(1-\alpha_k)+\alpha_k\varphi_i(x_k)}, \qquad (5.25)$$

$$\mathbf{F}_i(k+1)=\frac{\mathbf{F}_i(k)+\alpha_k[\varphi_i(x_k)[x_k-c_i(k)][x_k-c_i(k)]^T-\mathbf{F}_i(k)]}{(1-\alpha_k)+\alpha_k\varphi_i(x_k)}, \qquad (5.26)$$

где $\alpha_k=\frac{\alpha_0}{k}$ обозначает изменяющийся во времени коэффициент обучения, а α_0 – константа, подбираемая из интервала [0, 1] (чаще всего значение α_0 лежит в пределах [0,5 – 0,8]). В представляемом методе радиальная функция $\varphi(x)$ определяется в виде

$$\varphi(x)=\exp\left(-\frac{1}{2}[x-c_i(k)]^T\mathbf{F}_i^{-1}[x-c_i(k)]\right), \qquad (5.27)$$

где $\mathbf{F}_i = \mathrm{diag}[F_{11}, F_{22}, ..., F_{NN}]$. Ее значение соответствует условной вероятности того, что вектор $\boldsymbol{x}$ принадлежит к кластеру с центром $\boldsymbol{c}_i$. При таком определении матрицы $\mathbf{F}_i$ она связана с масштабирующей матрицей $\mathbf{Q}_i$, используемой в сети HRBF на рис. 5.4, соотношением

$$\mathbf{Q}_i = \sqrt{\frac{1}{2}\mathbf{F}_i^{-1}} \ . \tag{5.28}$$

Если обе матрицы имеют диагональную структуру, то $\mathbf{Q}_{jj} = \sqrt{\frac{1}{2\mathbf{F}_{jj}}}$.

В зависимостях (5.25) и (5.26) на каждом этапе выполнения алгоритма одновременно происходит адаптация и центров, и матрицы весов $\mathbf{F}$, причем уточняются параметры всех радиальных функций сети. Это существенно отличает описываемый метод от адаптивных зависимостей, реализуемых в описанном в предыдущем подразделе алгоритме K-усреднений, в соответствии с которым уточнялось значение только одного центра – победителя в конкурентной борьбе. Представленные формулы могут применяться и для сети RBF при условии, что $\mathbf{F}_i \equiv 1$ и что в соответствии с выражением (5.25) уточняются параметры только центра, имеющего наибольшее значение функции $\varphi_i(\boldsymbol{x})$.

5.3.3. Гибридный алгоритм обучения радиальных сетей

В гибридном алгоритме процесс обучения разделяется на два этапа:

1) подбор линейных параметров сети (веса выходного слоя) при использовании метода псевдоинверсии;
2) адаптация нелинейных параметров радиальных функций (центра $\boldsymbol{c}_i$ и ширины σ_i этих функций).

Оба этапа тесно переплетаются. При фиксации конкретных значений центров и ширины радиальных функций (в первый момент это будут начальные значения) за один шаг, с использованием декомпозиции SVD, подбираются величины линейных весов выходного слоя. Такая фиксация параметров радиальных функций позволяет определить значения самих функций $\varphi_i(\boldsymbol{x}_k)$ для $i = 1, 2, ..., K$ и $k = 1, 2, ..., p$, где i – это номер радиальной функции, а k – номер очередной обучающей пары $(\boldsymbol{x}_k, d_k)$. Очередные возбуждения $\boldsymbol{x}_k$ генерируют в скрытом слое сигналы, описываемые векторами $\varphi_k = [1, \varphi_1(\boldsymbol{x}_k), \varphi_2(\boldsymbol{x}_k), ..., \varphi_K(\boldsymbol{x}_k)]$, где 1 обозначает единичный сигнал поляризации. Им сопутствует выходной сигнал сети y_k, $y_k = \varphi_k \boldsymbol{w}$, причем вектор $\boldsymbol{w}$ содержит веса выходного слоя, $\boldsymbol{w} = [w_0, w_1, ..., w_K]^T$. При наличии p обучающих пар получаем систему уравнений

$$\begin{bmatrix} 1 & \varphi_1(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_1) & \cdots & \varphi_K(\boldsymbol{x}_1) \\ 1 & \varphi_1(\boldsymbol{x}_2) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_K(\boldsymbol{x}_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \varphi_1(\boldsymbol{x}_p) & \varphi_2(\boldsymbol{x}_p) & \cdots & \varphi_K(\boldsymbol{x}_p) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_K \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_p \end{bmatrix}, \tag{5.29}$$

которую в векторном виде можно записать как

$$\mathbf{G}\boldsymbol{w} = \boldsymbol{y}. \tag{5.30}$$

При использовании гибридного метода на этапе подбора выходных весов вектор $\boldsymbol{y}$ заменяется вектором ожидаемых значений $\boldsymbol{d} = [d_1, d_1, ..., d_p]^T$, и образованная при этом система уравнений $\mathbf{G}\boldsymbol{w} = \boldsymbol{d}$ решается за один шаг с использованием псевдоинверсии

$$\boldsymbol{w} = \mathbf{G}^{+}\boldsymbol{d}. \tag{5.31}$$

В алгоритме расчета псевдоинверсии применяется декомпозиция SVD, позволяющая получить текущее значение вектора $\boldsymbol{w}$ в соответствии с формулой (5.18).

На втором этапе при зафиксированных значениях выходных весов возбуждающие сигналы пропускаются по сети до выходного слоя, что позволяет рассчитать величину погрешности для последовательности векторов $\boldsymbol{x}_k$. Далее происходит возврат к скрытому слою (обратное распространение). По величине погрешности определяется вектор градиента целевой функции относительно конкретных центров c_{ij} и ширины σ_{ij}. Для последующего изложения предположим, что используется модель сети типа HRBF с диагональной формой масштабирующей матрицы Σ. Это означает, что каждая радиальная функция определяется в общем виде как

$$\varphi_i(\boldsymbol{x}_k) = \exp\left(-\frac{1}{2}u_{ik}\right), \tag{5.32}$$

где суммарный сигнал нейрона u_{ik} описывается выражением

$$u_{ik} = \sum_{j=1}^{N}\frac{(x_{jk} - c_{ij})^2}{\sigma_{ij}^2}. \tag{5.33}$$

При существовании p обучающих пар целевую функцию можно задать в виде

$$E = \frac{1}{2}\sum_{k=1}^{p}[y_k - d_k]^2 = \frac{1}{2}\sum_{k=1}^{p}\left[\sum_{i=0}^{K} w_i\varphi(\boldsymbol{x}_k) - d_k\right]^2. \tag{5.34}$$

В результате дифференцирования этой функции получаем:

$$\frac{\partial E}{\partial c_{ij}} = \sum_{k=1}^{p}\left[(y_k - d_k)w_i \exp\left(-\frac{1}{2}u_{ik}\right)\frac{(x_{jk} - c_{ij})}{\sigma_{ij}^2}\right], \tag{5.35}$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \sum_{k=1}^{p}\left[(y_k - d_k)w_i \exp\left(-\frac{1}{2}u_{ik}\right)\frac{(x_{jk} - c_{ij})^2}{\sigma_{ij}^3}\right]. \tag{5.36}$$

Применение градиентного метода наискорейшего спуска позволяет провести изменение центров и ширины радиальных функций согласно формулам:

$$c_{ij}(n+1) = c_{ij}(n) - \eta \frac{\partial E}{\partial c_{ij}}, \tag{5.37}$$

$$\sigma_{ij}(n+1) = \sigma_{ij}(n) - \eta \frac{\partial E}{\partial \sigma_{ij}}. \tag{5.38}$$

Уточнение нелинейных параметров радиальной функции завершает очередной цикл обучения. Многократное повторение обоих этапов ведет к полному и быстрому обучению сети, особенно когда начальные значения параметров радиальных функций близки к оптимальным.

На практике выделенные этапы в разной степени влияют на адаптацию параметров. Как правило, быстрее функционирует алгоритм SVD (он за один шаг находит локальный минимум функции). Для выравнивания этой диспропорции одно уточнение линейных параметров сопровождается обычно несколькими циклами адаптации нелинейных параметров.

5.3.4. Алгоритмы обучения, основанные на обратном распространении ошибки

Обособленный класс алгоритмов обучения радиальных функций составляют градиентные методы обучения с учителем, в которых используется алгоритм обратного распространения ошибки. Так же как и в сигмоидальных сетях, их основу составляет целевая функция, определенная для всех p пар обучающих выборок (x_j, d_j), в виде

$$E = \frac{1}{2} \sum_{j=1}^{p} \left[\sum_{i=1}^{K} w_i \varphi_i(x_j) - d_j \right]^2. \tag{5.39}$$

Для упрощения записи в дальнейшем будем учитывать только одну обучающую выборку (x, d), вследствие чего целевая функция принимает вид:

$$E = \frac{1}{2} \left[\sum_{i=1}^{K} w_i \varphi_i(x) - d \right]^2. \tag{5.40}$$

Такое упрощение ничем не ограничивает общность рассуждений, поскольку оно может означать обучение типа "онлайн", при котором на вход сети каждый раз подается только один обучающий вектор. Предположим, что применяется самая общая форма гауссовской радиальной функции $\varphi_i(x)$, соответствующей сети HRBF, в которой

$$\varphi_i(x) = \exp\left[-\frac{1}{2} [\mathbf{Q}_i(x - c_i)]^T [\mathbf{Q}_i(x - c_i)] \right], \tag{5.41}$$

а матрица $\mathbf{Q}_i$ имеет произвольную структуру. Независимо от выбираемого метода градиентной оптимизации необходимо прежде всего получить

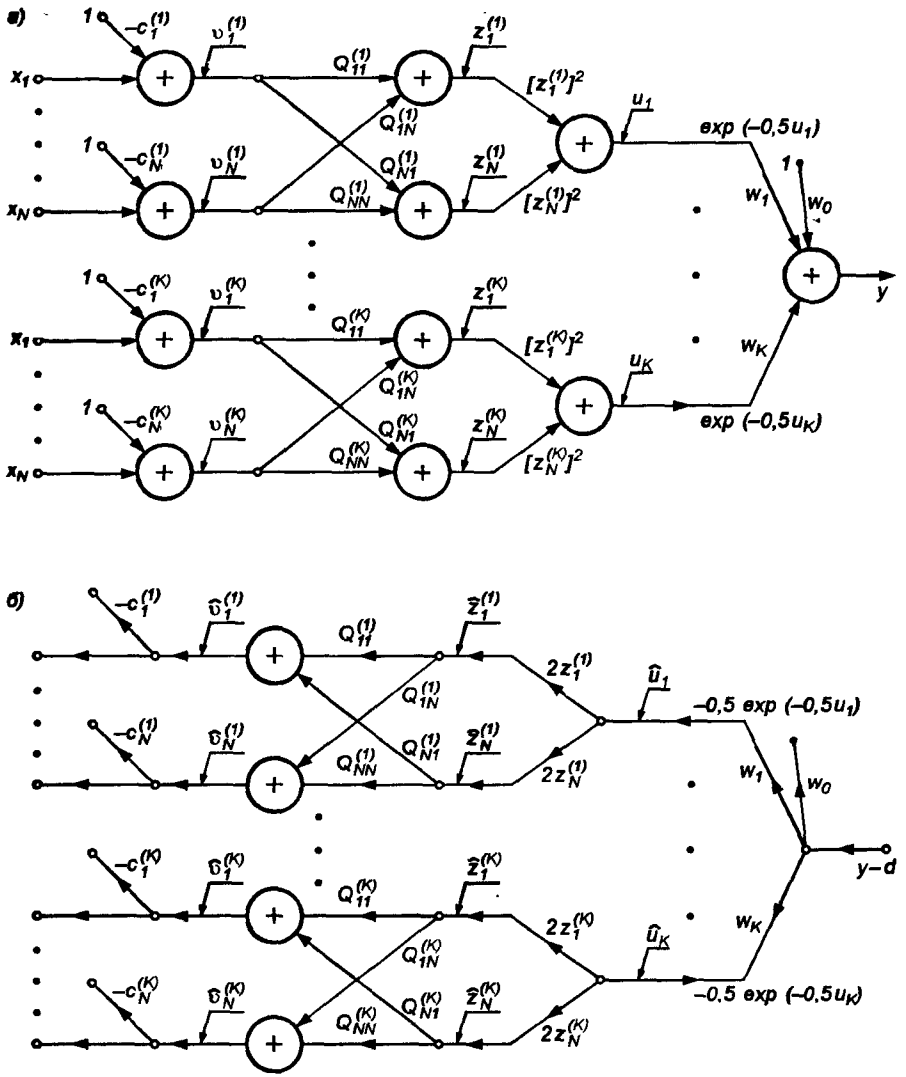

Рис. 5.8. Графы сети HRBF, используемые для генерации градиента:
а) исходная сеть; *б*) сопряженная сеть

вектор градиента целевой функции относительно всех параметров сети. Для расчета градиента будем использовать представленный в разделе 3 метод сопряженных графов, позволяющий определить любой компонент градиента на основе анализа исходного и сопряженного с ним графа сети. Граф сети HRBF с обозначенными на нем сигналами представлен на рис. 5.8.

В этой сети реализуются две нелинейные функции: квадратичная $f(z)=z^2$ и показательная $f(u)=\exp(-0{,}5u)$. В сопряженном графе, соответствующем исходному графу, обе эти функции линеаризуются

относительно значений $\frac{\partial f}{\partial z}$ и $\frac{\partial f}{\partial u}$, определенных в точках решения исходной системы, так, как это представлено на рис. 5.8 *б*. Направления всех дуг в *сопряженном графе противоположны их направлениям в исходном графе.* В качестве источника возбуждения в сопряженном графе выступает разностный сигнал $(y - d)$, представляющий величину фактического рассогласования. Конкретные составляющие градиента определяются непосредственно по информации об этих двух графах с использованием процедуры, описанной в разделе 3. Они принимают следующую форму:

$$\frac{\partial E}{\partial w_0} = y - d \ ; \tag{5.42}$$

$$\frac{\partial E}{\partial w_i} = \exp(-0{,}5u_i)(y-d); \tag{5.43}$$

$$\frac{\partial E}{\partial c_j^{(i)}} = \hat{v}_j^{(i)} = -\exp(-0{,}5u_i) w_i (y-d) \sum_{k=1}^{N} Q_{kj}^{(i)} z_k^{(i)} \ ; \tag{5.44}$$

$$\frac{\partial E}{\partial Q_{kj}^{(i)}} = v_j^{(i)} \hat{z}_j^{(i)} = -\exp(-0{,}5u_i) w_i (y-d)(x_k - c_k^{(i)}) \ z_j^{(i)}, \tag{5.45}$$

где

$$z_j^{(i)} = \sum_{k=1}^{N} Q_{kj}^{(i)} (x_k - c_k^{(i)}) \ , \tag{5.46}$$

$$u_i = \sum_{k=1}^{N} [z_k^{(i)}]^2. \tag{5.47}$$

Конкретизация компонентов градиента позволяет задействовать для подбора параметров любые градиентные методы оптимизации независимо от объекта обучения – будь то вес w_i либо центр $c_k^{(i)}$, либо коэффициент масштабирования $Q_{jk}^{(i)}$. Для обучения могут использоваться любые градиентные методы, представленные в разделе 3, а также любые способы подбора коэффициента обучения.

Главной проблемой, подлежащей разрешению, остается выбор начальных значений параметров. Если процесс обучения начинается со случайных значений, то вероятность попадания в точки локальных минимумов, далеких от искомого решения, оказывается более высокой, чем для сигмоидальных сетей, из-за нелинейности показательных функций. По этой причине случайный выбор начальных параметров радиальных функций применяется редко. Он заменяется специальной процедурой инициализации, основанной на анализе информации, содержащейся во множестве обучающих данных. Этой цели служат представленные в настоящем разделе алгоритмы самоорганизации, действие которых ограничивается несколькими циклами. Получаемые в результате значения параметров радиальных функций принимаются в качестве начальных

Стартовые величины весов w_i подбираются, как правило, случайным образом, так же как и в типовом алгоритме обучения сигмоидальных сетей.

5.4. Пример использования радиальной сети

Нейронные сети с радиальными базисными функциями находят применение как при решении задач классификации либо аппроксимации функций многих переменных, так и при прогнозировании, т.е. в тех прикладных сферах, в которых сигмоидальные сети имеют завоеванные позиции уже в течение многих лет. Они выполняют те же функции, что и сигмоидальные сети, однако реализуют иные методы обработки данных, связанные с локальными отображениями. Благодаря этой особенности обеспечивается значительное упрощение и, следовательно, ускорение процесса обучения.

В качестве примера рассмотрим аппроксимацию трехмерной функции, которая описывается формулой

$$d = f(x_1, x_2) = 3(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) +$$
$$-10\left(\frac{x1}{5} - x_1^3 - x_2^5\right)\exp(-x_1^2 - x_2^2) - \frac{1}{3}\exp(-(x_1+1)^2 - x_2^2) \ .$$

Пусть переменные изменяются в пределах $-3 \le x_1 \le 3$ и $-3 \le x_2 \le 3$. Обучающие данные имеют равномерное распределение в областях определения переменных x_1 и x_2. В общей сложности для обучения использовалось 625 обучающих выборок в виде пар данных ($[x_1, x_2]$, d). Для решения задачи была построена сеть со структурой 2–36–1 (2 входа для x_1 и x_2 соответственно,

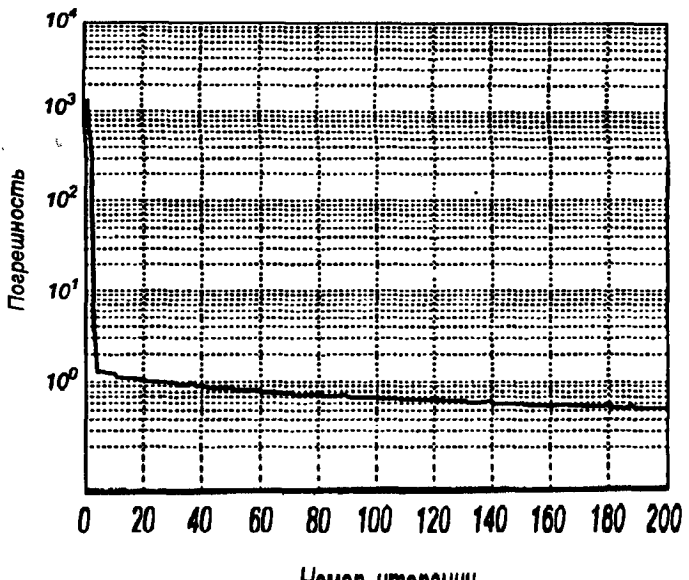

Рис. 5.9. График обучения радиальной сети RBF для примера восстановления трехмерной функции

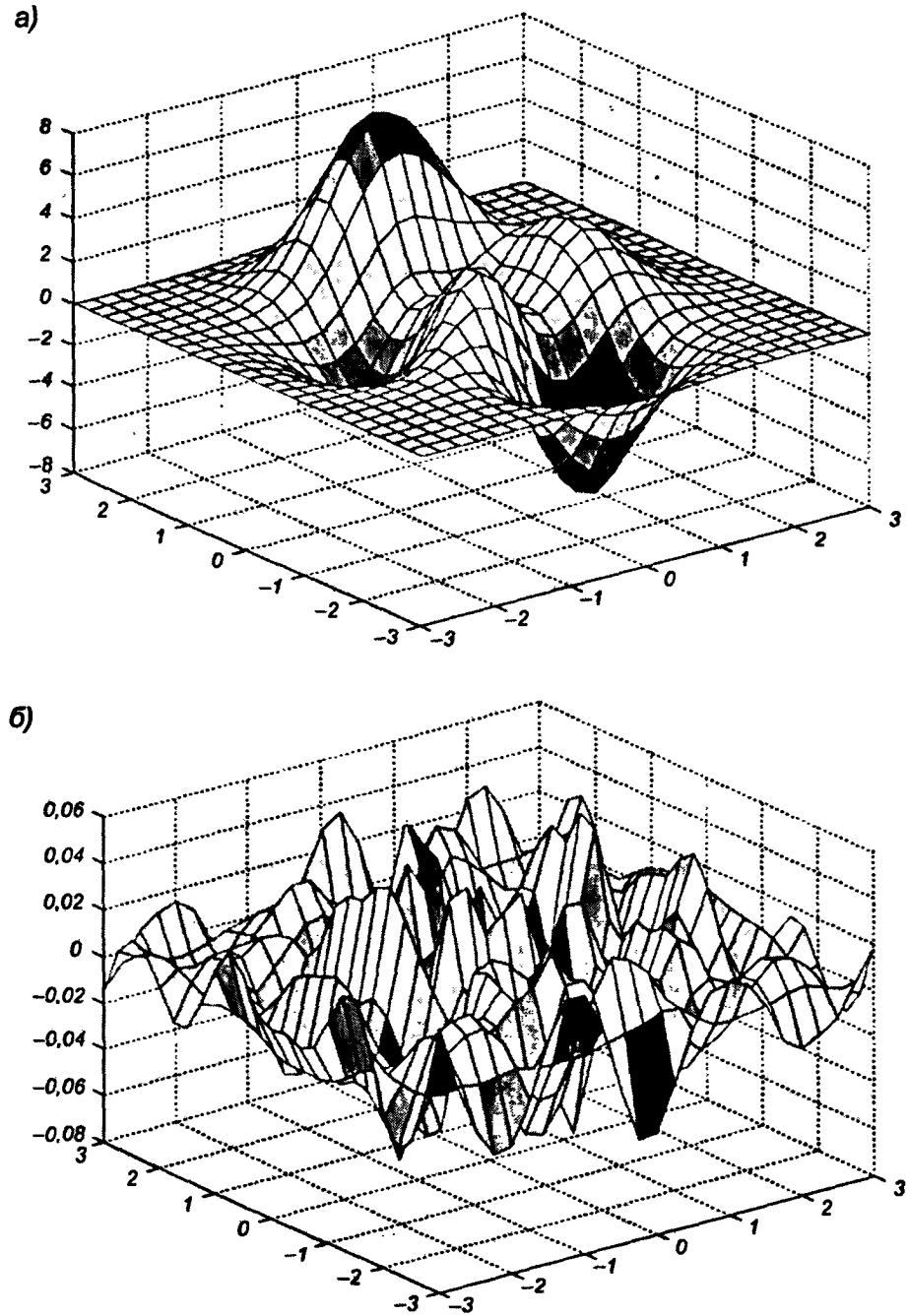

Рис. 5.10. Результаты восстановления трехмерной функции радиальной сетью RBF: *а*) восстановленная поверхность; *б*) погрешность восстановления

36 радиальных нейронов гауссовского типа и один выходной линейный нейрон, соответствующий значению d функции). Применялся гибридный алгоритм обучения со случайным выбором начальных значений параметров сети. На рис. 5.9 представлен график обучения сети (кривая изменения погрешности с увеличением количества итераций). Из графика видно, что прогресс в уменьшении погрешности достаточно велик, особенно в начальной стадии процесса.

На рис. 5.10 приведены графические представления восстановленной функции $f(x_1, x_2)$ и погрешности восстановления данных (характеристика обученности сети). Максимальная погрешность восстановления не превысила уровня 0,06, что составляет около 1 % от ожидаемого значения. Сравнение скорости обучения и обобщающих способностей радиальной сети с аналогичными показателями многослойного персептрона однозначно свидетельствует в пользу первой. Она быстрее обучается и гораздо менее чувствительна к начальным значениям параметров как базисных функций, так и весов выходного нейрона.

5.5. Методы подбора количества базисных функций

Подбор количества базисных функций, каждой из которых соответствует один скрытый нейрон, считается основной проблемой, возникающей при корректном решении задачи аппроксимации. Как и при использовании сигмоидальных сетей, слишком малое количество нейронов не позволяет уменьшить в достаточной степени погрешность обобщения множества обучающих данных, тогда как слишком большое их число увеличивает погрешность выводимого решения на множестве тестирующих данных. Подбор необходимого и достаточного количества нейронов зависит от многих факторов, в числе которых размерность задачи, объем обучающих данных и прежде всего – пространственная структура аппроксимируемой функции. Как правило, количество базисных функций K составляет определенную долю от объема обучающих данных p, причем фактическая величина этой доли зависит от размерности вектора $\boldsymbol{x}$ и от разброса ожидаемых значений d_i, соответствующих входным векторам $\boldsymbol{x}_i$, для $i = 1, 2, ..., p$.

5.5.1. Эвристические методы

Вследствие невозможности априорного определения точного количества скрытых нейронов применяются адаптивные методы, которые позволяют добавлять или удалять их в процессе обучения. Создано много эвристических методов, реализующих такие операции [10, 154]. Как правило, обучение сети начинается при каком-либо изначально принятом количестве нейронов, а впоследствии контролируется как степень уменьшения среднеквадратичной погрешности, так и

изменение значений подбираемых параметров сети. Если среднее изменение значений весов после определенного числа обучающих циклов слишком мало $\sum_i \langle \Delta w_i \rangle < \xi$, добавляются две базисные функций (2 нейрона) с центрами, соответствующими наибольшей и наименьшей погрешности адаптации, после чего обучение расширенной таким образом структуры продолжается. Одновременно контролируются абсолютные значения весов w_i всех отдельно взятых нейронов. Если они меньше установленного вначале порога δ, соответствующие им нейроны подлежат удалению из сети. Как добавление нейронов, так и их удаление начинается после выполнения определенного количества обучающих циклов и может происходить в течение всего процесса обучения вплоть до достижения требуемой точности отображения.

Другой подход к управлению количеством скрытых нейронов предложил Дж. Платт в работе [130]. Это метод, объединяющий элементы самоорганизации и обучения с учителем. После предъявления каждой очередной обучающей выборки определяется эвклидово расстояние между ней и центром ближайшей существующей радиальной функции. Если это расстояние превышает установленный порог $\delta(k)$, то создается центр новой радиальной функции (т.е. добавляется нейрон), после чего сеть подвергается стандартной процедуре обучения с использованием градиентных методов (обучение с учителем). Процесс добавления нейронов продолжается вплоть до достижения требуемого уровня погрешности отображения. Принципиально важным для этого метода считается подбор значения $\delta(k)$, в соответствии с которым принимается решение о расширении сети. Обычно $\delta(k)$ экспоненциально изменяется с течением времени (в зависимости от количества итераций) от значения $\delta_{\max}$ в начале процесса до $\delta_{\min}$ в конце его. Недостаток этого подхода состоит в невозможности уменьшения количества нейронов в процессе обработки информации даже тогда, когда в результате обучения какие-то из них дегенерируют (вследствие неудачного размещения центров) либо когда несколько нейронов начинают дублировать друг друга, выполняя одну и ту же функцию. Кроме того, этот метод очень чувствителен к подбору параметров процесса обучения, особенно значений $\delta_{\max}$ и $\delta_{\min}$.

5.5.2. Метод ортогонализации Грэма–Шмидта

Наиболее эффективным методом управления количеством скрытых нейронов остается применение специальной технологии обучения сети, основанной на методе ортогонализации наименьших квадратов, использующем классический алгоритм ортогонализации Грэма–Шмидта [8]. Отправная точка этого метода – представление задачи обучения в виде линейной адаптации вектора весов сети $\boldsymbol{w} = [w_0, w_1, ..., w_K]^T$, направленной на минимизацию значения вектора погрешности $\boldsymbol{e}$. Для p обучающих выборок вектор ожидаемых значений имеет вид: $\boldsymbol{d} = [d_0, d_1, ..., d_p]^T$. При использовании K базисных функций и p обучающих пар реакции скрытых нейронов образуют матрицу $\mathbf{G}$ вида

$$\mathbf{G}=\begin{bmatrix}\varphi_{11} & \varphi_{21} & \cdots & \varphi_{K1}\\ \varphi_{12} & \varphi_{22} & \cdots & \varphi_{K2}\\ \cdots & \cdots & \cdots & \cdots\\ \varphi_{1p} & \varphi_{2p} & \cdots & \varphi_{Kp}\end{bmatrix}, \tag{5.48}$$

в которой φ_{ji} обозначает реакцию i-й радиальной функции на j-ю обучающую выборку, $\varphi_{ji}=\varphi(\|x_j-c_i\|)$. Если вектор реакций i-й радиальной функции на все обучающие выборки обозначить $g_i=[\varphi_{i1}, \varphi_{i2},..., \varphi_{ip}]^T$, то матрицу $\mathbf{G}$ можно представить в форме

$$\mathbf{G}=[g_1, g_2,..., g_K]. \tag{5.49}$$

При таких обозначениях на каждом этапе обучения будет выполняться линейное равенство

$$d=\mathbf{G}w+e, \tag{5.50}$$

где w – вектор весов, а $e=[e_1, e_2, ..., e_p]^T$ – вектор фактической погрешности обучения. Квадрат произведения $\mathbf{G}w$ соответствует ожидаемой энергии, исходящей от сигналов, задаваемых вектором d, которая и подвергается максимизации в процессе обучения.

Метод ортогонализации наименьших квадратов основан на преобразовании векторов g_i во множество базисных ортогональных векторов, позволяющее оценить индивидуальный вклад каждого из них в общую энергию, представляемую произведением $\mathbf{G}w$. Это в свою очередь позволяет удалить те векторы, влияние которых на процесс оказывается минимальным.

В процессе обучения матрица $\mathbf{G}\in R^{p\times K}$ раскладывается на произведение матрицы $\mathbf{Q}\in R^{p\times K}$ с ортогональными столбцами q_i на верхнетреугольную матрицу $\mathbf{A}\in R^{K\times K}$ с единичными диагональными значениями:

$$\mathbf{G}=\mathbf{Q}\,\mathbf{A}, \tag{5.51}$$

где

$$\mathbf{A}=\begin{bmatrix}1 & a_{12} & a_{21} & \cdots & a_{1K}\\ 0 & 1 & a_{22} & \cdots & a_{2K}\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & 1\end{bmatrix},$$

а матрица $\mathbf{Q}$ соответствует условию

$$\mathbf{Q}^T\mathbf{Q}=\mathbf{H}.$$

При этом $\mathbf{H}$ – диагональная матрица с элементами $H_{ii}=q_i^Tq_i=\sum_{j=1}^{p}q_{ij}^2$. Решение зависимости (5.50) методом наименьших квадратов может быть спроецировано в пространство, образуемое ортогональными векторами q_i. Если ввести новую векторную переменную b, определенную как

$$b = \mathbf{A}\, w\,, \tag{5.52}$$

то из уравнения (5.50) получим:

$$d = \mathbf{Q}\, b + e\,. \tag{5.53}$$

Приближенное решение уравнения (5.53) (обозначаемое символом ^) методом наименьших квадратов имеет вид:

$$\hat{b} = [\,\mathbf{Q}^T\mathbf{Q}\,]^{-1}\,\mathbf{Q}^T d = \mathbf{H}^{-1}\mathbf{Q}^T d\,. \tag{5.54}$$

Принимая во внимание диагональный характер матрицы **H**, можно получить формулу, описывающую i-й компонент вектора $\hat{b}$:

$$\hat{b}_i = \frac{q_i^T d}{q_i^T q_i}\,. \tag{5.55}$$

Решение, определяющее вектор весов w, находится непосредственно из зависимости (5.52), которую можно переписать в форме

$$\hat{w} = \mathbf{A}^{-1}\hat{b}\,. \tag{5.56}$$

С учетом треугольной структуры матрицы **A** вычислительная сложность решения уравнения (5.56) относительно вектора w невелика.

Ортогонализация матрицы **Q**, описанная выражением (5.51), может быть проведена различными методами, наиболее эффективным из которых считается алгоритм Грэма–Шмидта. В соответствии с этим методом матрица **A** формируется последовательно, столбец за столбцом с одновременным формированием очередных столбцов ортогональной матрицы **Q**. На k-м шаге создается столбец q_k, ортогональный ко всем созданным ранее (k-1) столбцам q_i (i = 1, 2, ..., k-1). Процедура повторяется для значений k = 2, 3, ..., K. Математическая модель этой операции имеет вид:

$$q_1 = g_1\,, \tag{5.57}$$

$$a_{ik} = \frac{q_i^T g_k}{q_i^T q_i}\,, \tag{5.58}$$

$$q_k = g_k - \sum_{i=1}^{k-1} a_{ik} q_i\,, \tag{5.59}$$

для $1 \le i \le k$, k = 2, 3, ..., K. Многократно повторенная процедура ортогонализации позволяет сформировать все ортогональные векторы q_k и матрицу **A**, на основе которых можно получить методом наименьших квадратов приближенное решение $\hat{b}$ (уравнение (5.54)), а в дальнейшем из решения треугольной системы уравнений (5.56) найти вектор $\hat{w}$.

Однако важнейшим достоинством описываемого метода ортогонализации считается возможность селекции векторов q_i с учетом их важности для отображения обучающих данных. В случае априори определенного количества

K радиальных функций задача заключается в такой расстановке векторов q_i, чтобы отобрать из них первые K_r наиболее значимые в энергетическом плане, при этом, как правило, $K_r << K$. Использование в дальнейших вычислениях только K_r радиальных функций означает сокращение количества скрытых нейронов с начального их числа K до K_r. Принимая во внимание энергию сигналов, описываемых вектором d, в соответствии с выражением (5.53) получаем

$$d^T d = \sum_{i=1}^{K} b_i q_i^T q_i + \mathbf{E}^T e \ . \tag{5.60}$$

Если принять, что вектор ожидаемых реакций d имеет нулевое среднее значение, то произведение $b_i^2 q_i^T q_i$ может интерпретироваться как средний вклад, приходящийся на одну обучающую выборку вектора q_i, соответствующего i-й базисной функции. Относительная доля этого составляющего в общем энергетическом балансе может быть определена по формуле

$$\varepsilon_i = \frac{b_i^2 q_i^T q_i}{d^T d} \tag{5.61}$$

для $i = 1, 2, ..., K$. Расчет значений ε_i для всех базисных функций дает возможность оценить их важность для функционального отображения обучающих данных, что упрощает принятие решения о ликвидации тех, чей вклад оказывается наименьшим. После отбора наиболее значимой радиальной функции процесс ортогонализации повторяется для получения нового решения и выбора следующей по значимости радиальной функции. При фиксации начальной величины $K=p$ после многократного повторения ортогонализации Грэма–Шмидта можно отобрать K_r наиболее значащих базисных функций и исключить остальные. Таким образом количество скрытых нейронов уменьшается от начального числа K до K_r. Алгоритм отбора наиболее значимых базисных функций выглядит следующим образом [8]:

1. На первом этапе ($k = 1$) для $1 < i < K$ рассчитать

$$q_i(1) = g_i \ ,$$

$$b_i(1) = \frac{[q_i(1)]^T d}{[q_i(1)]^T q_i(1)} \ ,$$

$$\varepsilon_i(1) = \frac{[b_i(1)]^2 [q_i(1)]^T [q_i(1)]}{d^T d} \ .$$

Предполагается, что $\varepsilon_{i_1}(1) = \max\{\varepsilon_i(1)\}$ для $1 \le i \le K$, а вектор $q_1 = q_{i_1} = g_{i_1}$.

2. На следующих этапах ($k \ge 2$) для $1 \le i \le K$, $i \ne i_1 \ne ... \ne i_{k-1}$ следует провести очередные циклы ортогонализации:

$$a_{jk}^{(i)} = \frac{q_j^T g_i}{q_j^T q_j} \ ,$$

$$q_i(k) = g_i - \sum_{j=1}^{k-1} a_{jk}^{(i)} q_j \ ,$$

а также оценить влияние очередных радиальных функций на суммарное значение энергетической функции путем расчета:

$$b_i(k) = \frac{[q_i(k)]^T d}{[q_i(k)]^T [q_i(k)]} \ ,$$

$$\varepsilon_i(k) = \frac{[b_i(k)]^2 [q_i(k)]^T [q_i(k)]}{d^T d} \ .$$

Если наибольший вклад радиальной функции в общую энергию обозначить $\varepsilon_{i_k}(k)$, т.е.

$$\varepsilon_{i_k}(k) = \max\{\varepsilon_i(k)\}$$

для $1 \le i \le K$, $i \ne i_1 \ne ... \ne i_{k-1}$, тогда очередной выделенный вектор q_k будет соответствовать радиальной функции со следующим по важности вкладом в общую энергию. Этот вектор определяется выражением

$$q_k = q_{i_k}(k) = g_{i_k} - \sum_{j=1}^{k-1} a_{jk} q_j \ ,$$

в котором коэффициент $a_{jk} = a_{jk}^{(i_k)}$ для $1 \le j \le$ k-1.

3. Процедура выявления наиболее значимых для отображения радиальных функций завершается на этапе $k = K_r$, в момент выполнения условия

$$1 - \sum_{j=1}^{K_r} \varepsilon_j < \rho \ ,$$

где $0 < \rho < 1$ – это заранее установленный порог толерантности.

В результате выполнения процедуры в сети остается только K_r наиболее значимых радиальных функций, расположенных в ранее определенных центрах (например, путем самоорганизации). Одновременно при реализации алгоритма вычисляются конкретные составляющие вектора b, на основе которых по формуле (5.52) находятся значения весов w выходного слоя сети.

Геометрическая интерпретация представленной процедуры ортогонализации достаточно проста. На k-м этапе выполнения алгоритма размерность базисного пространства увеличивается на единицу, с $(k–1)$ до k, за счет введения дополнительной базисной функции. Вводя всякий раз наиболее значимую базисную функцию, мы получаем оптимальный их набор, что позволяет получить наилучшие результаты обучения.

Толерантность ρ, определяющая момент завершения процесса обучения, – это важный фактор, от которого зависит, с одной стороны, точность отображения обучающих данных, а с другой стороны, – уровень сложности нейронной сети. Во многих случаях ее значение можно оценить на основе статистического анализа

обучающих данных и фактических успехов в обучении. С методами подбора оптимальных значений ρ можно ознакомиться в [8].

Еще одно достоинство процесса ортогонализации – возможность избежать неудачной комбинации параметров процесса обучения. Выполнение условия $q_k^T q_k = 0$ означает, что соответствующий вектор g_k является линейной комбинацией векторов g_1, g_2, ..., g_{k-1}. Поэтому если в процессе ортогонализации произведение $q_k^T q_k$ меньше, чем заданное (пороговое) значение, то функцию g_k можно не включать во множество базисных функций.

5.6. Сравнение радиальных и сигмоидальных сетей

Радиальные нейронные сети относятся к той же категории сетей, обучаемых с учителем, что и многослойный персептрон. По сравнению с многослойными сетями, имеющими сигмоидальные функции активации, они отличаются некоторыми специфическими свойствами, обеспечивающими более простое отображение характеристик моделируемого процесса.

Сигмоидальная сеть, в которой ненулевое значение сигмоидальной функции распространяется от некоторой точки в пространстве до бесконечности, решает задачу глобальной аппроксимации заданной функции. В то же время радиальная сеть, основанная на функциях, имеющих ненулевые значения только в определенной области вокруг их центров, реализует аппроксимацию локального типа, сфера которой, как правило, более ограничена. Поэтому необходимо понимать, что обобщающие способности радиальных сетей несколько хуже, чем у сигмоидальных сетей, особенно на границах области обучающих данных. Вследствие глобального характера сигмоидальной функции многослойные сети не обладают встроенным механизмом идентификации области данных, на который сильнее всего реагирует конкретный нейрон. Из-за физической невозможности связать зону активности нейрона с соответствующей областью обучающих данных для сигмоидальных сетей сложно определить исходную позицию процесса обучения. Принимая во внимание полимодальность целевой функции, достижение глобального минимума в такой ситуации становится чрезвычайно трудным даже при самых совершенных методах обучения.

Радиальные сети решают эту проблему гораздо лучше. Наиболее часто применяемые на практике радиальные функции гауссовского типа по своей природе имеют локальный характер и принимают ненулевые значения только в зоне вокруг определенного центра. Это позволяет легко установить зависимость между параметрами базисных функций и физическим размещением обучающих данных в многомерном пространстве. Поэтому удается относительно просто найти удовлетворительные начальные условия процесса обучения с учителем. Применение подобных алгоритмов обучения при начальных

условиях, близких к оптимальным, многократно увеличивает вероятность достижения успеха с помощью радиальных сетей.

Считается [85, 154], что радиальные сети лучше, чем сигмоидальные, решают такие классификационные задачи, как обнаружение повреждений в различных системах, распознавание образов и т.п. Применение радиальных сетей для прогнозирования таких сложных временных процессов, как ежемесячные колебания занятости трудоспособного населения в масштабах страны [8], экономические тренды и т.д., также дает неплохие результаты, сравнимые или даже лучшие, чем получаемые с использованием сигмоидальных сетей.

Важное достоинство радиальных сетей – значительно упрощенный алгоритм обучения. При наличии только одного скрытого слоя и тесной связи активности нейрона с соответствующей областью пространства обучающих данных точка начала обучения оказывается гораздо ближе к оптимальному решению, чем это имеет место в многослойных сетях. Кроме того, можно отделить этап подбора параметров базисных функций от подбора значений весов сети (гибридный алгоритм), что сильно упрощает и ускоряет процесс обучения. Выигрыш во времени становится еще большим, если принять во внимание процедуру формирования оптимальной (с точки зрения способности к обобщению) структуры сети. При использовании многослойных сетей это очень трудоемкая задача, требующая, как правило, многократного повторения обучения или дообучения. Для радиальных сетей, особенно основанных на ортогонализации, формирование оптимальной структуры сети оказывается естественным этапом процесса обучения, не требующим никаких дополнительных усилий.

Раздел 6

СПЕЦИАЛИЗИРОВАННЫЕ СТРУКТУРЫ НЕЙРОННЫХ СЕТЕЙ

6.1. Сеть каскадной корреляции Фальмана

Сеть каскадной корреляции Фальмана – это специализированная многослойная нейронная конструкция, в которой подбор структуры сети происходит параллельно с ее обучением путем добавления на каждом этапе обучения одного скрытого нейрона. Таким образом, определение структуры сети и реализацию алгоритма ее обучения можно трактовать как выполнение процедуры подбора оптимальной архитектуры искусственной нейронной сети.

Архитектура сети каскадной корреляции представляет собой объединение нейронов взвешенными связями в виде развивающегося каскада (рис. 6.1). Каждый очередной добавляемый нейрон подключается к входным узлам и ко всем уже существующим скрытым нейронам. Выходы всех скрытых нейронов и входные узлы сети напрямую подключаются также и к выходным нейронам.

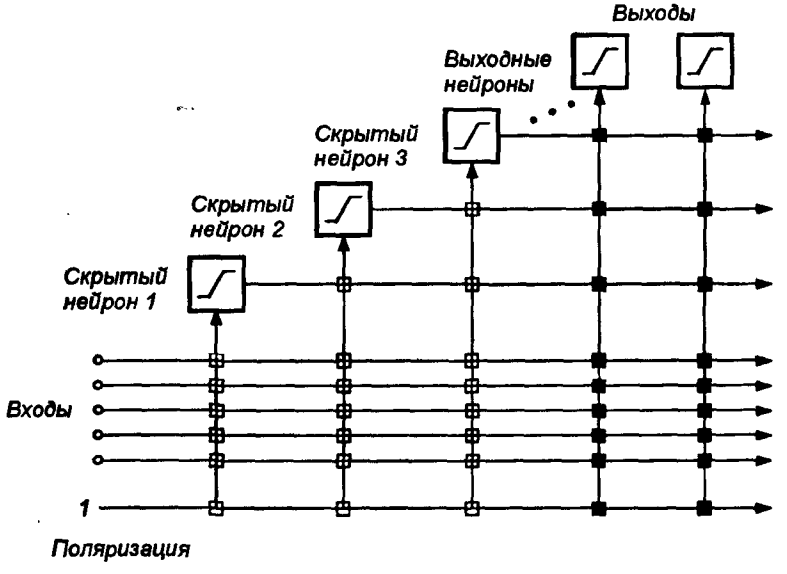

Рис. 6.1. Структура сети каскадной корреляции Фальмана

Каждый нейрон старается так адаптировать веса своих связей, чтобы быть востребованным для выполняемого сетью отображения данных. На начальном этапе формируется сеть, состоящая только из входных узлов и выходных нейронов. Количество входов и выходов зависит исключительно от специфики решаемой задачи и не подлежит модификации. Каждый вход соединен со всеми выходами посредством связей, веса которых уточняются в процессе обучения. Выходные нейроны могут быть как линейными, так и нелинейными, с практически произвольной функцией активации. Скрытые нейроны добавляются в сеть по одному. Каждый добавляемый нейрон подключается ко всем входам сети и ко всем ранее добавленным скрытым нейронам. В момент подключения нейрона к ранее созданной структуре фиксируются веса его входных связей, которые в дальнейшем не подлежат модификации, тогда как веса его связей с выходными нейронами постоянно уточняются. Каждый добавляемый скрытый нейрон образует отдельный одноэлементный слой.

Процесс обучения сети начинается до ввода в нее скрытых нейронов. Непосредственные соединения входов и выходов тренируются таким образом, чтобы минимизировать значение целевой функции. Для этого могут применяться любые методы обучения. В оригинальной работе Фальмана [34] использовался алгоритм *quickprop*, характеризующийся быстрой сходимостью к точке решения.

Если результат функционирования сети считается удовлетворительным с точки зрения ожидаемой или допустимой погрешности, процесс обучения и формирования структуры сети завершается. В противном случае следует расширить структуру сети добавлением одного скрытого нейрона. Для этого применяется специальная процедура, при выполнении которой вначале формируются и фиксируются входные веса нового нейрона, после чего он вводится в существующую сетевую структуру и его выход подключается ко всем выходным нейронам посредством связей с соответствующими весами. После подключения очередного скрытого нейрона происходит уточнение (с использованием выше описанной процедуры) весов выходных нейронов. Если полученный результат признается удовлетворительным, обучение завершается. В противном случае процедура включения в сеть очередного скрытого нейрона повторяется вплоть до достижения желаемых результатов обучения.

Формирование скрытого нейрона начинается с подключения его в качестве кандидата. Он представляет собой обособленный элемент, соединяемый со всеми входами сети и с выходами ранее введенных скрытых нейронов. Веса связей нейрона-кандидата подвергаются обучению, но его выходной сигнал на этом этапе никуда не подается. В таком состоянии вновь вводимый нейрон обучается с использованием множества обучающих выборок. Цель обучения состоит в таком подборе весов, при котором максимизируется корреляция между его активностью, определяемой выходным сигналом, и значением погрешности на выходе сети. Эта зависимость определяется коэффициентом корреляции S в виде

$$S = \sum_{j=1}^{M} \left| \sum_{k=1}^{p} (v^{(k)} - \bar{v})\,(e_j^{(k)} - \bar{e}_j) \right|, \qquad (6.1)$$

где p – количество обучающих выборок, M – количество выходных нейронов, $v^{(k)}$ - выходной сигнал нейрона-кандидата при k-й обучающей выборке, $e_j^{(k)}$ – значение погрешности j-го выходного нейрона для k-й обучающей выборки, $e_j^{(k)} = d_j^{(k)} - y_j^{(k)}$. $\bar{v}$ и $\bar{e}_j$ обозначены средние значения соответственно v и e, рассчитанные по всему множеству обучающих выборок.

Для максимизации значения функции S следует определить ее производную $\frac{\partial S}{\partial w_i}$ относительно весов w_i всех входных связей нейрона-кандидата. На основе правила разложения составной функции и с учетом физической интерпретации отдельных ее компонентов, по аналогии с методом обратного размножения получаем

$$\frac{dS}{dw_i} = \sum_j \sum_k \sigma_j (e_j^{(k)} - \bar{e}_j) f'^{(k)} I_i^{(k)}, \qquad (6.2)$$

где σ_i – обозначение корреляции между нейроном-кандидатом и j-м выходом сети, $f'^{(k)}$ – рассчитанная для k-й обучающей выборки производная функции активации нейрона-кандидата относительно взвешенной суммы его выходных сигналов, а $I_i^{(k)}$ – импульс, получаемый от i-го возбуждающего сигнала (входного сигнала сети либо выходного сигнала от ранее введенных скрытых нейронов). После определения вектора градиента функции S относительно всех подбираемых весов нейрона выполняется максимизация S (на практике обычно минимизируется значение функции S) на основе любого метода оптимизации. В оригинальной работе Фальмана это тот же самый алгоритм *quickprop*, который применяется для подбора весов выходных нейронов. После достижения максимального значения S нейрон-кандидат включается в существующую структуру нейронной сети, подобранные веса его входных связей фиксируются, и возобновляется процесс подбора значений весов выходных нейронов за счет минимизации целевой функции. Следует подчеркнуть, что, несмотря на большое количество слоев, в сети Фальмана не требуется использовать алгоритм обратного распространения ошибок, поскольку в процессе минимизации целевой функции задействованы только весовые коэффициенты выходного слоя, для которых погрешность рассчитывается непосредственно.

Для достижения наилучших результатов корреляционного обучения, как правило, одновременно обучается не один, а несколько нейронов-кандидатов. Принимая во внимание, что обучение начинается со случайных значений, каждый кандидат получает различные конечные значения весов, характеризующиеся различными значениями коэффициента корреляции. Среди таких обученных кандидатов выбирается имеющий наибольшее значение S, который и вводится в структуру сети. Параллельное корреляционное обучение нескольких нейронов-кандидатов уменьшает вероятность попадания в точку локального минимума и ввода в сеть нейрона с плохо подобранными весами, которые на последующих

этапах обучения уже невозможно будет откорректировать. Стандартное количество одновременно обучаемых нейронов-кандидатов, рекомендуемое авторами метода, составляет от 5 до 10.

Каждый нейрон-кандидат, претендующий на включение в сетевую структуру, может иметь свою функцию активации: сигмоидальную (биполярную или униполярную), гауссовскую, радиальную и т.п. Побеждает тот нейрон, который лучше приспосабливается к условиям, созданным множеством обучающих выборок. Вследствие такого подхода сеть каскадной корреляции может объединять нейроны с различными функциями активации, подобранными обучающим алгоритмом с учетом той роли, которую они играют в структуре сети.

Алгоритм каскадной корреляции был реализован авторами на языке *C* и известен под названием *Cascor* [34]. После некоторых изменений, упростивших его применение, этот алгоритм был откомпилирован в Институте теоретической электротехники и электроизмерений Варшавского политехнического университета с использованием компилятора *Watcom*. В ходе всесторонних исследований он продемонстрировал прекрасные качества как средство обучения и построения сети.

В качестве численного примера рассмотрим аппроксимацию функции двух переменных, заданную выражением

$$f(x,y) = 0{,}5\sin(\pi x^2)\sin(2\pi y)$$

для значений $-1 \le x \le 1$ и $-1 \le y \le 1$.

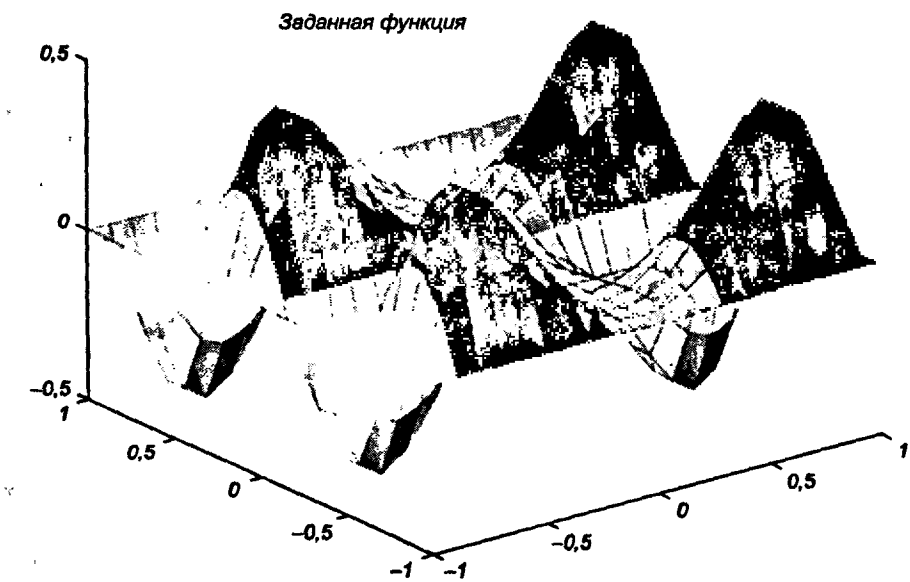

Рис. 6.2. Пример заданной трехмерной функции, преобразуемой сетью каскадной корреляции

На рис. 6.2 представлен график аппроксимируемой функции. В качестве обучающих данных использовалось 500 значений этой функции, равномерно распределенных по всему диапазону. В качестве тестирующих данных были сгенерированы 1000 значений функции в других точках того же диапазона. Сеть обучалась исходя из условия, что значение целевой функции должно быть меньше 0,01. Кривая обучения сети (график изменения погрешности обучения в зависимости от номера итерации) представлена на рис. 6.3. Ожидаемое

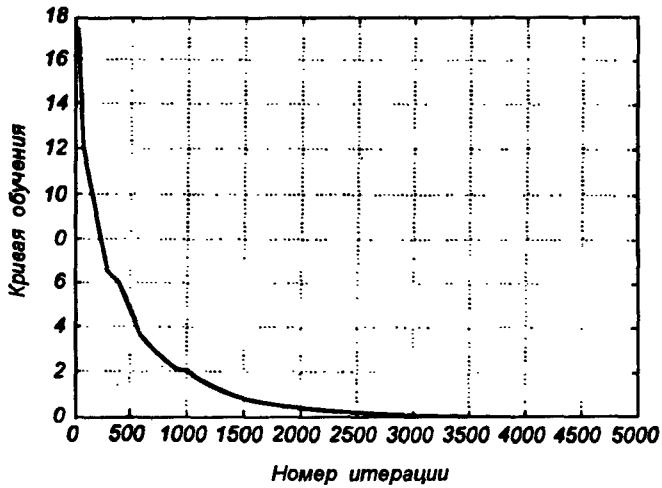

Рис. 6.3. *График кривой обучения сети каскадной корреляции на примере* преобразования трехмерной функции

значение погрешности обучения было получено на выходе сети при введении в ее структуру 41-го скрытого нейрона. Результаты тестирования подтвердили хорошие обобщающие способности сети. График функции, восстановленной по результатам тестирования, приведен на рис. 6.4*а*, а график погрешности восстановления структуры поверхности – на рис. 6.4*б*.

6.2. Сеть Вольтерри

Сеть Вольтерри – это динамическая сеть для нелинейной обработки последовательности сигналов, задержанных относительно друг друга. Возбуждением для сети в момент n служит вектор $x = [x_n, x_{n-1}, ..., x_{n-L}]^T$, где L – количество единичных задержек, а $(L+1)$ означает длину вектора. В соответствии с определением ряда Вольтерри выходной сигнал y генерируется по формуле [123, 137].

$$y(n) = \sum_{i_1=1}^{L} w_i x(n-i) + \sum_{i_1=1}^{L}\sum_{i_2=1}^{L} w_{i_1 i_2} x(n-i_1)x(n-i_2) + \\ + \sum_{i_1=1}^{L}\cdots\sum_{i_K=1}^{L} w_{i_1 i_2 \ldots i_K} x(n-i_1)x(n-i_2)\ldots x(n-i_K), \qquad (6.3)$$

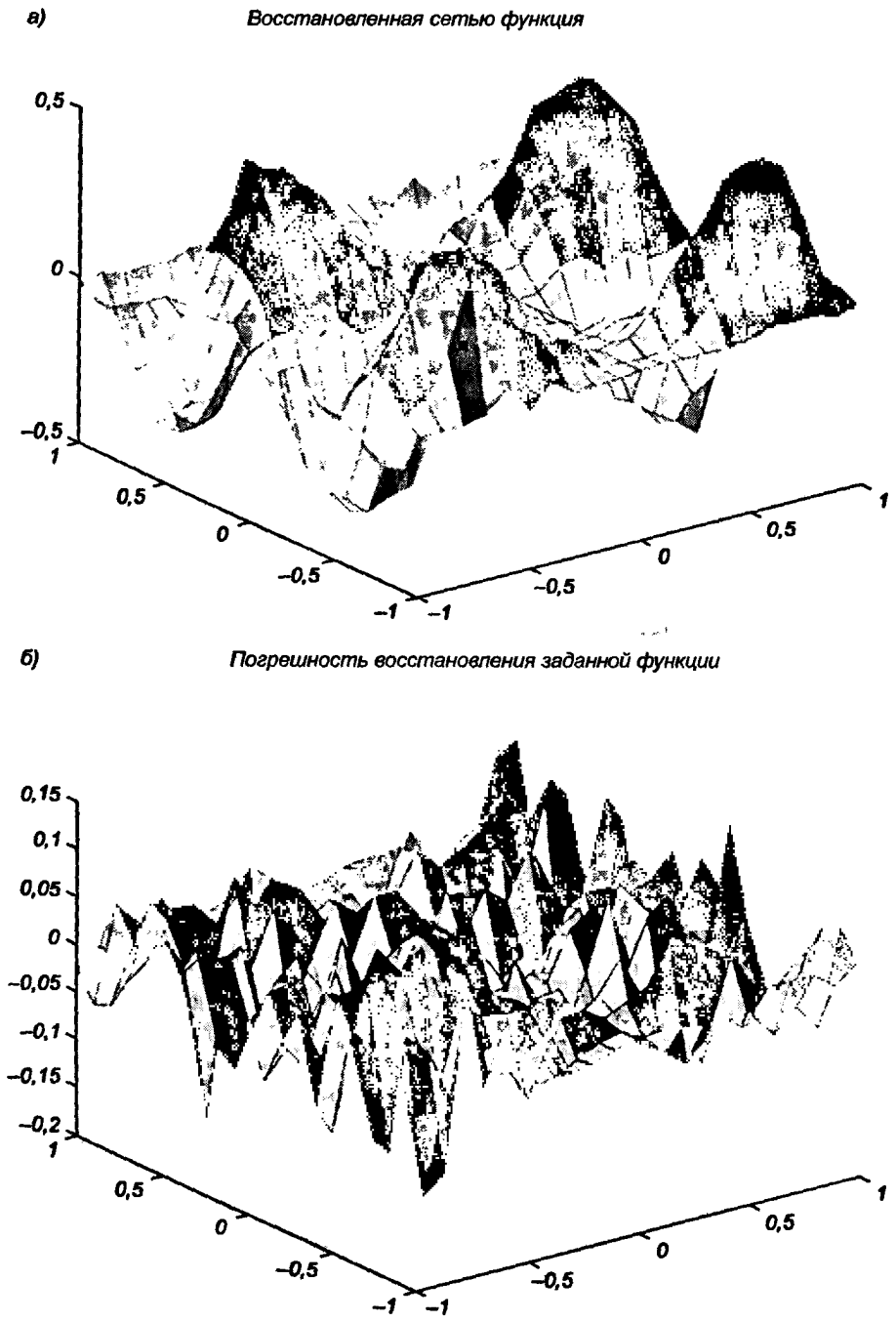

Рис. 6.4. Результаты восстановления трехмерной функции сетью каскадной корреляции Фальмана:
а) восстановленная поверхность; *б*) погрешность восстановления

где x обозначает входной сигнал, а веса w_i, w_{ij} , ..., w_{ijk} и т.д., называемые ядрами Вольтерри, соответствуют реакциям высших порядков. Нелинейная функциональная зависимость Вольтерри является естественным полиномиальным обобщением описания линейного фильтра FIR [137]. Порядок этого полинома K также называется степенью ряда Вольтерри. В случае адаптации реакции системы Вольтерри к заданной последовательности значений необходимо определить соответствующую целевую функцию, например $E = 0{,}5[y(n) - d(n)]^2$, и минимизировать ее значение с использованием универсальных способов оптимизации нейронных сетей, сводящихся к решению системы дифференциальных уравнений, описываемых выражением (2.16), которое в данном случае приобретает вид

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = -\mu\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{w}}. \qquad (6.4)$$

Вектор $\boldsymbol{w}$ обозначает вектор весов сети, E – целевую функцию, а $\mathrm{d}E/\mathrm{d}\boldsymbol{w}$ – градиент. Легко показать, что при ограничении в разложении $K = 3$ система дифференциальных уравнений может быть записана в виде

$$\frac{\mathrm{d}w_i(n)}{\mathrm{d}t} = -\mu[y(n) - d(n)]x(n\text{-}1), \qquad (6.5)$$

$$\frac{\mathrm{d}w_{ij}(n)}{\mathrm{d}t} = -\mu[y(n) - d(n)]x(n\text{-}i)x(n\text{-}j), \qquad (6.6)$$

$$\frac{\mathrm{d}w_{ijl}(n)}{\mathrm{d}t} = -\mu[y(n) - d(n)]x(n\text{-}i)x(n\text{-}j)x(n\text{-}l) \qquad (6.7)$$

для $i, j, l = 1, 2, ..., L$. Нейронная сеть, основанная на такой модели, минимизирует целевую функцию $E=0{,}5[y(n)-d(n)]^2$, которая, как следует из принятого определения, является квадратичной относительно весов w_i, w_{ij}, w_{ijl}, ··· . Если эта стратегия реализуется техническими средствами, следует считаться со значительной сложностью системы, вызванной огромным количеством весов сети. Уже при длине фильтра $L = 6$ и порядке $K = 3$ количество подбираемых весов составит 84. Это количество геометрически возрастает с увеличением длины L и порядка K.

Еще один недостаток непосредственного подхода к подбору весов системы Вольтерри – ухудшение обусловленности задачи с возрастанием порядка K и длины L. Следует отметить, что поскольку расчет значений ядер Вольтерри относится к процедурам линейного типа, обусловленность задачи является неудовлетворительной, так как содержащая многочисленные произведения выборок $x(n-i_k)$ матрица корреляции системы имеет, как правило, очень большое число обусловленности.

6.2.1. Структура и особенности обучения сети

Для упрощения структуры сети и уменьшения ее вычислительной сложности разложение Вольтерри (6.3) можно представить в следующей форме:

$$y_n = \sum_{i=0}^{L} x_{n-i}\left[w_i + \sum_{j=0}^{L} x_{n-j}\left[w_{ij} + \sum_{k=0}^{L} x_{n-k}(w_{ijk} + \ldots)\right]\right], \qquad (6.8)$$

где используются обозначения $y_n = y(n)$, $x_{n-i} = x(n - i)$ и т.д. Каждое слагаемое в квадратных скобках представляет собой линейный фильтр первого порядка, в котором соответствующие веса представляют импульсную реакцию другого линейного фильтра следующего уровня. Количество уровней, на которых создаются фильтры, равно порядку K. На рис. 6.5 показано распространение сигналов по сети, реализующей зависимость (6.8), при ограничении $K = 3$. Система представляет собой структуру типичной многослойной однонаправленной динамической нейронной сети. Это сеть с полиномиальной нелинейностью. Подбор весов производится после-

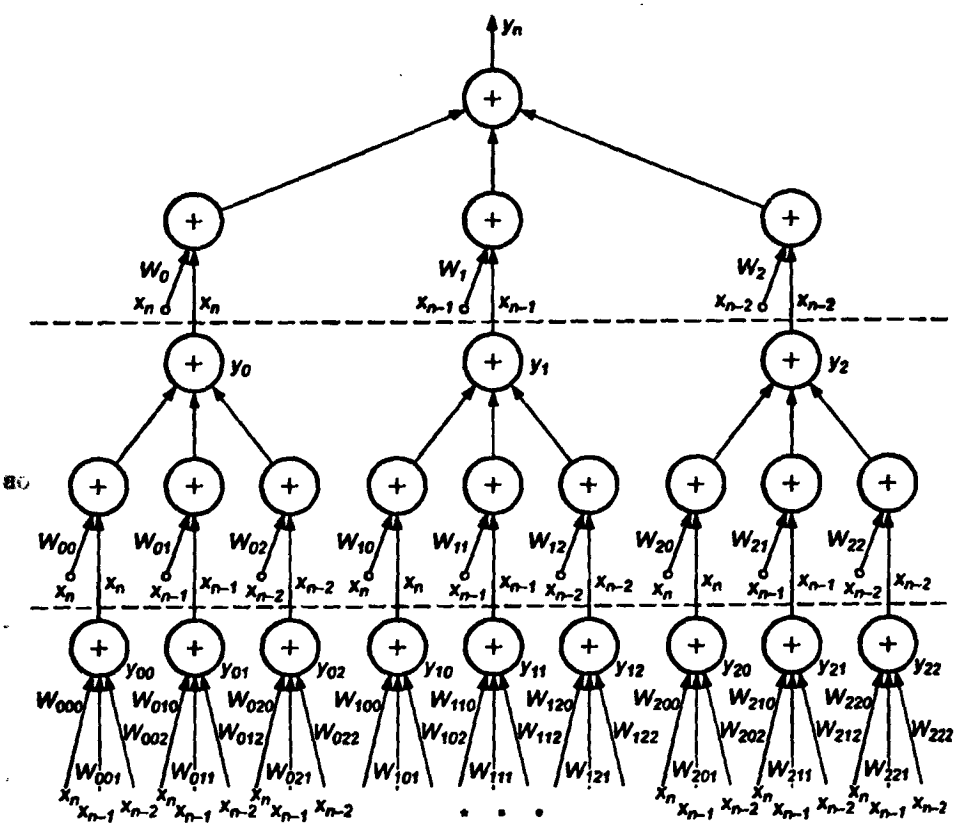

Рис. 6.5. Граф сети Вольтерри

довательно слой за слоем, причем эти процессы независимы друг от друга, и, следовательно, увеличение как количества весов в слое, так и количества самих слоев в сети в незначительной степени сказывается на обусловленности задачи. Это дает возможность существенно увеличить длину L и порядок K системы при ее практической реализации. Обучение нейронной сети, структура которой изображена на рис. 6.5, лучше всего проводить с использованием технологии сопряженных графов, представленной в разделе 3.

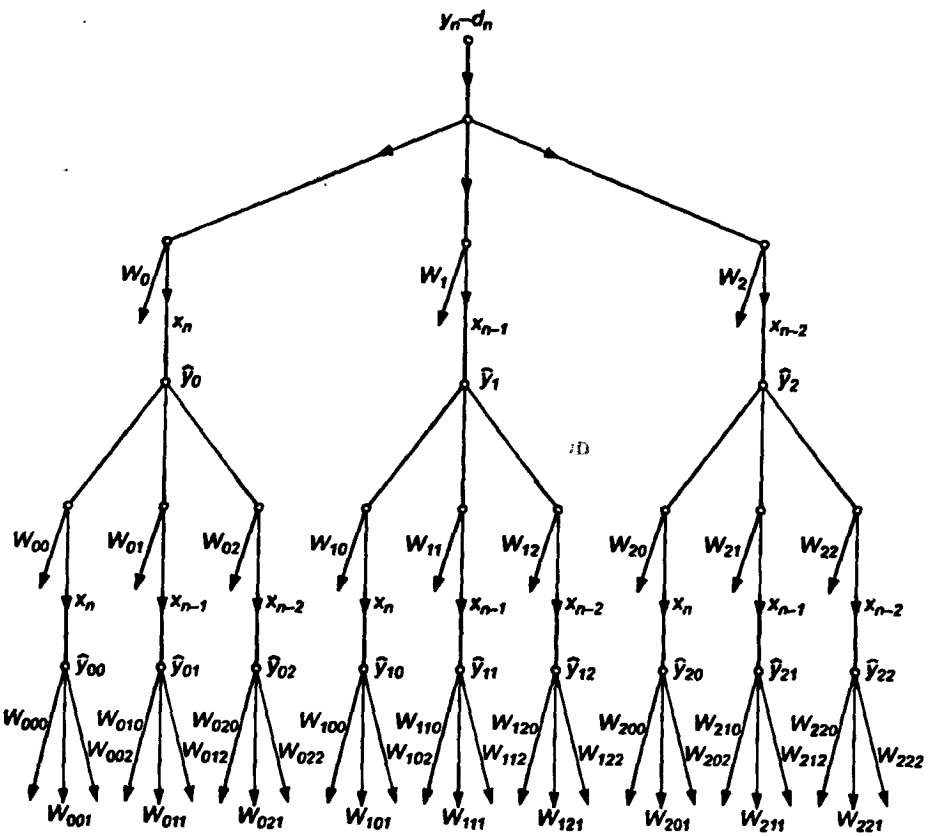

Рис. 6.6. Сопряженный граф сети Вольтерри

Сопряженный граф для сети, представленной на рис. 6.5, строится без особого труда. После его создания можно получить достаточно простые формулы, определяющие компоненты вектора градиента, составляющие основу процесса обучения. Сопряженный граф сети изображен на рис. 6.6. В соответствии с обозначениями, принятыми на этих рисунках, возбуждением сопряженного графа служит разностный сигнал $(y_n - d_n)$, где d_n обозначает ожидаемое, а y_n – фактическое значение в выходном узле системы в момент n. Принимая во внимание выражение (3.22), определяющее компоненты градиента, на основе

исходного и сопряженного с ним графа можно простым образом вывести конкретные компоненты этого вектора. В частности,

$$\frac{\partial E}{\partial w_i} = x_{n-i}(y_n - d_n), \tag{6.9}$$

$$\frac{\partial E}{\partial w_{ij}} = x_{n-j}\hat{y}_i, \tag{6.10}$$

$$\frac{\partial E}{\partial w_{ijk}} = x_{n-k}\hat{y}_{ij}, \tag{6.11}$$

$$\frac{\partial E}{\partial w_{i_1 i_2 \ldots i_{K-1} i_K}} = x_{n-i_K}\hat{y}_{i_1 i_2 \ldots i_{K-1}}. \tag{6.12}$$

В приведенных формулах сигналы, обозначенные символом $\hat{}$, соответствуют сопряженному, а остальные – исходному графу системы. После определения конкретных компонентов градиента обучение сети с применением оптимизационного метода наискорейшего спуска может быть сведено к решению системы дифференциальных уравнений:

$$\frac{dw_i}{dt} = -\mu\frac{\partial E}{\partial w_i}, \tag{6.13}$$

$$\frac{dw_{ij}}{dt} = -\mu\frac{\partial E}{\partial w_{ij}}, \tag{6.14}$$

$$\frac{dw_{ijk}}{dt} = -\mu\frac{\partial E}{\partial w_{ijk}}, \tag{6.15}$$

$$\frac{dw_{i_1 i_2 \ldots i_{K-1} i_K}}{dt} = -\mu\frac{\partial E}{\partial w_{i_1 i_2 \ldots i_{K-1} i_K}}, \tag{6.16}$$

где m обозначен коэффициент обучения. Важным достоинством метода сопряженных графов считается простота учета равных значений весов в различных ветвях сети. Легко заметить, что симметрия ядер Вольтерри приводит к равенству весов $w_{i_1 i_2 \ldots i_K}$ для всех перестановок индексов i_1, i_2, ..., i_K [137]. Это означает, что в случае двухиндексных весов наблюдается равенство $w_{ij} = w_{ji}$, а в случае трехиндексных весов – $w_{ijk} = w_{jik} = w_{ikj} = w_{jki} = w_{kij} = w_{kji}$. Подобное соотношение, только еще более громоздкое, относится к четырехиндексным, пятииндексным и т.д. весам. Анализ обозначений весов сети, изображенной на рис. 6.5, позволяет легко найти веса, которые должны иметь одни и те же значения. После расчета градиента относительно таких весов следует обратить внимание, что они присутствуют на различных позициях дифференцируемого выражения. При использовании правила суперпозиции дифференцирования такой функции можно заметить, что расчет градиента потребует повторения операции дифференцирования относительно всех ветвей сети, описанных общим

весом, с последующим суммированием отдельных компонентов. С учетом симметрии ядер Вольтерри выражения, описывающие компоненты градиента относительно весов w_{ij}, w_{ijk} и т.д., могут быть определенным образом модифицированы по сравнению с введенными ранее формулами. Их можно представить в виде (см. формулу 3.28)

$$\frac{\partial E}{\partial w_{ij}} = x_{n-j}\hat{y}_i + x_{n-i}\hat{y}_j, \tag{6.17}$$

$$\frac{\partial E}{\partial w_{ijk}} = x_{n-k}\hat{y}_{ij} + x_{n-j}\hat{y}_{ik} + x_{n-k}\hat{y}_{ji} + x_{n-i}\hat{y}_{jk} + x_{n-j}\hat{y}_{ki} + x_{n-i}\hat{y}_{kj}. \tag{6.18}$$

При использовании ядер Вольтерри высших порядков будет действовать аналогичное правило, обусловленное существованием ветвей, которые описываются весами с одинаковыми значениями. Выделение соответствующих компонентов градиента позволяет реализовать процесс оптимизации путем сведения его к решению системы дифференциальных уравнений, описываемых формулами (6.13) – (6.16).

6.2.2. Примеры использования сети Вольтерри

Идентификация нелинейного объекта

В процессе идентификации объекта одна и та же последовательность входных сигналов $x(n)$ подавалась параллельно на объект и его модель так, как это показано на рис. 6.7. Разность фактических реакций модели $y(n)$ и объекта $d(n)$ (последняя рассматривалась как ожидаемое значение) воспринималась как сигнал погрешности $\varepsilon(n) = y(n) - d(n)$, управляющий адаптивным алгоритмом, который подбирает параметры модели таким образом, чтобы уменьшить сигнал рассогласования $\varepsilon(n)$ до нуля. Использование в качестве модели нелинейной системы, описываемой полиномом Вольтерри, позволило значительно расширить класс идентифнцируемых объектов. При этом следует учитывать, что сеть

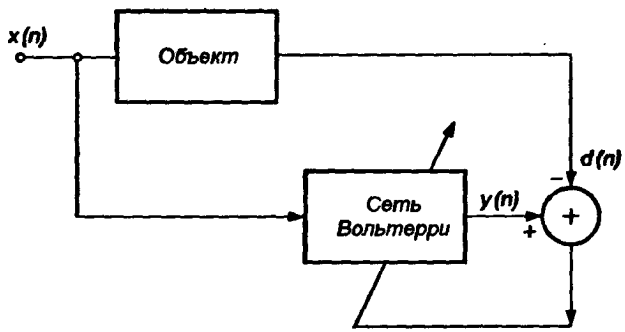

Рис. 6.7. Схема включения сети Вольтерри в адаптивной системе идентификации динамического объекта

Вольтерри является обобщением линейного фильтра типа FIR, поэтому она накрывает классы как линейных, так и нелинейных систем. При применении ее для идентификации объекта реализовалась описанная выше общая стратегия адаптации, основанная на разложении Вольтерри (6.8) и на концепции сопряженных графов. В экспериментальных исследованиях учитывался принцип естественной симметрии ядер Вольтерри, вследствие которой все веса $w_{i_1 i_2 \ldots i_K}$ имеют одни и те же значения для каждой комбинации индексов i_1, i_2, ..., i_K. Уравнения адаптации весов были представлены в форме дифференциальных уравнений (6.13) – (6.16) при описании компонентов вектора градиента выражениями (6.9), (6.17), (6.18). Вся адаптивная система была реализована на языке Simulink со значением коэффициента обучения m = 106. На рис. 6.8

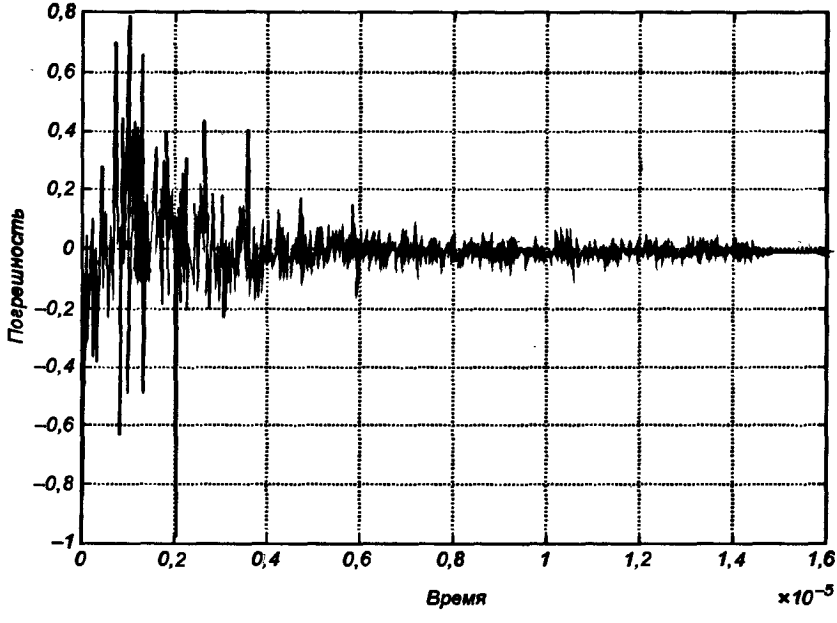

Рис. 6.8. Процесс обучения сети Вольтерри в примере идентификации

представлен график обучения $\varepsilon(n)$ как функция от времени для системы, идентифицирующей нелинейный динамический объект, который описывается зависимостью

$$d_n = x_{n-1} + x_{n-1}x_{n-2} + x_n x_{n-1} x_{n-2} \ .$$

За время, меньшее 15 микросекунд, погрешность идентификации была уменьшена до нуля, а реакции объекта и модели стали совпадать с точностью до второго знака после запятой. Параметры объекта были идентифицированы следующим образом:

$$w_{012} = w_{021} = w_{102} = w_{120} = w_{201} = w_{210} = 0{,}1649,$$
$$w_{12} = w_{21} = 0{,}4999,$$
$$w_1 = 1{,}02 \ .$$

Значения остальных весов фильтра были равны нулю с толерантностью 10^{-3}. В итоге объект был идентифицирован в виде

$$y_n = 1{,}02x_{n-1} + 0{,}999x_{n-1}x_{n-2} + 0{,}989x_n x_{n-1}x_{n-2},$$

который с высокой точностью обеспечивает получение принятых в качестве исходных значений d_n.

Устранение интерференционных шумов

Общая структура адаптивной системы для устранения интерференционных шумов представлена на рис. 6.9. Полезный сигнал s смешан с некоррелируемым с ним шумом n_0. Сигнал n является установочным, он не коррелирован с s, однако неизвестным образом коррелирует с сигналом помехи n_0. Считается, что s, n

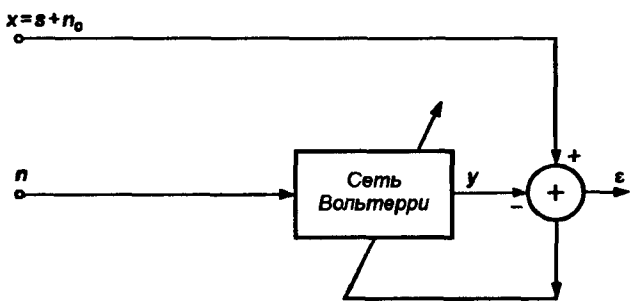

Рис. 6.9. Схема включения сети Вольтерри в адаптивной системе исключения интерференционных шумов

и n_0 статистически стационарны, а их средние значения равны нулю. Задача сети Вольтерри состоит в такой обработке сигнала n, чтобы сигнал y на выходе сети был как можно более близок к сигналу помехи n_0. Сигнал погрешности ε, вырабатываемый сумматором (рис. 6.9), определяется как

$$\varepsilon = s + n_0 - y. \tag{6.19}$$

Целевую функцию можно представить в форме ожидаемого значения E квадратичной погрешности

$$\xi = 0{,}5E[\varepsilon^2] = 0{,}5E[s^2 + (n_0 - y)^2 + 2s(n_0 - y)]. \tag{6.20}$$

Если принять во внимание, что сигнал s не коррелирует с сигналами помехи, то ожидаемое значение $E[s(n_0 - y)] = 0$ и целевая функция упрощаются до выражения

$$\xi = 0{,}5(E[s^2] + E[n_0 - y]^2). \tag{6.21}$$

Поскольку фильтр не изменяет сигнал s, минимизация функции погрешности ξ обеспечивается таким подбором его параметров, чтобы значение $E[n_0 - y]^2$ было минимальным. Таким образом, достижение минимума целевой

функции означает наилучшую адаптацию значения y к помехе n_0. Минимально возможное значение x равно $E[s^2]$, при котором $y = n_0$. В этом случае выходной сигнал ε соответствует полностью очищенному от шума полезному сигналу s.

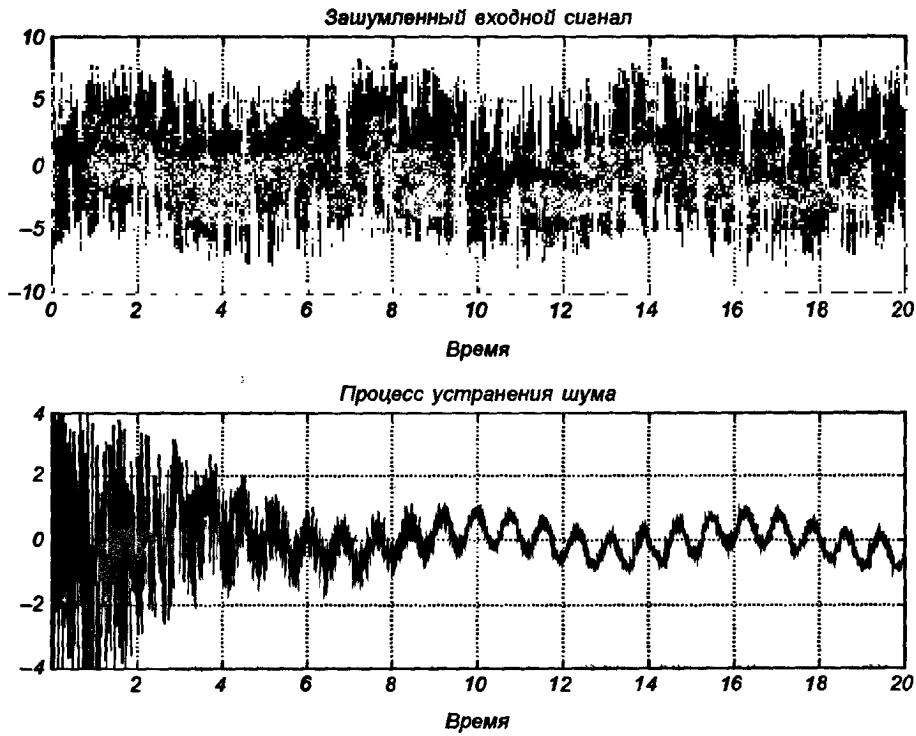

Рис. 6.10. Иллюстрация процесса устранения интерференционного шума

При использовании сети Вольтерри в качестве адаптивной системы следует помнить, что сигнал d равен измеренному сигналу $x = s + n_0$, а в качестве входного сигнала сети в данном случае используется установочный сигнал n. Поэтому выходной сигнал y_n может быть определен выражением

$$y_n = \sum_{i=0}^{L} n_{n-i}\left[w_i + \sum_{j=0}^{L} n_{n-j}\left[w_{ij} + \sum_{k=0}^{L} n_{n-k}(w_{ijk} + \ldots)\right]\right]. \qquad (6.22)$$

Если в рассуждениях ограничиться $K = 3$ и предположить эргодичность сигналов[1], то можно получить следующие уравнения адаптации весов нейронного фильтра Вольтерри ($d_n = s_n + n_{0n}$):

$$\frac{\mathrm{d}w_i}{\mathrm{d}t} = -\mu(n_{n-i}(y_n - d_n)), \qquad (6.23)$$

[1] Сигнал считается эргодическим, если при расчете его ожидаемого значения усреднение по многим реализациям можно заменить усреднением по одной реализации стохастического процесса.

$$\frac{dw_{ij}}{dt} = -\mu(n_{n-j}\hat{y}_i + n_{n-i}\hat{y}_j), \tag{6.24}$$

$$\frac{dw_{ijk}}{dt} = -\mu(n_{n-k}\hat{y}_{ij} + n_{n-j}\hat{y}_{ik} + n_{n-k}\hat{y}_{ji} + n_{n-i}\hat{y}_{jk} + n_{n-j}\hat{y}_{ki} + n_{n-i}\hat{y}_{kj}). \tag{6.25}$$

На базе приведенных зависимостей было выполнено моделирование нейронного фильтра Вольтерри с $K = 3$ и $L = 3$ с помощью программы *Simulink*, в ходе которого получены удовлетворительные результаты для таких сигналов, как синусоидальный, прямоугольный, треугольный, а также для суперпозиции синусоидальных сигналов. Применение описанного фильтра позволило устранять шум даже тогда, когда полезный сигнал частично присутствовал в установочном сигнале.

На рис. 6.10 иллюстрируется результат фильтрации сетью Вольтерри, содержащей три скрытых нейрона, сигнала *s*, образованного суммированием двух синусоидальных сигналов с сильно отличающимися частотами, на который накладывалась помеха, имеющая равномерное распределение. В системе наблюдалось частичное проникновение полезного сигнала в установочный сигнал.

Прогнозирование переменных во времени нестационарных сигналов

Блок-схема адаптивной системы для прогнозирования сигналов представлена на рис. 6.11. Введение нелинейности в адаптивное устройство обогащает его внутреннюю структуру и увеличивает способность к адаптации при решении более сложных задач. При использовании обозначений сигналов, приведенных на рис. 6.11, выходной сигнал фильтра Вольтерри описывается формулой

$$y_n = \sum_{i=0}^{L} h_{n-i}\left[w_i + \sum_{j=0}^{L} h_{n-j}\left[w_{ij} + \sum_{k=0}^{L} h_{n-k}(w_{ijk} + \ldots)\right]\right], \tag{6.26}$$

в которой $h_n = h(n)$ обозначает задержанный сигнал $x(n)$. Решение задачи адаптации весов находится из дифференциальных уравнений (6.13) – (6.16), причем конкретные компоненты градиента (при ограничении порядком $K = 3$) в этом случае имеют вид:

$$\frac{\partial E}{\partial w_i} = h_{n-i}(y_n - d_n), \tag{6.27}$$

$$\frac{\partial E}{\partial w_{ij}} = h_{n-j}\hat{y}_i + h_{n-i}\hat{y}_j, \tag{6.28}$$

$$\frac{\partial E}{\partial w_{ijk}} = h_{n-k}\hat{y}_{ij} + h_{n-j}\hat{y}_{ik} + h_{n-k}\hat{y}_{ji} + h_{n-i}\hat{y}_{jk} + h_{n-j}\hat{y}_{ki} + h_{n-i}\hat{y}_{kj}. \tag{6.29}$$

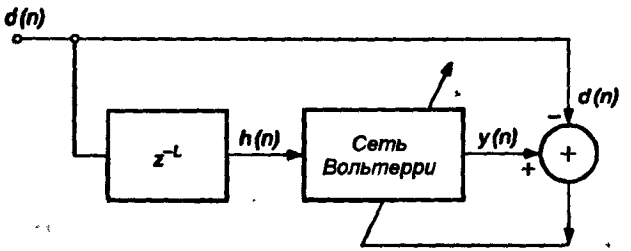

Рис. 6.11. Схема включения сети Вольтерри в качестве прогнозирующей системы

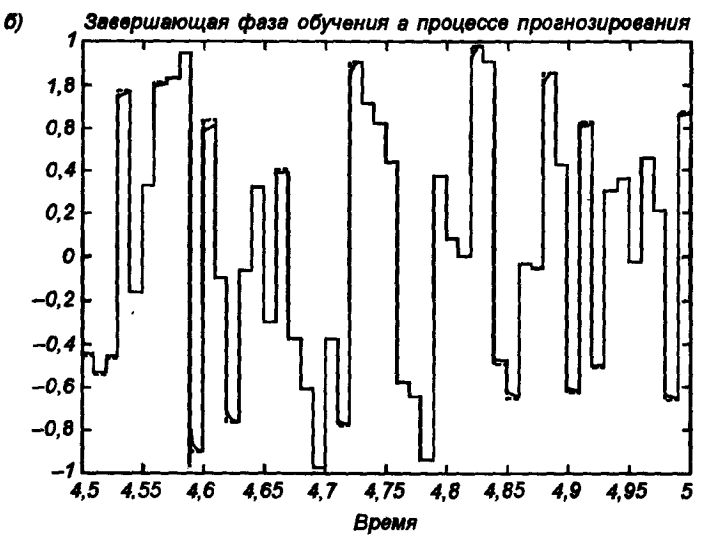

Рис. 6.12. Иллюстрация адаптационных возможностей сети Вольтерри, решающей задачу прогнозирования псевдослучайной последовательности:
а) начальная фаза обучения; *б*) завершающая фаза обучения

Как показали исследования, учет нелинейности фильтра значительно повышает качество прогнозирования. Это хорошо заметно на примере псевдослучайного сигнала, представляющего собой временной ряд, сформированный дискретизацией белого шума. На рис. 6.12 представлен процесс адаптации системы, предназначенной для прогнозирования этого сигнала. График на рис. 6.12 *а* демонстрирует временное распределение дискретизированного шума: фактическое (пунктирная линия) и спрогнозированное (сплошная линия) на первой стадии адаптации (от 0 до 1-й секунды). Видны существенные отличия между этими временными рядами. На рис. 6.12 *б* показано фактическое и спрогнозированное адаптивной системой распределение шума по истечении 4,5 сек. Заметно значительное улучшение результатов прогнозирования. Погрешности наблюдаются на границах скачкообразных изменений значений сигнала и имеют небольшую амплитуду. Сравнение полученных результатов с результатами линейного прогнозирования свидетельствует о резком повышении качества прогноза вследствие использования нелинейных элементов.

Раздел 7

РЕКУРРЕНТНЫЕ СЕТИ КАК АССОЦИАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

7.1. Введение

Отдельную группу нейронных сетей составляют сети с обратной связью между различными слоями нейронов. Это так называемые *рекуррентные сети*. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя во входной слой.

Главная особенность, выделяющая эти сети среди других нейронных сетей, – динамические зависимости на каждом этапе функционирования. Изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа "один ко многим". В сети возникает некоторый переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего. Если, как и прежде, функцию активации нейрона обозначить $f(u)$, где u – это взвешенная сумма его возбуждений, то состояние нейрона можно определить выходным сигналом $y_i = f(u_i) = f(\sum_{j=1}^{N} w_{ij} x_j)$. Принимая во внимание, что при обратной связи типа "один ко многим" роль возбуждающих импульсов для нейрона играют выходные сигналы других нейронов, изменение его состояния может быть описано системой дифференциальных нелинейных уравнений [46, 51]

$$\tau_i \frac{du_i}{dt} = \sum_{j=1, j\neq i}^{N} w_{ij} f(u_j) - u_i - b_i \qquad (7.1)$$

для $i = 1, 2, ..., N$, где b_i представляет собой пороговое значение, заданное внешним источником (в однонаправленных сетях это показатель поляризации). Коэффициент τ_i является численной константой, а его интерпретация аналогична постоянной времени в уравнениях, описывающих динамические состояния. Состояние нейрона рассчитывается в результате решения дифференциального уравнения (7.1) как $y_i = f(u)$. При определенном уровне возбуждения нейронов, описываемом значениями их выходных

сигналов y_i, рекуррентной сети можно сопоставить энергетическую функцию Ляпунова [46, 51, 54]

$$E = -\frac{1}{2}\sum_{j}\sum_{i,i\neq j} w_{ij}y_i y_j + \sum_{i=1}^{N}\frac{1}{R_i}\int_0^{x_i} f_i^{-1}(y_i)dy_i + \sum_{i=1}^{N} b_i y_i \ . \qquad (7.2)$$

Она связана с каждым возбужденным состоянием сети и имеет тенденцию убывания с течением времени. Изменение состояния какого-либо нейрона инициализирует изменение энергетического состояния всей сети в направлении минимума ее энергии вплоть до его достижения. Обычно существует множество локальных минимумов, каждый из которых представляет одно из состояний системы, сформированных на этапе обучения сети. В пространстве состояний локальные энергетические минимумы E представлены точками стабильности, называемыми *аттракторами* из-за тяготения к ним ближайшего окружения.

В настоящем разделе будут рассмотрены только отдельные избранные классы сетей, функционирующих в качестве ассоциативных запоминающих устройств. Ассоциативная память играет роль системы, определяющей взаимную зависимость векторов. В случае, когда на взаимозависимость исследуются компоненты одного и того же вектора, говорят об ассоциативной памяти. Если же взаимозависимыми оказываются два различных вектора $\boldsymbol{a}$ и $\boldsymbol{b}$, можно говорить о памяти гетероассоциативного типа. Типичным представителем первого класса является сеть Хопфилда, а второго – сеть Хемминга и сеть типа BAM (англ.: *Bidirectional Associative Memory* – двунаправленная ассоциативная память).

Главная задача ассоциативной памяти сводится к запоминанию входных (обучающих) выборок таким образом, чтобы при представлении новой выборки система смогла сгенерировать ответ – какая из запомненных ранее выборок наиболее близка к вновь поступившему образу. Наиболее часто в качестве меры близости отдельных множеств применяется мера Хемминга.

При использовании двоичных значений (0, 1) расстояние Хемминга между двумя векторами $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$ и $\boldsymbol{d} = [d_1, d_2, ..., d_n]^T$ определяется в виде [46]

$$d_H(\boldsymbol{y},\boldsymbol{d}) = \sum_{i=1}^{n}[d_i(1-y_i)+(1-d_i)y_i] \ . \qquad (7.3)$$

При биполярных (±1) значениях элементов обоих векторов расстояние Хемминга рассчитывается по формуле

$$d_H(\boldsymbol{y},\boldsymbol{d}) = \frac{1}{2}\left[n - \sum_{j=1}^{n} y_j d_j\right] . \qquad (7.4)$$

Мера Хемминга равна нулю только тогда, когда $\boldsymbol{y} = \boldsymbol{d}$. В противном случае она равна количеству битов, на которое различаются оба вектора.

7.2. Автоассоциативная сеть Хопфилда

7.2.1. Основные зависимости

Одним из наиболее известных типов ассоциативной памяти является сеть Хопфилда. Обобщенная структура этой сети представляется, как правило, в виде системы с непосредственной обратной связью выхода со входом (рис. 7.1). Характерная особенность такой системы состоит в том, что выходные сигналы нейронов являются одновременно входными сигналами сети: $x_i(k) = y_i(k-1)$, при этом возбуждающий вектор особо не выделяется. В классической системе Хопфилда отсутствует связь нейрона с собственным выходом, что соответствует $w_{ii} = 0$, а матрица весов является симметричной: $\mathbf{W} = \mathbf{W}^T$.

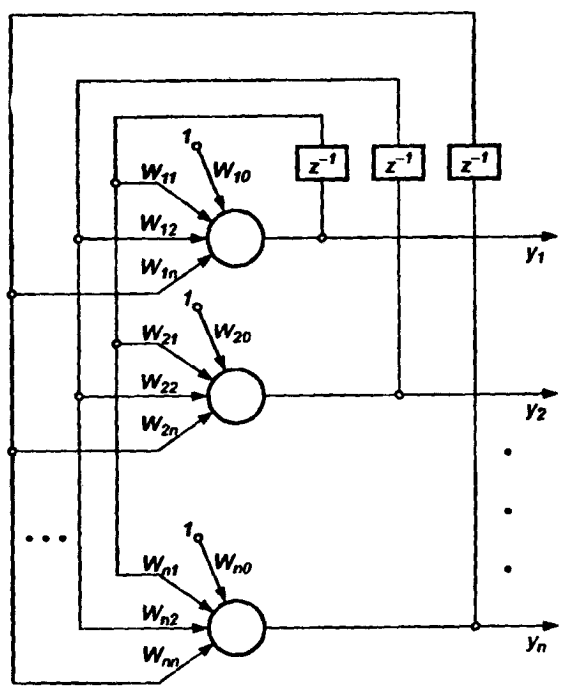

Рис. 7.1. Обобщенная структура сети Хопфилда

Процесс обучения сети формирует зоны притяжения (аттракции) некоторых точек равновесия, соответствующих обучающим данным. При использовании ассоциативной памяти мы имеем дело с обучающим вектором $\boldsymbol{x}$ либо с множеством этих векторов, которые в результате проводимого обучения определяют расположение конкретных аттракторов (точек притяжения). В последующих рассуждениях в соответствии с рекомендацией Хопфилда будем предполагать, что каждый нейрон имеет функцию активации типа *signum* со

значениями ±1. Это означает, что выходной сигнал i-го нейрона определяется функцией

$$y_i = \operatorname{sgn}\left(\sum_{j=0}^{N} w_{ij} x_j + b_i\right), \qquad (7.5)$$

где N обозначает количество нейронов, $N = n$.

Для упрощения дальнейших рассуждений допустим, что постоянная составляющая (поляризация), определяющая порог срабатывания отдельных нейронов, является компонентом вектора x. Без учета единичных задержек сети, представляющих собой способ синхронизации процесса передачи сигналов, основные зависимости, определяющие сеть Хопфилда, можно представить в виде

$$y_i(k) = \operatorname{sgn}\left(\sum_{j=1, i\neq j}^{N} w_{ij} y_j(k-1)\right), \qquad (7.6)$$

с начальным условием $y_i(0) = x_j$. В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных обучающих выборок x подбираются весовые коэффициенты w_{ij}. В режиме классификации при зафиксированных значениях весов и вводе конкретного начального состояния нейронов $y(0) = x$ возникает переходный процесс, протекающий в соответствии с выражением (7.6) и завершающийся в одном из локальных минимумов, для которого $y(k) = y(k-1)$.

При вводе только одной обучающей выборки x процесс изменений продолжается до тех пор, пока зависимость (7.6) не начнет соблюдаться для всех нейронов. Это условие автоматически выполняется в случае выбора значений весов, соответствующих отношению

$$w_{ij} = \frac{1}{N} x_i x_j , \qquad (7.7)$$

поскольку только тогда $\frac{1}{N}\left(\sum_{j=1}^{N} x_i x_j x_j\right) = x_i$ (вследствие биполярных значений элементов вектора x всегда $x_j^2 = (\pm 1)^2 = 1$). Следует отметить, что зависимость (7.7) представляет введенное ранее правило Хебба.

При вводе большего количества обучающих выборок $x(k)$ для $k = 1, 2, ..., p$ веса w_{ij} подбираются согласно обобщенному правилу Хебба, в соответствии с которым

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{p} x_i^{(k)} x_j^{(k)} . \qquad (7.8)$$

Благодаря такому режиму обучения веса принимают значения, определяемые усреднением множества обучающих выборок.

В случае множества обучающих выборок становится актуальным фактор стабильности ассоциативной памяти. Для стабильного функционирования сети необходимо, чтобы реакция i-го нейрона $y_i^{(l)}$ на l-ю обучающую выборку $x^{(l)}$ совпадала с ее i-й составляющей $x_i^{(l)}$. Это означает, что с учетом выражения (7.8) мы получаем

$$y_i^{(l)} = \operatorname{sgn}\left(\sum_{j=0}^{N} w_{ij} x_j^{(l)}\right) = \operatorname{sgn}\left(\frac{1}{N}\sum_{j=0}^{N}\sum_{k=1}^{p} x_i^{(k)} x_j^{(k)} x_j^{(l)}\right) = x_i^{(l)} . \qquad (7.9)$$

Если взвешенную сумму входных сигналов i-го нейрона обозначить $u_i^{(l)}$, то можно выделить в ней ожидаемое значение $x_i^{(l)}$ и остаток, называемый *диафонией*:

$$u_i^{(l)} = x_i^{(l)} + \frac{1}{N}\sum_{j=0}^{N}\sum_{k\neq l} x_i^{(k)} x_j^{(k)} x_j^{(l)} . \qquad (7.10)$$

Вследствие применения функции активации типа *signum* выполнение условия (7.9) возможно тогда, когда значение диафонии настолько мало, что оно не в состоянии изменить знак $x_i^{(l)}$. Это означает, что несмотря на определенное несовпадение битов (значение диафонии не равно нулю), переходный процесс завершается на нужном аттракторе. Ассоциативная память демонстрирует способности к коррекции. При представлении тестовой выборки, отличающейся некоторым количеством битов на отдельных позициях вектора, нейронная сеть может откорректировать эти биты и завершить процесс классификации на нужном аттракторе.

Важным параметром ассоциативной памяти считается ее емкость. Под этим термином следует понимать максимальное количество запомненных образов, которые классифицируются с допустимой погрешностью $\varepsilon_{\max}$. В [51] показано, что при использовании для обучения правила Хебба и при выборе значения $\varepsilon_{\max}$ = 1% (1% битов образца отличается от нормального состояния) максимальная емкость памяти (количество запомненных образцов составит всего лишь около 13,8 % от количества нейронов, образующих ассоциативную память. Это свидетельствует о невысокой продуктивности хеббовского обучающего правила. Именно по этой причине оно применяется редко. В качестве альтернативы используются методы обучения, основанные на псевдоинверсии, которые характеризуются гораздо более высокой эффективностью обучения.

7.2.2. Режим обучения сети Хопфилда

Фаза обучения сети Хопфилда ориентирована на формирование таких значений весов, при которых в режиме функционирования задание начального состояния нейронов, близкого к одному из обучающих векторов x, при соблюдении зависимости (7.6) приводит к стабильному состоянию, в котором реакция нейронов $y = x$ остается неизменной в любой момент времени.

Выше было показано, что применение правила Хебба для обучения малоэффективно, а в режиме функционирования при наличии шума (когда начальные выборки отличаются от запомненных значений) оно приводит к многочисленным неточностям в виде локальных минимумов, далеких от искомого решения.

Гораздо лучшие результаты можно получить, если для обучения используется псевдоинверсия. Отправной точкой этого метода считается предположение, что при правильно подобранных весах каждая поданная на вход выборка x генерирует на выходе саму себя, мгновенно приводя к искомому состоянию (зависимость (7.6)). В матричной форме это можно представить в виде

$$\mathbf{W}\,\mathbf{X} = \mathbf{X}, \tag{7.11}$$

где $\mathbf{W}$ – матрица весов сети размерностью $N \times N$, а $\mathbf{X}$ – прямоугольная матрица размерностью $N \times p$, составленная из p последовательных обучающих векторов $x^{(i)}$, т.е. $\mathbf{X} = [x^{(1)}, x^{(2)}, ..., x^{(p)}]$. Решение такой линейной системы уравнений имеет вид [42]:

$$\mathbf{W} = \mathbf{X}\,\mathbf{X}^{+}, \tag{7.12}$$

где знак + обозначает псевдоинверсию. Если обучающие векторы линейно независимы, последнее выражение можно упростить и представить в форме [42, 100]:

$$\mathbf{W} = \mathbf{X}\,(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T. \tag{7.13}$$

Псевдоинверсия матрицы размерностью $N \times p$ в этом выражении заменена обычной инверсией квадратной матрицы $\mathbf{X}^T\mathbf{X}$ размерностью $p \times p$. Дополнительное достоинство выражения (7.13) – возможность записать его в итерационной форме, не требующей расчета обратной матрицы. В этом случае (7.13) принимает вид функциональной зависимости от последовательности обучающих векторов $x^{(i)}$ для $i = 1, 2, ..., p$:

$$\mathbf{W}^{(i)} = \mathbf{W}^{(i-1)} + \frac{1}{[x^{(i)}]^T x^{(i)} - [x^{(i)}]^T \mathbf{W}^{(i-1)} x^{(i)}} \times [\mathbf{W}^{(i-1)} x^{(i)} - x^{(i)}] \times$$

$$\times [\mathbf{W}^{(i-1)} x^{(i)} - x^{(i)}]^T. \tag{7.14}$$

при начальных условиях $\mathbf{W}^{(0)} = 0$. Такая форма предполагает однократное предъявление всех p обучающих выборок, в результате чего матрица весов сети принимает фиксированное значение $\mathbf{W} = \mathbf{W}^{(p)}$. Зависимость (7.13) либо ее итерационная форма (7.14) называется *методом проекций*. Следует подчеркнуть, что применение метода псевдоинверсии увеличивает максимальную емкость сети Хопфилда, которая в этом случае становится равной $N-1$.

Модифицированный вариант метода проекций – так называемый *метод Δ-проекций* – это градиентная форма алгоритма минимизации определенной особым образом целевой функции. В соответствии с этим способом веса подбираются рекуррентно с помощью циклической процедуры, многократно повторяемой на всем множестве обучающих выборок:

$$\mathbf{W} \leftarrow \mathbf{W} + \frac{\eta}{N}[x^{(i)} - \mathbf{W}x^{(i)}][x^{(i)}]^T . \qquad (7.15)$$

Коэффициент η – это константа обучения, выбираемая обычно из интервала [0,7 – 0,9]. Его смысл тот же, что и в случае многослойных сетей. В отличие от обычного метода проекций метод Δ-проекций предполагает многократное предъявление всех p обучающих выборок вплоть до стабилизации значений весов. Процесс обучения завершается, когда изменения вектора весов становятся меньше априорно принятого значения толерантности ε.

Созданная в Институте электротехники и электроизмерений Варшавского политехнического университета компьютерная программа *Hfnet*, реализующая алгоритм, основанный на методах псевдоинверсии, продемонстрировала значительные преимущества над методом обучения по Хеббу. В режиме распознавания она правильно воспринимала значительно большие отклонения начального вектора от соответствующего ему аттрактора (одного из векторов, использованных для обучения).

7.2.3. Режим распознавания сети Хопфилда

По завершении подбора весов сети их значения "замораживаются", и сеть может использоваться в режиме распознавания. В этой фазе на вход сети подается тестовый вектор x и рассчитывается ее отклик в виде

$$y(i) = \mathrm{sgn}(\mathbf{W}y(i-1)) \qquad (7.16)$$

(в начальный момент $y(0) = x$), причем итерационный процесс повторяется для последовательных значений $y(i)$ вплоть до стабилизации отклика. Итерационный процесс стабилизации отклика системы состоит из определенного количества циклов и в значительной степени зависит от размеров сети и от распределения локальных минимумов.

В процессе распознавания образа по зашумленным сигналам, образующим начальное состояние нейронов сети Хопфилда, возникают проблемы с определением искомого конечного состояния, соответствующего одному из запомненных образов. Неоднократно итерационный процесс будет сходиться не к искомому, а к ошибочному решению. Этому есть много объяснений. Во-первых, значение энергетической функции, заданной выражением (7.2), зависит от произведения состояний двух нейронов и симметрично относительно поляризации. Одно и то же энергетическое состояние приписывается обеим поляризациям $\pm y_i$, $\pm y_j$ при условии, что они одновременно изменяют свои значения на противоположные. Поэтому для трехнейронной сети состояния (+1, –1, +1) и (–1, +1, –1) характеризуются идентичной энергией, и оба состояния считаются одинаково хорошим решением задачи. Переход из одного состояния в другое возможен при простой одновременной замене поляризации всех нейронов.

Другая причина выработки сетью Хопфилда ошибочных решений заключается в возможности перемешивания различных компонентов

запомненных образов и формирования стабильного состояния, воспринимаемого как локальный минимум. Следовательно, смешанное состояние соответствует такой линейной комбинации нечетного количества образов, которая сопровождается стабильным состоянием сети. Оно характеризуется более высоким энергетическим уровнем нейронов, чем искомое состояние.

При большом количестве образов образуются косвенные локальные минимумы, не соответствующие ни одному из запомненных образов, но определяемые сформированной структурой энергетической функции сети. Процесс распознавания может сойтись к одному из таких локальных минимумов, вследствие чего полученное решение не будет соответствовать состоянию ни одного из нейронов, принимавших участие в процессе обучения.

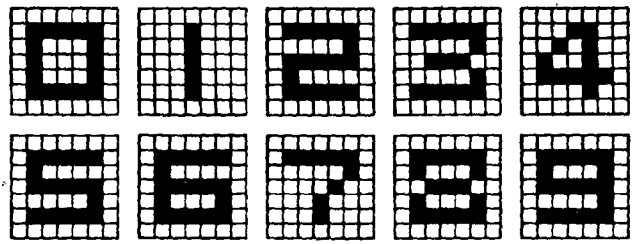

Рис. 7.2. Образы цифр, использованные для обучения сети Хопфилда

На рис. 7.2 и 7.3 демонстрируется эффективность функционирования сети Хопфилда на примере образцов 10 цифр, представленных в пиксельной форме размерностью 7 × 7. Поэтому количество нейронов сети Хопфилда составляет 49, а количество обучающих выборок – 10. Обучение проводилось с использованием программы *Hfnet* с применением трех описанных выше методов: по Хеббу, методов проекций и D-проекций. На этапе обучения

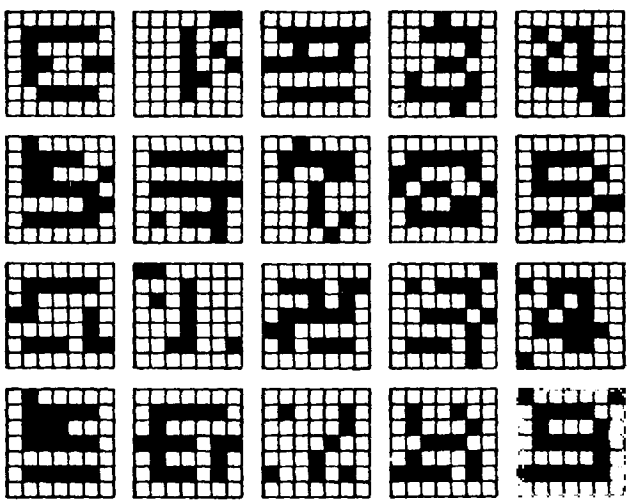

Рис. 7.3. Зашумленные образы цифр, использованные для тестирования сети Хопфилда

обрабатывались представленные на рис. 7.2 идеальные (незашумленные) образцы, дающие безошибочное восстановление. Обученная сеть подверглась тестированию на 20 сильно зашумленных образах, показанных на рис. 7.3.

Результаты распознавания сильно отличались в зависимости от применяемого метода обучения. В случае обучения по Хеббу только один образ был распознан безошибочно, а остальные не привели к искомому решению, поскольку процесс распознавания завершался в точках локальных минимумов, очень далеких от образов, использованных для обучения. Методы проекций и Δ-проекций дали возможность почти безошибочно распознать каждый из запомненных образов.

В завершение обсуждения сети Хопфилда следует упомянуть, что, помимо упомянутой выше программной реализации, существуют также ее аппаратурные реализации на основе стандартных элементов микроэлектронной технологии. Исходной точкой являются описание сети в виде дифференциального уравнения (7.1) и его реализация в виде специализированной аналоговой цепи. Мы не будем подробно останавливаться на сети Хопфилда этого типа, а интересующимся ею можно порекомендовать такие публикации, как [46, 53, 54, 55, 113, 182].

7.3. Сеть Хемминга

Предложенная Р. Липпманном в работе [91] сеть Хемминга – это трехслойная рекуррентная структура, которую можно считать развитием сети Хопфилда. Она позиционируется как специализированное гетероассоциативное запоминающее устройство. Основная идея функционирования этой сети состоит в минимизации расстояния Хемминга между тестовым вектором, подаваемым на вход сети, и векторами обучающих выборок, закодированными в структуре сети.

На рис. 7.4 представлена обобщенная схема сети Хемминга. Первый ее слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой, MAXNET, состоит из нейронов, связанных обратными связями по принципу "каждый с каждым", при этом в отличие от структуры Хопфилда существует ненулевая связь входа нейрона со своим собственным выходом. Веса нейронов в слое MAXNET также постоянны. Разные нейроны связаны отрицательной (подавляющей) обратной связью с весом $-\varepsilon$, при этом обычно величина ε обратно пропорциональна количеству образов. С собственным выходом нейрон связан положительной (возбуждающей) обратной связью с весом, равным +1. Веса поляризации нейронов принимают значения, соответствующие нулю. Нейроны этого слоя функционируют в режиме WTA, при котором в каждой фиксированной ситуации активизируется только один нейрон, а остальные пребывают в состоянии покоя. Выходной однонаправленный слой формирует выходной вектор, соответствующий входному вектору. Веса нейронов этого слоя подбираются в зависимости от входных обучающих выборок.

В процессе функционирования сети можно выделить три фазы. В первой из них на ее вход подается N-элементный вектор x. После предъявления этого вектора на выходах нейронов первого слоя генерируются сигналы, задающие начальные состояния нейронов второго слоя, т.е. MAXNET'а.

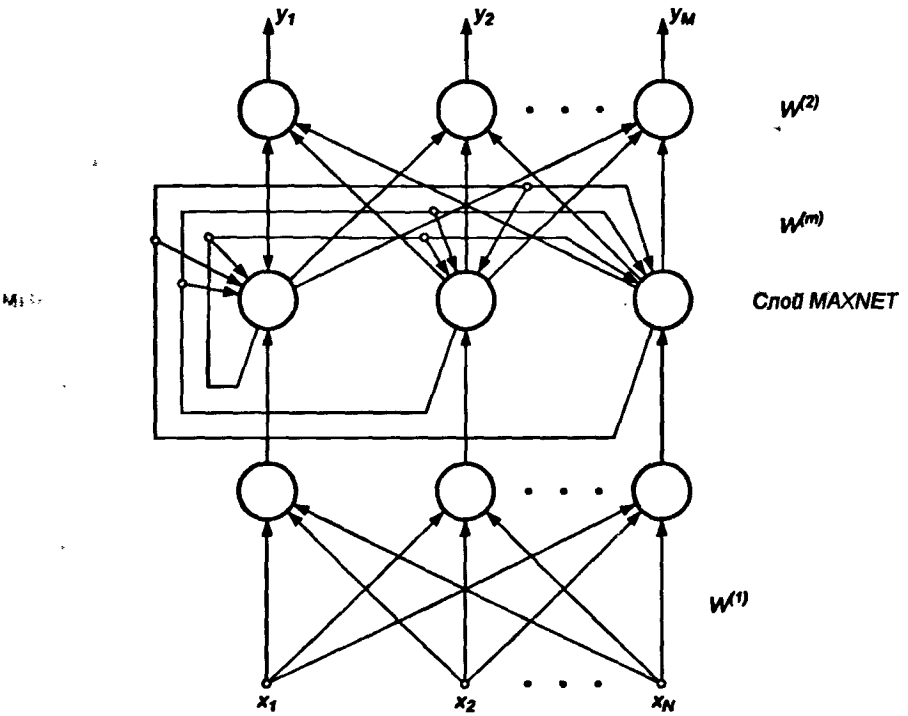

Рис.7.4. Структура сети Хемминга

Во второй фазе инициировавшие MAXNET сигналы удаляются, и из сформированного ими начального состояния запускается итерационный процесс внутри этого слоя. Итерационный процесс завершается в момент, когда все нейроны, кроме одного (победителя с выходным сигналом, равным 1), перейдут в нулевое состояние. Нейрон-победитель с ненулевым выходным сигналом становится представителем класса данных, к которому принадлежит входной вектор.

В третьей фазе этот же нейрон посредством весов, связывающих его с нейронами выходного слоя, формирует на выходе сети отклик в виде вектора y, соответствующий возбуждающему вектору x.

Сеть Хемминга считается гетероассоциативным запоминающим устройством с парой связанных между собой векторов (y, x), где x и y – это соответственно входной и выходной биполярные векторы сети со значениями элементов ± 1. Входные узлы сети 1, 2, ..., N принимают значения, задаваемые аналогичными компонентами вектора x. Нейроны первого слоя рассчитывают расстояние

Хемминга между фактически предъявленным входным вектором x и каждым из p закодированных векторов-образцов $x^{(i)}$, образующих веса нейронов первого слоя. Нейроны в слое MAXNET выбирают вектор с наименьшим расстоянием Хемминга, определяя таким образом класс, к которому принадлежит предъявленный входной вектор x. Веса нейронов выходного слоя формируют вектор, соответствующий предъявленному входному вектору. При p нейронах первого слоя емкость запоминающего устройства Хемминга также равна p, поскольку каждый нейрон представляет единственный класс.

Подбор весов сети Хемминга оказывается чрезвычайно простым. Веса первого слоя соответствуют очередным векторам образов $x^{(i)}$, поэтому

$$w_{ij}^{(1)} = x_j^{(i)} \tag{7.17}$$

для $i = 1, 2, ..., p$. Аналогично веса выходного слоя соответствуют очередным векторам образов $y^{(i)}$, связанным с $x^{(i)}$:

$$w_{ji}^{(2)} = y_j^{(i)} . \tag{7.18}$$

В случае нейронов слоя MAXNET, функционирующих в режиме WTA, веса сети должны усиливать собственный сигнал нейрона и ослаблять остальные. Для достижения этого эффекта принимается

$$w_{ii}^{(m)} = 1 , \tag{7.19}$$

а также

$$-\frac{1}{p-1} < w_{ij}^{(m)} < 0 \tag{7.20}$$

для $i \neq j$. Для обеспечения абсолютной сходимости алгоритма веса $w_{ij}^{(m)}$ должны отличаться друг от друга. Р. Липпманн в своей работе принял

$$w_{ij}^{(m)} = -\frac{1}{p-1} + \xi , \tag{7.21}$$

где ξ – случайная величина с достаточно малой амплитудой.

Нейроны различных слоев сети Хемминга функционируют по-разному. Нейроны первого слоя рассчитывают расстояния Хемминга между поданными на вход сети вектором x и векторами весов $w^{(i)} = x^{(i)}$ отдельных нейронов этого слоя ($i = 1, 2, ..., p$). Значения выходных сигналов этих нейронов определяются по формуле [35]

$$\hat{y}_i = 1 - \frac{d_H(x^{(i)}, x)}{N} , \tag{7.22}$$

где $d_H(x^{(i)}, x)$ обозначает расстояние Хемминга между входными векторами x и $x^{(i)}$, т.е. количество битов, на которое различаются эти два вектора. Значение $\hat{y}_i = 1$, если $x = x^{(i)}$, и $\hat{y}_i = 0$, если $x = -x^{(i)}$. В остальных случаях значения $\hat{y}_i$ располагаются в интервале [0, 1].

Сигналы $\hat{y}_i$ нейронов первого слоя становятся начальными состояниями нейронов слоя MAXNET на второй фазе функционирования сети. Задача нейронов этого слоя состоит в определении победителя, т.е. нейрона, уровень возбуждения которого наиболее близок к 1. Такой нейрон указывает на вектор образа с минимальным расстоянием Хемминга до входного вектора x. Процесс *определения победителя* – это рекуррентный процесс, выполняемый согласно формуле

$$y_i(k) = f\left(\sum_j w_{ij}^{(m)} y_j(k-1)\right) = f\left(y_i(k-1) + \sum_{j \neq i} w_{ij}^{(m)} y_j(k-1)\right), \qquad (7.23)$$

при начальном значении $y_j(0) = \hat{y}_i$. Функция активации $f(y)$ нейронов слоя MAXNET задается выражением

$$f(y) = \begin{cases} y & \text{для } y \geq 0 \\ 0 & \text{для } y < 0 \end{cases}. \qquad (7.24)$$

Итерационный процесс (7.23) завершается в момент, когда состояние нейронов стабилизируется и активность продолжает проявлять только один нейрон, тогда как остальные пребывают в нулевом состоянии. Активный нейрон становится победителем и через веса $w_{ij}^{(2)}$ линейных нейронов выходного слоя представляет вектор $y^{(i)}$, который соответствует вектору $x^{(i)}$, признанному слоем MAXNET в качестве ближайшего к входному вектору x.

Важным достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Например, 100-входовая сеть Хопфилда, кодирующая 10 различных векторных классов, должна содержать 10000 взвешенных связей с подбираемыми значениями весов. При построении аналогичной сети Хемминга количество взвешенных связей уменьшается до 1100, из которых 1000 весов находятся в первом слое и 100 – в слое MAXNET. Выходной слой в этом случае не учитывается, поскольку сеть Хемминга, аналогичная сети Хопфилда, является ассоциативной.

В результате многочисленных экспериментов доказано, что рекуррентная сеть Хемминга дает лучшие результаты, чем сеть Хопфилда, особенно в ситуациях, когда взаимосвязанные векторы x и y являются случайными. В частности, реализованная в программе *Mathlab* сеть Хемминга, протестированная на 10 цифрах, изображенных на рис. 7.3, позволила почти безошибочно распознать все представленные зашумленные образы. Достигнутая эффективность распознавания зашумленных образов составила 100%. На рис. 7.5 и 7.6 изображены искаженные образы цифр 0 – 9, поданные на вход натренированной сети Хемминга, и соответствующие им образы, распознанные этой сетью. Для цифр с рис. 7.5 только искаженным образам цифр 0, 3 и 6 были ошибочно приписаны другие оригиналы. Однако такое решение не может считаться результатом неправильного функционирования сети, поскольку распознанные образы соответствовали

эталонам с наименьшим расстоянием Хемминга до искаженных образов (после повреждения эталонов шумом они стали подобны остальным обучающим выборкам).

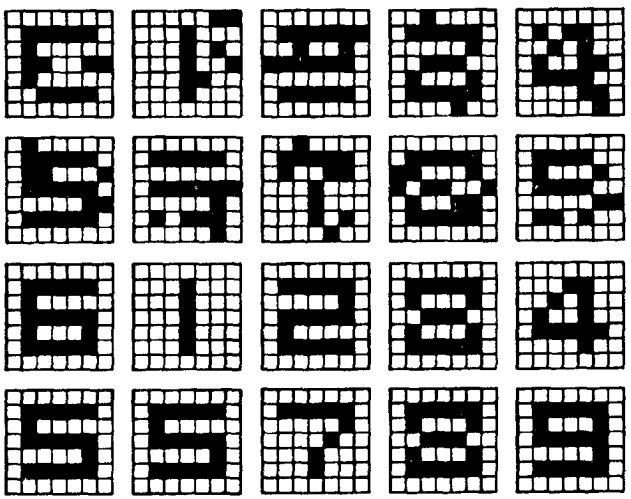

Рис.7.5. Тестовые (сверху) и распознанные сетью Хемминга (снизу) образы цифр при обработке первой группы искаженных входных данных

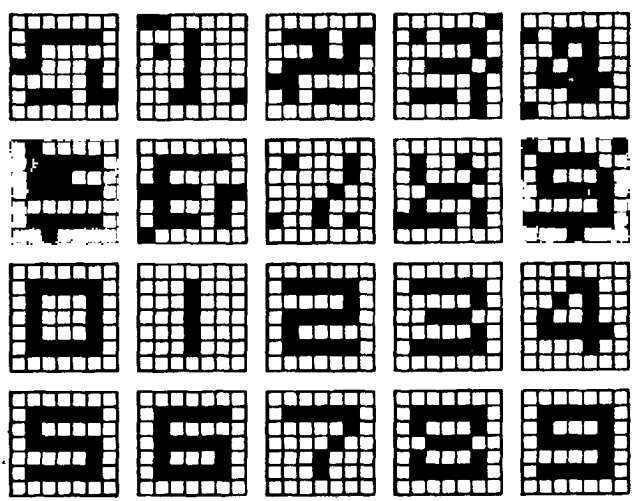

Рис.7.6. Тестовые (сверху) и распознанные сетью Хемминга (снизу) образы цифр при обработке второй группы искаженных входных данных

Единственная проблема, связанная с сетью Хемминга, проявляется в случае, когда зашумленные образы находятся на одинаковом (в смысле Хемминга) расстоянии от двух или более эталонов. В этом случае выбор сетью Хемминга одного из этих эталонов становится совершенно случайным.

7.4. Сеть типа BAM

7.4.1. Описание процесса функционирования сети

Обобщением сети Хопфилда на случай двухслойной рекуррентной структуры, позволяющей кодировать множества двух взаимосвязанных векторов, считается двунаправленное ассоциативное запоминающее устройство, называемое BAM (англ.: *Bidirectorial Associative Memory*), предложенное Б. Коско в работе [78]. Его обобщенная структура представлена на рис. 7.7. Сигналы распространяются в двух направлениях: от входа к выходу и обратно. Функционирование имеет синхронный характер. Это означает, что если в первом цикле сигналы вначале проходят в одну сторону для определения состояния нейронов-получателей, то в следующем цикле они сами становятся источником, высылающим сигналы в обратную сторону. Этот процесс повторяется до достижения состояния равновесия.

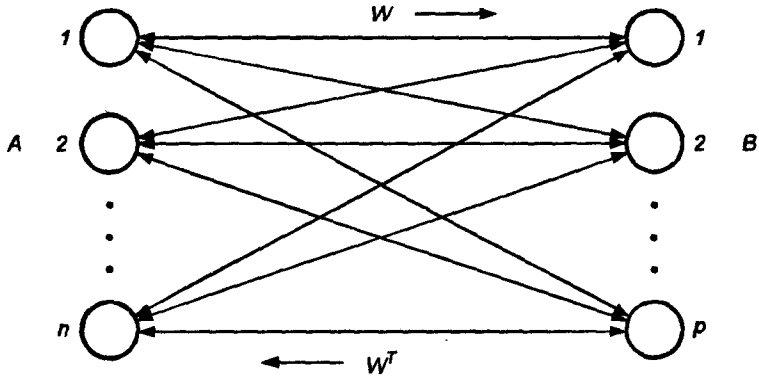

Рис.7.7. Структура сети BAM

Функция активации нейронов имеет пороговый характер: она может быть двоичной со значениями 1 или 0 либо биполярной со значениями ±1. При нулевом сигнале возбуждения нейрона его текущее состояние остается равным предыдущему состоянию. Для обеспечения лучших характеристик сети в режиме распознавания на этапе обучения используются только биполярные сигналы. Матрица весов **W**, связывающая обе части сети, является действительной и несимметричной. С учетом симметрии связей входного и выходного слоев сети при прямом направлении распространения сигналов веса описываются матрицей **W**, а при противоположном направлении – матрицей $\mathbf{W}^T$. Предположим, что входные обучающие данные определены в виде множества из m биполярных пар $\{(\boldsymbol{a}_i, \boldsymbol{b}_i)\}$, где $\boldsymbol{a}_i = [a_{i1}, a_{i2}, ..., a_{in}]$, $\boldsymbol{b}_i = [b_{i1}, b_{i2}, ..., b_{ip}]$ (векторы-строки). Этому множеству сопоставляется множество биполярных пар $\{(x_i, y_i)\}$, где x_i – это биполярное представление $\boldsymbol{a}_i$ ($0 \to -1$, $1 \to 1$), а y_i – биполярное представление $\boldsymbol{b}_i$.

В соответствии с определением Б. Коско [78] матрица весов **W** формируется на основе множества $\{(x_i, y_i)\}$ как матрица корреляции

$$\mathbf{W} = \sum_{i=1}^{m} x_i^T y_i \,. \tag{7.25}$$

Показано, что использование биполярных обучающих векторов дает лучшие результаты на стадии распознавания. Определение весов межнейронных связей позволяет проследить процесс стабилизации состояния на обоих концах сети. Если допустить, что начальное состояние сети было задано парой (x_0, y_0), то процесс двунаправленной обработки сигналов состоит из последовательных циклов

$$f(x_0\mathbf{W}) = y_1 \to f(y_1\mathbf{W}^T) = x_1 \to f(x_1\mathbf{W}) = y_2 \to$$
$$\to f(y_2\mathbf{W}^T) = x_2 \to f(x_2\mathbf{W}) = y_3 \to$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$\to f(y_f\mathbf{W}^T) = x_f \to f(x_f\mathbf{W}) = y_f \,,$$

в результате чего формируются две стабильные величины x_f и y_f, свидетельствующие о достижении стабильного состояния сети. В случае бинарного описания начального состояния в виде (a_0, b_0) биполярным величинам (x_f, y_f) сопоставляются бинарные представления (a_f, b_f). Каждой промежуточной точке процесса (x_k, y_k) можно сопоставить энергетическую функцию E_k, определяемую в виде [78]

$$E_k = -x_k\mathbf{W}y_k^T \,. \tag{7.26}$$

Доказано [78], что каждое очередное изменение состояния переходного процесса ведет к уменьшению значения энергетической функции сети вплоть до достижения локального минимума. Этот минимум достигается за конечное количество итераций, и он имеет значение

$$E_{\min} = -x_f\mathbf{W}y_f^T \,. \tag{7.27}$$

Иными словами, любое другое решение (в том числе и ближайшее, отличающееся лишь на 1 в смысле меры Хемминга от (x_f, y_f)) будет характеризоваться большим значением энергетической функции. При выполнении некоторых дополнительных условий парой (x_f, y_f) становится одна из обучающих пар, участвующих в формировании матрицы **W**, которая наиболее подобна (наиболее близка по мере Хемминга) паре, определившей начальное состояние (x_0, y_0).

В качестве примера рассмотрим обучение по правилу Коско сети BAM, имеющей 4 входа (векторы x состоят из 4 элементов) и 5 выходов (5-элементные векторы y). Задача сети состоит в запоминании множества из пяти сопряженных векторов x и y, заданных в биполярной форме. Обучающие векторы

сгруппированы в приведенные ниже матрицы **X** и **Y**. Каждая строка матрицы **X** представляет собой один обучающий вектор, сопряженный с соответствующей строкой матрицы **Y**.

$$\mathbf{X}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

$$\mathbf{Y}=\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Матрица весов сети, сформированная согласно формуле $\mathbf{W}=x_1^T y_1+x_2^T y_2+x_3^T y_3+x_4^T y_4+x_5^T y_5$, имеет вид:

$$\mathbf{W}=\begin{bmatrix} 3 & 1 & -1 & -3 & -5 \\ 1 & -1 & -3 & -5 & -3 \\ -1 & -3 & -5 & -3 & -1 \\ -3 & -5 & -3 & -1 & 1 \end{bmatrix}.$$

В режиме распознавания при начальных значениях векторов, совпадающих с использованными при обучении, сеть распознает их безошибочно. Значения энергии, соответствующие конечному состоянию, равны: $E_1=-x_1\mathbf{W}y_1^T=-40$, $E_2=-x_2\mathbf{W}y_2^T=-34$, $E_3=-x_3\mathbf{W}y_3^T=-32$, $E_4=-x_4\mathbf{W}y_4^T=-34$ и $E_5=-x_5\mathbf{W}y_5^T=-40$. При искажении значений векторов x и y, использовавшихся в процессе распознавания, спроектированная по алгоритму Коско сеть BAM не всегда способна откорректировать эти векторы, и распознает их с определенными погрешностями. Оригинальное решение, предложенное Б. Коско, характеризуется относительно невысоким качеством распознавания. Если размерности векторов x и y обозначить соответственно n и p, то удовлетворительное качество распознавания можно получить при выполнении зависимости $m<\sqrt{\min(n,p)}$.

7.4.2. Модифицированный алгоритм обучения сети BAM

В работе [161] показано, что если сопоставленная i-й обучающей паре энергия $E_i=-a_i\mathbf{W}b_i^T$ не составляет локальный минимум, то обучающая пара (a_i, b_i) не может быть распознана сетью даже тогда, когда начальные значения также равны (a_i, b_i).

Помимо того, BAM показывает неважные результаты, если в процессе обучения используются не похожие друг на друга векторы (например, подобным

векторам x сопоставляются не подобные друг другу векторы y, при этом степень подобия векторов измеряется расстоянием Хемминга $d_H(a_i, a_j)$, т.е. исследуется степень выполнения условия $\frac{1}{n}d_H(a_i,a_j) \approx \frac{1}{p}d_H(b_i,b_j)$ для всех значений i и j).

В работе [163] предложена модификация правила Коско, обеспечивающая распознавание вектора (a_i, b_i) независимо от того, образует он локальный минимум или нет. Вместо выражения (7.25) предлагается использовать

$$\mathbf{W} = \sum_{i=1}^{m} x_i^T y_i + (q-1)x_j^T y_j . \qquad (7.28)$$

Поправочный компонент $(q - 1)\, x_j^T y_j$ равнозначен увеличению в $(q - 1)$ раз участия пары (x_j, y_j) в процессе обучения. Подбор значения q важен для достижения ассоциативным запоминающим устройством хорошего качества распознавания. Процедуру добавления поправок можно повторять для каждой пары, не соответствующей условиям критерия минимизации энергетической функции, с использованием зависимости

$$\mathbf{W}^{(i)} = \mathbf{W}^{(i-1)} + (q-1)x_j^T y_j , \qquad (7.29)$$

принимая в качестве $\mathbf{W}^{(i-1)}$ матрицу, полученную на предыдущем цикле обучения. В энергетическом смысле предложенная Й. Вангом поправка уменьшает значение связанной с i-й парой энергетической функции с $E_i(a_i, b_i) = -a_i \mathbf{W} b_i^T$ до

$$E_i'(a_j,b_i) = -a_i \mathbf{W} b_i^T - (q-1)a_i x_i^T y_i b_i^T ,$$

соответствующего локальному минимуму. Предложенный Й. Вангом в работе [161] метод подбора значения q основан на формуле

$$q \geq \max\left(1,\ \frac{\varepsilon_{0i}^A}{2p}+1,\ \frac{\varepsilon_{0i}^B}{2n}+1\right), \qquad (7.30)$$

где ε_{0i}^A и ε_{0i}^B равны максимальным разностям энергии i-го оригинального вектора и векторов, отстоящих от него на расстояние Хемминга, равное 1, во множествах A и B соответственно, $\varepsilon_{0i}^A = \max(E_{0i}^A - E_{0i}')$, $\varepsilon_{0i}^B = \max(E_{0_i}^B - E_{0i}')$. Как показали исследования, обсуждаемая модификация также не обеспечивает 100%-ной безошибочности функционирования сети на стадии распознавания. Полную достоверность гарантирует только модификация матрицы весов, пред-ложенная в работе [163].

7.4.3. Модифицированная структура сети BAM

Авторы работы [163] предложили заменить матрицу W расширенной матрицей $\hat{\mathbf{W}}_f$ вида

$$\hat{\mathbf{W}}_f = [\,\mathbf{W}\mathbf{W}_{ij}] \qquad (7.31)$$

при передаче сигнала в направлении от x, и матрицей $\hat{\mathbf{W}}_b$ вида

$$\hat{\mathbf{W}}_b = [\mathbf{W}^T \mathbf{W}_x], \tag{7.32}$$

при передаче сигнала в направлении от y. Вводимая таким образом поправка разрушает симметрию передачи сигналов в противоположных направлениях. Дополнительные матрицы $\mathbf{W}_x$ и $\mathbf{W}_y$ конструируются так, что при нормальной работе алгоритма Коско их влияние нивелируется; они включаются в работу только при возникновении ошибок распознавания.

Пусть p' и n' обозначают количество обучающих пар, для которых в процессе распознавания получены неправильные ответы для векторов y и x соответственно. Индексами y и x будем обозначать процессы, приводящие к формированию ошибочных векторов y и x соответственно. Если (x_i, y_i) является очередной k-й обучающей парой, для которой $f(x_i\mathbf{W}) \neq y_i$, то принимается $\bar{y}_{ik} = 1$, $\bar{y}_{ij} = 0$ для $j \neq k$ $(k = 1, 2, ..., p')$. Если для (x_i, y_i) выполняется условие $f(x_i\mathbf{W}) = y_i$, то $\bar{y}_{ik} = 0$ для $k = 1, 2, ..., p'$. Компоненты $\bar{y}_{ik}$ образуют вектор $\bar{y}_i$ длиной p'. Аналогичным образом для процессов, распространяющихся в противоположном направлении, при замене векторов y на x можно получить векторы $\bar{x}_i$ длиной n'. Корректирующие матрицы $\mathbf{W}_x$ и $\mathbf{W}_y$ формируются согласно формулам [163]:

$$\mathbf{W}_y = \sum_{i=1}^{m} x_i^T \bar{y}_i, \tag{7.33}$$

$$\mathbf{W}_x = \sum_{i=1}^{m} y_i^T \bar{x}_i. \tag{7.34}$$

На следующем шаге создаются матрицы дополнительных узлов сети $\mathbf{T}_y$ и $\mathbf{T}_x$, причем

$$\mathbf{T}_y = \sum_{j=1}^{m} q_y \bar{y}_j^T y_j, \tag{7.35}$$

$$\mathbf{T}_x = \sum_{j=1}^{m} q_x \bar{x}_j^T x_j. \tag{7.36}$$

Параметры q_x и q_y подбираются таким образом, чтобы они соответствовали условиям:

$$q_y > n(m-2) - 2\min_i \left\{ \sum_{j \neq i} d_H(a_i, a_j) \right\}, \tag{7.37}$$

$$q_y > n(m-2) - 2\min_i \left\{ \sum_{j \neq i} d_H(b_i, b_j) \right\}. \tag{7.38}$$

Модифицированная структура сети BAM, в которой учитываются связи через матрицы $\mathbf{T}_x$ и $\mathbf{T}_y$, представлена на рис. 7.8. Зачернеиные нейроны увеличивают размерность сети, они корректируют неточности функционирования связей,

задаваемых матрицей **W**. После предъявления на вход сети тестовой пары (x_0, y_0) осуществляется аналогичный протекающему в сети Коско рекуррентный процесс, приводящий к получению конечных значений (x_f, y_f)

$$(x_0, y_0) \rightarrow (x_1, y_1) \rightarrow \cdots (x_f, y_f),$$

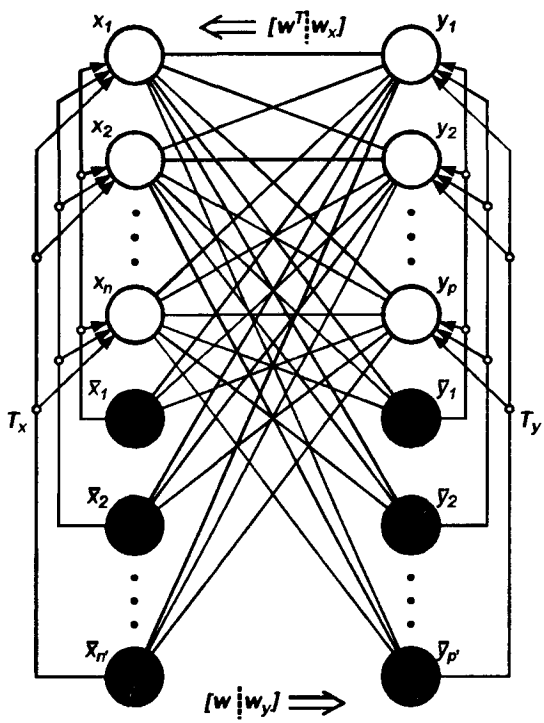

Рис.7.8. Структура расширенной сети BAM

при этом описание отдельных его этапов должно содержать дополнительные связи, показанные на рис. 7.8. При использовании введенных обозначений получаем

$$y_1 = f(x_0\mathbf{W} + g_y(x_0\mathbf{W}_y)\mathbf{T}_y) \ ,$$

$$x_1 = f(y_1\mathbf{W} + g_x(y_1\mathbf{W}_x)\mathbf{T}_x) \ ,$$

$$\cdots = \cdots$$

$$y_f = f(x_{f-1}\mathbf{W} + g_y(x_{f-1}\mathbf{W}_y) \ ,$$

$$y_1 = f(x_0\mathbf{W} + g_y(x_0\mathbf{W}_y)\mathbf{T}_y) \ ,$$

где $g_x()$ и $g_y()$ обозначают векторы функций активации дополнительных корректирующих нейронов. В алгоритме Ванга эти функции подбирают

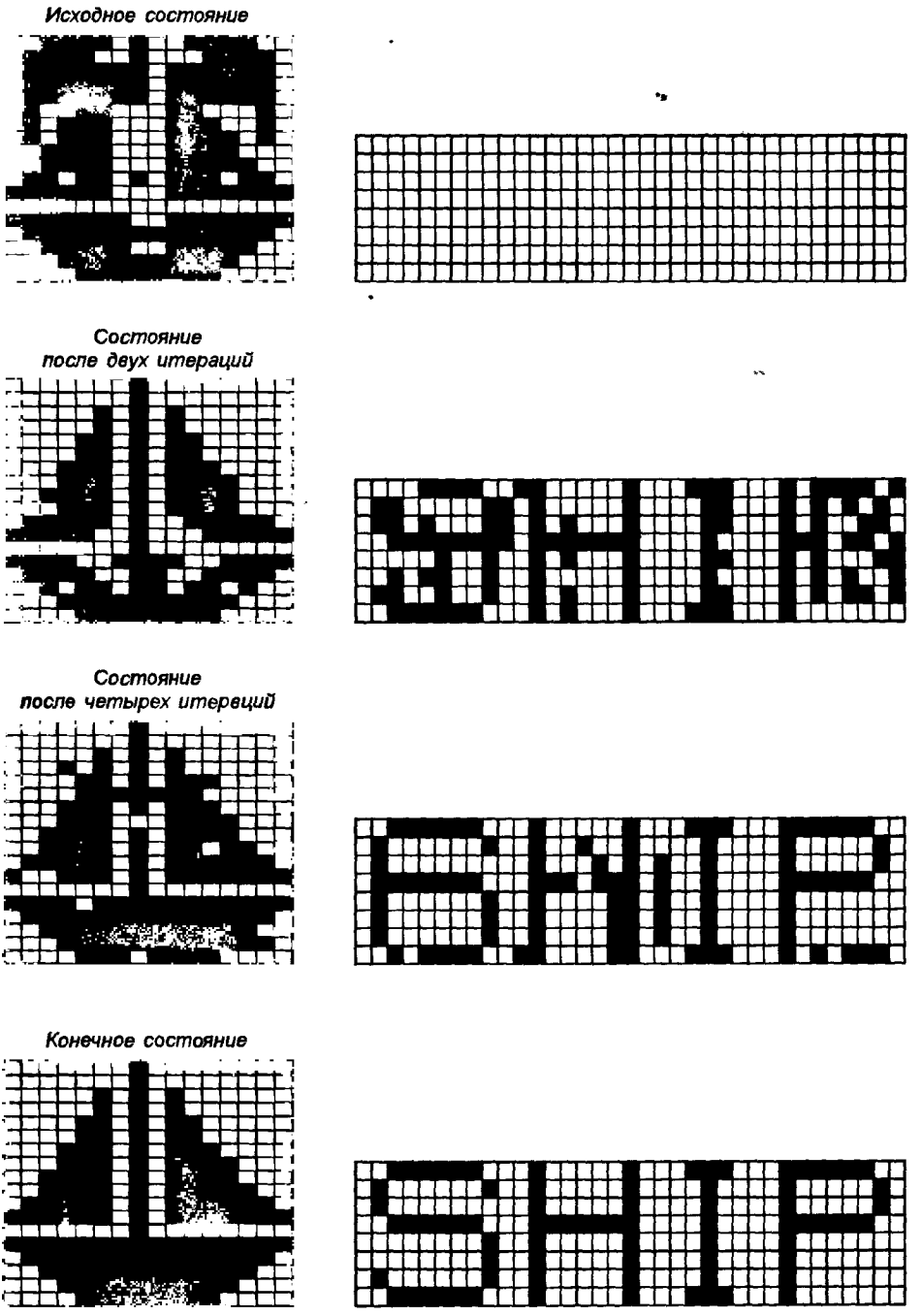

Рис.7.9. Иллюстрация последовательности распознавания сетью BAM двух сопряженных образов, предварительно искаженных шумом

следующим образом:

$$g_y(v) = [g_y(v_1), g_y(v_2), \ldots, g_y(v_{p'})],$$
$$g_x(v) = [g_x(v_1), g_x(v_2), \ldots, g_x(v_{n'})],$$

где элементы векторов g_x и g_y определяются выражениями:

$$g_y(v_i) = \begin{cases} 1 & \text{для } v_i > n - \varepsilon_1 \\ 0 & \text{для других} \end{cases},$$

$$g_x(v_i) = \begin{cases} 1 & \text{для } v_i > p - \varepsilon_2 \\ 0 & \text{для других} \end{cases}.$$

Величины ε_1 и ε_2 имеют положительные значения, удовлетворяющие условиям

$$0 << \varepsilon_1 << 2 \min_{i \neq j} \{d_H(a_i, a_j)\}, \quad (7.39)$$

$$0 << \varepsilon_2 << 2 \min_{i \neq j} \{d_H(b_i, b_j)\}. \quad (7.40)$$

В работе [163] доказано, что при подобной модификации сеть BAM всегда обеспечивает хорошее распознавание запомненных сигналов независимо от того, образуют они локальные минимумы или нет.

На рис. 7.9 показаны последовательные циклы функционирования модифицированной сети BAM на примере распознавания зашумленного схематического образа корабля и связанной с ним надписи *ship*. Обучающие данные, составляющие векторы $\boldsymbol{a}$ и $\boldsymbol{b}$, формировались на базе пиксельных карт, представляющих упрощенный образ корабля (вектор $\boldsymbol{a}$) и надпись SHIP (вектор $\boldsymbol{b}$). Размерность вектора $\boldsymbol{a}$ равнялась 288, а вектора $\boldsymbol{b}$ – 280. Процесс распознавания исходного идеального образа оказывается совсем не простым, и обычная структура BAM выполнить его не в состоянии. Модификация Ванга позволяет получить правильное решение, однако и в этом случае важную роль играет грамотный подбор коэффициентов ε_1 и ε_2. Слишком малые или слишком большие значения этих коэффициентов приводят к тому же эффекту, вызывая снижение фильтрационных способностей сети и невозможность получения образа, очищенного от шума.

На рис. 7.10 приведен тестовый набор из пяти различных образов (вертолет, танк, самолет, корабль и лицо), связанных с соответствующими надписями на английском языке. Векторы $\boldsymbol{x}_i$ описывают образы, а векторы $\boldsymbol{y}_i$ – надписи. Размерности всех векторов $\boldsymbol{x}_i$ равны 288, а векторов $\boldsymbol{y}_i$ – 280. На рис. 7.11 представлены последовательные этапы распознавания этих образов после предъявления их сети в искаженном виде (шум повреждал также и надписи). После двухкратного прохождения сигналов через модифицированную сеть BAM произошло безошибочное распознавание как образов, соответствующих изображениям (векторов $\boldsymbol{x}$), так и связанных с ними надписей (векторов $\boldsymbol{y}$).

ункционирование системы огромное влияние оказывает подбор параметров полнительной части сети, который в значительной степени зависит от степени кажения образов. Это считается определенным неудобством метода,

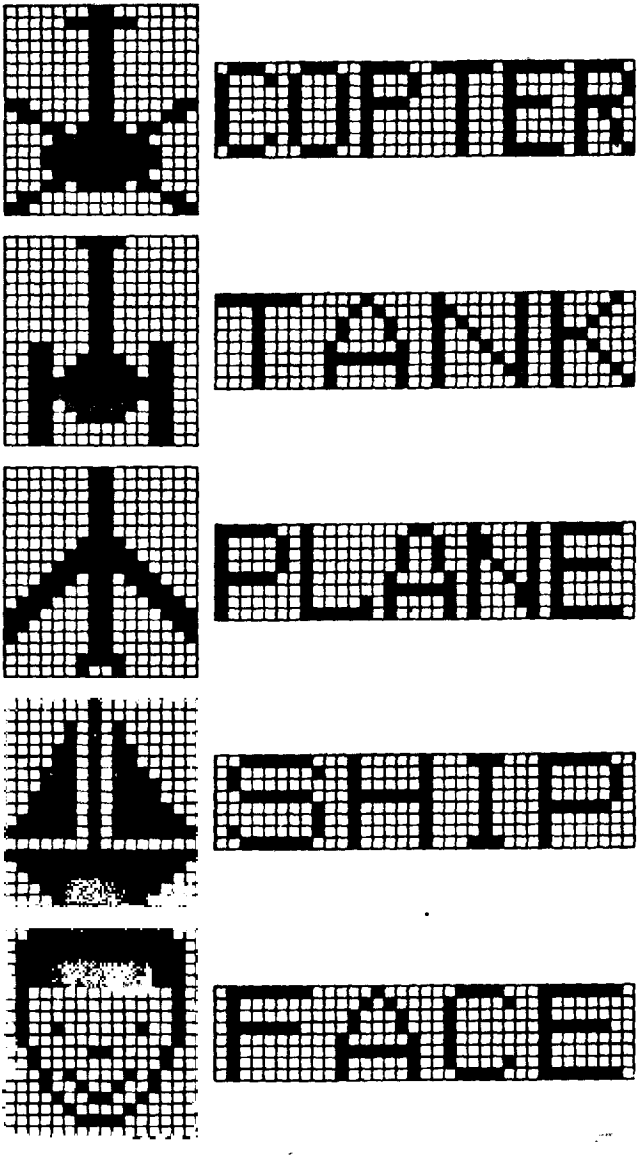

Рис.7.10. Набор данных для тестирования сети BAM с расширенной структурой

поскольку параметры дополнительной части сети, подобранные оптимальным образом для одного уровня шума, необязательно будут эффективны при изменении этого уровня. Для набора образов, исследовавшихся в описываемом

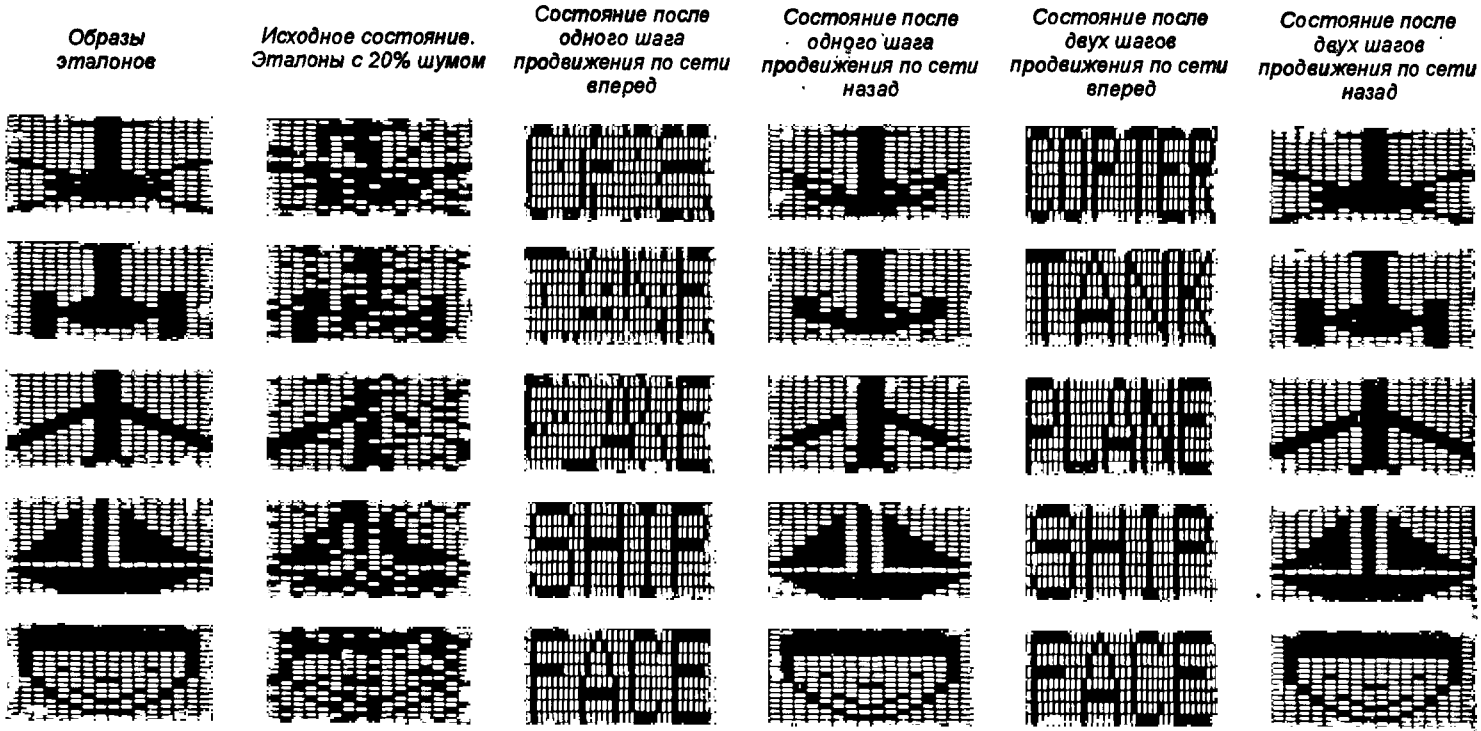

Рис. 7.11. Последовательность распознавания сетью BAM с модифицированной структурой всех пар образов при 20%-ном искажении входных изображений

численном эксперименте, обученная при 20%-ном шуме (20% пикселов находилось в искаженных состояниях) сеть также обеспечивала безошибочное распознавание образов, искажение которых достигало 50%.

Интересным представляется сравнение емкости сети BAM при использовании различных алгоритмов обучения. Оригинальная процедура Коско характеризуется относительно невысоким качеством распознавания. Если размерности векторов x и y обозначить n и p соответственно, то распознавание будет считаться удовлетво-рительным при емкости $m < \sqrt{\min(n, p)}$. При использовании модифицированной Вангом структуры сети какие-либо условия и ограничения распознаваемости входных векторов отсутствуют. Однако это достигается за счет увеличения размерности сети и количества межнейронных соединений. При $m \gg \sqrt{\min(n, p)}$ дополнительные связи, вводимые применяемым алгоритмом, становятся доминирующими и оказывают решающее влияние на функционирование сети.

Раздел 8

РЕКУРРЕНТНЫЕ СЕТИ НА БАЗЕ ПЕРСЕПТРОНА

8.1. Введение

Рекуррентные сети, рассматриваемые в настоящем разделе, представляют собой развитие однонаправленных сетей персептронного типа за счет добавления в них соответствующих обратных связей. Обратная связь может исходить либо из выходного, либо из скрытого слоя нейронов. В каждом контуре такой связи присутствует элемент единичной задержки, благодаря которому поток сигналов может считаться однонаправленным (выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размерность входного вектора x сети). Представленная подобным образом рекуррентная сеть с учетом способа формирования выходного сигнала функционирует как однонаправленная персептронная сеть. Тем не менее алгоритм обучения такой сети, адаптирующий значения синаптических весов, является более сложным вследствие зависимости сигналов в момент времени t от их значений в предыдущие моменты и соответственно ввиду более громоздкой формулы для расчета вектора градиента.

При обсуждении рекуррентных сетей, в которых в качестве выходного элемента используется многослойный персептрон, мы обсудим наиболее известные структуры сетей и разработанные для них алгоритмы обучения. В этом разделе мы ограничимся сетями RMLP, RTRN Вильямса–Зипсера и сетью Эльмана. Будут рассмотрены примеры реализации таких сетей и результаты численного моделирования при решении конкретных тестовых задач.

8.2. Персептронная сеть с обратной связью

8.2.1. Структура сети RMLP

Один из простейших способов построения рекуррентной сети на базе однонаправленной ИНС состоит во введении в персептронную сеть обратной связи.

В дальнейшем мы будем сокращенно называть такую сеть RMLP (англ.: *Recurrent MultiLayer Perceptron* – рекуррентный многослойный персептрон). Ее обобщенная структура представлена на рис. 8.1.

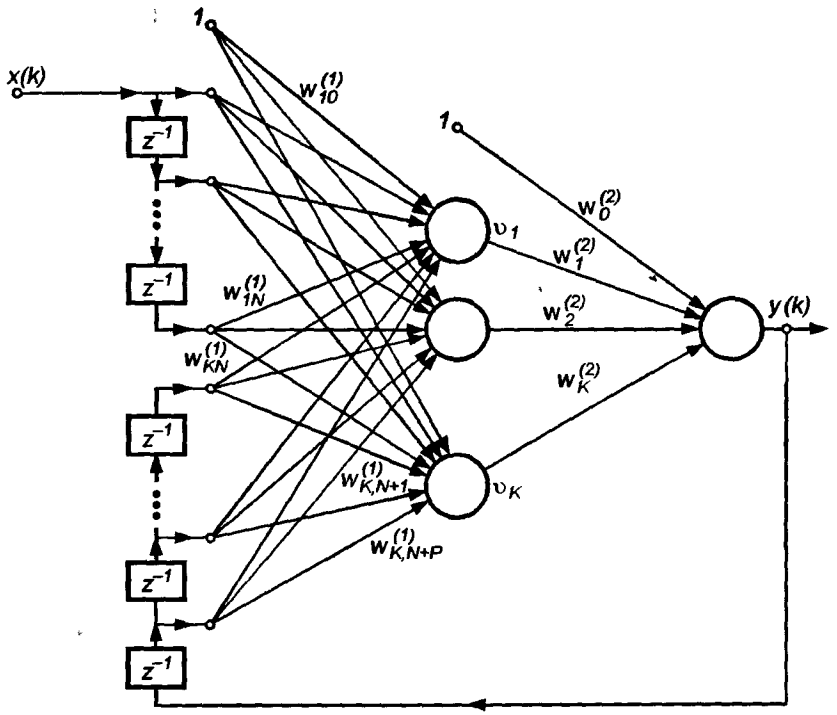

Рис. 8.1. Структура сети RMLP

Это динамическая сеть, характеризующаяся запаздыванием входных и выходных сигналов, объединяемых во входной вектор сети. Рассуждения будут касаться только одного входного узла $x(k)$ и одного выходного нейрона, а также одного скрытого слоя. Такая система реализует отображение:

$$y(k+1)=f(x(k),\ x(k-1),\ ...,\ x(k-(N-1)),\ y(k-1),\ y(k-2),\ ...,\ y(k-P)\,, \qquad (8.1)$$

где $N-1$ – количество задержек входного сигнала, а P - количество задержек выходного сигнала. Обозначим K количество нейронов в скрытом слое. В этом случае сеть RMLP можно характеризовать тройкой чисел (N, P, K). Подаваемый на вход сети вектор $\boldsymbol{x}$ имеет вид: $\boldsymbol{x}(k) = [1,\ x(k),\ x(k-1),\ ...,\ x(k-(N-1)),\ y(k-P),\ y(k-P+1),\ ...,\ y(k-1)]^T$. Допустим, что все нейроны имеют сигмоидальную функцию активации. Обозначим u_i взвешенную сумму сигналов i-го нейрона скрытого слоя, а g – взвешенную сумму сигналов выходного нейрона. При введенных обозначениях выходные сигналы конкретных нейронов описываются зависимостями:

$$u_i = \sum_{j=0}^{N+P} w_{ij}^{(1)} x_j\ , \qquad (8.2)$$

$$v_i = f(u_i)\ , \qquad (8.3)$$

$$g = \sum_{i=0}^{K} w_i^{(2)} f(u_i), \tag{8.4}$$

$$y = f(g). \tag{8.5}$$

8.2.2. Алгоритм обучения сети RMLP

Сеть RMLP адаптируется с применением градиентного алгоритма обучения. Как и в ситуации с однонаправленной сетью, рассчитывается градиент целевой функции относительно каждого веса. Для упрощения будем рассматривать сеть с одним выходным нейроном. В этом случае целевую функцию в момент t можно определить в виде

$$E(k) = \frac{1}{2}[y(k) - d(k)]^2. \tag{8.6}$$

Дифференцируя эту функцию относительно произвольного веса $w_\alpha^{(2)}$ ($\alpha = 0, 1, ..., K$) выходного слоя сети, получаем:

$$\frac{\partial E(k)}{\partial w_\alpha^{(2)}} = [y(k) - d(k)]\frac{dy(k)}{dw_\alpha^{(2)}} = [y(k) - d(k)]\frac{df(g(k))}{dg(k)}\frac{dg(k)}{dw_\alpha^{(2)}}. \tag{8.7}$$

С учетом зависимостей (8.2) – (8.5) получаем

$$\frac{\partial E(k)}{\partial w_\alpha^{(2)}} = [y(k) - d(k)]\frac{df(g(k))}{dg(k)}\sum_{i=0}^{K}\frac{d(w_i^{(2)} v_i(k))}{dw_\alpha^{(2)}}, \tag{8.8}$$

где $v_i = f(u_i)$. Производная $\frac{d(w_i^{(2)})}{dw_\alpha^{(2)}}$ равна 1 только при $i = a$ и равна 0 во всех остальных случаях. С учетом этого факта

$$\frac{\partial E(k)}{\partial w_\alpha^{(2)}} = [y(k) - d(k)]\frac{df(g(k))}{dg(k)}\left[v_\alpha(k) + \sum_{i=0}^{K} w_i^{(2)}\frac{dv_i(k)}{dw_\alpha^{(2)}}\right], \tag{8.9}$$

причем

$$\frac{dv_i(k)}{dw_\alpha^{(2)}} = \frac{df(u_i(k))}{du_i(k)}\sum_{j=0}^{N+P} w_{ij}^{(1)}\frac{dx_j}{dw_\alpha^{(2)}} =$$

$$= \frac{df(u_i(k))}{du_i(k)}\sum_{j=N+1}^{N+P} w_{ij}^{(1)}\frac{dy(k-P-1+(j-N))}{dw_\alpha^{(2)}} =$$

$$= \frac{df(u_i(k))}{du_i(k)}\sum_{j=1}^{P} w_{i,j+N}^{(1)}\frac{dy(k-P-1+j)}{dw_\alpha^{(2)}}. \tag{8.10}$$

С учетом зависимостей (8.6) – (8.10) получаем

$$\frac{dy(k)}{dw_\alpha^{(2)}} = \frac{df(g(k))}{dg(k)}\left[v_\alpha(k) + \sum_{i=0}^{K} w_i^{(2)} \frac{df(u_i(k))}{du_i^{(k)}} \sum_{j=1}^{P} w_{i,j+N}^{(1)} \frac{dy(k-P-1+j)}{dw_\alpha^{(2)}}\right]. \quad (8.11)$$

Рекуррентная формула (8.11) позволяет рассчитать значение производной $\frac{dy(k)}{dw_\alpha^{(2)}}$ в произвольный момент времени по ее значениям в предыдущие моменты. Она связывает значения производных в момент t со значениями тех же функций в моменты $t-1$, $t-2$, ..., $t-P$. Можно предположить, что начальные значения производных от сигналов перед началом обучения равны, т.е.

$$\frac{dy(0)}{dw_\alpha^{(2)}} = \frac{dy(-1)}{dw_\alpha^{(2)}} = ... = \frac{dy(-P)}{dw_\alpha^{(2)}} = 0.$$

При использовании в процессе обучения метода наискорейшего спуска адаптация весов выходного слоя определяется формулой

$$\Delta w_\alpha^{(2)} = -\eta[y(k) - d(k)]\frac{dy(k)}{dw_\alpha^{(2)}}. \quad (8.12)$$

Актуализация весов скрытого слоя происходит аналогичным образом. После расчета производной сигнала $y(k)$ относительно веса $w_{\alpha,\beta\beta}^{(1)}$ скрытого слоя получаем ($\delta_{i\alpha}$ здесь означает дельту Кронекера)

$$\frac{dy(k)}{dw_{\alpha,\beta}^{(1)}} = \frac{df(g(k))}{dg(k)} \sum_{i=1}^{K} w_i^{(2)} \frac{df(u_i(k))}{du_i^{(k)}}\left[\sum_{j=1}^{P} w_{i,j+N}^{(1)} \frac{dy(k-P-1+j)}{dw_\alpha^{(2)}} + \delta_{i\alpha} x_\beta\right]. \quad (8.13)$$

Следовательно, формула, определяющая адаптацию веса $w_{\alpha,\beta\beta}^{(1)}$ скрытого слоя, при использовании метода наискорейшего спуска принимает вид

$$\Delta w_{\alpha\beta}^{(1)} = -\eta[y(k) - d(k)]\frac{dy(k)}{dw_{\alpha\beta}^{(1)}}. \quad (8.14)$$

В окончательном виде алгоритм обучения сети RMLP можно сформулировать следующим образом.

1. Выполнить инициализацию случайным способом весов нейронов скрытого и выходного слоев.
2. Для каждого момента t при заданном возбуждении в виде вектора x рассчитать состояние всех нейронов сети в соответствии с формулами (8.2) – (8.5).
3. С помощью зависимостей (8.11) и (8.13) определить значения производных $\frac{dy(k)}{dw_\alpha^{(2)}}$ и $\frac{dy(k)}{dw_{\alpha\beta}^{(1)}}$ для всех значений α и β, соответствующих весам сети с изначально выбранной структурой.
4. Актуализировать веса в соответствии с формулами (8.12) и (8.14), после чего вернуться к п.2 настоящего алгоритма.

Представленный алгоритм функционирует в режиме "онлайн", принимая поступающие входные данные и соответствующие им значения ожидаемого вектора d и оперативно корректируя значения весов.

8.2.3. Подбор коэффициента обучения

При обучении нейронной сети по методу обратного распространения ошибок решающее влияние на скорость обучения и на получаемые конечные результаты оказывает коэффициент обучения η. Значение этого коэффициента в процессе обучения может оставаться постоянным либо подбираться адаптивным способом. Сохранение постоянного значения коэффициента обучения считается самой простой формой определения η. Такой способ имеет много недостатков, в том числе медленную сходимость, высокую вероятность расходимости процесса при слишком большом значении η, легкость попадания в точки локальных минимумов. Тем не менее до настоящего времени он остается наиболее простым и эффективным методом, используемым при обучении в режиме "онлайн". Адаптивный подбор коэффициента η позволяет контролировать погрешности обучения, в результате чего проводится увеличение или уменьшение его значения. Для ускорения процесса обучения предусматривается непрерывное возрастание коэффициента η, если уровень фактической погрешности по сравнению с погрешностью предыдущей итерации находится в допустимых пределах. Если обозначить ε_{i-1} и ε погрешности адаптации на i-м и $(i-1)$-м шаге, а η_{i-1} и η_i – соответствующие им коэффициенты обучения, то в случае $\varepsilon_i > k_w\varepsilon_{i-1}$ (k_w – коэффициент допустимого прироста погрешности) производится уменьшение значения η по формуле

$$\eta_{i+1} = \eta_i \alpha_d, \tag{8.15}$$

где α_d является коэффициентом уменьшения значения η. В противном случае, когда $\varepsilon_i \le k_w\varepsilon_{i-1}$, значение этого коэффициента увеличивается по формуле

$$\eta_{i+1} = \eta_i \alpha_i, \tag{8.16}$$

где α_i является коэффициентом увеличения значения η.

8.2.4. Коэффициент усиления сигнала

Применение выходных нейронов с сигмоидальной функцией активации дает возможность минимизировать структуру рекуррентной нейронной сети. В сети RMLP стандартной структуры, описываемой в большинстве литературных источников, как правило, используются выходные нейроны с линейной функцией активации, что облегчает приведение сигнала к любому числовому диапазону. Опубликованные в последнее время работы наводят на мысль о возможности их замены сигмоидальными нейронами, позволяющими значительно сократить раз-

мерность сети. Так, для сети, предложенной Нарендрой и содержащей линейные выходные нейроны, необходимо большое количество скрытых нейронов, например $K = 10$. Тот же эффект может быть достигнут в сети с сигмоидальным выходным нейроном и всего двумя скрытыми нейронами. Однако следует учитывать, что значения сигнала сигмоидального нейрона ограничены интервалом от −1 до +1. Чтобы обеспечить любой требуемый диапазон значений, на выходе сети добавляется линейный блок, усиливающий сигнал в M раз ($0 < M < \infty$). При грамотном подборе козффициента усиления M подобная сеть демонстрирует такие же хорошие возможности адаптации при значительно меньшем количестве скрытых нейронов.

8.2.5. Результаты компьютерного моделирования

Сеть RMLP повсеместно применяется для моделирования динамических процессов в режиме "онлайн". Типичным примером ее приложения может служить имитация нелинейных динамических объектов, для которых сеть RMLP выступает в роли модели, а алгоритм уточнения весов – в роли процедуры идентификации параметров этой модели. Идентифицированная модель объекта может в последующем использоваться для управления данным объектом. Именно по этой причине сети RMLP наиболее популярны для имитации систем управления машинами, устройствами и динамическими процессами [156]. В настоящем разделе мы обсудим подход к моделированию нелинейных динамических систем, предложенный в работах К. Нарендры [107]. В отличие от работ Нарендры будем использовать сеть с нелинейным выходным нейроном, описываемым сигмоидальной функцией активации. На первый взгляд нелинейность выходного нейрона осложняет проблему идентификации нелинейного объекта (из-за ограниченности выходного сигнала диапазоном (−1, 1), меньшей динамики изменения сигнала, более сложного процесса обучения и т.п.). В действительности она имеет другие достоинства, которые отсутствуют у сети с линейным выходным нейроном: количество скрытых нейронов может быть существенно уменьшено, например, с $K = 20$ в работах Нарендры до $K = 2$ при обсуждаемом подходе; намного сокращается длительность обучения; в начальных фазах обучения и тестирования возникает меньше ошибок, связанных с нулевыми начальными значениями. Ограниченность выходного сигнала легко преодо-левается включением в структуру сети усилителя, масштабирующего значения, изначально нормализованные в интервале (−1, 1).

Сеть обучалась с использованием программы RMLP, приспособленной для обучения в режиме "онлайн". Обучение было основано на адаптивной идентификации нелинейных динамических объектов. Объект, описываемый известной нелинейной функцией, генерировал последовательность заданных сигналов $d(n)$ в качестве реакции на возбуждение в виде векторов $\boldsymbol{x}$, формируемых случайным образом. Сеть RMLP со структурой, изображенной

на рис. 8.1, использовалась в качестве модели этого объекта. В результате сравнения выходного сигнала этой модели $y(n)$ с заданным сигналом $d(n$ рассчитывалось значение погрешности $\varepsilon(n)$:

$$\varepsilon(n) = y(n) - d(n), \tag{8.17}$$

управляющей процессом уточнения параметров нейронной сети. На рис. 8.2 показан способ включения сети при проведении экспериментов. Символом M обозначен постоянный коэффициент усиления модуля, масштабирующего выходной сигнал сети таким образом, чтобы его динамический уровень лежал в том же диапазоне, что и уровень заданного сигнала $d(n)$.

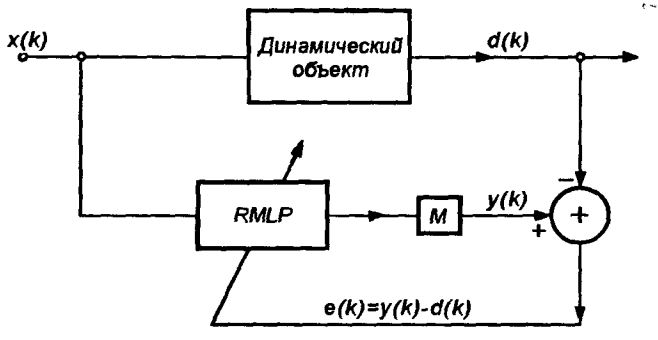

Рис. 8.2. Схема включения сети RMLP при решении задачи идентификации

Во всех численных экспериментах использовалась сеть со структурой 2-2-1. Вход системы состоял из одного входного узла $x(n)$ и одного контекстного узла, вырабатывавшего копию задержанного на один такт выходного сигнала. Скрытый слой состоял всего из двух нейронов, а выходной слой – из одного нейрона. При реализации процесса обучения выполнялся описанный выше адаптивный подбор коэффициента обучения η. Уточнение весов проводилось в двух режимах.

В первом режиме предъявление каждой новой обучающей выборки сопровождалось однократным уточнением значений всех весов сети и переходом к следующей выборке. Этот режим будем называть в дальнейшем *однократной адаптацией*.

Во втором режиме каждая обучающая выборка вызывала многократное уточнение весов сети (предъявление обучающей выборки на вход сети сопровождалось изменением выходного сигнала, после этого уточнялись значения весов; повторная подача на вход сети сигнала обратного распространения ошибок при неизменной обучающей выборке приводила к очередному изменению выходного сигнала с соответствующим уточнением весов и т.д.). Этот режим обучения сети будем называть *многократной адаптацией*. Каждый процесс обучения сети начинался со случайных значений весов, равномерно распределенных в заданном интервале. В проводимых экспериментах это был интервал (–0,1, 0,1).

Первый численный эксперимент был связан с динамической системой, предложенной в работе [107] и описываемой выражением

$$y_{k+1} = 0{,}3y_k + 0{,}6y_{k-1} + 0{,}6\sin(\pi u_k) + 0{,}3\sin(3\pi u_k) + 0{,}1\sin(5\pi u_k). \quad (8.18)$$

Дискретный входной сигнал задавался функцией $u_k = \sin\left(\frac{2\pi k}{250}\right)$.

На рис. 8.3 представлена форма заданных сигналов, генерируемых динамической системой, определенной выражением (8.18). Из этого уравнения следует, что выходной сигнал системы будет ограничен при условии, что на входной сигнал также наложены ограничения. В экспериментах применялись обе методики уточнения весов – как однократной, так и многократной адаптации [156].

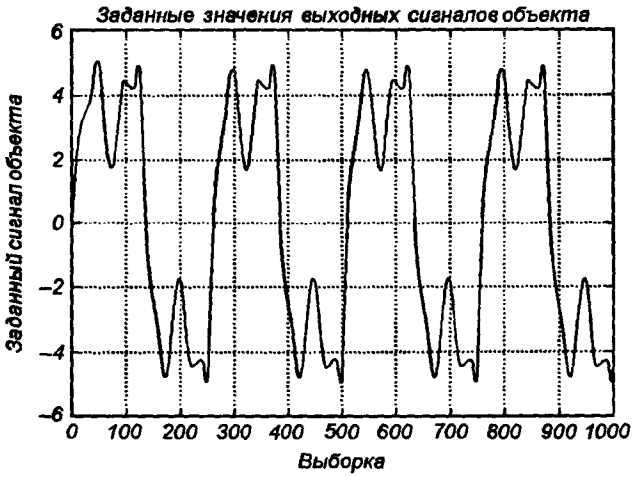

Рис. 8.3. Заданные сигналы динамического объекта, определенного выражением (8.18).

При использовании первой методики значения весов уточнялись после предъявления каждой обучающей выборки по алгоритму наискорейшего спуска с постоянным коэффициентом обучения η = 0,085 (адаптивный подбор коэффициента обучения при однократной адаптации не имеет смысла). Результаты процесса обучения в виде изменений погрешности $\varepsilon(n) = y(n) - d(n)$ представлены на рис. 8.4. Они свидетельствуют, что погрешность обучения (уже после 20 циклов) быстро уменьшилась до несущественной величины, которая воспринималась только благодаря высокой точности идентификации системы.

Согласно второй методике значения весов уточнялись трижды на протяжении каждого цикла с применением адаптивного коэффициента обучения η и значений коэффициентов k_d = 0,7 и k_w = 1,03. График погрешности обучения для этого случая представлен на рис. 8.5.

На графике видно, что погрешность, особенно в первой фазе обучения, оказалась меньше, а процесс адаптации модели к реакциям объекта протекал быстрее, особенно в начале обучения. Следует подчеркнуть, что и в первом,

и во втором случае остаточная погрешность обучения стабилизировалась на определенном, достаточно низком уровне, являясь движущей силой механизма адаптации параметров модели.

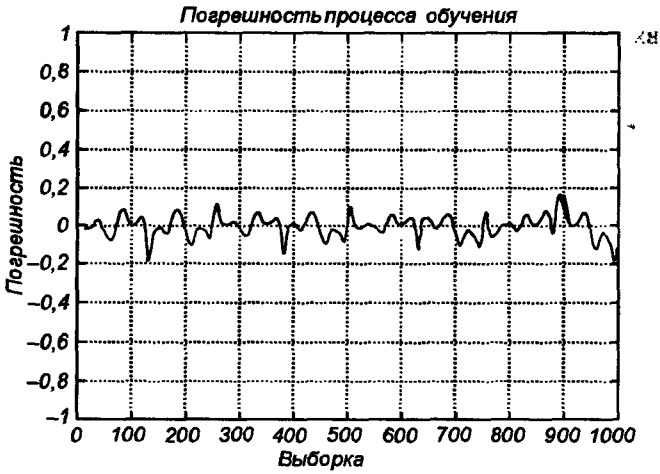

Рис. 8.4. График обучения сети RMLP при однократной адаптации весов в каждом цикле для динамического объекта из первого эксперимента

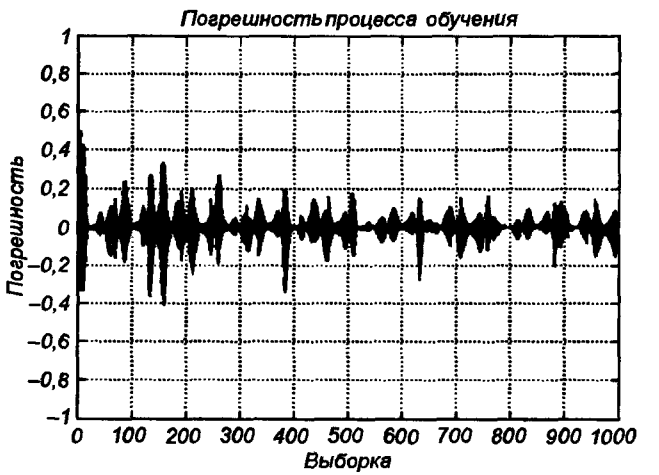

Рис. 8.5. График обучения сети RMLP при трехкратной адаптации весов в каждом цикле для динамического объекта из первого эксперимента

Во втором эксперименте исследовалась нелинейная динамическая система, описываемая следующей зависимостью:

$$y_{k+1} = \frac{y_k y_{k-1}(y_k + 2{,}5)}{1 + y_k^2 + y_{k-1}^2} + u_k \qquad (8.19)$$

со входным сигналом $u_k = \sin\left(\frac{2\pi k}{250}\right)$. Это динамический объект с нелинейностью измерительного характера, неудобный для численного моделирования.

На рис. 8.6 приведен график изменения выходного сигнала объекта (заданных значений), описываемого выражением (8.19). Результаты обучения в виде

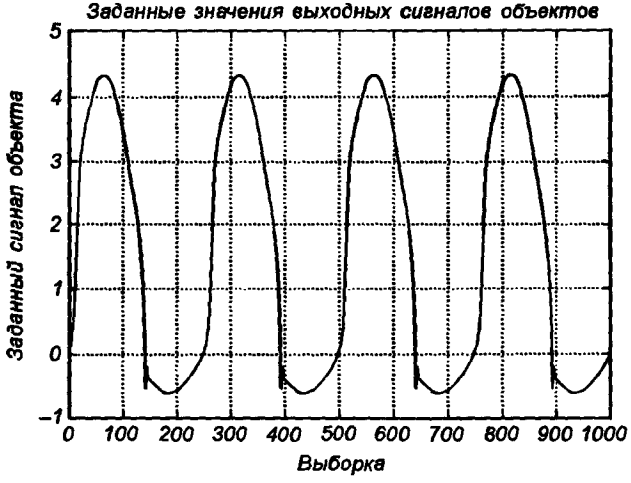

Рис. 8.6. Заданные сигналы динамического объекта, определенного выражением (8.19).

графика погрешности при однократной адаптации весов представлены на рис. 8.7. Погрешность обучения, принимавшая в начале процесса значения 4 – 5-го порядка, очень быстро (примерно за пять циклов) сократилась до остаточной величины, уменьшающейся в ходе обучения.

График погрешности обучения, соответствующий трехкратной адаптации весов, изображен на рис. 8.8. Погрешность обучения при трехкратной адаптации

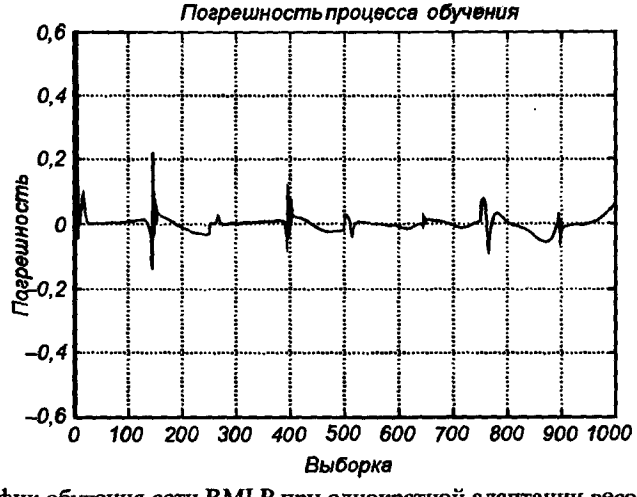

Рис. 8.7. График обучения сети RMLP при однократной адаптации весов в каждом цикле для динамического объекта из второго эксперимента

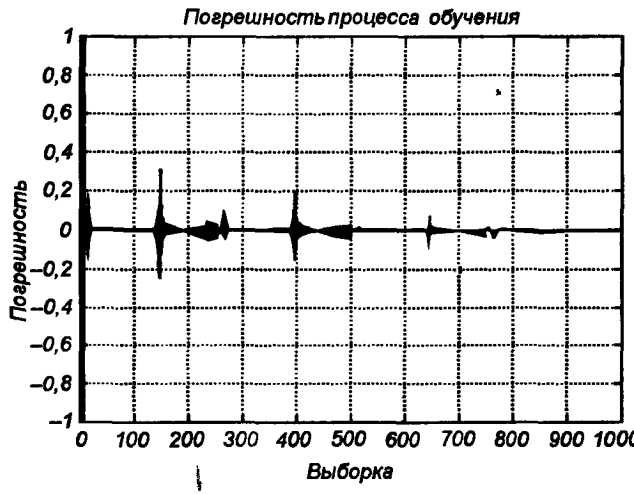

Рис. 8.8. График обучения сети RMLP при трехкратной адаптации весов в каждом цикле для динамического объекта из второго эксперимента

намного меньше, чем при однократной, а процесс обучения оказывается более коротким и приводит к сокращению величины погрешности до уровня $10^{-2} - 10^{-3}$.

8.3. Рекуррентная сеть Эльмана

8.3.1. Структура сети

Рекуррентная сеть Эльмана характеризуется частичной рекуррентностью в форме обратной связи между скрытым и входным слоем [46, 114], реализуемой с помощью единичных элементов запаздывания z^{-1}. Обобщенная структура этой сети представлена на рис. 8.9. Каждый скрытый нейрон имеет свой аналог в контекстном слое, образующем совместно с внешними входами сети входной слой. Выходной слой состоит из нейронов, однонаправленно связанных только с нейронами скрытого слоя, подобно сети MLP. Обозначим внутренний вектор возбуждения сети x (в его состав входит также единичный сигнал поляризации), состояния скрытых нейронов – $v \in R^K$, а выходные сигналы сети – $y \in R^M$. При таких обозначениях входной вектор сети в момент t имеет форму

$$x(k) = [x_0(k), x_1(k), ..., x_N(k), v_1(k-1), v_2(k-1), ..., v_K(k-1)] . \qquad (8.20)$$

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя – $w_{ij}^{(2)}$. Если взвешенную сумму i-го нейрона скрытого

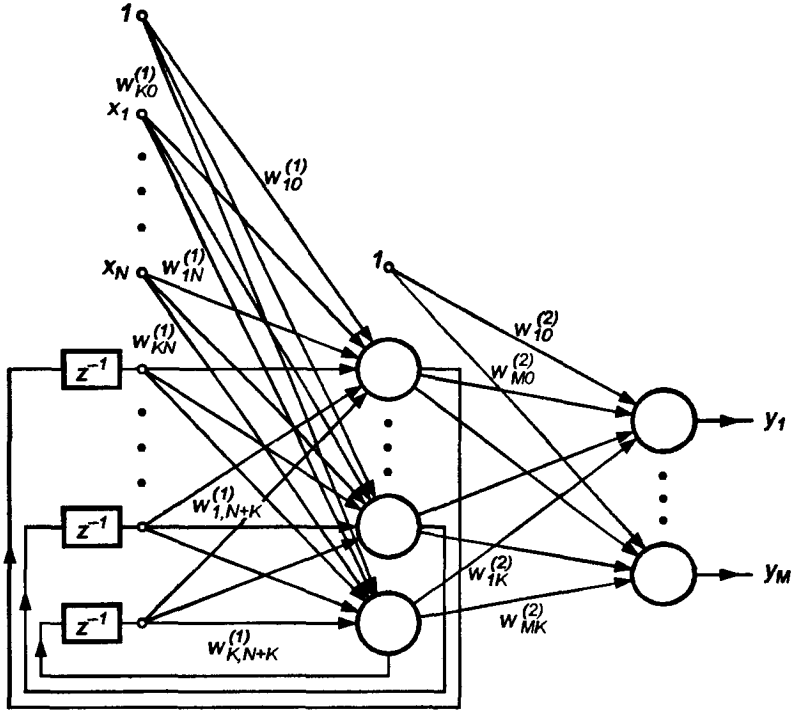

Рис. 8.9. Структура сети Эльмана

слоя обозначить u_i, а его выходной сигнал – v_i, то

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(k) \,, \tag{8.21}$$

$$v_i(k) = f_1(u_i(k)) \,. \tag{8.22}$$

Веса $w_{ij}^{(1)}$ образуют матрицу $\mathbf{W}^{(1)}$ синаптических связей скрытого слоя, а $f_1(u_i)$ – функция активации i-го нейрона этого слоя. Аналогично можно обозначить взвешенную сумму i-го нейрона выходного слоя g_i, а соответствующий ему выходной сигнал сети – y_i. Эти сигналы описываются формулами

$$g_i(k) = \sum_{j=0}^{K} w_{ij}^{(2)} v_j(k) \,, \tag{8.23}$$

$$y_i(k) = f_2(g_i(k)) \,. \tag{8.24}$$

В свою очередь, веса $w_{ij}^{(2)}$ образуют матрицу $\mathbf{W}^{(2)}$, описывающую синаптические связи нейронов выходного слоя, а $f_2(g_i)$ – функция активации i-го нейрона выходного слоя.

8.3.2. Алгоритм обучения сети Эльмана

Для обучения сети Эльмана будем использовать градиентный метод наискорейшего спуска, основанный на работах Р. Вильямса и Д. Зипсера [171]. Для этого метода необходимо задать формулы, позволяющие рассчитывать градиент целевой функции в текущий момент времени. Целевая функция в момент t определяется как сумма квадратов разностей между значениями выходных сигналов сети и их ожидаемыми значениями для всех M выходных нейронов:

$$E(k)=\frac{1}{2}\sum_{i=1}^{M}[y_i(k)-d_i(k)]^2=\frac{1}{2}\sum_{i=1}^{M}e_i(k)^2. \qquad (8.25)$$

При дифференцировании целевой функции относительно весов выходного слоя получаем:

$$\nabla^{(2)}_{\alpha\beta}E(k)=\frac{\partial E(k)}{\partial w^{(2)}_{\alpha\beta}}=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\frac{dg_i(k)}{dw^{(2)}_{\alpha\beta}}. \qquad (8.26)$$

С учетом выражений (8.23) и (8.24) зависимость (8.26) можно представить в виде

$$\nabla^{(2)}_{\alpha\beta}E(k)=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\sum_{j=0}^{K}\frac{d(w^{(2)}_{ij}v_j(k))}{dw^{(2)}_{\alpha\beta}}= \qquad (8.27)$$

$$=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\sum_{j=0}^{K}\left(\frac{dv_j}{dw^{(2)}_{\alpha\beta}}w^{(2)}_{ij}+\frac{dw^{(2)}_{ij}}{dw^{(2)}_{\alpha\beta}}v_j(k)\right).$$

Связи между скрытым и выходным слоем однонаправленные (см. рис. 8.9), поэтому $\frac{dv_j}{dw^{(2)}_{\alpha\beta}}=0$. С учетом этого факта

$$\nabla^{(2)}_{\alpha\beta}E(k)=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\sum_{j=0}^{K}\frac{dw^{(2)}_{ij}}{dw^{(2)}_{\alpha\beta}}v_j(k)=e_\alpha(k)\frac{df_2(g_\alpha(k))}{dg_\alpha(k)}v_\beta(k). \qquad (8.28)$$

При использовании метода наискорейшего спуска веса $w^{(2)}_{\alpha,\beta}$ уточняются по формуле $w^{(2)}_{\alpha,\beta}(k+1)=w^{(2)}_{\alpha,\beta}(k)+\Delta w^{(2)}_{\alpha,\beta}$, где

$$\Delta w^{(2)}_{\alpha,\beta}=-\eta\nabla^{(2)}_{\alpha,\beta}E(k). \qquad (8.29)$$

Простое сравнение с персептронной сетью показывает, что веса выходного слоя сети Эльмана подвергаются точно такой же адаптации, как и веса выходных нейронов персептрона.

Формулы уточнения весов скрытого слоя сети Эльмана более сложны из-за наличия обратных связей между скрытым и контекстным слоями. Расчет компонентов вектора градиента целевой функции относительно весов скрытого слоя требует выполнения значительно большего количества математических операций.

Зависимости,

$$\nabla^{(1)}_{\alpha\beta}E(k)=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\sum_{j=1}^{K}\frac{d(v_j w^{(2)}_{ij})}{dw^{(1)}_{\alpha\beta}}=$$

$$=\sum_{i=1}^{M}e_i(k)\frac{df_2(g_i(k))}{dg_i(k)}\sum_{j=1}^{K}\frac{dv_j(k)}{dw^{(1)}_{\alpha\beta}}w^{(2)}_{ij}\ . \qquad (8.30)$$

С учетом зависимости (8.22) получаем:

$$\frac{dv_j(k)}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_j)}{du_j}\sum_{k=0}^{N+K}\frac{d(x_k(k)w^{(1)}_{jk})}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_j)}{du_j}\left[\delta_{j\alpha}x_\beta+\sum_{k=0}^{N+K}\frac{dx_k(k)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{jk}\right]. \qquad (8.31)$$

Из определения входного вектора x (формула (8.20)) в момент t следует, что

$$\frac{dv_j(k)}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_j)}{du_j}\left[\delta_{j\alpha}x_\beta+\sum_{k=N+1}^{N+K}\frac{dv_{(k-N)}(k-1)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{jk}\right]=$$

$$=\frac{df_1(u_j)}{du_j}\left[\delta_{j\alpha}x_\beta+\sum_{k=1}^{K}\frac{dv_k(k-1)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{j,k+N}\right]. \qquad (8.32)$$

Это выражение позволяет рассчитать производные целевой функции относительно весов скрытого слоя в момент t. Следует отметить, что это рекуррентная формула, определяющая производную в момент t в зависимости от ее значения в предыдущий момент $t-1$. Начальные значения производных в исходный момент $t = 0$ считаются нулевыми: $\frac{dv_1(0)}{dw^{(1)}_{\alpha\beta}}=\frac{dv_2(0)}{dw^{(1)}_{\alpha\beta}}=...=\frac{dv_K(0)}{dw^{(1)}_{\alpha\beta}}=0$. Таким образом, алгоритм обучения сети Эльмана можно представить в следующем виде.

1. Начать обучение с присвоения весам случайных начальных значений, имеющих, как правило, равномерное распределение в определенном интервале (например, между –1 и 1).
2. Для очередного момента t ($t = 0, 1, 2, ...$) определить состояние всех нейронов сети (сигналы v_i и y_i). На этой основе можно сформировать входной вектор $x(k)$ для произвольного момента t.
3. Определить вектор погрешности обучения $e(n)$ для нейронов выходного слоя как разность между фактическим и ожидаемыми значениями сигналов выходных нейронов.
4. Сформировать вектор градиента целевой функции относительно весов выходного и скрытого слоя с использованием формул (8.28), (8.30) и (8.32).
5. Уточнить значения весов сети согласно правилам метода наискорейшего спуска:
   - для нейронов выходного слоя сети

$$w^{(2)}_{\alpha,\beta}(k)=w^{(2)}_{\alpha,\beta}(k-1)-\eta\nabla^{(2)}_{\alpha,\beta}E(k); \qquad (8.33)$$

- для нейронов скрытого слоя сети

$$w_{\alpha,\beta}^{(1)}(k) = w_{\alpha,\beta}^{(1)}(k-1) - \eta \nabla_{\alpha,\beta}^{(1)} E(k). \qquad (8.34)$$

После уточнения значений весов вернуться к пункту 2 алгоритма для расчетов в очередной момент t.

Для упрощения вычислений при выводе формул (8.33) и (8.34) была предложена онлайн-версия алгоритма обучения, согласно которой значения весов уточнялись после предъявления каждой обучающей пары (x, d). Обычный офф-лайн-алгоритм несколько более эффективен, поскольку при его выполнении значения весов уточняются только после предъявления всех обучающих пар. Реализация такой версии алгоритма аналогична рассмотренной выше с той разницей, что в ней суммируются компоненты градиента последовательно предъявляемых обучающих пар, а процесс изменения весов выполняется в каждом цикле только один раз.

Представленный алгоритм считается нелокальным, поскольку уточнение каждого отдельного веса требует знания значений всех остальных весов сети и сигналов конкретных нейронов. Его требования к размерам памяти компьютера дополнительно повышаются в связи с необходимостью хранить все значения $\frac{dv_j(k)}{dw_{\alpha\beta}^{(1)}}$.

8.3.3. Обучение с учетом момента

Практические реализации алгоритма обучения сети Эльмана строятся на методе наискорейшего спуска, усиленном моментом. Это значительно повышает эффективность обучения и вероятность достижения глобального минимума целевой функции. При использовании такого подхода уточнение вектора весов сети в момент t выполняется в соответствии с формулой

$$\Delta w(k) = -\eta \nabla E(k) + \alpha(k)\, \Delta w(k-1), \qquad (8.35)$$

где $\alpha(k)$ - это коэффициент момента, выбираемый из интервала (0, 1). Первое слагаемое этого выражения соответствует обычному методу обучения, тогда как второе, учитывающее фактор момента, отражает последнее изменение весов и не зависит от фактического значения градиента. Чем больше величина α, тем большее влияние на подбор весов оказывает слагаемое момента. Его значение существенно возрастает на плоских участках целевой функции и около локального минимума, где значение градиента близко к нулю.

В окрестности локального минимума не связанный с градиентом фактор момента может вызвать изменение весов, ведущее к росту значения целевой функции и к выходу из зоны притяжения этого минимума с возобновлением поиска области, в которой целевая функция имеет меньшее значение. Фактор момента не может доминировать при уточнении весов, поскольку в этой ситуации

процесс обучения и, следовательно, поиска минимума никогда бы не завершился. Обычно для управления процессом обучения вводится понятие допустимого прироста погрешности, например, 3%. В таком случае, если в k-й итерации значение целевой функции удовлетворяет зависимости $E(k)<1{,}03E(k-1)$, то шаг принимается и значения весов уточняются, в противном случае фактор момента игнорируется, и приннмается $\alpha(k)=0$. Выбор оптимального значения коэффициента момента – это непростая задача. Для ее решения требуется провести значительное количество численных экспериментов, цель которых состоит в адаптации значения этого коэффициента к решаемой проблеме. Обычно удовлетворительным считается субоптимальное значение, которое обеспечивает достижение (хотя, возможно, и не самое быстрое) хороших показателей обучения.

8.3.4. Пример компьютерного моделирования сети Эльмана

Сеть Эльмана имеет рекуррентную структуру с обратной связью между скрытым и входным слоями. С учетом непосредственного влияния сигналов в момент $(n-1)$ на ее поведение в момент n такая сеть естественным образом предрасположена к моделированию временных рядов.

В практической реализации алгоритма обучения сети Эльмана (программа Elman) использовался способ адаптации весов типа "оффлайн" и применялся метод наискорейшего спуска с учетом момента со значениями коэффициента момента в интервале от 0 до 1 (значение по умолчанию равно 0,95). Для ускорения обучения использовался адаптивный коэффициент обучения η, подстраиваемый к фактическим изменениям значений целевой функции в процессе обучения.

В настоящем подразделе будут представлены результаты применения сети Эльмана для распознавания амплитуд двух синусоидальных сигналов по каждой реализации сигнала, поданного на вход сети. Задача состояла в предсказании амплитуды сигнала на основе текущего значения входного сигнала и запомненных значений из предыдущего временного цикла. В итоге сеть должна сформировать на своем выходе сигнал, соответствующий амплитуде входного сигнала.

В численном эксперименте входной сигнал представлял собой последовательность из 80 синусоидальных сигналов с двумя различными амплитудами, равными 1 и 2 соответственно, описываеме. зависимостью

$$x_k=\begin{cases}\sin(k) & \text{для } 0<k\le 20\\ 2\sin(k-20) & \text{для } 20<k\le 40\\ \sin(k-40) & \text{для } 40<k\le 60\\ 2\sin(k-60) & \text{для } 60<k\le 80\end{cases}. \qquad (8.36)$$

Для решения задачн была построена сеть со структурой 11–10–1. Вход сети образован одним истинным входным узлом и 10 контекстными узлами, так как

скрытый слой состоит из 10 нейронов. Каждый нейрон скрытого слоя характеризуется сигмоидальной (биполярной) функцией активации, а выходной нейрон является линейным. На рис. 8.10 представлены сигналы: ожидаемый (сплошная линия) и фактически сгенерированный сетью (пунктирная линия) после 1000 циклов обучения (верхний график), а также график погрешности распознавания амплитуды конкретных выборок. Анализ графиков свидетельствует,

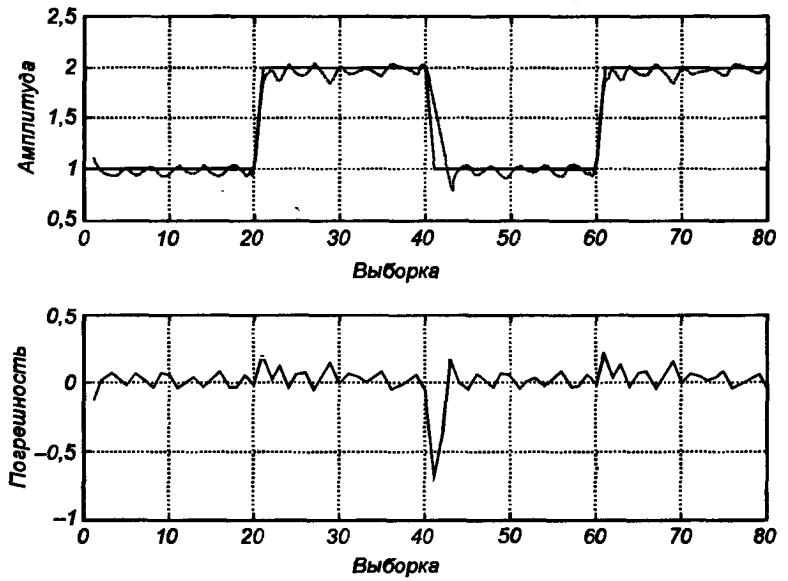

Рис. 8.10. Результаты обучения сети Эльмана распознаванию амплитуды двух сигналов

что эффективность отслеживания сетью входного сигнала может считаться удовлетворительной. Среднее значение модуля погрешности за время проведения эксперимента по результатам обучения составило 0,069. Для проверки корректности функционирования сети в случаях генерации последовательностей с другими амплитудами, с зашумленными сигналами, а также при потерях отдельных фрагментов данных и обработке сигналов с частотой, отличающейся от использованной для обучения, была организована серия обобщающих тестов натренированной сети.

В качестве зашумленных сигналов использовалась последовательность данных, описываемых зависимостью $x'_k = x_k + s$, где s – это шум из интервала $(-N, N)$, а x_k – сигнал, определенный выражением (8.36). На рис. 8.11 приведены результаты тестирования сети для $N = 0{,}1$ и $N = 0{,}5$. Оказалось, что сеть неплохо "справляется" с сигналами, имеющими небольшую зашумленность, например, $N = 0{,}1$. Однако ее эффективность в распознавании амплитуды сигнала, сильно искаженного шумом (например, $N=0{,}5$), значительно ниже, хотя и в этом случае наблюдается отслеживание изменений амплитуды входных сигналов.

Для проверки работы сети при другой амплитуде синусоидальных сигналов натренированная ранее сеть тестировалась на последовательности сигналов,

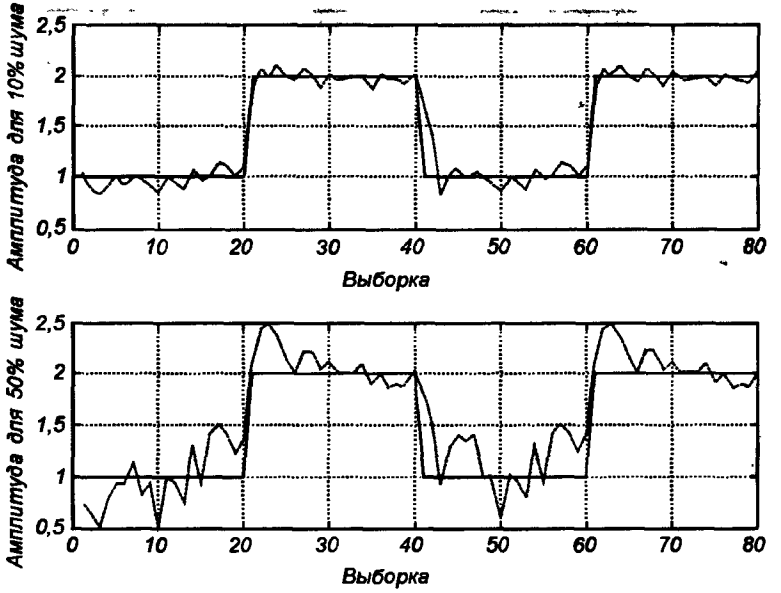

Рис. 8.11. Иллюстрация влияния шума на распознавание сетью Эльмана амплитуды двух сигналов

описываемых зависимостью

$$x_k = \begin{cases} 1{,}2\sin(k) & \text{для} \quad 0 < k \le 20 \\ 1{,}6\sin(k-20) & \text{для} \quad 20 < k \le 40 \\ 1{,}2\sin(k-40) & \text{для} \quad 40 < k \le 60 \\ 1{,}6\sin(k-60) & \text{для} \quad 60 < k \le 80 \end{cases} \tag{8.37}$$

На рис. 8.12 представлены результаты функционирования сети в этом эксперименте. Поскольку обучение проводилось только на амплитудах со значениями 1 и 2, сеть сохранила определенные способности к обобщению, позволившие ей отслеживать значения других сигналов, хотя и со значительно большей погрешностью. Для проверки устойчивости натренированной сети к изменениям частоты сигнала были проведены очередные тесты с данными, имеющими характеристики

$$x_k = \begin{cases} \sin(Nk) & \text{для} \quad 0 < k \le 20 \\ 2\sin(N(k-20)) & \text{для} \quad 20 < k \le 40 \\ \sin(N(k-40)) & \text{для} \quad 40 < k \le 60 \\ 2\sin(N(k-60)) & \text{для} \quad 60 < k \le 80 \end{cases} \tag{8.38}$$

при $N = 0{,}8$ и $N = 1{,}5$. Результаты моделирования в виде соответствующих графиков амплитуд: ожидаемой (сплошная линия) и распознанной сетью (пунк-

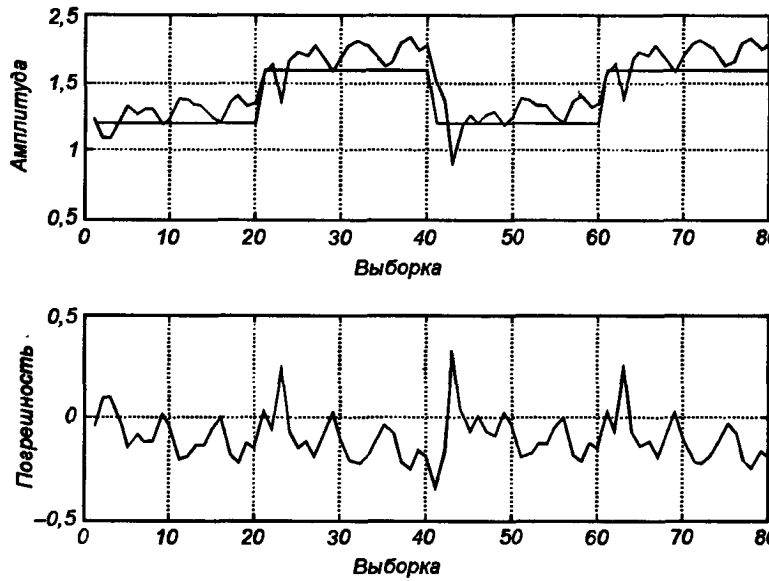

Рис. 8.12. Результаты тестирования сети Эльмана на данных с амплитудой, измененной по сравнению с обучающими сигналами

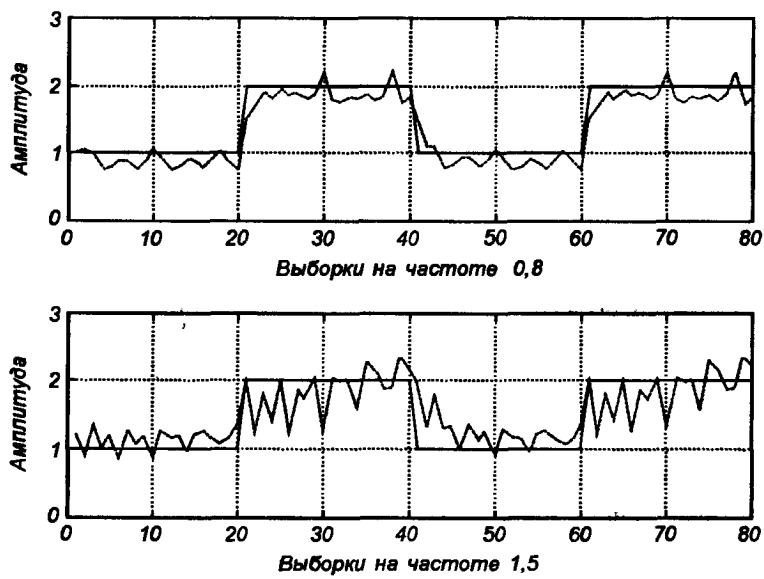

Рис. 8.13. Иллюстрация влияния изменения частоты на распознавание сетью Эльмана амплитуды двух сигналов

тирная линия) представлены на рис. 8.13. Анализ этих графиков позволяет сделать вывод, что сеть корректно распознает амплитуды сигналов с частотой, отличающейся от частоты обучающих данных даже при изменении частоты на 20%. Неудовлетворительные результаты распознавания наблюдаются при значи-

тельных отклонениях частоты от ожидаемой величины (результаты при 50%-ном отклонении частоты приведены на нижнем графике рис. 8.13). Во всех описываемых экспериментах структура данных была симметричной: 20 выборок имели одну амплитуду, а 20 – другую, причем их последовательности повторялись. Такое чередование тестовых сигналов наследовало структуру данных, использованную на стадии обучения сети. Для проверки способности сети к обобщению были также проведены тесты с данными другой длины и с иной пропорцией распределения выборок. В первом тесте были удалены 10 первых выборок с амплитудой 1, а во втором – еще дополнительно 10 первых выборок с амплитудой 2. Результаты этих тестов представлены на рис. 8.14. Можно

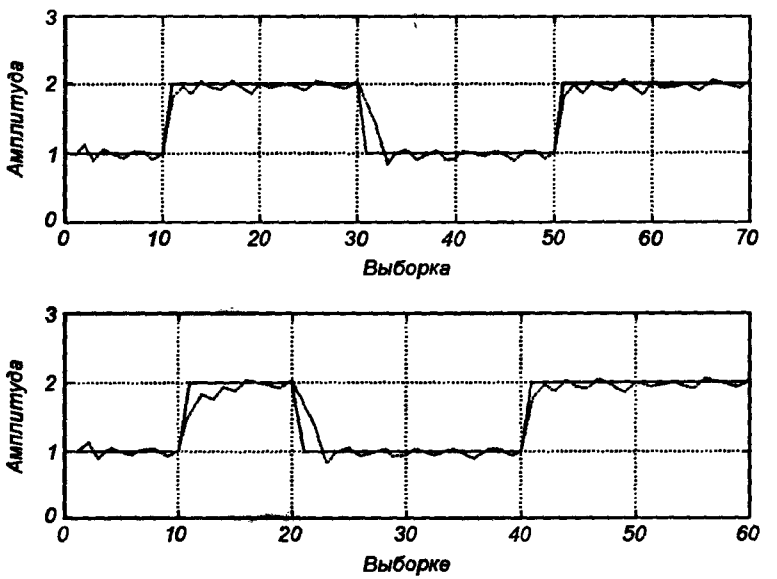

Рис. 8.14. Результаты тестирования сети Эльмана на данных с несимметричной структурой

отметить, что удаление части сигналов не оказало существенного воздействия на качество отклика. Система функционировала корректно, демонстрируя удовлетворительное качество распознавания амплитуды.

8.4. Сеть RTRN

8.4.1. Структура сети и алгоритм обучения

Среди рекуррентных сетей особого внимания заслуживает сеть типа RTRN (англ.: *Real Time Recurrent Network*), предложенная Р. Вильямсом и Д. Зипсером в работе [171] и предназначенная для обработки сигналов в реальном времени. Обобщенная структура сети представлена на рис. 8.15. Сеть содержит N входных

узлов, K скрытых нейронов и K соответствующих им узлов контекстного слоя. Из K скрытых нейронов только M составляют выход сети. Обозначим взвешенную сумму i-го нейрона скрытого слоя u_i, а выход этого нейрона – y_i. Вектор $\boldsymbol{x}(k)$ и смещенный (задержанный) на один цикл вектор $\boldsymbol{y}(k-1)$ образуют расширенный вектор активации $\boldsymbol{x}(k)$, возбуждающий нейроны сети:

$$\boldsymbol{x}(k) = [1, x_1(k), x_2(k), \ldots, x_N(k), y_1(k-1), \ldots, y_K(k-1)]^T. \tag{8.39}$$

После описания входного вектора сети в момент t можно определить состояние всех нейронов согласно зависимостям:

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij} x_j(k), \tag{8.40}$$

$$y_i(k) = f(u_i(k)), \tag{8.41}$$

причем $f(\ )$ обозначает непрерывную функцию активации нейрона (как правило, сигмоидальную). На рис. 8.15 видно, что сеть RTRN представляет собой частный

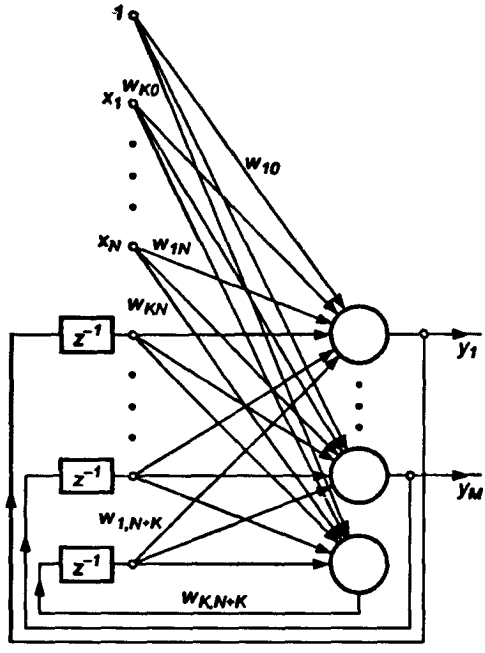

Рис. 8.15. Структура сети RTRN

случай сети Эльмана, в которой веса выходного слоя постоянны и равны дельте Кронекера, т.е. $w_{ij} = \delta_{ij} = 1$ для $i = j$ или 0 для $i \neq j$. В этом случае можно применять алгоритм обучения Вильямса–Зипсера, представленный в одном из преды-дущих подразделов, поскольку обучение сети связано с уточнением весов единственного существующего слоя. При модификации соответствующих формул расчета вектора градиента, выведенных для сети Эльмана (при фик-

зации значений весов выходного слоя), получим следующий алгоритм обучения сети RTRN, называемый алгоритмом Вильямса–Зипсера [171].

1. Выбрать случайные начальные значения весов сети, составляющих матрицу W и равномерно распределенных в заданном интервале (обычно в диапазоне от −1 до 1).
2. Рассчитать состояние всех K нейронов для очередного момента t ($t = 0, 1, 2, ...$) с использованием зависимостей (8.40) и (8.41). На этой основе можно определить расширенный входной вектор $x(k)$, возбуждающий нейроны в момент t.
3. Рассчитать значения $\frac{\partial y_j(k)}{\partial w_{\alpha\beta}}$ в соответствии с формулой

$$\frac{\mathrm{d}y_j(k)}{\mathrm{d}w_{\alpha\beta}} = \frac{\mathrm{d}f_1(u_j)}{\mathrm{d}u_j}\left[\delta_{j\alpha}x_\beta + \sum_{k=1}^{K}\frac{\mathrm{d}y_k(k-1)}{\mathrm{d}w_{\alpha\beta}}w_{j,k+N}\right]. \qquad (8.42)$$

4. Уточнить значения весов по алгоритму наискорейшего спуска согласно формуле

$$w_{\alpha\beta}(k+1) = w_{\alpha\beta}(k) - \eta\sum_{j=1}^{M}[y_j(k) - d_j(k)]\frac{\mathrm{d}y_j(k)}{\mathrm{d}w_{\alpha\beta}}, \qquad (8.43)$$

для $\alpha = 1, 2, ..., K$ и $\beta = 0, 1, 2, ..., N + K$. Циклы (2 – 4) повторять вплоть до стабилизации значений всех весов сети.

Приведенный алгоритм обучения сети RTRN был реализован в программе, названной также *RTRN*. В ней процесс обучения оптимизирован за счет применения адаптивного коэффициента обучения и обучения с учетом момента, аналогично тому, как это делалось в программе *Elman*.

8.4.2. Результаты вычислительных экспериментов

Как мы уже отмечали, сеть RTRN – это частный случай сети Эльмана, в которой веса выходного слоя имеют бинарные значения, т.е. $w_{ij} = \delta_{ij} = 1$ для $i = j$ или 0 для $i \neq j$. Для изучения возможностей ее практического применения исследовалось функционирование этой сети в качестве системы линейной идентификации динамического объекта, описываемого матричным уравнением состояния [156]:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \mathbf{A}\boldsymbol{x}(k) + \mathbf{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \mathbf{C}\boldsymbol{x}(k), \end{aligned} \qquad (8.44)$$

где

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}.$$

Вектор $\boldsymbol{x}$ называется вектором состояния, $\boldsymbol{y}$ – выходным вектором, а $\boldsymbol{u}$ – вектором возбуждения. Квадратная матрица состояния A имеет размерность $K \times K$, матрица

B – размерность $K \times N$, а матрица **C** – размерность $M \times K$. В этом приложении сеть абсолютно линейна (в том числе и скрытые нейроны). При сопоставлении структуры сети (рис. 8.15) со схемой распространения сигналов, соответствующей зависимости (8.44), видно, что матрицу A образуют веса w_{ij} для $i = 1, ..., K$ и $j = N + 1, ..., N + K$, матрицу **B** – веса w_{ij} для $i = 1, ..., K$ и $j = 1, ..., N$, а матрицу **C** – единичные или нулевые веса выходного слоя (сеть RTRN состоит из одного слоя, поэтому ее выход могут определять только переменные состояния, а в матрице **C** ненулевыми, т.е. единичными, могут быть лишь диагональные элементы).

Результаты вычислительных экспериментов можно представить на примере линейной динамической системы третьего порядка с одним входом и двумя выходами, описываемой матрицами **A**, **B** и **C** вида

$$\mathbf{A} = \begin{bmatrix} 0{,}3 & 0{,}1 & 0{,}1 \\ 0 & 0{,}4 & 0{,}3 \\ 0{,}5 & 0{,}1 & 0{,}5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0{,}5 \\ 0{,}2 \\ 0{,}1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Для определенных таким образом матриц состояния генерировались обучающие данные в форме последовательности пар "вход – выход" (рис. 8.16), где входной сигнал $u(k)$ был случайным, а заданный вектор $\boldsymbol{d}$ составлялся из обоих компонентов выходного сигнала и рассчитывался по формуле $\boldsymbol{d} = \mathbf{C}(s\mathbf{1} - \mathbf{A})^{-1}\mathbf{B}u$. Таким образом, реализованная нейронная сеть RTRN имела структуру 4–3–2 (4 входных узла: один для внешнего сигнала $u(k)$ и три

(11.8)

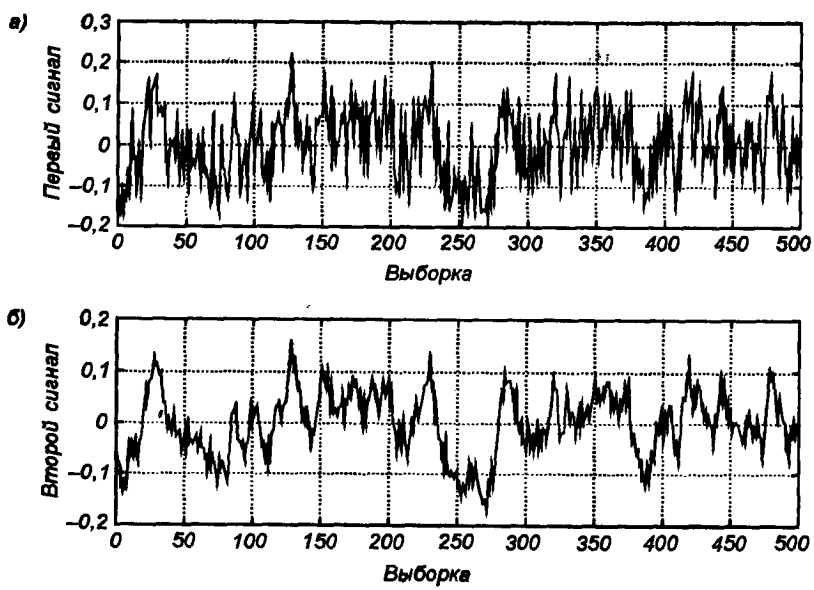

Рис. 8.16. Графики обучающих данных (заданные значения) для сети RTRN для примера идентификации матрицы переменных состояния:
а) сигналы первого контура; *б*) сигналы второго контура

контекстных узла, 3 скрытых нейрона и 2 выходных нейрона с априори известными весами $w_{ij} = \delta_{ij}$). Сеть RTRN обучалась на множестве из 500 пар данных. Распределение входных данных, использованных для обучения сети, представлено на рис. 8.16. Верхний график представляет заданные значения первого контура, а нижний график – данные второго контура.

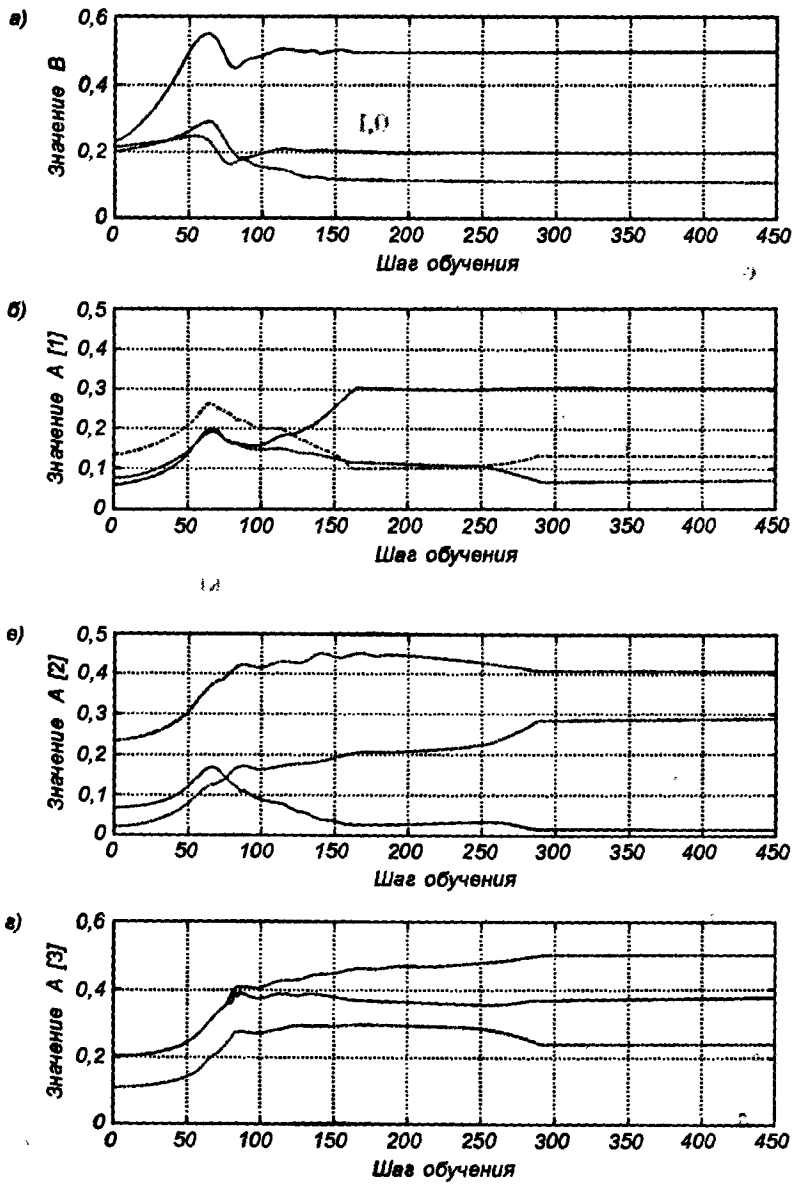

Рис. 8.17. Графики изменения значений отдельных элементов матриц **A** и **B** в процессе обучения сети RTRN

На рис. 8.17 показан процесс адаптации трех весов сети, составляющих матрицу **B** (рис. 8.17*а*) и девяти весов, составляющих матрицу **A** (рис. 8.17*б*, *в*, *г*). Для матрицы **A** каждый график относится к трем весам соответствующей строки матрицы. Достигнутое состояние с фиксированными значениями весов свидетельствует об успешной адаптации сети в качестве модели динамической системы с заданной временной характеристикой, определенной обучающими данными. Идентифицированные значения матриц **A** и **B** имеют вид:

$$\hat{\mathbf{A}} = \begin{bmatrix} 0{,}305 & 0{,}0675 & 0{,}13 \\ 0{,}012 & 0{,}286 & 0{,}404 \\ 0{,}373 & 0{,}236 & 0{,}506 \end{bmatrix}, \quad \hat{\mathbf{B}} = \begin{bmatrix} 0{,}5 \\ 0{,}2 \\ 0{,}112 \end{bmatrix}$$

Они отличаются от оригинала, несмотря на то, что характеризуются такими же временными реакциями (одинаковые временные характеристики может иметь множество систем различной структуры – система идентификации имеет много решений). После обучения сеть подверглась тестированию на данных, основанных на синусоидальном возбуждающем сигнале. Проводимый тест должен был проверить, насколько хорошо сеть отражает свойства динамического объекта, в качестве модели которого она используется. Несмотря на то, что полученные матрицы **A** и **B** отличались от значений, при которых генерировались обучающие данные, идентифицированная система демонстрировала такие же свойства, что и оригинальный объект. Временные реакции как

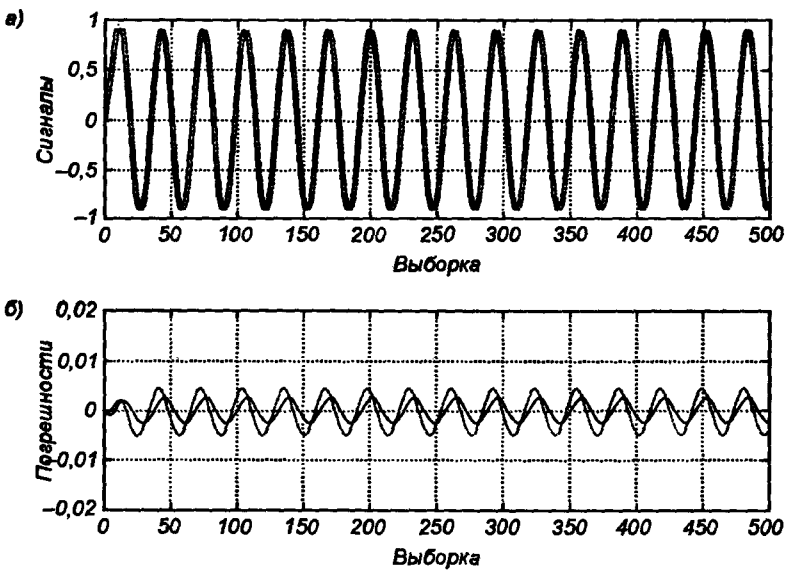

Рис. 8.18. Реакция натренированной сети RTRN на синусоидальное возбуждение: *а*) график значений, фактически полученных на двух выводах модели; *б*) графики соответствующих погрешностей вычислений

на пороговое, так и на синусоидальное возбуждение с высокой точностью совпадали с ожидаемыми значениями. На рис. 8.18 в качестве примера представлена реакция системы на синусоидальное возбуждение. График на рис. 8.18 *а* относится к реакции в обоих входных контурах, а график на рис. 8.18 *б* – к разностям по отношению к ожидаемым значениям. Минимальное отклонение реакции модели от реакции объекта не превысило значения 0,005 в обоих контурах, что с учетом амплитуды выходного сигнала, равной 1, может считаться очень хорошим достижением.

Раздел 9

СЕТИ С САМООРГАНИЗАЦИЕЙ НА ОСНОВЕ КОНКУРЕНЦИИ

Основу самоорганизации нейронных сетей составляет подмеченная закономерность, что глобальное упорядочение сети становится возможным в результате самоорганизующих операций, независимо друг от друга проводящихся в различных локальных сегментах сети. В соответствии с поданными входными сигналами осуществляется активация нейронов, которые вследствие изменения значений синаптических весов адаптируются к поступающим обучающим выборкам. В процессе обучения наблюдается тенденция к росту значений весов, из-за которой создается своеобразная положительная обратная связь: более мощные возбуждающие импульсы – более высокие значения весов – большая активность нейронов. При этом происходит естественное расслоение нейронов на различные группы. Отдельные нейроны или их группы сотрудничают между собой и активизируются в ответ на возбуждение, создаваемое конкретными обучающими выборками, подавляя своей активностью другие нейроны. При этом можно говорить как о сотрудничестве между нейронами внутри группы, так и о конкуренции между нейронами внутри группы и между различными группами.

Среди механизмов самоорганизации можно выделить два основных класса: самоорганизация, основанная на ассоциативном правиле Хебба, и механизм конкуренции между нейронами на базе обобщенного правила Кохонена.

Независимо от способов обучения самоорганизующихся сетей важное значение имеет избыточность обучающих данных, без которой обучение просто невозможно.

Широкий спектр обучающих данных, включающий многократные повторения похожих друг на друга выборок, образует "базу знаний" для обучения сети, из которой путем соответствующих сопоставлений выводятся решения по формированию на входе сети определенного отклассифицированного вектора.

9.1. Отличительные особенности сетей с самоорганизацией на основе конкуренции

В этом подразделе представлены сети с самоорганизацией, основу обучения которых составляет конкуренция между нейронами. Как правило, это однослойные сети, в которых каждый нейрон соединен со всеми компонентами N-мерного входного вектора x так, как это схематически изображено для $N = 2$ на рис. 9.1.

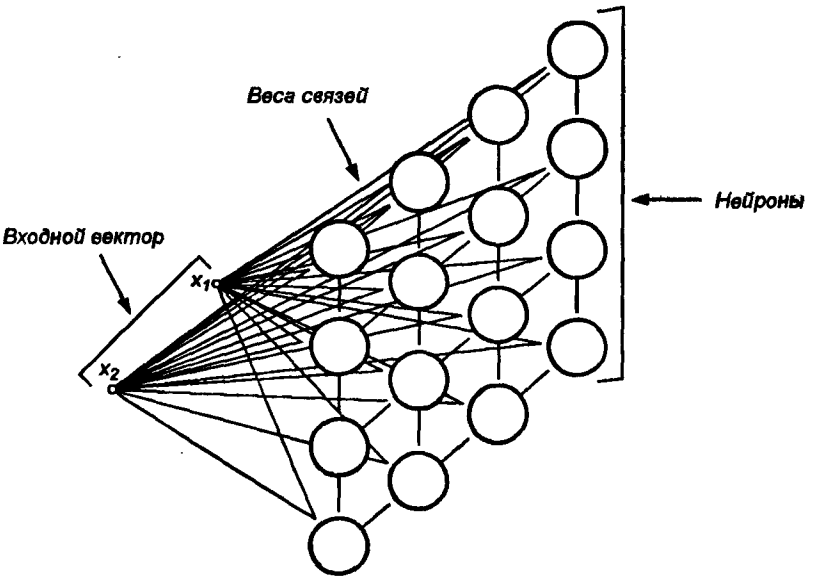

Рис. 9.1. Структура самоорганизующейся сети Кохонена

Веса синаптических связей нейронов образуют вектор $w_i = [w_{i1}, w_{i2}, ..., w_{iN}]^T$. После нормализации входных векторов при активации сети вектором x в конкурентной борьбе побеждает тот нейрон, веса которого в наименьшей степени отличаются от соответствующих компонентов этого вектора. Для w-го нейрона-победителя выполняется отношение

$$d(x, w_w) = \min_{1 \le i \le n} d(x, w_i) , \qquad (9.1)$$

где $d(x, w)$ обозначает расстояние (в смысле выбранной метрики) между векторами x и w, а n – количество нейронов. Вокруг нейрона-победителя образуется топологическая окрестность $S_w(k)$ с определенной энергетикой, уменьшающейся с течением времени. Нейрон-победитель и все нейроны, лежащие в пределах его окрестности, подвергаются адаптации, в ходе которой их векторы весов изменяются в направлении вектора x по правилу Кохонена [74]:

$$w_i(k+1) = w_i(k) + \eta_i(k) [x - w_i(k)] \qquad (9.2)$$

для $i \in S_w(k)$, где $\eta_i(k)$ обозначен коэффициент обучения i-го нейрона из окрестности $S_w(k)$ в k-й момент времени. Значение $\eta_i(k)$ уменьшается с увеличением расстояния между i-м нейроном и победителем. Веса нейронов, находящихся за пределами $S_w(k)$, не изменяются. Размер окрестности и коэффициенты обучения нейронов являются функциями, значения которых уменьшаются с течением времени. Х.Риттер и К.Шультен в [134] доказали, что адаптация по формуле (9.2) эквивалентна градиентному методу обучения, основанному на минимизации целевой функции

$$E(\bar{w}) = \frac{1}{2}\sum_{i,j,k} S_i(x(k))[x_j(k) - w_{ij}(k)]^2 , \qquad (9.3)$$

а $S_i(x(k))$ представляет собой функцию определения окрестности, изменяющуюся в процессе обучения. Доказано [73, 74, 134], что при таком способе обучения функция плотности распределения векторов w_i нейронов сводится к дискретизированной плотности распределения вынужденных векторов.

После предъявления двух различных векторов x, например x_1 и x_2, активизируются два нейрона сети, веса которых наиболее близки к координатам соответствующих векторов x_1 и x_2. Эти веса, обозначенные в векторной форме w_1 и w_2, могут отображаться в пространстве как две точки. Сближение векторов x и x_2 вызывает соответствующее изменение в расположении векторов w_1 и w_2. В пределе равенство $w_1 = w_2$ выполняется тогда и только тогда, когда x_1 и x_2 совпадают или практически неотличимы друг от друга. Сеть, в которой эти условия выполняются, называется топографической картой, или картой Кохонена

9.1.1. Меры расстояния между векторами

Процесс самоорганизации предполагает определение победителя каждого этапа, т.е. нейрона, вектор весов которого в наименьшей степени отличается от поданного на вход сети вектора x. В этой ситуации важной проблемой становится выбор метрики, в которой будет измеряться расстояние между векторами x и w_i. Чаще всего в качестве меры расстояния используются:

- эвклидова мера

$$d(x, w_i) = \| x - w_i \| = \sqrt{\sum_{j=1}^{N}(x_j - w_{ij})^2}; \qquad (9.4)$$

- скалярное произведение

$$d(x, w_i) = 1 - x \cdot w_i = 1 - \| x \| \| w_i \| \cos(x, w_i); \qquad (9.5)$$

- мера относительно нормы L1 (Манхэттен)

$$d(x, w_i) = \sqrt{\sum_{j=1}^{N} | x_j - w_{ij} |}; \qquad (9.6$$

- мера относительно нормы L_∞

$$d(x, w_i) = \max_j |x_j - w_{ij}| . \tag{9.7}$$

При использовании эвклидовой меры разбиение пространства на зоны доминирования нейронов равносильно мозаике Вороного, в которой пространство вокруг центральных точек образует окрестность доминирования данного нейрона. Использование при самоорганизации любой другой меры вызывает несколько иное разделение сфер влияния нейронов. Следует отметить, что если векторы не нормализуются, то использование в качестве меры скалярного произведения может привести к несвязному разделению пространства, при котором в одном регионе может находиться несколько нейронов, а в другом не будет ни одного [25].

9.1.2. Нормализация векторов

Доказано, что если хотя бы один из векторов x или w подвергается нормализации, то процесс самоорганизации всегда приводит к связному разделению пространства данных. При нормализованных обучающих векторах x стремящиеся к ним векторы весов w нормализуются автоматически (норма вектора равна 1). Однако нормализация вектора весов приводит к тому, что если $\|w_i\| = \text{const}$, то для всех нейронов при фиксированном значении x произведение $\|x\|\|w_i\|$ также становится постоянной величиной. Поэтому активация нейрона определяется значением $\cos(x, w_i)$, которое становится общей мерой для всей сети. Следует отметить, что при нормализации вектора весов эвклидова мера и скалярное произведение равнозначны друг другу, поскольку $\|x - w_i\|^2 = \|x\|^2 + \|w_i\|^2 - 2x^T w_i$. Поэтому $\min\|x - w_i\|^2 = \max(x^T w_i)$ при $\|w_i\| = \text{const}$.

Экспериментальные исследования подтвердили необходимость применения нормализации векторов при малой размерности пространства, например при $n = 2$ или $n = 3$. Такая нормализация может проводиться двумя способами:

- переопределением компонентов вектора в соответствии с формулой

$$x_i \leftarrow \frac{x_i}{\sqrt{\sum_{i=1}^{N} x_i^2}} , \tag{9.8}$$

- увеличением размерности пространства на одну координату ($R^N \rightarrow R^{N+1}$) с таким выбором значения ($N+1$)-го компонента вектора, чтобы

$$\sum_{i=1}^{N+1} x_i^2 = 1 . \tag{9.9}$$

При использовании этого способа нормализации, как правило, возникает необходимость предварительного масштабирования компонентов вектора x в пространстве R^N для того, чтобы могло выполняться равенство (9.9).

С увеличением размерности входного вектора эффект нормализации становится все менее заметным, и при больших объемах сети ($N > 200$) она перестает оказывать влияние на процесс самоорганизации.

9.1.3. Проблема мертвых нейронов

При инициализации весов сети случайным способом часть нейронов может оказаться в области пространства, в которой отсутствуют данные или их количество ничтожно мало. Эти нейроны имеют мало шансов на победу и адаптацию своих весов, поэтому они остаются мертвыми. Таким образом, входные данные будут интерпретироваться меньшим количеством нейронов (мертвые нейроны не принимают участие в анализе), а погрешность интерпретации данных, иначе называемая погрешностью квантования, увеличится. Поэтому важной проблемой становится активация всех нейронов сети.

Такую активацию можно осуществить, если в алгоритме обучения предусмотреть учет количества побед каждого нейрона, а процесс обучения организовать так, чтобы дать шанс победить и менее активным нейронам. Идея такого подхода к обучению возникла при наблюдении за поведением биологических нейронов. Отмечен факт, что нейрон-победитель сразу после победы на некоторое время теряет активность, "отдыхая" перед следующим этапом конкурентной борьбы [143]. Такой способ учета активности нейронов будет называться в дальнейшем *механизмом утомления*[1].

Существуют различные механизмы учета активности нейронов в процессе обучения. Часто используется метод подсчета потенциала p_i каждого нейрона. значение которого модифицируется всякий раз после представления очередной реализации входного вектора x в соответствии со следующей формулой (в ней предполагается, что победителем стал w-й нейрон):

$$p_i(k+1) = \begin{cases} p_i(k) + \dfrac{1}{n} & (i \neq w) \\ p_i(k) - p_{\min} & (i = w) \end{cases}. \qquad (9.10)$$

Значение коэффициента $p_{\min}$ определяет минимальный потенциал, разрешающий участие в конкурентной борьбе. Если фактическое значение потенциала p_i падает ниже $p_{\min}$, i-й нейрон "отдыхает", а победитель ищется среди нейронов, для которых выполняется отношение

$$d(x, w_w) = \min\{d(x, w_i)\}, \qquad (9.11)$$

для $1 \le i \le N$ и $p_i \ge p_{\min}$. Максимальное значение потенциала ограничивается на уровне, равном 1. Выбор конкретного значения $p_{\min}$ позволяет установить порог готовности нейрона к конкурентной борьбе. При $p_{\min} = 0$ утомляемость нейронов не возникает, и каждый из них сразу после победы будет готов к продолжению

[1] Оригинальное название *conscience mechanism* иногда переводится как *механизм совести*.

соперничества (стандартный алгоритм Кохонена). При $p_{\min} = 1$ возникает другая крайность, вследствие которой нейроны побеждают по очереди, так как в каждый момент только один из них оказывается готовым к соперничеству. На практике хорошие результаты достигаются, когда $p_{\min} \approx 0{,}75$.

В другом очень удачном алгоритме обучения количество побед нейрона учитывается при подсчете эффективного расстояния между вектором весов и реализацией обучающего вектора x. Это расстояние модифицируется пропорционально количеству побед данного нейрона в прошлом. Если обозначить количество побед i-го нейрона N_i, такую модификацию можно представить в виде

$$\mathrm{d}(x, w_i) \leftarrow N_i\, d(x, w_i). \tag{9.12}$$

Активные нейроны с большим значением N_i штрафуются искусственным завышением этого расстояния. Следует обратить внимание, что модификация расстояния производится только при выявлении победителя. В момент уточнения весов учитывается фактическое расстояние. Модификация этой характеристики имеет целью активизировать все нейроны путем введения их в область с большим количеством данных. После решения этой задачи (обычно после двух или трех циклов обучения) модификация прекращается, что позволяет продолжить "честную" конкуренцию нейронов [139].

9.2. Алгоритмы обучения сетей с самоорганизацией

Целью обучения сети с самоорганизацией на основе конкуренции нейронов считается такое упорядочение нейронов (подбор значений их весов), которое минимизирует значение ожидаемого искажения, оцениваемого погрешностью аппроксимации входного вектора x, значениями весов нейрона-победителя в конкурентной борьбе. При p входных векторах x и применении эвклидовой метрики эта погрешность, называемая также *погрешностью квантования*, может быть выражена в виде

$$E_q = \frac{1}{p}\sum_{i=1}^{p} \| x_i - w_{w(i)} \|^2, \tag{9.13}$$

где $w_{w(i)}$ – это вес нейрона-победителя при предъявлении вектора x_i.

Этот подход также называется *векторным квантованием* (англ.: *Vector Quantization – VQ*). Номера нейронов-победителей при последовательном предъявлении векторов x образуют так называемую кодовую таблицу. При классическом решении задачи кодирования применяется алгоритм *K*-усреднений (англ.: *K-means*), представленный в разделе 5 и носящий имя обобщенного алгоритма Ллойда [89].

Для нейронных сетей аналогом алгоритма Ллойда считается алгоритм WTA (англ.: *Winner Takes All* – Победитель получает все). В соответствии с ним после предъявления вектора x рассчитывается активность каждого нейрона. Победителем признается нейрон с самым сильным выходным сигналом, т.е. тот, для которого скалярное произведение $(x^T w)$ оказывается наибольшим. В предыдущем разделе было показано, что при использовании нормализованных векторов это равнозначно наименьшему эвклидовому расстоянию между входным вектором и вектором весов нейронов. Победитель получает право уточнить свои веса в направлении вектора x согласно правилу

$$w_w \leftarrow w_w + \eta(x - w_w) \ . \tag{9.14}$$

Веса остальных нейронов уточнению не подлежат. Алгоритм позволяет учитывать усталость нейронов путем подсчета количества побед каждого из них и поощрять элементы с наименьшей активностью для выравнивания их шансов. Как уже отмечалось ранее, такая модификация применяется чаще всего на начальной стадии обучения с последующим отключением после активизации всех нейронов. Подобный способ обучения реализован в программе *Kohon* в виде режима CWTA[1] и считается одним из наилучших и наиболее быстрых алгоритмов самоорганизации.

Помимо алгоритмов WTA, в которых в каждой итерации может обучаться только один нейрон, для обучения сетей с самоорганизацией широко применяются алгоритмы типа WTM (англ.: *Winner Takes Most* – Победитель получает больше), в которых, кроме победителя, уточняют значения своих весов и нейроны из его ближайшего окружения. При этом чем дальше какой-либо нейрон находится от победителя, тем меньше изменяются его веса. Процесс уточнения вектора весов может быть определен обобщенной зависимостью (9.2), которая здесь представляется в виде

$$w_i \leftarrow w_i + \eta_i G(i,x)[x - w_i] \tag{9.15}$$

для всех i нейронов, расположенных в окрестности победителя. В приведенной формуле коэффициент обучения η_i каждого нейрона отделен от его расстояния до предъявленного вектора x функцией $G(i,x)$. Если $G(i,x)$ определяется в форме

$$G(i,x) = \begin{cases} 1 & \text{для } i = w \\ 0 & \text{для } i \neq w \end{cases}, \tag{9.16}$$

где w обозначает номер победителя, то мы получаем классический алгоритм WTA. Существует множество вариантов алгоритма WTM, отличающихся прежде всего формой функции $G(i,x)$. Для дальнейшего изучения выберем два из них: классический алгоритм Кохонена и алгоритм нейронного газа.

[1] Название CWTA происходит от английского названия алгоритма *Conscience Winner Takes All.*

9.2.1. Алгоритм Кохонена

Алгоритм Кохонена относится к наиболее старым алгоритмам обучения сетей с самоорганизацией на основе конкуренции, и в настоящее время существуют различные его версии. В классическом алгоритме Кохонена сеть инициализируется путем приписывания нейронам определенных позиций в пространстве и связывания их с соседями на постоянной основе. В момент выбора победителя уточняются не только его веса, но также и веса его соседей, находящихся в ближайшей окрестности. Таким образом, нейрон-победитель подвергается адаптации вместе со своими соседями. В классическом алгоритме Кохонена функция соседства $G(i, x)$ определяется в виде

$$G(i,x)=\begin{cases}1 & \text{для } i=w\\ 0 & \text{для } i\neq w\end{cases}. \qquad (9.17)$$

В этом выражении $d(i,w)$ может обозначать как эвклидово расстояние между векторами весов нейрона-победителя w и i-го нейрона, так и расстояние, измеряемое количеством нейронов. Коэффициент 1 выступает в роли уровня соседства, его значение уменьшается в процессе обучения до нуля. Соседство такого рода называется *прямоугольным*.

Другой тип соседства, часто применяемый в картах Кохонена, – это соседство гауссовского типа, при котором функция $G(i, x)$ определяется формулой

$$G(i,x)\exp\left(-\frac{d^2(i,w)}{2\lambda^2}\right). \qquad (9.18)$$

Степень адаптации нейронов-соседей определяется не только эвклидовым расстоянием между i-м нейроном и победителем (w-м нейроном), но также и уровнем соседства λ. В отличие от соседства прямоугольного типа, где каждый нейрон, находящийся в окрестности победителя, адаптировался в равной степени, при соседстве гауссовского типа уровень адаптации отличается и зависит от значения функции Гаусса. Как правило, гауссовское соседство дает лучшие результаты обучения и обеспечивает лучшую организацию сети, чем прямоугольное соседство.

9.2.2. Алгоритм нейронного газа

Значительно лучшую самоорганизацию сети и ускорение сходимости алгоритма WTM можно получить применением метода, предложенного М. Мартинесом, С. Берковичем и К. Шультеном в работе [94] и названного авторами алгоритмом нейронного газа из-за подобия его динамики движению молекул газа.

В этом алгоритме на каждой итерации все нейроны сортируются в зависимости от их расстояния до вектора x. После сортировки нейроны размечаются в последовательности, соответствующей увеличению удаленности

$$d_0 < d_1 < d_2 < ... < d_{n-1}, \tag{9.19}$$

где $d_k = \|x - w_{m(i)}\|$ обозначает удаленность i-го нейрона, занимающего в результате сортировки m-ю позицию в последовательности, возглавляемой нейроном-победителем, которому сопоставлена удаленность d_0. Значение функции соседства для i-го нейрона $G(i,x)$ определяется по формуле

$$G(i,x) = \exp\left(-\frac{m(i)}{\lambda}\right), \tag{9.20}$$

в которой $m(i)$ обозначает очередность, полученную в результате сортировки ($m(i) = 0, 1, 2, ..., n - 1$), а λ – параметр, аналогичный уровню соседства в алгоритме Кохонена, уменьшающийся с течением времени. При $\lambda = 0$ адаптации подвергается только нейрон-победитель, и алгоритм превращается в обычный алгоритм WTA, но при $\lambda \neq 0$ уточнению подлежат веса многих нейронов, причем уровень уточнения зависит от величины $G(i,x)$. Алгоритм нейронного газа напоминает стратегию нечетких множеств, в соответствии с которой каждому нейрону приписывается значение функции принадлежности к окрестности, определенной соотношением (9.20).

Для достижения хороших результатов самоорганизации процесс обучения должен начинаться с большого значения λ, однако с течением времени его величина уменьшается до нуля. Изменение $\lambda(k)$ может быть линейным или показательным. В работе [134] предложено изменять значение $\lambda(k)$ в соответствии с выражением

$$\lambda(k) = \lambda_{\max}\left(\frac{\lambda_{\min}}{\lambda_{\max}}\right)^{k/k_{\max}}, \tag{9.21}$$

где $\lambda(k)$ обозначает значение λ на k-й итерации, а $\lambda_{\min}$ и $\lambda_{\max}$ – принятые минимальное и максимальное значения λ соответственно. Коэффициент $k_{\max}$ определяет максимальное заданное количество итераций.

Коэффициент обучения i-го нейрона $\eta_i(k)$ тоже может изменяться как линейно, так и показательно, причем его степенная изменчивость определяется формулой

$$\eta_i(k) = \eta_i(0)\left(\frac{\eta_{\min}}{\eta_i(0)}\right)^{k/k_{\max}}, \tag{9.22}$$

в которой $\eta_i(0)$ обозначает начальное значение коэффициента обучения, а $\eta_{\min}$ – априорно заданное минимальное значение, соответствующее $k = k_{\max}$. На практике наилучшие результаты самоорганизации достигаются при линейном изменении $\eta_i(k)$, и именно эта стратегия реализована в режиме NGAS[1] программы *Kohon*.

[1] Название NGAS происходит от английского названия нейронного газа *Neural gas*.

Для сокращения объема вычислений, необходимых для реализации алгоритма нейронного газа, можно применить определенное упрощение, состоящее в учете при сортировке только нейронов с наиболее значимой величиной функции $G(i,x)$. При этом используется зависимость (9.20), в соответствии с которой если $m(i) >> 1$, то значение $G(i,x) \cong 0$, Например, если принять $K = 3\lambda$, то при сортировке нейронов, а в последующем – и при их адаптации можно ограничиться только первыми K элементами.

Алгоритм нейронного газа наряду с алгоритмом WTA, учитывающим активность нейронов (в режиме CWTA), считается одним из наиболее эффективных средств самоорганизации нейронов в сети Кохонена. При соответствующем подборе параметров управления процессом можно добиться очень хорошей организации сети при скорости функционирования, превышающей достижимую в классическом алгоритме Кохонена.

9.2.3. Сравнение алгоритмов самоорганизации

Приведенные выше алгоритмы сравнивались при решении задачи восстановления двухмерных обучающих данных сложной структуры, представленной на рис. 9.2. Для восстановления данных использовались два множества нейронов, включающих 20 и 400 элементов, которые при идеальном упорядочении позиции нейронов будут отражать распределение обучающих данных. Они должны группироваться в областях максимальной концентрации данных. На рис. 9.3 приведены результаты самоорганизации 40 нейронов при использовании трех алгоритмов, представленных в настоящем разделе: CWTA (рис. 9.3*а*), нейронного газа (рис. 9.3*б*) и алгоритма Кохонена (рис. 9.3*в*). Для сравнения на рис. 9.4 приведены те же самые отображения сетью, состоящей из 200 нейронов.

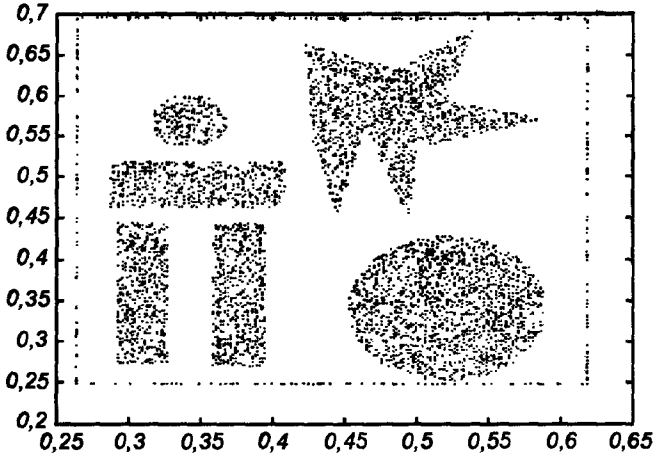

Рис. 9.2. Распределение двухмерных данных, использованных для тестирования

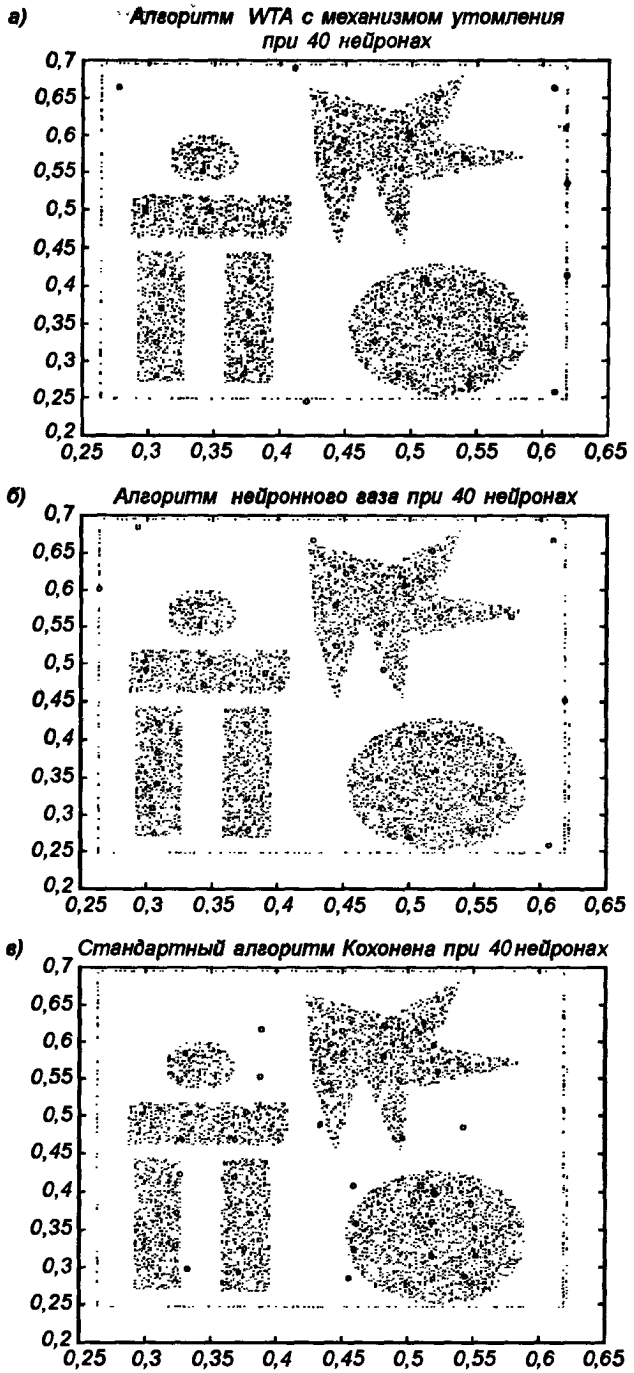

Рис. 9.3. Восстановление данных с рис. 9.2 самоорганизующейся сетью Кохонена, состоящей из 40 нейронов, при использовании:

а) алгоритма CWTA; *б*) алгоритма нейронного газа; *в*) классического алгоритма Кохонена

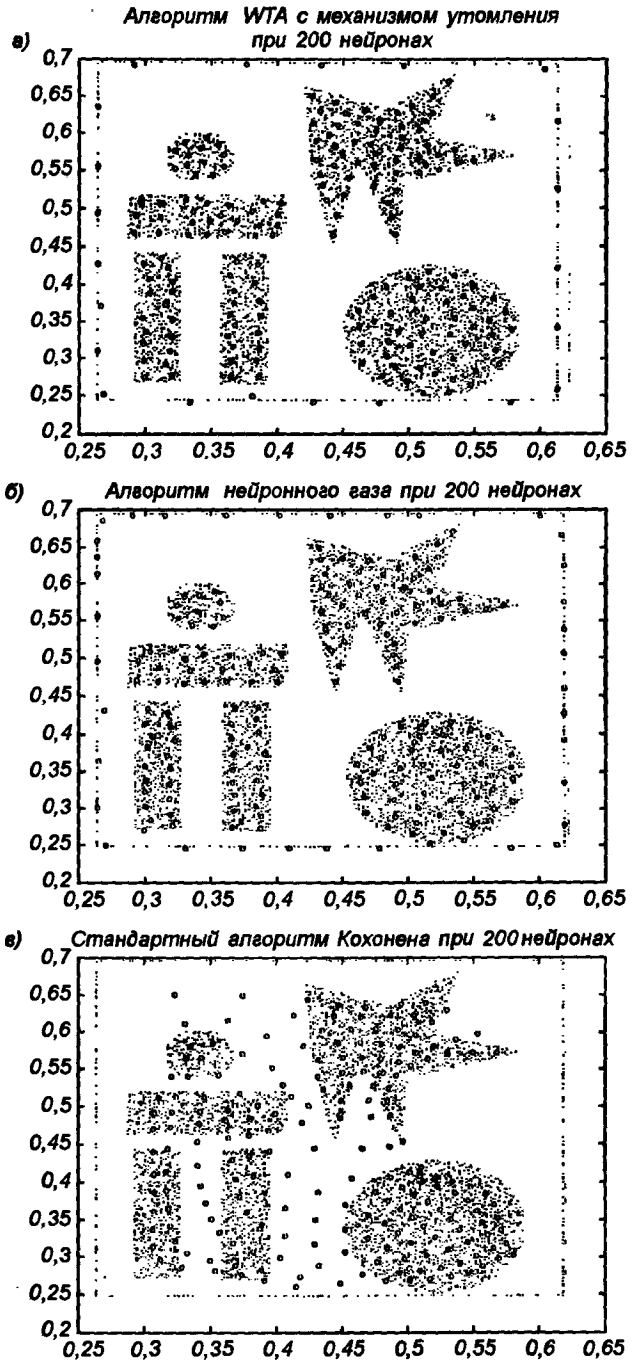

Рис. 9.4. Восстановление данных с рис. 9.2 самоорганизующейся сетью Кохонена, состоящей из 200 нейронов, при использовании:
а) алгоритма CWTA; *б*) алгоритма нейронного газа; *в*) классического алгоритма Кохонена

Независимо от количества нейронов наилучшие результаты самоорганизации были получены с использованием алгоритмов CWTA и нейронного газа, причем последний из-за необходимости сортировки оказался значительно более медленным, чем CWTA. Оригинальный алгоритм Кохонена в обоих случаях оставался наихудшим, не обеспечивая хорошего восстановления данных (определенное количество нейронов размещалось в областях, свободных от данных).

Объективное количественное сравнение результатов самоорганизации можно получить при сопоставлении расчетной погрешности квантования E_q (формула (9.13)) для каждого случая. При 200 нейронах получены значения $E_q = 0{,}007139$ – для CWTA, $E_q = 0{,}007050$ – для нейронного газа и $E_q = 0{,}010763$ – для алгоритма Кохонена. При 40 нейронах результаты следующие: $E_q = 0{,}017416$ (CWTA), $E_q = 0{,}017599$ (нейронный газ) и $E_q = 0{,}02539$ (алгоритм Кохонена). Численные показатели подтверждают зрительное восприятие качества восстановления данных, согласно которому алгоритмы CWTA и нейронного газа дают сходные (и наилучшие) результаты, а алгоритм Кохонена наименее эффективен.

9.3. Сеть восстановления одно- и двухмерных данных

Для хорошего восстановления данных нейронной сетью требуется, чтобы нейроны группировались в областях наибольшей концентрации данных, а не там, где они отсутствуют. При оценке качества нейронной самоорганизующейся сети важное значение имеет восстановление одно- и двухмерных данных, с учетом четкости и очевидности интерпретации результатов на плоскости (x, y). Если принять во внимание, что веса нейронов соответствуют координатам центров кластеров, на которые подразделяется множество данных, то каждому вектору весов можно приписать соответствующую точку на плоскости. Объединяя эти точки с ближайшими соседями, можно получить регулярную сетку, отображающую топографическое распределение данных.

При равномерном распределении обучающих векторов $\boldsymbol{x}$ на плоскости ожидаемое распределение восстановленных весов конкретных нейронов, спроецированное на эту плоскость, также будет равномерным. Если же данные распределены неравномерно, то концентрация будет наблюдаться в тех областях, для которых вероятность предъявления обучающих векторов оказалась наибольшей.

На рис. 9.5 представлены примеры восстановления данных, образующих фигуры различной формы: эллиптическую, треугольную, прямоугольную и нерегулярную. Сеть обучалась программой *Kohon* по алгоритму CWTA. Нейроны группировались в областях с наибольшей концентрацией данных. О качестве самоорганизации свидетельствует равномерное распределение нейронов в пространстве при отображении равномерного распределения обучающих данных.

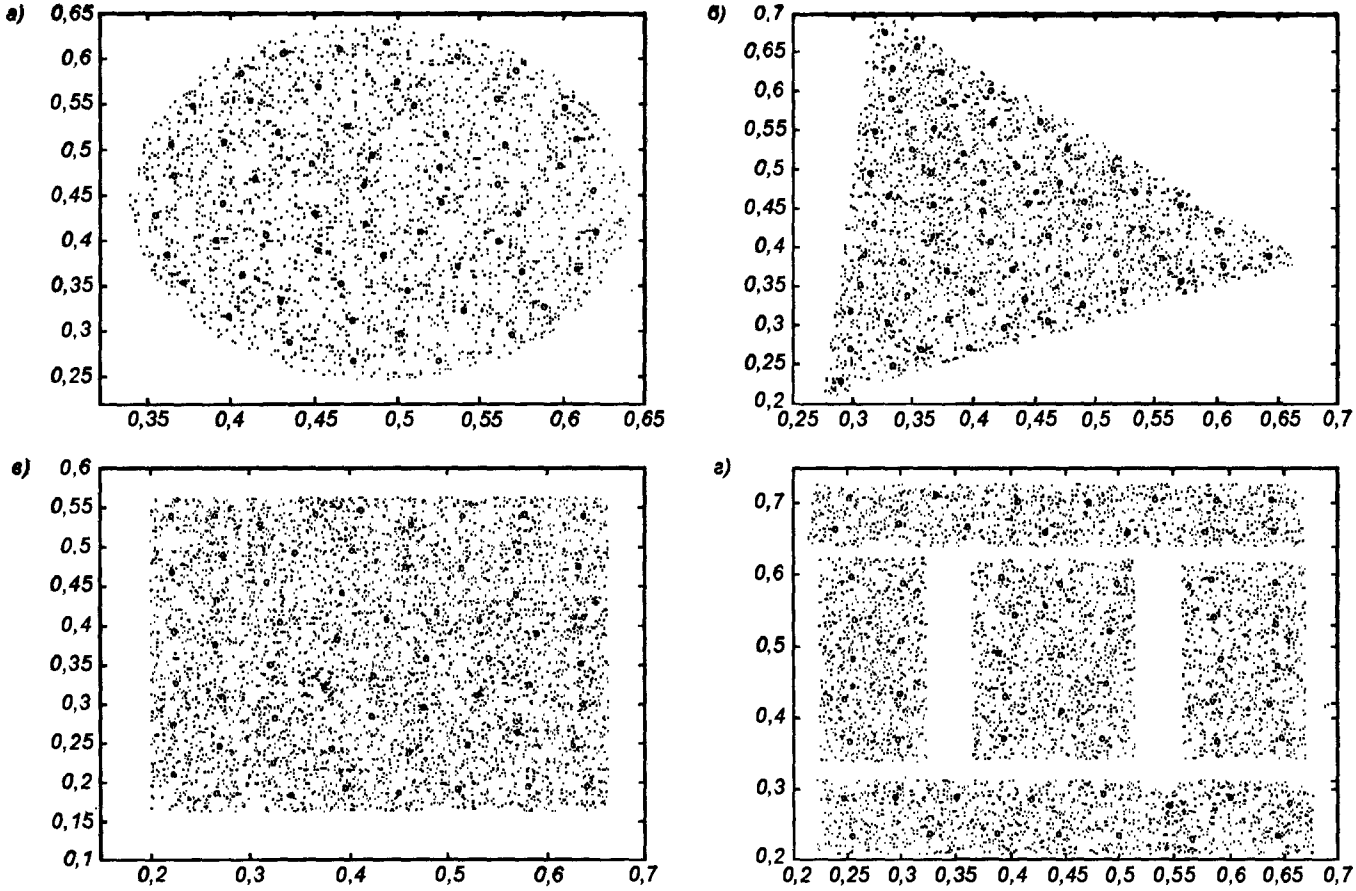

Рис. 9.5. Иллюстрация восстановления сетью Кохонена равномерно распределенных двухмерных данных: *а*) эллиптическая фигура; *б*) треугольная фигура; *в*) прямоугольная фигура; *г*) составная фигура

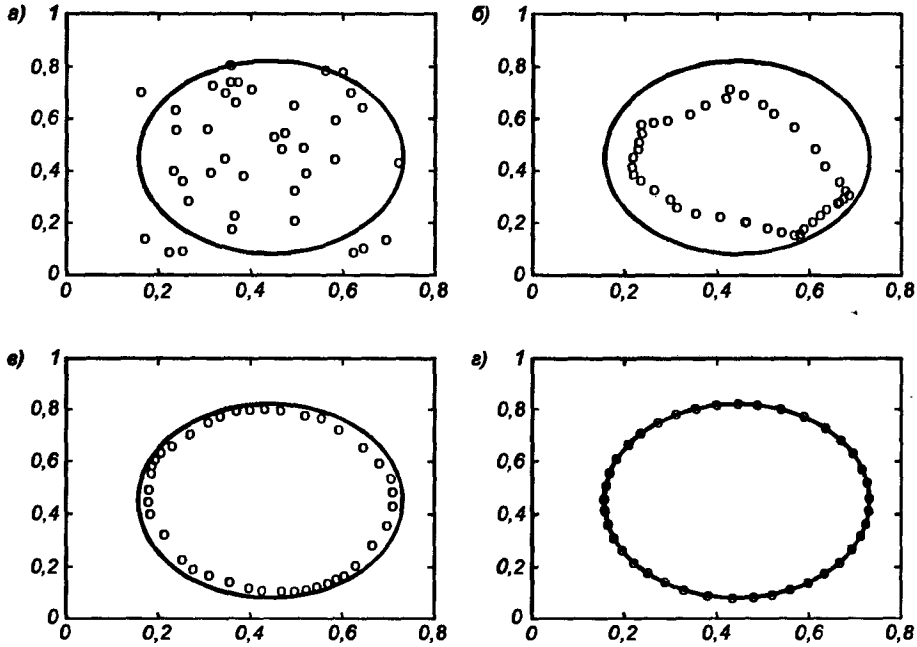

Рис. 9.6. Иллюстрация последовательности обучения самоорганизующейся сети Кохонена при использовании алгоритма нейронного газа

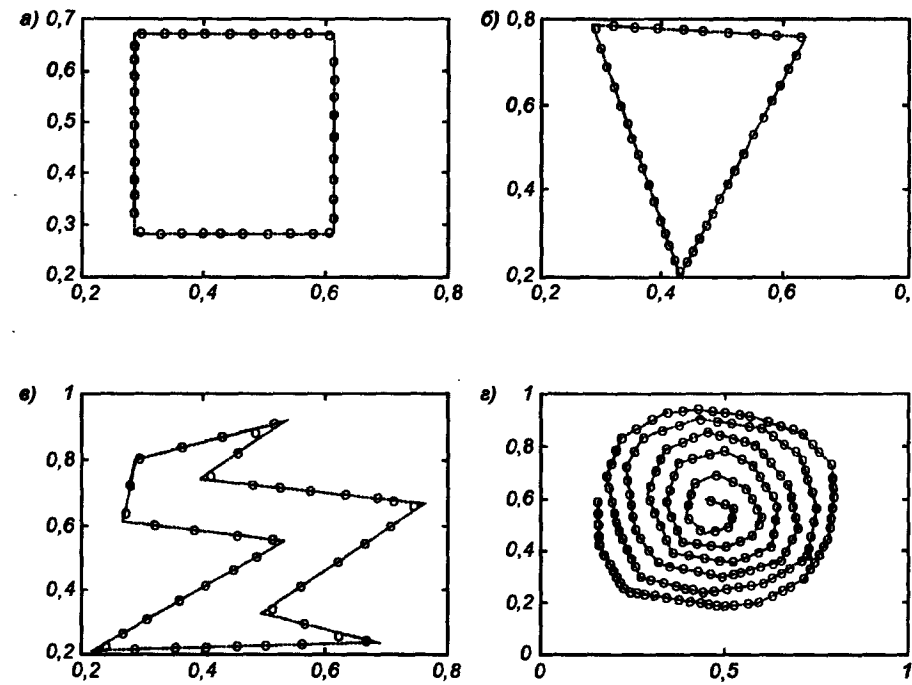

Рис. 9.7. Восстановление сложных кривых самоорганизующимися нейронами Кохонена

Также хорошие результаты получаются при восстановлении самоорганизующейся сетью одномерных данных. На рис. 9.6 для примера проиллюстрирован процесс самоорганизации 40 нейронов, отображающих эллипсовидную структуру (позиции нейронов обозначены окружностями). Рис. 9.6*а* представляет начальное состояние весов нейронов, рис. 9.6*б* – состояние после двух циклов обучения, рис. 9.6*в* – состояние после пяти циклов обучения, рис. 9.6*г* – конечное состояние (после десяти циклов) весов на фоне обучающих данных, образующих эллипсовидную структуру. Сеть обучалась программой *Kohon* по алгоритму нейронного газа. На рис. 9.7 приведены результаты упорядочения нейронов при восстановлении фигур различной структуры, образованных данными на плоскости (x, y). В каждом случае веса нейронов группировались в областях присутствия данных таким образом, чтобы погрешность квантования была минимальной.

Главное достоинство сетей с самоорганизацией становится заметным только при обработке многомерных данных, пространственное размещение которых человек уже не в состоянии себе представить. Механизмы самоорганизации, встроенные в алгоритмы обучения таких сетей, функционируют независимо от размерности задачи. Важнейшая функция, реализуемая при этом сетью, – векторное квантование, состоящее в том, что огромное количество данных, образующих кластер, отображается вектором весов нейрона, представляющего центр кластера. Поэтому пространственное размещение нейронов позволяет определить зоны концентрации данных в многомерном пространстве и основные характеристики их распределения, существенные с точки зрения пользователя.

9.4. Восстановление Сэммона

При обработке многомерных данных возникает задача такой графической визуализации расположения нейронов, которая была бы четкой и понятной для пользователя. Это предполагает проекцию распределения данных из многомерного пространства в двух- или в крайнем случае трехмерное пространство при сохранении основных характеристик распределения в многомерном пространстве.

Пусть имеется n N-мерных векторов x_i (i = 1, 2, ..., n). В соответствии с ними определяются n векторов в M-мерном пространстве (M = 2, 3), обозначаемых y_i. Расстояния между векторами в N-мерном пространстве будем обозначать $d_{ij}^* = (y_i, y_j)$, а в M-мерном пространстве $d_{ij} = (y_i, y_j)$. Для определения расстояния между векторами можно использовать любую метрику, в частности эвклидову. Задача нелинейного восстановления Сэммона состоит в таком подборе векторов y, чтобы минимизировать функцию

погрешности E, определяемую по формуле

$$E = \frac{1}{c}\sum_{i<j}^{n}\frac{[d_{ij}^{*} - d_{ij}]^2}{d_{ij}^{*}}, \tag{9.23}$$

где

$$c = \sum_{i<j}^{n} d_{ij}^{*}, \tag{9.24}$$

$$d_{ij} = \sqrt{\sum_{k=1}^{M}[y_{ik} - y_{jk}]^2}, \tag{9.25}$$

а y_{ij} обозначает j-й компонент вектора y_i.

Для минимизации функции погрешности E Сэммон применил метод минимизации Ньютона, упрощенный до вида

$$y_{pq}(k+1) = y_{pq} - \eta\Delta_{pq}(k), \tag{9.26}$$

где

$$\Delta_{pq}(k) = \frac{\dfrac{\partial E}{\partial y_{pq}}}{\left|\dfrac{\partial^2 E}{\partial y_{pq}^2}\right|} \tag{9.27}$$

представляет частное от деления соответствующего компонента градиента на диагональный элемент гессиана, определенный на k-й итерации. Коэффициент η аналогичен константе обучения, выбираемой из интервала [0,3, 0,4]. При определении функции погрешности в виде (9.23) соответствующие компоненты градиента и гессиана описываются выражениями:

$$\frac{\partial E}{\partial y_{pq}} = -\frac{2}{c}\sum_{j=1, j\neq p}^{n}\left[\frac{d_{pj}^{*} - d_{pj}}{d_{pj}d_{pj}^{*}}\right][y_{pq} - y_{jq}], \tag{9.28}$$

$$\frac{\partial^2 E}{\partial y_{pq}^2} = -\frac{2}{c}\sum_{j=1, j\neq p}^{n}\frac{1}{d_{pj}^{*}d_{pj}} \times$$

$$\times\left[(d_{pj}^{*} - d_{pj}) - \frac{(y_{pq} - y_{jq})^2}{d_{pj}}\left(1 + \frac{d_{pj}^{*} - d_{pj}}{d_{pj}}\right)\right]. \tag{9.29}$$

9.5. Применение сетей с самоорганизацией

Главным свойством сети Кохонена считается компрессия данных, состоящая в том, что образующие кластер большие группы данных представляются единственным вектором весов нейрона-победителя. При разделении p данных на P кластеров и представлении каждого кластера одним из n нейронов достигается значительное сокращение количества информации, которое и называется

компрессией. Это компрессия с потерями, которая сопровождается определенной погрешностью квантования, описываемой зависимостью (9.13).

9.5.1. Компрессия данных

Примером использования компрессионных свойств сети Кохонена может считаться сжатие изображений, предназначенное для уменьшения количества информации, представляющей конкретный образ, при сохранении погрешности восстановления на заданном уровне (обеспечении достаточно большого значения коэффициента PSNR).

Предположим, что изображение размером $N_x \times N_y$ пикселов разделяется на одинаковые кадры размером $n_x \times n_y$ пикселов. Образующие кадр пикселы представляют собой компоненты входных векторов $\boldsymbol{x}$. Каждый вектор состоит из $n_x \times n_y$ компонентов, определяющих интенсивность конкретного пиксела в кадре. Соотнесение пикселам вектора может проводиться либо объединением соответствующих строк кадра в единую последовательность, либо растровым способом. Сеть с самоорганизацией содержит n нейронов, каждый из которых связан синаптическими дугами со всеми компонентами входного вектора $\boldsymbol{x}$. Обучение сети при помощи одного из алгоритмов самоорганизации состоит в подборе таких весов конкретных нейронов, при которых минимизируется погрешность квантования (9.13). В результате обучения формируется структура сети, при которой вектору $\boldsymbol{x}$ каждого кадра соответствует вектор весов нейрона-победителя. При аналогичных структурах вектора $\boldsymbol{x}$ для разных кадров побеждать будет один и тот же нейрон либо их группа с похожими векторами весов. В процессе предъявления очередного кадра выбирается номер нейрона-победителя, например, 1, 1, 3, 80 и т.д. Номера нейронов-победителей образуют кодовую таблицу, а веса этих нейронов представляют средние значения, соответствующие конкретным компонентам вектора $\boldsymbol{x}$ (т.е. уровням интенсивности пикселов, составляющих кадр). Если принять во внимание, что количество нейронов n обычно намного меньше количества кадров N_r, то можно получить существенное сокращение объема информации, описывающей исходное изображение. При определении степени компрессии следует учитывать также и конечное число битов, необходимых для кодирования номеров нейронов-победителей конкретного кадра. В итоге коэффициент компрессии изображения определяется в виде [46, 114]

$$K_\gamma = \frac{N_\gamma n_x n_y T}{N_\gamma \lg_2 n + n n_x n_y t}, \qquad (9.30)$$

где n_x и n_y обозначают размеры кадра в осях x и y, N_r – количество кадров; n – количество нейронов, а T и t – количество битов, используемых для представления соответственно градаций интенсивности пиксела и значений весов. Этот подход позволяет получить степень компрессии изображений порядка 16 при значениях коэффициента PSNR около 26–28 дБ.

На рис. 9.8*а* представлены результаты обучения сети Кохонена для образа "Барбара" размером 512 × 512 пикселов, разделенного на 16-элементные кадры (4 × 4 пиксела). Сеть Кохонена состояла из 512 нейронов. При 8-битовом представлении данных достигнута степень компрессии $K_r \cong 9.8$. На рис. 9.8 приведено изображение, восстановленное по весам нейронов, побеждавших пр предъявлении очередных кадров. Коэффициент PSNR для восстановленног изображения составил 26,2 дБ. Различия в качестве оригинального (рис. 9.8*а*) · восстановленного (рис. 9.8*б*) изображений относительно невелики. Это заметн на рис. 9.8*в*, представляющего собой графическую интерпретацию погрешност в виде разности между оригинальным и восстановленным после компрессии изображениями.

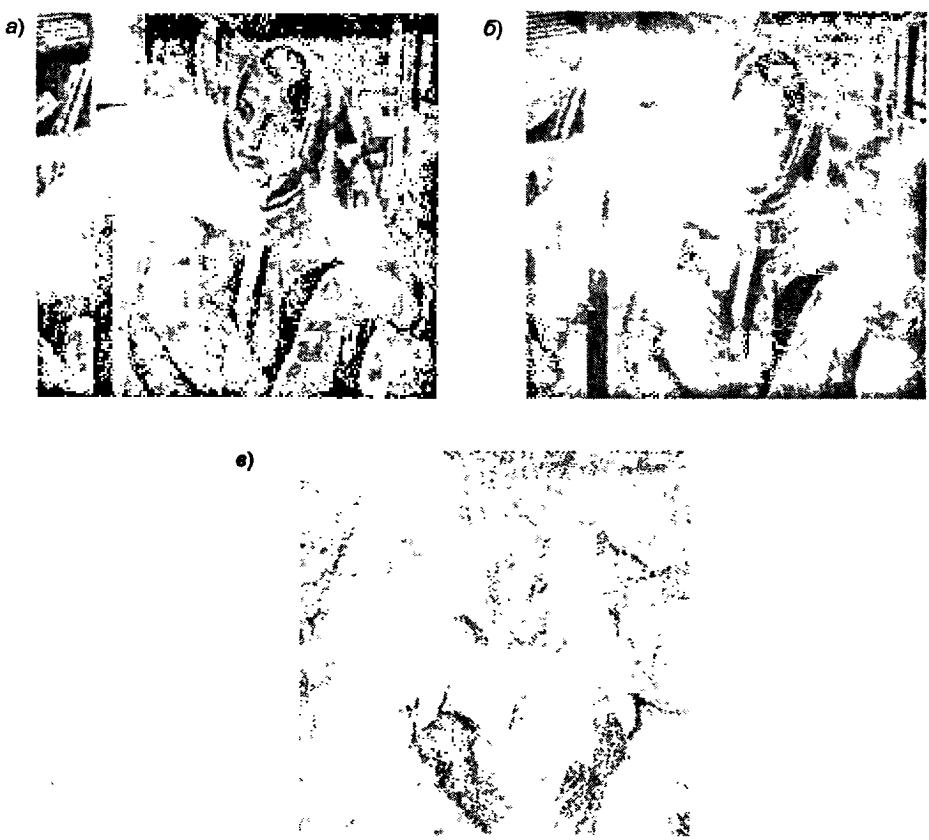

Рис. 9.8. Образ "Барбара", использованный для компрессии сетью Кохонена: *а*) оригинальное изображение; *б*) восстановленное изображение; *в*) разностное изображение

Важнейшее свойство нейронных сетей, со всей очевидностью проявляющееся при компрессии изображений, – это способность к обобщению и, следовательно, возможность сжатия (путем сопоставления отдельным кадрам нового образа в

виде номеров нейронов-победителей, входящих в натренированную ранее сеть) и декомпрессии (сопоставления векторов весов нейронов-победнтелей соответствующим кадрам) информации. Качество восстановленного изображения, которое не использовалось ранее для обучения, не сильно отличается от качества образа, участвовавшего в обучении, при условии, что степень сложности обоих изображений примерно одинакова. Для примера на рис. 9.9 представлено изображение папиллярных линий, подвергнутое сжатию и декомпрессии с помощью сети Кохонена, натренированной на изображении, приведенном на рис. 9.8. Рисунок 9.9*а* – это оригинальное изображение, рис. 9.9*б* – образ, восстановленный после сжатия, а рис. 9.9*в* иллюстрирует погрешность декомпрессии. Степень искажения восстановленного изображения сопоставима с искажением образа, на котором проводилось обучение сети, а коэффициент искажения PSNR ≅ 26,4 дБ.

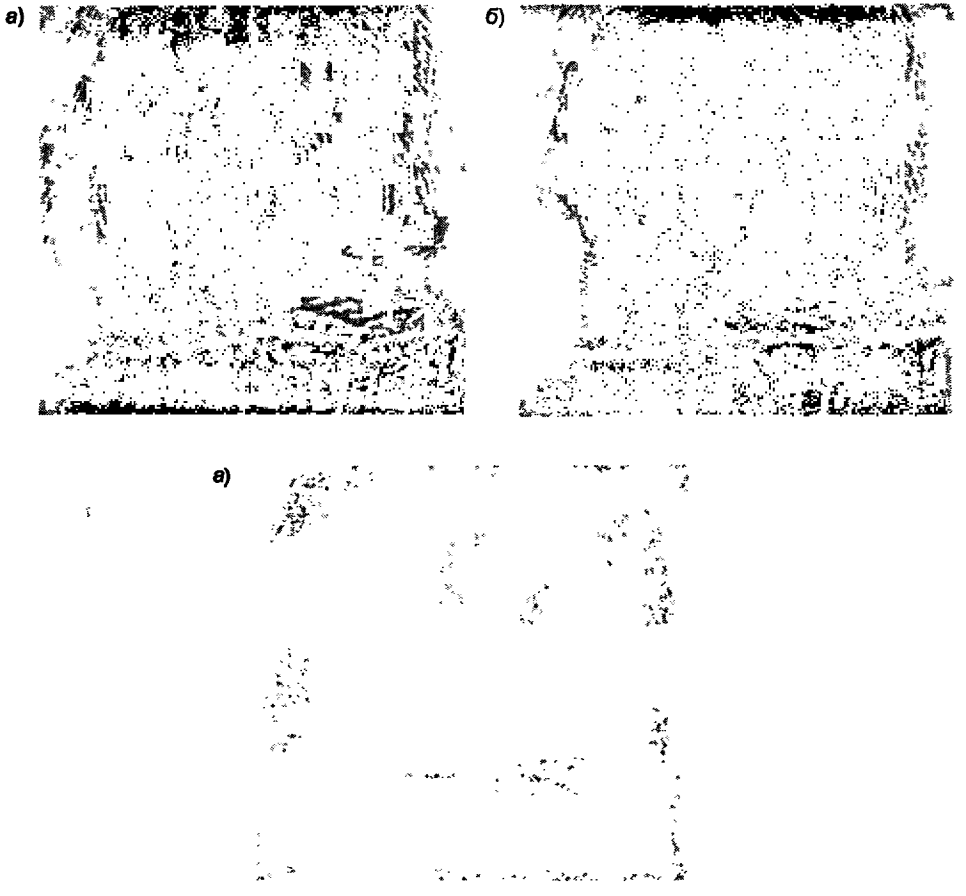

Рис. 9.9. Изображение папиллярных линий, использованных для тестирования сети Кохонена при сжатии данных: *а*) оригинальное изображение; *б*) восстановленное изображение; *в*) разностное изображение

9.5.2. Диагностирование неисправностей оборудования

Сети с самоорганизацией также находят применение для диагностирования неисправностей различного оборудования. При этом используется их способность к компрессии, т.е. к представлению множества точек вектором весов единственного нейрона. Множество контрольных точек устройства, в которых снимаются его характеристики в различных режимах работы, может считаться вектором x (каждый вектор соответствует определенному режиму работы), подаваемым на вход сети. В зависимости от условий работы, вида неисправного элемента и степени повреждения получаются различные характеристики одного и того же устройства. Как правило, неисправность каждого вида связана со специфическим изменением характеристик устройства, свойственным только этой неисправности. Нейрон, побеждающий в конкуренции при определенной комбинации характеристик устройства, представляет впоследствии либо нормальный режим работы, либо определенную неисправность, позволяя тем самым локализовать ее. Типовая схема обнаружения неисправностей представлена на рис. 9.10. База данных состоит из множества характеристик.

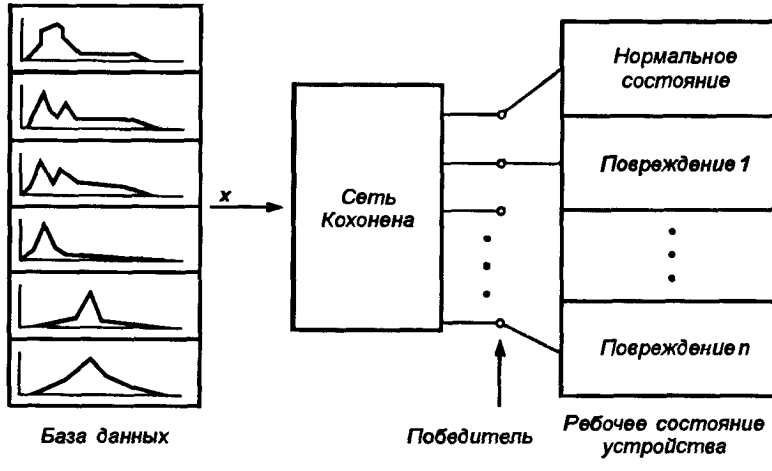

Рис. 9.10. Схема применения сети Кохонена для обнаружения неисправностей в системе

отвечающих различным нормальным и аварийным состояниям в определенных режимах работы, в которых, как правило, устройство подвергается диагностированию. Главное условие корректного функционирования системы – дифференциация характеристик при различных аварийных состояниях. Если две разные аварии имеют идентичные признаки, их различение будет невозможным. Подготовка соответствующей базы данных, по которой будет проводиться обучение, а в последующем эксплуатация сети (собственно диагностирование неисправностей) требуют проведения таких измерений, которые будут однозначно свидетельствовать о фактическом состоянии устройства. При этом следует выделять те фрагменты характеристик, которые отличаются друг от

друга. Для достижения этой цели могут выполняться любые операции (как линейные, так и нелинейные) на всей базе данных. Подобный подход был реализован для диагностирования типовых неисправностей, возникающих в силовых электрических трансформаторах на основе измерений характеристик проводимости $|y| = f(u_{we})$.

В работах [121, 125] рассмотрен пример использования сети Кохонена для диагностирования неисправностей активного электрического фильтра RC на основе частотной характеристики двух функций цепи: падения напряжения и входной проводимости. Сеть, обученная на характеристиках идеальных состояний короткого замыкания и разрыва цепи, оказалась способной определять неисправности при частичных повреждениях (например, значение сопротивления, равное 10% номинала, распознавалось как короткое замыкание).

Способность сети Кохонена активизировать единственный нейрон, ответственный за целый кластер данных, можно использовать для локализации поврежденного элемента независимо от состояния остальных. Для примера продемонстрируем применение сети для диагностирования коротких замыканий в линиях электропередач на основе измерений амплитуды напряжения в определенных точках этой линии [125]. Предполагается, что доступно посекционное измерение напряжения. В этом случае можно определить место аварии с точностью до секции.

Входная информация для сети – это векторы измеренных напряжений. Номер активизированного нейрона указывает на локализацию короткого замыкания. Для линии, на которой доступно большое количество контрольных точек, во

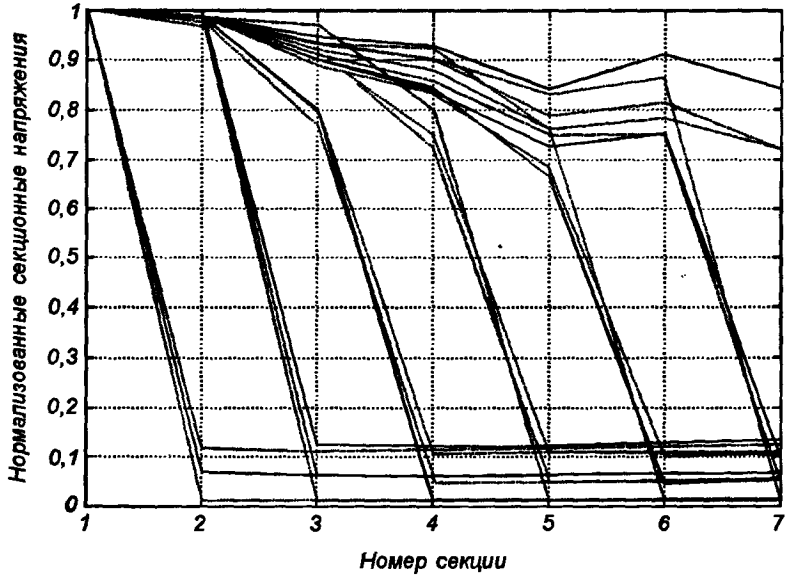

Рис. 9.11. Примерные нормализованные характеристики линии электропередач при различных замыканиях в системе

входных векторах можно ограничиться информацией об амплитуде измеренных напряжений, причем условием корректного функционирования системы считается как можно большее различие входных векторов, соответствующих разным аварийным ситуациям. Для проявления этих различий применяются многочисленные приемы предварительной обработки измерительных данных, отдаляющие эти векторы друг от друга. Из анализа снятой с линий электропередач измерительной информации следует, что эффективным решением может считаться нормализация истинных значений напряжения в контрольных точках, отражающая только разницу между напряжениями в соседних секциях.

В работе [125] предложено проводить нормализацию согласно формуле

$$x_i = \frac{v_i - v_{i-1}}{E}, \qquad (9.31)$$

где v_i обозначает измерение в контрольной точке i-й секции, а E – входное напряжение линии. На рис. 9.11 представлены характеристики распределения нормализованных таким образом измеренных значений для шести секций линии при трех различных закорачивающих резисторах, включенных в узловых точках секций. Они образуют обучающие векторы для сети Кохонена, использованной при решении задачи. Исследовалась секционная модель кабельной линии со следующими параметрами на 1 км длины: R = 70 Ω, L = 80 mH, C = 47 nF, G = 0,01 S. Эксперименты проводились для частоты 3825 Гц с возможностью неполного закорачивания в секциях (путем подключения к соответствующим участкам линии резистора с сопротивлением от 0 Ω при идеальном коротком замыкании до 1/3 значения модуля волновой импедантности). При исследовании замыканий (одиночных) предполагалась относительная нечувствительность к ним (до 10%) значений других характеристик неповрежденных элементов линии. Анализ характеристик линии при изменении сопротивления перемычки свидетельствует о близости (в пространстве параметров) точек, соответствующих нормализованным характеристикам аварий одного типа, группирующихся в виде кластеров. Они с успехом могут представляться одиночными нейронами Кохонена либо выходными нейронами персептронной сети. Оба метода проверялись в работе [125], показавшей высокую эффективность их применения для локализации места замыкания. При 10%-ной нечувствительности неповрежденных элементов 100%-ная результативность диагностирования и локализации замыкания обеспечивается обеими сетями при сопротивлении перемычки, доходящем до 0,2 значения волновой импедантности. При большей неидеальности замыкания (до 0,4 значения волновой импедантности) эффективность диагностирования начинает изменяться. При 10%-ной нечувствительности неповрежденных элементов обеспечивается 98%-ная результативность локализации места замыкания многослойным персептроном и только 85%-ная – сетью Кохонена. Причина этого скорее всего кроется в существенной нечеткости признаков замыкания, связанных с его неидеальностью, поэтому обучение с учителем, которому подвергается многослойный персептрон, дает лучшие результаты.

9.5.3. Краткосрочное прогнозирование нагрузок энергетической системы

Сеть с самоорганизацией также с успехом может использоваться для прогнозирования, например, нагрузок в электроэнергетической системе. В настоящем подразделе будут представлены подробности решения задачи прогнозирования часовых нагрузок в электроэнергетической системе на 24-часовом интервале. Использованный подход во многом совпадает с методикой распознавания образов, представленной в предыдущих подразделах. Сеть обучается распознавать параметры часовых нагрузок, характерные для различных дней года. В разделе 4 было показано, что каждый день года имеет свою специфику распределения часовых нагрузок, которая от года к году меняется лишь в незначительной степени. Некоторые дни, относящиеся к одной поре года и имеющие один и тот же тип, различаются минимально. В многомерном пространстве они образуют компактные группы данных, каждая из которых может отображаться весами единственного нейрона-победителя. Чтобы отстроиться от определенного общего тренда, обусловленного ежегодным промышленным ростом страны, принимаются во внимание только переменные части характеристик, остающиеся после вычитания среднего значения. Если обозначить среднее значение нагрузки системы в j-й день $P_m(j)$, а его вариацию – $\sigma(j)$, то можно определить часовой профиль j-го дня в виде

$$p(j,h)=\frac{P(j,h)-P_m(j)}{\sigma(j)}, \qquad (9.32)$$

для $h=$ 1, 2, ..., 24, где $P(j,h)$ обозначает фактический отбор мощности в электроэнергетической системе в h-й час j-го дня. Значения $p(j,h)$ составляют вектор профильных нагрузок дня, $p_j = [p(j,1), p(j,2), ..., p(j,24)]$. За начальную точку отсчета времени можно выбрать часы с наиболее стабильной нагрузкой в масштабах всего года. Это, как правило, ночное время (между 3 и 5 часами). В этом случае компонент $p(j,1)$ будет соответствовать действительной нагрузке в первый час этого интервала, $p(j,2)$ – во второй час и т.д. Для каждого дня года, представленного в базе данных, формируется профильный вектор в соответствии с формулой (9.32). Для уменьшения влияния случайных нагрузок база данных должна охватывать несколько последних лет. Множество профильных векторов подается на вход сети Кохонена, состоящей из n нейронов. Процесс самоорганизации сети приводит к автоматической кластеризации данных и к сопоставлению каждому кластеру одного из нейронов сети. Этот нейрон считается победителем, а его веса наилучшим образом адаптируются к усредненным весам профильных векторов, составляющих кластер. Характерная особенность состоит в том, что соседние векторы имеют сходные профильные характеристики.

Близость весов нейронов, расположенных недалеко друг от друга, легко объяснить, если принять во внимание механизм соседства в алгоритмах самоорганизации. Это означает, что один и тот же день в разные годы при небольших отличиях в часовых нагрузках может возбуждать различные нейроны, расположенные недалеко друг от друга и образующие своего рода кластеры, группирующие данные сходных классов. Если нанести веса нейронов на плоскость (x, y) и приписать каждому из них виды дней, в которые они становились победителями, можно наглядно выделить обширные области, характерные для праздничных и для рабочих дней. Это подтверждает известный в энергетике факт подобия профилей нагрузок для рабочих дней и близости профилей для праздничных дней.

Знание таблицы распределения побед конкретных нейронов сети позволяет относительно легко предвидеть профили часовых нагрузок для произвольного дня года. С этой целью создаются таблицы принадлежности каждого дня года к области доминирования определенного нейрона с обозначением количества его побед для всех дней в прошлом. Для примера в табл. 9.1 представлены данные, касающиеся июльских вторников на протяжении последних пяти лет (для моделирования использовалось 100 нейронов, упорядоченных в табл. 10×10).

Таблица 9.1

Распределение побед нейронов при прогнозе нагрузок в июльские вторники

Месяц	День	Нейрон	Количество побед
Июль	Вторник	34	5
Июль	Вторник	35	6
Июль	Вторник	43	4
Июль	Вторник	44	6

Для выбора прогнозируемого профиля нагрузок актуального дня (например, вторника) в требуемом месяце (например, в июле) рассчитываются усредненные значения весов нейронов-победителей, которые указывали в прошлом на требуемый день. Если количество побед i-го нейрона, соответствующее j-му дню, обозначить k_{ji}, а соответствующие векторы весов класса – w_i, то прогнозируемый профильный вектор j-го дня рассчитывается по формуле

$$\hat{p}_j = \frac{\sum_{i=1}^{n} k_{ji} w_i}{\sum_{i=1}^{n} k_{ji}}, \qquad (9.33)$$

где $k_{ji} = 0$, если соответствующий нейрон никогда не побеждал в данной классификации. На рис. 9.12 в качестве примера представлены профильные характеристики, полученные этим методом для четырех дней: 18.02.1994,

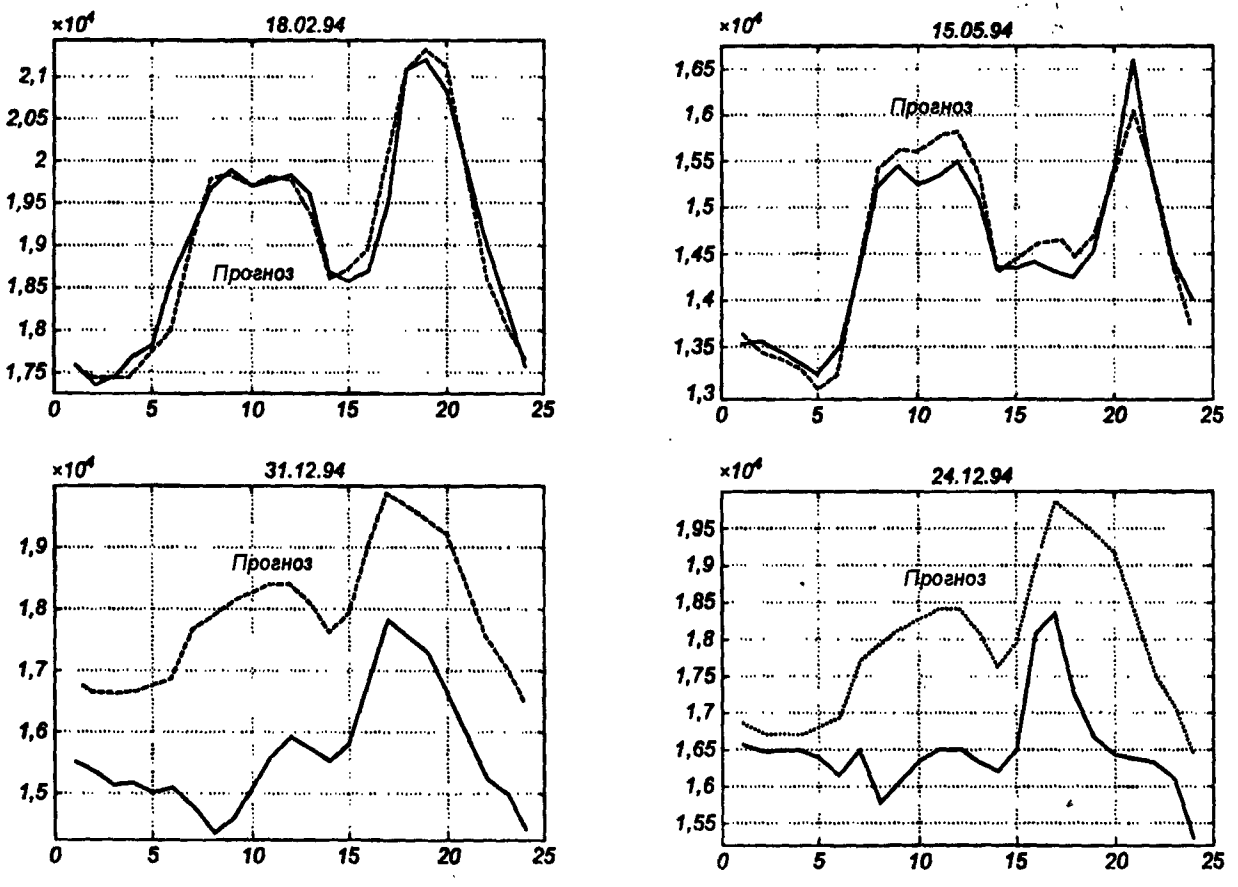

Рис. 9.12. Характеристики профилей нагрузок Польской энергетической системы для избранных дней года: сплошная линия – фактические значения, пунктирная линия – прогноз

15.05.1994, 31.12.1994 и 24.12.1994. Сплошная линия соответствует фактическим значениям, а пунктирная – прогнозу. Для типичных дней эти кривые с высокой точностью совпадают (два верхних графика). Значительные отличия наблюдаются для нетипичных дней, например последнего дня года и Сочельника[1] (два нижних графика). Обычная методика применения сети Кохонена в таких случаях оказывается недостаточной. Возникает необходимость введения для особенного дня отдельной категории, по которой прогноз строится на основе таких же дней в прошлом.

После определения профильного вектора фактические нагрузки, соответствующие конкретным часам данного дня, рассчитываются на основе формулы (9.32) как

$$\hat{P}(j) = \sigma(j)\hat{p}_j + P_m(j) . \qquad (9.34)$$

Комплексный прогноз нагрузок требует дополнительного предсказания среднего значения и вариации для каждого дня, на который этот прогноз составляется. Такое предсказание может быть выполнено с использованием статистического анализа прошлых средних значений и их вариаций либо путем применения специальной персептронной сети, обученной решению только этой задачи (см. раздел 4).

Благодаря использованию для предсказания средних значений и их вариаций отлаженной нейронной технологии, реализованной в корректно спроектированной и обученной сети с самоорганизацией, стало возможным обеспечить в польских условиях точность прогнозирования нагрузок, характеризующуюся погрешностью MAPE порядка 2,5%.

9.6. Гибридная сеть

Главная особенность сети с самоорганизацией на основе конкуренции – это очень высокая скорость обучения, многократно большая, чем у сетей, тренируемых с учителем. Их недостатком считается сложность отображения пар обучающих данных (x, d), поскольку сеть с самоорганизацией, выполняющая обработку только входного вектора x, не обладает свойствами хорошего аппроксиматора, присущими многослойному персептрону или радиальной сети RBF. Очень хорошие результаты удается получить при объединении самоорганизующегося слоя и персептронной сети, что позволяет совместить способности сети Кохонена к локализации и возможности аппроксимации, свойственные многослойному персептрону. Подобная структура, которая в последующем будет называться гибридной сетью, изображена на рис. 9.13. Она представляет собой каскадное подключение слоя Кохонена и персептронной сети (часто оказывается достаточным один персептронный слой). Самоорганизующийся слой улавливает

[1] В католической Польше Рождество Христово отмечается 25 декабря – *Примеч. перев.*

значимые признаки процесса (локализует их на основе входных данных x), после чего им приписывается входной вектор в персептронном слое. Вследствие хорошей локализации признаков процесса первым слоем сети в большинстве приложений бывает достаточным применение персептрона, содержащего только один слой нейронов (зачастую линейных). Обучение гибридной сети состоит из двух отдельных этапов, следующих друг за другом.

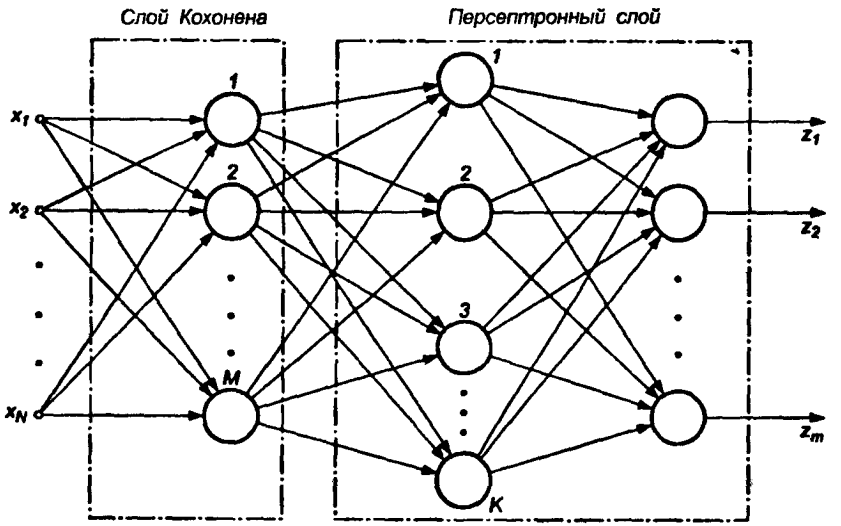

Рис. 9.13. Структура гибридной сети

Вначале на множестве входных векторов x обучается слой Кохонена. В результате нейроны этого слоя организуются таким образом, что векторы их весов наилучшим образом (с минимальной погрешностью квантования) отображают распределение данных обучающих векторов x. Тренинг слоя Кохонена проводится по одному из реализованных в программе *Kohon* алгоритмов обучения сети с самоорганизацией (например, нейронного газа или WTA с механизмом учета усталости). По завершении обучения слоя Кохонена веса его нейронов замораживаются и проводится анализ их выходных сигналов при подаче на вход сети последовательности сигналов x из обучающего множества. Победитель (нейрон с наибольшим значением суммарного сигнала u_{max}) переводится в единичное состояние, а остальным нейронам приписываются состояния из интервала $(0-1)$. Переход от фактических суммарных сигналов u_i этих нейронов к выходным нормализованным сигналам может проводиться по различным формулам. Опыт показывает, что хорошие результаты можно получить с использованием выражения

$$y_j = \exp\left(-\frac{|u_{\max} - u_i|^2}{\sigma^2}\right), \tag{9.35}$$

при значениях σ, индивидуально подбираемых для каждой решаемой задачи.

- Персептронная сеть обучается с учителем по завершении тренинга самоорганизующегося слоя. Обучающими сигналами для нее является множество пар (y_i, d_i), где y_i – это вектор, составленный из выходных сигналов нейронов слоя Кохонена, а d_i – вектор ожидаемых значений оригинального отображения (x_i, d_i), которому соответствует вектор y_i. Сеть обучается как по алгоритму обратного распространения, так и градиентным методом (программа *Netteach* применяется так же, как и для обычной персептронной сети). Процесс обучения в этом случае может протекать во много раз быстрее, чем для одиночной персептронной сети, благодаря хорошей локализации данных, обеспеченной первым слоем Кохонена. Это особенно заметно на сети с одним персептронным слоем. Если принять во внимание, что каждому возбуждению x соответствует конкретный доминирующий нейрон (победитель), то можно осуществить предварительную инициализацию весов персептронного слоя таким образом, что значения весов, соединяющих нейрон-победитель с выходными нейронами гибридной сети, будут усредняться по тем векторам d, на которых были одержаны победы. При такой предварительной инициализации весов обучение персептронного слоя сводится к незначительной корректировке их значений, отражающей влияние проигравших нейронов Кохонена на окончательный результат. Эта корректировка, как правило, требует небольшого количества итераций и ведет к достижению глобального минимума функции погрешности.

Следует отметить, что представленная гибридная сеть может считаться обобщением сети обратного распространения Хехта–Нильсена [50, 113, 118]. В отличие от нее в гибридной сети допускается дробная активность нейронов в слое Кохонена от значения 1 для победителя до $0 \le y_i < 1$ для остальных нейронов. Активность нейронов, проигравших в конкурентной борьбе, рассчитывается согласно формуле (9.35). Учет активности многих нейронов при функционировании персептронной сети позволяет лучше локализовать вектор x в многомерном пространстве и получить лучшее отображение данных нейронной сетью в целом.

Это оригинальное свойство гибридной сети может иметь различные практические приложения. Например, ее применение для предсказания профилей нагрузок в электроэнергетической системе позволило значительно уменьшить как погрешность MAPE, так и максимальную погрешность самого прогноза.

В табл. 9.2 приведены результаты сравнения точности прогиозов нагрузок для Польской электроэнергетической системы, полученных на основе классической сети Кохонена и гибридной сети [124]. Во втором случае результаты были получены при подкреплении прогнозирования персептронной сетью на этапе расчета средних значений и их вариаций, используемых в формуле (9.34). Из приведенных данных следует, что гибридная сеть позволила уменьшить как погрешность MAPE, так и максимальную погрешность прогноза.

Таблица 9.2

Погрешности MAPE и MAX прогноза нагрузок при использовании сети Кохонена и гибридной сети

Год	Сеть Кохонена		Гибридная сеть	
	MAPE	MAX	MAPE	MAX
1991	2,38%	20,02%	2,24%	17,19%
1992	2,42%	21,07%	2,27%	16,16%
1993	2,73%	19,79%	2,54%	14,08%
1994	2,49%	20,03%	2,28%	14,93%
1995	2,56%	16,85%	2,41%	13,97%

Хорошие результаты получаются при применении гибридной сети для анализа состава газовых смесей и оценивания концентрации отдельных компонентов по показаниям полупроводниковых датчиков (так называемый искусственный нос) [118]. Полупроводниковые датчики характеризуются относительно слабой селективностью; они в разной степени реагируют на присутствие в смеси различных газов. В технических решениях применяются матрицы датчиков, по-разному реагирующих на присутствие конкретного газа. В этой ситуации задача нейронной сети состояла в калибровке прибора, т.е. в

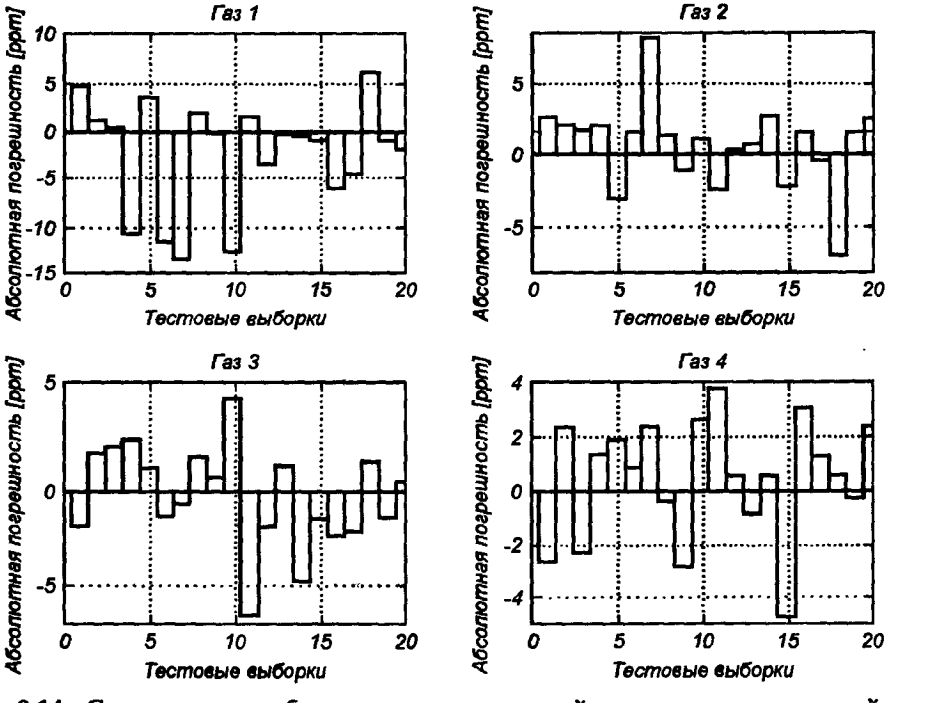

Рис. 9.14. Сопоставление абсолютных погрешностей оценки состава смесей из четырех газов (количество измерений равно 20), полученных в результате применения гибридной сети

сопоставлении показаниям каждого датчика соответствующих значений концентрации конкретных компонентов газовой смеси. Для достижения поставленной цели была создана гибридная нейронная сеть, в которой слой Кохонена определял тип газовых компонентов, а персептронный слой оценивал их концентрацию.

На рис. 9.14 представлены графики погрешностей оценок присутствия четырех газов (в ppm): углекислого газа, метана, метанола и пропана/бутана, полученных при использовании гибридной нейронной сети [118]. Обучение сети проводилось на 400 обучающих выборках, а ее тестирование – на 20 других выборках.

Максимальная погрешность для тестовых примеров не превышала 12 ppm, а средняя абсолютная погрешность оставалась на уровне 2,59 ppm. На рис. 9.15 представлены графики относительных погрешностей оценок по 20 тестовым

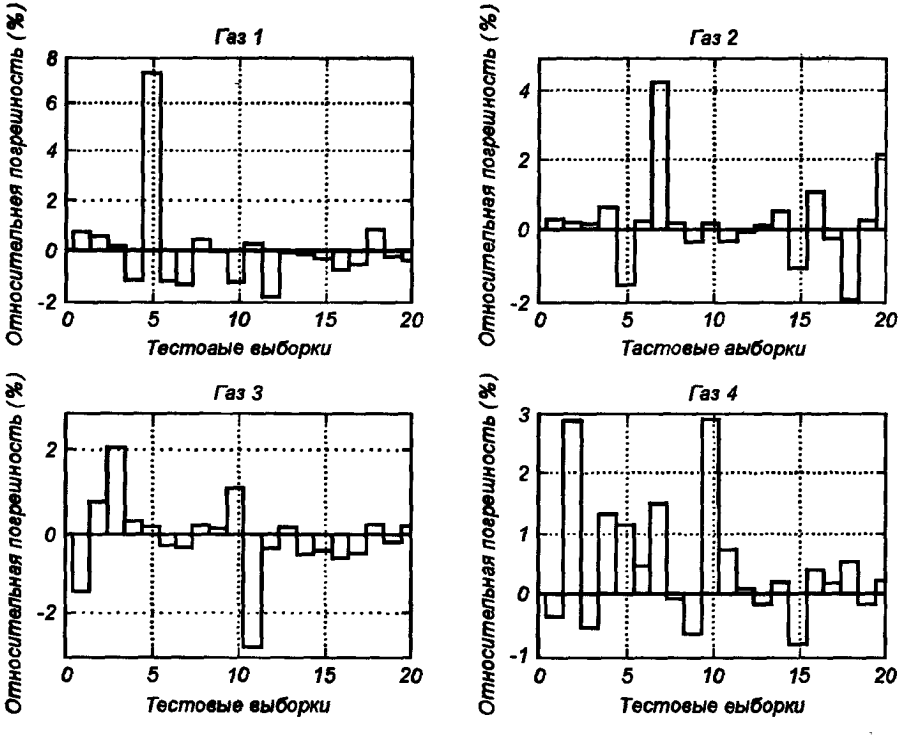

Рис. 9.15. Сопоставление относительных погрешностей оценки состава смесей из четырех газов (количество измерений равно 20), полученных в результате применения гибридной сети

выборкам, соответствующие результатам, представленным на рис. 9.14. Значение максимальной относительной погрешности не превысило 7%, а абсолютное среднее значение относительной погрешности распознавания всех компонентов исследованных газовых смесей составило 0,78%.

Раздел 10

СЕТИ С САМООРГАНИЗАЦИЕЙ КОРРЕЛЯЦИОННОГО ТИПА

Другой важный класс сетей с самоорганизацией составляют сети, в процессе обучения которых используется информация о зависимостях между сигналами. Эти сети называются *корреляционными*, или *хеббовскими*. В процессе обучения они выявляют значимые корреляционные зависимости между сигналами, подстраиваясь под них путем адаптации значений синаптических весов.

В этом разделе будут обсуждаться различные аспекты самоорганизации на основе обобщенных правил Хебба. Мы представим два вида хеббовских сетей: сеть, выполняющую декомпозицию данных по главным компонентам, называемую *сетью PCA* (англ.: *Principal Component Analysis* – анализ главных компонентов), и сеть, декомпозирующую обучающие данные на независимые компоненты, называемую *сетью ICA* (англ.: *Independent Component Analysis* – анализ независимых компонентов). Обе сети по своей природе линейны (линейны как сами нейроны, так и межнейронные связи), хотя алгоритмы обучения имеют нелинейный характер.

10.1. Энергетическая функция корреляционных сетей

Применение основного правила Хебба связано с использованием модели линейного нейрона, в соответствии с которой выходной сигнал y_i нейрона равен взвешенной сумме входных сигналов:

$$y_i = \sum_{i=0}^{N} w_{ij} x_i \ . \qquad (10.1)$$

В этой модели используется вектор $x = [x_0, x_1, ..., x_N]$, дополненный нулевым компонентом $x_0 = 1$, обозначающим поляризацию. В соответствии с постулатом Хебба изменение веса нейрона после предъявления вектора x производится по формуле

$$\Delta w_{ji} = \eta(y_i - y_i^{(0)})(x_k - x_k^{(0)}) \ , \qquad (10.2)$$

где $x_k^{(0)}$, $y_i^{(0)}$ являются определенными константами, а η – коэффициент обучения. С учетом всего множества обучающих выборок изменение значений весов сети во времени может быть выражено обобщенной формулой [46]

$$\frac{dw_{jk}}{dt} = \sum_{i=1}^{N} w_{ji} C_{ik} + k_1 + \frac{k_2}{N} \sum_{i=1}^{N} w_{ji} , \tag{10.3}$$

в которой k_1 и k_2 – это определенные константы, связанные с $x_k^{(0)}$, $y_i^{(0)}$ и с η, тогда как C_{ik} – это усредненная ковариация активности i-го и k-го нейронов, определяемая в виде

$$C_{ik} = \frac{1}{p} \sum_{j=1}^{p} \left(x_i^{(j)} - \bar{x}_i\right)\left(x_k^{(j)} - \bar{x}_k\right). \tag{10.4}$$

Константой $\bar{x}_i$ обозначено усредненное значение входных выборок по i-му компоненту усредненного вектора $\bar{x}$, где $\bar{x} = \frac{1}{p} \sum_{k=1}^{p} x^{(k)}$. Если принять, что изменения весов производятся в соответствии с правилом максимального уменьшения значения энергетической функции E сети, получаем

$$\frac{dE}{dw_{jk}} = -\frac{dw_{jk}}{dt} = -\sum_{i=1}^{N} w_{ji} C_{ik} - k_1 - \frac{k_2}{N} \sum_{i=1}^{N} w_{ji} . \tag{10.5}$$

После решения этого дифференциального уравнения получаем энергетическую функцию в виде [46]

$$E = E_v + E_k , \tag{10.6}$$

где

$$E_v = -\frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} w_{ji} C_{ik} w_{kj} , \tag{10.7}$$

$$E_k = -k_1 \sum_{i=1}^{N} w_{ji} - \frac{k_2}{2N} \left(\sum_{i=1}^{N} w_{ji} \right)^2 . \tag{10.8}$$

Энергетическая функция содержит два слагаемых: E_v и E_k. Первый элемент определяет вариацию σ_j^2 активности j-го нейрона; так как $\sigma_j^2 = \left\langle (y_j - \bar{y})^2 \right\rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ji} C_{ik} w_{jk}$. Второе слагаемое можно отождествить со штрафной компонентой энергетической функции, иначе называемой в теории оптимизации стоимостной или целевой функцией. При зафиксированных значениях весов функция E будет принимать минимальные значения тогда, когда вариация σ_j^2 будет максимальной. В связи с этим обучение по Хеббу приводит к такой организации нейронов (подбором их весов), которая максимизирует

вариацию активности нейронов при определенном ограничении значений их весов. Поскольку в общем случае эти весовые ограничения не уточняются, функция может иметь несколько локальных максимумов.

10.2. Нейронные сети PCA

10.2.1. Математическое введение

Анализ главных компонентов (PCA) – это статистический метод, определяющий линейное преобразование $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$. Оно трансформирует описание стационарного стохастического процесса, представленного вектором $\boldsymbol{x} \in R^N$, в вектор $\boldsymbol{y} \in R^K$ посредством матрицы $\mathbf{W} \in R^{K \times N}$ при $K<N$ таким образом, что выходное пространство редуцированного размера сохраняет наиболее важную информацию об исходном процессе. Другими словами, преобразование по методу PCA позволяет заменить большое количество информации, основанной на взаимно коррелирующих входных данных, множеством статистически независимых компонентов с учетом их важности. Поэтому оно считается одной из форм компрессии с потерей информации, известной в теории связи как преобразование Карьюнена–Лёве [29, 82].

Пусть $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$ обозначает случайный вектор с нулевым средним значением, а $\mathbf{R}_{xx} = E[\boldsymbol{x}\boldsymbol{x}^T] = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$ обозначает ожидаемое (среднее) значение матрицы автокорреляции по всем векторам $\boldsymbol{x}$. Эту матрицу при конечном количестве p векторов $\boldsymbol{x}$ можно описать зависимостью

$$\mathbf{R}_{xx} \approx \frac{1}{p} \sum_{k=1}^{p} \boldsymbol{x}_k \boldsymbol{x}_k^T = \frac{1}{p} \mathbf{X}\mathbf{X}^T , \tag{10.9}$$

где матрица данных $\mathbf{X}$ образована последовательностью обучающих векторов $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p]$.

Обозначим λ_i собственные значения матрицы автокорреляции $\boldsymbol{R}_{xx}$, а $\boldsymbol{w}_i$ – сопряженные с ними ортогональные векторы собственных значений, где $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{iN}]^T$. Собственные значения и собственные векторы связаны зависимостью

$$\mathbf{R}_{xx} \boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i , \tag{10.10}$$

где $i = 1, 2, \ldots, N$. Собственные значения симметричной неотрицательной матрицы корреляции являются рациональными и неотрицательными. Упорядочим их в порядке убывания: $\lambda_1 > \lambda_2 > \ldots > \lambda_N \geq 0$. В аналогичной последовательности расположим собственные векторы w_i, сопряженные с λ_i. Если ограничиться K максимальными собственными значениями, матрица $\mathbf{W}$ преобразования PCA может быть определена в форме $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]^T$, при $\mathbf{W} \in R^{KXN}$. Эта матрица определяет преобразование PCA как линейное преобразование

$$\boldsymbol{y} = \mathbf{W}\boldsymbol{x}. \tag{10.11}$$

Вектор $y = [y_1, y_2, ..., y_K]^T$ представляет собой вектор главных компонентов PCA, имеющих наибольшее влияние на реконструкцию вектора данных $x = [x_1, x_2, ..., x_N]^T$.

Таким образом, преобразование PCA тесно связано с разложением матрицы корреляции в соответствии с собственными значениями. Если обозначить L диагональную матрицу, сформированную из использованных в отображении собственных значений λ_i, т.е. $\mathbf{L} = [\lambda_1, \lambda_2, ..., \lambda_K]$, то матрицу корреляции можно представить в виде следующей декомпозиции:

$$\mathbf{R}_{xx} = \mathbf{W}^T\mathbf{L}\mathbf{W}. \tag{10.12}$$

С позиций статистики преобразование PCA определяет множество K ортогональных векторов (строк матрицы W), имеющих наибольшее влияние на вариацию входных данных. Первый главный компонент $y_1 = w_1^T x$ определяет нормализованную линейную комбинацию тех компонентов входных векторов, которые дают наибольшее среднее значение вариации, равное λ_1: $\mathrm{var}(w_1^T x) = E[\|w_1^T x\|^2] = E[w_1^T R_{xx} w_1] = \lambda_1$. Задача алгоритмов PCA состоит в определении направлений $w_1, w_2, ..., w_K$ (называемых главными собственными векторами) таким образом, чтобы максимизировать значение $E(\|w_1^T x\|^2)$ при выполнении условия ортогональности $w_1^T w_j = 0$ для $j \geq i$ $(i = 1, 2, ..., K)$ и $w_1^T w_j = 1$.

Реконструкция x на основе вектора y и ортогональной матрицы W проводится в соответствии с выражением [29]

$$\hat{x} = W^T y. \tag{10.13}$$

Матрица разложения PCA (W) и матрица реконструкции ($\mathbf{W}^T$) составляют взаимную транспозицию. PCA минимизирует значение ожидаемой погрешности реконструкции данных, которая описывается формулой

$$E_r = E[\|x - \hat{x}\|^2]. \tag{10.14}$$

Из практики известно [29], что при ограничении K наибольшими собственными значениями (K главными компонентами) эту погрешность можно выразить зависимостью

$$E_r = \sum_{i=K+1}^{N} \lambda_i. \tag{10.15}$$

Минимизация погрешности реконструкции данных равнозначна максимизации вариации проекции

$$\max E_v = \sum_{i=1}^{K} \lambda_i. \tag{10.16}$$

Как E_r, так и E_v являются неотрицательными, поскольку все собственные значения матрицы корреляции, как матрицы симметричной и неотрицательно определенной, являются положительными или нулевыми.

Представление вектора x наибольшими главными компонентами $y_1, y_2, ..., y_K$, составляющими вектор y, равнозначно сохранению информации о наибольшей части энергии, содержащейся во множестве данных. Первый (наибольший) главный компонент, сопряженный с λ_1 своим собственным вектором w_1, определяет направление в многомерном пространстве, в котором вариация данных максимальна. Последний наименьший главный компонент (англ.: *Minor Principal Component*) указывает направление, в котором вариация минимальна.

На рис. 10.1 представлена геометрическая интерпретация наиболее значимого и наименее значимого главных компонентов преобразования PCA. Первый главный компонент соответствует направлению наибольшей вариации (энергии) сигналов. При представлении данных только с помощью единственного главного компонента и сопряженного с ним собственного вектора с последующим выбором в качестве результата наибольшего из главных компонентов (y_1) допускается наименьшая погрешность реконструкции и одновременно максимизируется вариация преобразования. Наименее значимый главный компонент оказывает наименьшее влияние на точность восстановления данных. Поэтому для сжатия данных (уменьшения количества информации с минимальными потерями для ее реконструкции) необходимо их представление множеством наибольших главных компонентов. Игнорирование наименьших компонентов оказывает минимальное воздействие на точность реконструкции данных.

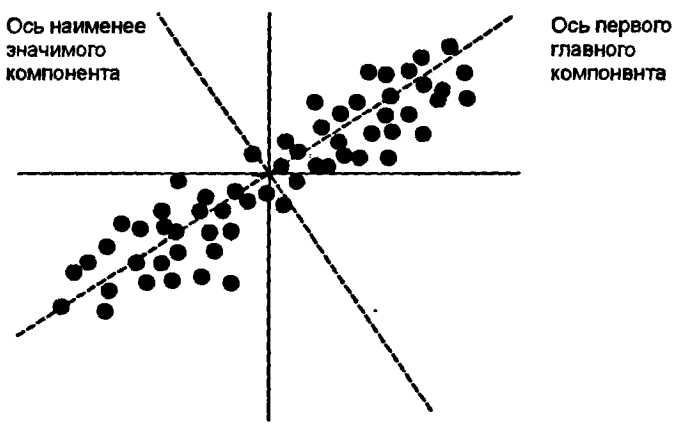

Рис. 10.1. Иллюстрация главных компонентов для группы результатов измерений

Преобразование PCA позволяет определить корреляцию, возникающую между различными переменными, составляющими множество данных. Если эти переменные коррелируют между собой, то знание только части из них будет достаточным для определения остальных. Поэтому такое множество данных может представляться меньшим количеством переменных. В случае, когда переменные не коррелируются, восстановление части из них на основании знания других данных становнтся невозможным.

В качестве примера, иллюстрирующего разложение на главные компоненты, рассмотрим фактическую корреляцию, существующую между длиной, шириной и высотой особей, составляющих популяцию черепах [29]. Вектор измерений $\boldsymbol{x}$ в этой ситуации образуют три компонента: длина d, ширина s и высота w: $\boldsymbol{x} = [d, s, w]$. Т. П.Жоликур и Ж.Мосманн [29] провели измерения этих параметров для популяции размером $p = 24$ особи. Была получена матрица корреляции $\mathbf{R}_{xx}$ в виде

$$\mathbf{R}_{xx} = \begin{bmatrix} 451{,}3 & 271{,}2 & 168{,}7 \\ 271{,}2 & 171{,}7 & 103{,}3 \\ 168{,}7 & 103{,}3 & 66{,}65 \end{bmatrix}.$$

Декомпозируя эту матрицу по собственным значениям, получаем $\lambda_1 = 680{,}37$, $\lambda_2 = 6{,}43$, $\lambda_3 = 2{,}81$, а также сопряженные с ними собственные векторы:

$$w_1 = \begin{bmatrix} 0{,}8126 \\ 0{,}4955 \\ 0{,}3068 \end{bmatrix}, \quad w_2 = \begin{bmatrix} -0{,}5454 \\ 0{,}8322 \\ 0{,}1003 \end{bmatrix}, \quad w_3 = \begin{bmatrix} 0{,}2056 \\ 0{,}2488 \\ -0{,}9465 \end{bmatrix}.$$

На их основе определяется матрица $\mathbf{W} = [w_1, w_2, w_3]^T$ преобразования PCA в виде

$$\mathbf{W} = \begin{bmatrix} 0{,}8126 & 0{,}4955 & 0{,}3068 \\ -0{,}5454 & 0{,}8322 & 0{,}1003 \\ 0{,}2056 & 0{,}2488 & -0{,}9465 \end{bmatrix},$$

а также диагональная матрица $\mathbf{L}$, образованная собственными значениями $\lambda_1, \lambda_2, \lambda_3$ матрицы $\mathbf{R}_{xx}$, $\mathbf{L} = \mathrm{diag}[680{,}37,\ 6{,}43,\ 2{,}81]$.

Наибольшее собственное значение $\lambda_1 = 680{,}37$ определяет первый главный компонент, сопряженный с собственным вектором w_1, составляющим первую строку матрицы $\mathbf{W}$. Этот компонент при входном векторе $\boldsymbol{x}$, состоящем из трех элементов (длина d, ширина s и высота w), описывается выражением $y_1 = w_1^T x$. которое в нашем случае приобретает конкретный вид: $y_1 = 0{,}8126d + 0{,}4955s + 0{,}3068w$. Каждое из собственных значений λ_i соответствует вариации, которую представляет i-й главный компонент. Относительный вклад каждого главного компонента в общую вариацию данных (энергию) можно определить выражением $m_i = \lambda_i / \sum_{j=1}^{3} \lambda_j$. В рассматриваемом примере этот вклад составляет: $m_1 = 0{,}9866$, $m_2 = 0{,}0093$, $m_3 = 0{,}0041$. Анализ полученных значений говорит о том, что доля первого главного компонента в суммарной вариации данных составляет 98,66%. При восстановлении длины, ширины и высоты черепахи на основании вектора $\boldsymbol{y}$ можно ограничиться его наибольшей составляющей y_1 и проигнорировать остальные, так как они не несут существенной информационной нагрузки. Это означает возможность трехкратного уменьшения количества обрабатываемой информации.

При обсуждении преобразования PCA следует подчеркнуть связь между собственными значениями матрицы автокорреляции $\mathbf{R}_{xx}$ и особенными значениями s_i матрицы $\mathbf{X}$, составляющими матрицу $\mathbf{R}_{xx}$ ($\mathbf{R}_{xx} = \frac{1}{p}\mathbf{X}\mathbf{X}^T$). Особенные значения s_i образуют псевдодиагональную матрицу $\mathbf{S}$, которая является одной из составляющих SVD-разложения матрицы $\mathbf{X}$. SVD-разложение этой матрицы определяется формулой

$$\mathbf{X} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T. \tag{10.17}$$

Матрицы $\mathbf{U} \in R^{N\times N}$ и $\mathbf{V} \in R^{p\times p}$ являются ортогональными, а псевдодиагональная матрица $\mathbf{S} \in R^{N\times p}$ содержит неотрицательные диагональные элементы. Разложение $\mathbf{R}_{xx}$ на K главных компонентов соответствует выделению при SVD-разложении матрицы $\mathbf{X}$ только K наибольших особенных значений и сопряженных с ними K столбцов матриц $\mathbf{U}$ и $\mathbf{V}$.

Кроме того, существует тесная связь между собственными значениями матрицы $\mathbf{R}_{xx}$ и особенными значениями матрицы $\mathbf{X}$. Если определить SVD-разложение матрицы $\mathbf{X}$ в форме $\mathbf{X} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$ при правостороннем умножении $\mathbf{X}$ на $\mathbf{X}^T$, то получим:

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T\mathbf{V}\,\mathbf{S}^T\mathbf{U}^T = \mathbf{U}\,\mathbf{S}\,\mathbf{S}^T\mathbf{U}^T. \tag{10.18}$$

Вследствие псевдодиагональности матрицы $\mathbf{S}$ произведение этой матрицы на $\mathbf{S}^T$ дает диагональную матрицу $\mathbf{D}$ с элементами, равными квадратам элементов s_i матрицы $\mathbf{S}$, т.е.

$$\mathbf{D} = \mathrm{diag}[\, s_1^2\ , s_2^2\ , \ldots,\ s_N^2\ ].$$

В результате получаем:

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{D}\mathbf{U}^T. \tag{10.19}$$

Принимая во внимание, что $\mathbf{R}_{xx} = \frac{1}{2}\mathbf{X}\mathbf{X}^T$, SVD-разложение матрицы $\mathbf{X}$ точно соответствует разложению PCA матрицы корреляции, определенному выражением (10.12) при $\mathbf{L} = \frac{1}{2}\mathbf{D}$ и $\mathbf{W} = \mathbf{U}^T$. Главные векторы w_i отождествляются со столбцами ортогональной матрицы $\mathbf{U}$, полученной в результате SVD-разложения матрицы данных $\mathbf{X}$.

Аналогичным образом можно доказать, что при левостороннем умножении матрицы $\mathbf{X}$ на матрицу $\mathbf{X}^T$ получаем:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^T. \tag{10.20}$$

В этом случае роль матрицы $\mathbf{U}$ принимает на себя матрица $\mathbf{V}$, также полученная в результате SVD-разложения матрицы $\mathbf{X}$.

Стандартные методы определения собственных векторов (продолжение декомпозиции QR [42]) матрицы $\mathbf{R}_{xx}$ при больших размерностях векторов x имеют значительную вычислительную сложность, поэтому на практике более эффективными оказываются адаптивные методы, основанные на обобщенном

правиле Хебба и непосредственно преобразующие входные векторы x без явного определения матрицы $\mathbf{R}_{xx}$. Адаптивные методы особенно незаменимы при поступлении данных в режиме "онлайн", когда создание явной формы матрицы корреляции просто невозможно.

В развитии метода *PCA* важную роль *играют хеббовские искусственные* нейронные сети, выполняющие это преобразование в режиме онлайн непосредственно на последовательности векторов x. Это преобразование является адаптивным и производится однослойной нейронной сетью, линейной при использовании обобщенного алгоритма Хебба. Созданы различные варианты алгоритмов, в каждом из которых учитывается корреляция между векторами, представляющими входные данные. Значительное упрощение вычислений достигается в результате определения только одного (наибольшего) главного компонента. Поэтому первым будет представлен алгоритм PCA именно для этого случая.

10.2.2. Определение первого главного компонента

Для определения первого главного компонента y_1 и связанного с ним вектора w_1, соответствующего матрице $\mathbf{R}_{xx}$, Е. Ойя предложил систему, состоящую из одного линейного нейрона (рис. 10.2), для которого

$$y_1 = w_1^T x = \sum_{j=0}^{N} w_{1j} x_j. \qquad (10.21)$$

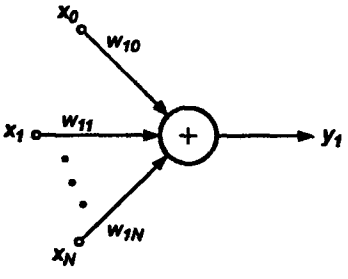

Рис. 10.2. Нейронная сеть PCA для определения одного (важнейшего) главного компонента

Веса вектора w_1 подбираются согласно нормализованному правилу Хебба, называемому правилом Ойя, которое может быть записано в скалярной форме как

$$w_{1j}(k+1) = w_{1j}(k) + \\ + \eta\, y_1[x_j(k) - w_{1j}(k) y_1(k)] \qquad (10.22)$$

или в векторной форме:

$$w_1(k+1) = w_1(k) + \eta(k) y_1(k)\,[x(k) - w_1(k)\, y_1(k)], \qquad (10.23)$$

где $\eta(k)$ обозначает коэффициент обучения. Первое слагаемое формулы соответствует обычному правилу Хебба, а второе обеспечивает самонормализацию векторов весов, т.е. $\|w_1\|^2 = 1$ [51, 111]. Подбор значения η оказывает существенное влияние на сходимость алгоритма. Хорошие результаты достигаются, когда значение $\eta(k)$ уменьшается с течением времени обучения. Широко применяется методика изменения $\eta(k) = \frac{\eta(0)}{k^{\gamma}}$, где $\eta(0) = 0{,}5[X^T X]$; $0{,}5 \le \gamma < 1$. В процессе обучения одни и те же обучающие выборки предъявляются многократно вплоть до стабилизации весов сети.

10.2.3. Алгоритмы определения множества главных компонентов

Определение следующих компонентов PCA предполагает использование в выходном слое большого количества нейронов. Сеть содержит столько нейронов, сколько должно учитываться главных компонентов разложения. Они располагаются в одном слое, поэтому сеть PCA считается однослойной с линейными функциями активации нейронов (рис. 10.3). Обобщенное правило Ойя для такой сети становится нелокальным и малопривлекательным с вычислительной точки зрения. Лучшие результаты дает применение правила Сенгера [141]. Если K линейных нейронов выходного слоя генерируют выходные сигналы согласно выражению

$$y_i(k) = \sum_{i=0}^{N} w_{ij}(k) x_j(k), \qquad (10.24)$$

то уточнение весов сети производится по формуле [141]

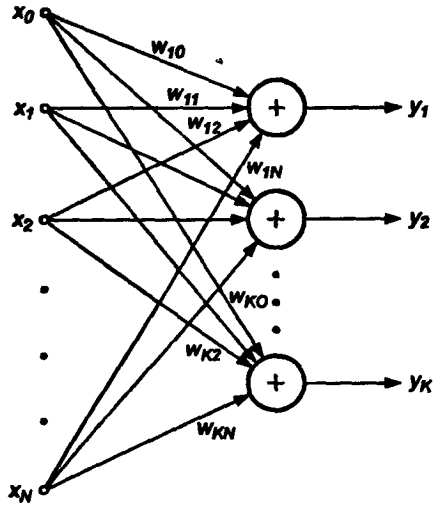

Рис. 10.3. Линейная нейронная сеть PCA для определения K главных компонентов

$$w_{ij}(k+1) = w_{ij}(k) + \eta y_i(k)[x_j(k) - \sum_{h=1}^{i} w_{hj}(k) y_h(k)] \qquad (10.25)$$

для $j = 0, 1, 2, ..., N$, $i = 1, 2, ..., K$. Если принять обозначение

$$x'_j(k) = x_j(k) - \sum_{h=1}^{i-1} w_{hj}(k) y_h(k), \qquad (10.26)$$

то выражение (10.25) можно представить в форме

$$w_{ij}(k+1) = w_{ij}(k) + \eta y_i(k)[x'_j(k) - w_{ij}(k) y_i(k)], \qquad (10.27)$$

аналогичной формуле Ойя (10.22), соответствующей только одному нейрону. Поэтому даже при наличии в выходном слое K нейронов правило обучения все равно остается локальным при условии модификации значения входного сигнала x'_j. Скалярные зависимости (10.26) и (10.27) можно записать в векторной форме

$$x'(k) = x(k) - \sum_{h=1}^{i-1} w_h(k) y_h(k), \qquad (10.28)$$

$$w_i(k+1) = w_i(k) + \eta y_i(k)[x'(k) - y_i(k) w_i(k)] \qquad (10.29)$$

для $i = 1, 2, ..., K$. Правило имеет локальный характер, поскольку для уточнения весов одного нейрона не требуется решать уравнения, описывающие всю сеть. Следует заметить, что для первого нейрона (первого главного компонента PCA) $\boldsymbol{x}'(k) = \boldsymbol{x}(k)$. Для второго нейрона получаем: $\boldsymbol{x}'(k) = \boldsymbol{x}(k) - \boldsymbol{w}_1(k)y_1(k)$ – в этой формуле присутствуют только уже известные веса первого нейрона. Аналогично для третьего нейрона: $\boldsymbol{x}'(k) = \boldsymbol{x}(k) - \boldsymbol{w}_1(k)y_1(k) - \boldsymbol{w}_2(k)y_2(k)$ и т.д. – модификация вектора $\boldsymbol{x}$ выражается через ранее определенные величины, и процесс обучения осуществляется, подобно алгоритму Ойи, с самонормализующимися векторами $\boldsymbol{w}_i$, $\| \boldsymbol{w}_i \| = 1$ [141].

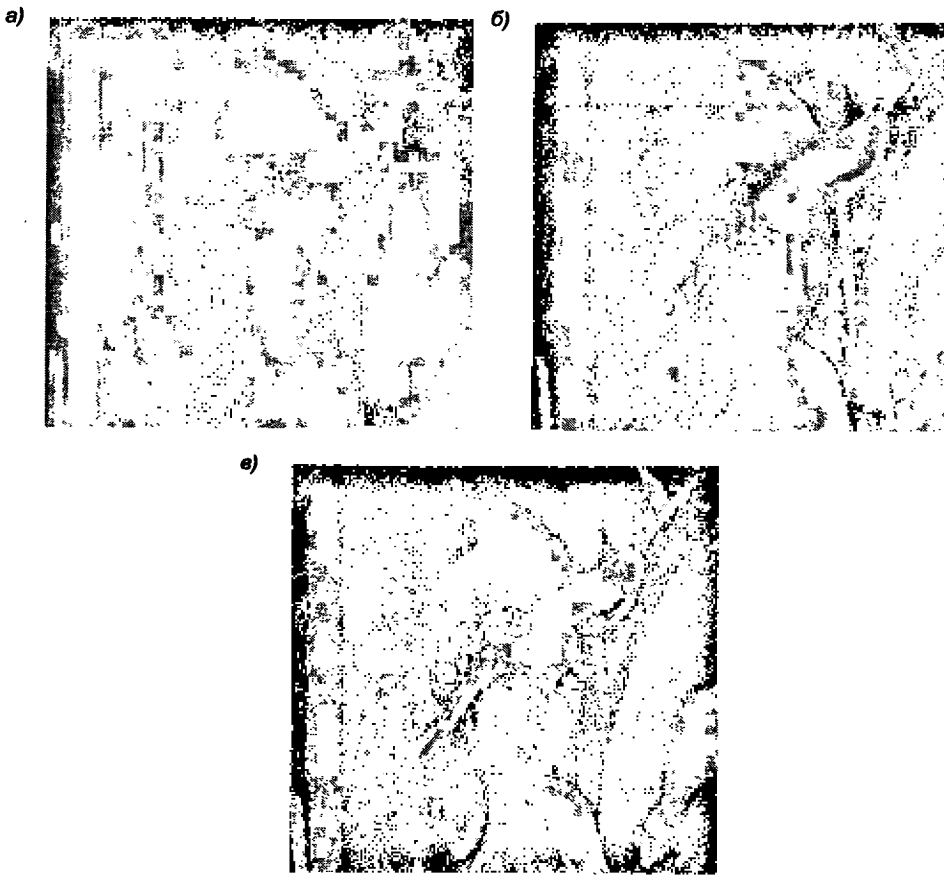

Рис. 10.4. Образ "Лена", восстановленный сетью PCA при использовании различного количества главных компонентов:
а) один главный компонент; *б*) три главных компонента; *в*) пять главных компонентов

В настоящее время существует много различных нейронных алгоритмов, позволяющих осуществить преобразование PCA. К наиболее серьезным, помимо алгоритма Сенгера, относятся алгоритмы Фолдяка, Рабнера, а также APEX

(англ.: *Adaptive Principal component EXtraction*). Подробности их реализации можно найти в книге Диамантараса и Кунга [29].

Преобразование PCA чаще всего применяется для компрессии данных, при котором большое количество входной информации заменяется уменьшенной дозой, содержащейся в векторах y и w_i. В зависимости от степени сжатия (количества главных компонентов PCA) можно получить различное качество восстановления данных.

Для примера на рис. 10.4 представлены три изображения "Лена", реконструированные на основе 1, 3 и 5 главных компонентов PCA [92]. Образ, подвергнутый компрессии, имел размер 512×512 пикселов и был разделен на кадры размером 8×8. Качество восстановленного изображения сильно зависит от количества K главных компонентов, учитываемых при восстановлении. Чем больше этих компонентов, тем выше качество изображения и одновременно тем меньше коэффициент компрессии. Изображение на рис. 10.4*а* соответствует коэффициенту компрессии около 64, на рис. 10.4*б* – около 21, а на рис. 10.4*в* – около 12. При наибольшей степени сжатия (при одном главном компоненте) на изображении сильно заметны отдельные кадры. Изображение, восстановленное на основе пяти главных компонентов, зрительно не отличается от оригинала. Коэффициенты PSNR, полученные для этих образов, равны соответственно 18,80 дБ, 25, 43 дБ и 27, 58 дБ.

10.3. Нейронные ICA-сети Херольта–Джуттена

10.3.1. Предварительные пояснения

Сети Херольта–Джуттена [62] – это линейные сети с самоорганизацией, использующие обобщенное правило Хебба и относящиеся к классу корреляционных сетей. Их концепция была сформулирована в середине восьмидесятых годов XX века профессорами Дж. Херольтом и К. Джуттеном из Гренобля [62, 63]. Первоначально эти сети применялись для так называемой слепой сепарации сигналов. В настоящее время они выполняют и многие другие функции, в том числе анализ главных компонентов PCA, анализ независимых компонентов ICA (англ.: *Independent Component Analysis*), сглаживание и т.п. Первичная структура сети была рекуррентной. В настоящее время часто используются также однонаправленные сети. Независимо от способа соединения нейронов между собой, эти сети обычно имеют адаптивную линейную структуру, обрабатывающую сигналы в режиме реального времени (онлайн). Нелинейные функции, применяемые в алгоритмах обучения, играют очень важную роль при уточненни весов, не оказывая влияния на саму структуру взвешенных связей.

Оригинальное решение Херольта–Джуттена касалось проблемы сепарации сигналов $s_i(t)$ на основе информации, содержащейся в их линейной суперпозиции.

Пусть имеются n независимых сигналов $s_i(t)$ и смешивающая матрица **A**

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}. \qquad (10.30)$$

Для измерений доступны только сигналы $x_i(t)$, представляющие собой линейную суперпозицию $s_i(t)$, причем

$$x_i(t) = \sum_{j=1}^{n} a_{ij} s_j(t) \qquad (10.31)$$

для $i = 1, 2, ..., n$. Главная трудность заключается в том, что как a_{ij}, так и $s_i(t)$ не известны. На основании гипотезы о статистической независимости сигналов Дж. Херольт и К. Джуттен предложили решать эту задачу с применением нейронной сети. Обобщенная схема включения этой сети в измерительную систему представлена на рис. 10.5.

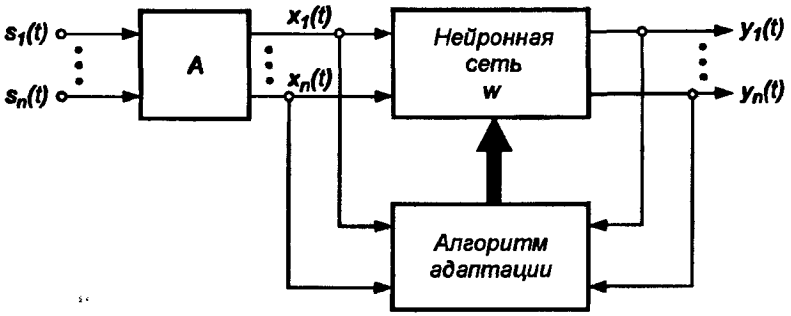

Рис. 10.5. Обобщенная схема включения нейронной сети в систему разделения сигналов

10.3.2. Статистическая независимость сигналов

Статистическая независимость случайных сигналов – это более общее понятие, чем некоррелируемость. В общем случае две случайные переменные y_i и y_j будут статистически независимыми, если информация об одной переменной ничего не говорит о другой. С математической точки зрения статистическая независимость означает, что двухмерная плотность вероятности $p(y_i, y_j)$ равна произведению одномерных функций плотности

$$p(y_i, y_j) = p(y_i)p(y_j) . \qquad (10.32)$$

Для статистически независимых сигналов обобщенная матрица ковариации

функций $f(y_i)$ и $g(y_j)$ (обе функции должны быть нечетными) представляет собой неособенную диагональную матрицу, имеющую вид:

$$E[f(\mathbf{y})\mathbf{g}^T(\mathbf{y})] - E[f(\mathbf{y})]E[\mathbf{g}^T(\mathbf{y})] = \\ = \begin{bmatrix} E[f(y_1)g(y_1)] - E[f(y_1)]E[g(y_1)] & & \\ & E[f(y_2)g(y_2)] - E[f(y_2)]E[g(y_2)] & \\ & & \ddots \\ & & E[f(y_n)g(y_n)] - E[f(y_n)]E[g(y_n)] \end{bmatrix}. \quad (10.33)$$

В этом выражении символом E обозначается ожидаемое значение. Из условия статистической независимости следует, что все обобщенные взаимные являются нулевыми, поэтому $E[f(y_i)g(y_j)] - E[f(y_i)]E[g(y_j)] = 0$, а собственные ковариации – ненулевыми, т.е. $E[f(y_i)g(y_i)] - E[f(y_i)]E[g(y_i)] \neq 0$. Условие статистической независимости сигналов отождествляется в статистике с обнулением взаимных кумулянтов высшего порядка [12, 63].

10.3.3. Рекуррентная структура разделяющей сети

Для решения задачи сепарации статистически независимых сигналов Дж. Херольт и К. Джуттен предложили линейную нейронную сеть с обратной связью, представленную на рис. 10.6. Сеть состоит из n линейных нейронов, связанных между собой взаимными обратными связями. Синаптические веса w_{ij} в оригинальном решении Херольта и Джуттена отличны от нуля только при взаимных связях.

Собственные связи в оригинальном решении, представленном в работе [62], отсутствуют. Каждый нейрон сети генерирует выходной сигнал

$$y_i(t) = x_i(t) - \sum_{j=1, j\neq i}^{n} w_{ij} y_j(t). \quad (10.34)$$

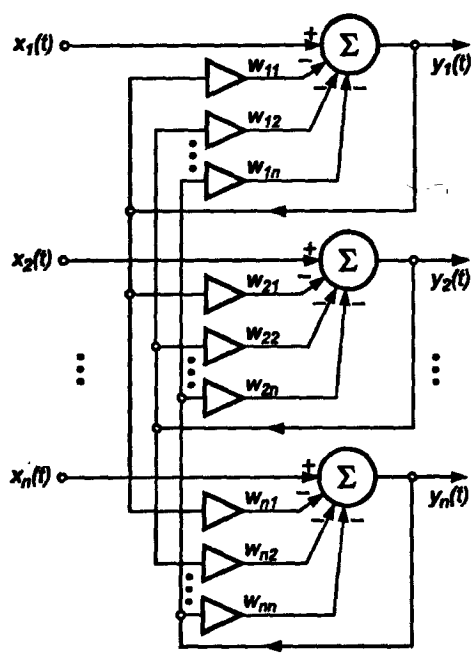

Рис. 10.6. Структура рекуррентной сети Херольта–Джуттена для разделения сигналов

Если обозначить **A** смешивающую матрицу (10.30), **W** – матрицу весов:

$$\mathbf{W} = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix}, \tag{10.35}$$

$\boldsymbol{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]^T$ – вектор наблюдаемых сигналов, преобразованных в соответствии с выражением (10.31), $\boldsymbol{s}(t) = [s_1(t), s_2(t), ..., s_n(t)]^T$ – вектор исходных сигналов, $\boldsymbol{y}(t) = [y_1(t), y_2(t), ..., y_n(t)]^T$ – вектор выходных сигналов, то функционирование сети, изображенной на рис. 10.6, можно описать матричными уравнениями:

$$\boldsymbol{x}(t) = \mathbf{A}\boldsymbol{s}(t), \tag{10.36}$$

$$\boldsymbol{y}(t) = \boldsymbol{x}(t) - \mathbf{W}\boldsymbol{y}(t). \tag{10.37}$$

Если матрица **A** и вектор $\boldsymbol{s}(t)$ не известны, то при выдвижении гипотезы о статистической независимости компонентов вектора $\boldsymbol{s}(t)$ задача сети сводится к такому определению вектора решения

$$\boldsymbol{y}(t) = (\mathbf{1} + \mathbf{W})^{-1}\boldsymbol{x}(t), \tag{10.38}$$

которое позволит восстановить первичные сигналы $s_i(t)$, составляющие вектор $\boldsymbol{s}(t)$ с конкретной, но не определенной заранее степенью точности d_i

$$\boldsymbol{y}(t) = \mathbf{D}\boldsymbol{s}(t), \tag{10.39}$$

где **D** – диагональная матрица, $\mathbf{D} = [d_1, d_2, ..., d_n]$, и с сохранением произвольной последовательности отдельных компонентов в векторе

$$\boldsymbol{y}(t) = \mathbf{P}\boldsymbol{s}(t), \tag{10.40}$$

где **P** – элементарная матрица перестановок, задающая различные комбинации компонентов вектора $\boldsymbol{s}(t)$.

Решение, определяющее вектор $\boldsymbol{y}(t)$, который отвечает условиям (10.39) и (10.40), может быть получено при произвольном количестве нейронов n. Если количество источников больше двух, а значения коэффициентов смешивающей матрицы a_{ij} заранее не известны, сигналы можно разделить только с помощью адаптивного алгоритмического метода подбора весов нейронной сети.

10.3.4. Алгоритм Херольта–Джуттена для рекуррентной сети

Решение задачи разделения сигналов на основе рекуррентной сети было сведено Дж. Херольтом и К. Джуттеном к решению системы дифференциальных уравнений, описывающих изменения весов этой сети. Они предложили простой

адаптивный алгоритм, использующий критерий статистической независимости сигналов и функционирующий в режиме "онлайн", который можно представить в виде системы дифференциальных уравнений

$$\frac{\mathrm{d}\, w_{ij}}{\mathrm{d}t} = \eta(t) f(y_i(t)) g(y_j(t)) \ , \tag{10.41}$$

для $i = 1, 2, ..., n$ и $i = 1, 2, ..., n$ при $i \neq j$ (в оригинальном решении собственные обратные связи отсутствуют, $w_{ii} \equiv 0$). Значение коэффициента обучения $\eta(t)$, как правило, уменьшается в процессе обучения до нуля. Функции $f(y)$ и $g(y)$ нечетны и не равны между собой, $f(y) \neq g(y)$. Следует отметить, что зависимость (10.41) представляет собой нелинейное обобщение простого правила Хебба.

На практике применяются различные виды функций $f(x)$ и $g(x)$, чаще всего одна из них имеет выпуклую, а вторая – вогнутую форму. Наиболее популярны представления $f(x) = x^3$, $f(x) = x^5$. Относительно функции $g(x)$ можно сказать, что хорошие результаты достигаются при $g(x) = \tanh(x)$, $g(x) = \mathrm{arctg}(x)$, $g(x) = x$, $g(x) = \mathrm{sgn}(x)$ и т.д.

В работе [62] было доказано, что обе функции $f(\ )$ и $g(\ )$ соответствуют статистическим моментам высших порядков, что в случае статистической независимости сигналов автоматически обеспечивает равенство нулю средних значений $\left\langle f(y_i(t)) g(y_j(t)) \right\rangle$, гарантирующее сходимость алгоритма обучения.

Правило обучения, определенное выражением (10.41), может быть записано в обобщенной матричной форме

$$\frac{\mathrm{d}\, \mathbf{W}}{\mathrm{d}t} = \eta(t) \boldsymbol{f}(\boldsymbol{y}(t)) \boldsymbol{g}^T(\boldsymbol{y}(t)) \ , \tag{10.42}$$

где $\boldsymbol{f}(\boldsymbol{y}(t)) = [f(y_1(t)), f(y_2(t)), ..., f(y_n(t))]^T$, $\boldsymbol{g}(\boldsymbol{y}(t)) = [g(y_1(t)), g(y_2(t)), ..., g(y_n(t))]^T$. Чаще всего коэффициент обучения $\eta(t)$ имеет в начальный момент фиксированную величину, уменьшающуюся по показательному закону в зависимости от времени обучения t.

Следует обратить внимание, что адаптивная зависимость (10.42) относится к переменным компонентам сигналов. При наличии постоянной составляющей ее следует отфильтровать. Для этого обычно применяется фильтр первого или второго порядка, выходной сигнал которого воспринимается как переменная составляющая $x_j(t)$ $x_j(t) * \eta(t)$, где $\eta(t)$ обозначает импульсную реакцию фильтра, а $*$ – обозначение свертки.

Экспериментальные исследования сети Херольта–Джуттена, проведенные как с помощью компьютерного моделирования, так и в процессе ее технической реализации, подтвердили хорошую сходимость алгоритма и возможность разделения многих статистически независимых сигналов различной структуры, но при ограниченных соотношениях амплитуд отдельных компонентов сигналов, подлежащих сепарации.

10.3.5. Обобщенный алгоритм обучения рекуррентной сети

На практике метод Херольта–Джуттена оказывается эффективным при небольшом разбросе амплитуд отдельных сигналов $s_i(t)$, обычно меньшем, чем 1 : 100.

При сильных отличиях сигналов значительно более эффективным считается модифицированный алгоритм А.Чихотского [12], в котором вводятся собственные обратные связи нейронов с весами $w_{ii} \neq 0$. Эти обратные связи вызывают самонормализацию выходных сигналов, что приводит их к одинаковому численному уровню и облегчает тем самым процесс сепарации.

В соответствии с модификацией Чихотского [12] адаптивные механизмы уточнения весов (при условии, что $y_i(t)$ не содержат постоянных составляющих) описываются формулами

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \eta(t)\, f(y_i(t))\, g(y_j(t)) \tag{10.43}$$

для $i \neq j$ и

$$\frac{\mathrm{d}w_{ii}}{\mathrm{d}t} = \eta(t)\,[f(y_i(t))\, g(y_i(t)) - 1] \tag{10.44}$$

для $i = 1, 2, ..., n$. Обе формулы можно представить в обобщенной матричной форме

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} = \eta(t)[f(\boldsymbol{y}(t))\, \boldsymbol{g}^T(\boldsymbol{y}(t)) - \mathbf{1}], \tag{10.45}$$

в которой используются те же обозначения, что и в формуле (10.42). Вектор $\boldsymbol{y}(t)$ рассчитывается в каждый момент времени согласно (10.38), $\boldsymbol{y}(t) = (\mathbf{1} + \mathbf{W})^{-1}\boldsymbol{x}(t)$, для текущих значений матрицы весов $\mathbf{W}$ и вектора смешанных сигналов $\boldsymbol{x}(t)$.

Главный источник повышенной эффективности алгоритма – самонормализация до единичных значений выходных сигналов $y_i(t)$. Поскольку в стабилизированном состоянии $\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} = \mathbf{0}$, следовательно, $\langle f(y_i(t))g(y_i(t))\rangle = 1$, и независимо от уровня сигналов $s_i(t)$ происходит масштабирование всех сигналов в сети до единичного уровня. Имитационные исследования сети с модифицированным алгоритмом продемонстрировали возможность разделения сигналов с амплитудами, различающимися даже в отношении $1 : 10^{10}$. Сеть с модифицированным правилом обучения также менее чувствительна к коэффициенту обусловленности смешивающей матрицы **A**.

На рис. 10.7 представлен процесс сепарации четырех сигналов $s_i(t)$ со значительно отличающимися амплитудами:

$$s_1 = 0{,}001 \sin(300t + 6\cos 60t);$$

$$s_2 = \mathrm{rand}(0{,}00001,\ t + 6\cos 60t);$$

$$s_3 = 0{,}001\ \mathrm{sgn}(\cos 155t);$$

$$s_4 = 0{,}00001 \sin(1200t)\sin(50t);$$

смешанными посредством матрицы **A**.

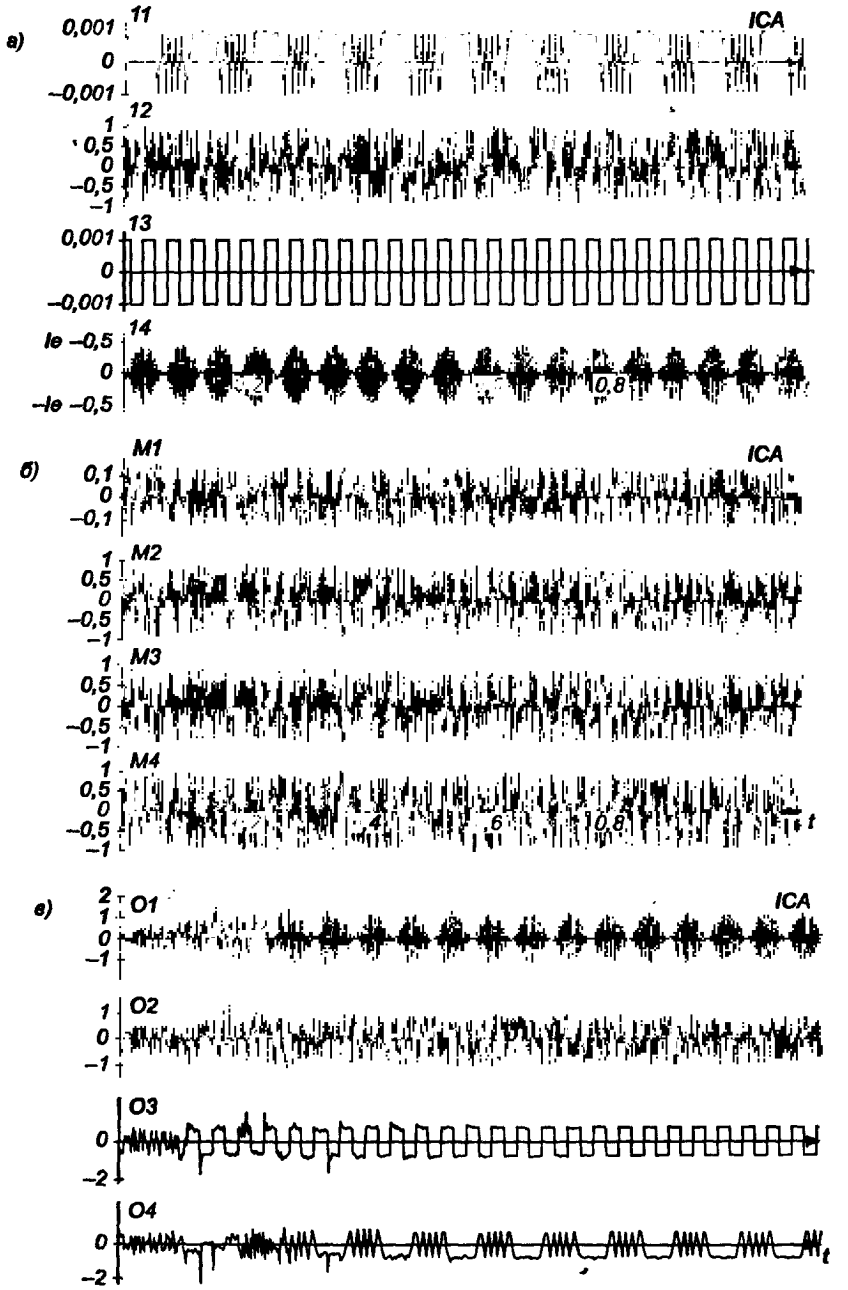

Рис. 10.7. Графическая иллюстрация процесса разделения сигналов сетью Херольта–Джуттена:

а) исходные сигналы; *б*) сигналы, смешанные посредством матрицы A; *в*) выходные сигналы нейронной сети в процессе их разделения

$$\mathbf{A} = \begin{bmatrix} 0{,}78 & 0{,}15 & -0{,}22 & 0{,}12 \\ -0{,}92 & -0{,}90 & 0{,}27 & -0{,}93 \\ 0{,}40 & -0{,}91 & 0{,}60 & -0{,}78 \\ -0{,}88 & 0{,}99 & 0{,}10 & 0{,}61 \end{bmatrix}.$$

На рис. 10.7*а* изображены исходные сигналы $s_i(t)$, на рис. 10.7*б* – смешанные сигналы $x_i(t)$, а на рис. 10.7*в* – сигналы, выделенные нейронной сетью в процессе сепарации (это независимые сигналы, являющиеся реакцией на исходные сигналы). В качестве входных используются смешанные сигналы (три средних графика на рис. 10.7). Из-за огромного различия в амплитудах исходных сигналов на графиках подаваемых на ввод сети смешанных сигналов видны только наибольшие сигналы шума, тогда как сигналы с малой амплитудой практически незаметны. Процесс сепарации осуществлялся с помощью программы *BS* [13] с применением нелинейных функций $f(x) = x^3$, $g(x) = \tanh(10x)$, а также коэффициента обучения $\eta(t)$, адаптивно уменьшающегося от начального значения, равного 2000. Это позволило разделить все сигналы независимо от их амплитуд (три нижних графика на рис. 10.7). Выделенные сетью сигналы характеризовались одинаковым уровнем амплитуды, достигнутым благодаря собственным обратным связям нейронов.

10.3.6. Однонаправленная сеть для разделения сигналов

Серьезный недостаток рекуррентной сети Херольта–Джуттена, с трудом устраняемый на практике, состоит в сложности обеспечения стабильности процесса разделения сигналов, особенно тогда, когда матрица **A** плохо обусловлена, а исходные сигналы сильно отличаются друг от друга по амплитуде. Также следует отметить, что в рекуррентной сети на каждом шаге возникает необходимость инвертировать матрицу весов (формула (10.38)), что заметно увеличивает вычислительную сложность алгоритма. Устранить эти проблемы позволяет применение однонаправленной сети без обратных связей.

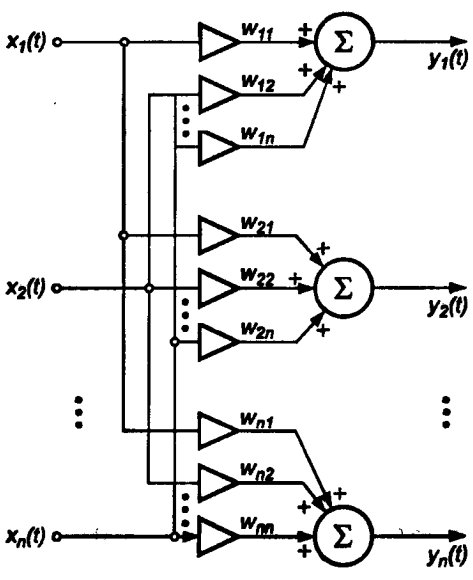

Рис. 10.8. Структура однонаправленной сети для разделения сигналов

Обобщенная структура такой сети представлена на рис. 10.8. Источниками информации для сети являются только смешанные сигналы $x_i(t)$. В результате

их преобразования линейной системой синаптических весов w_{ij} формируется вектор y

$$y = Wx \tag{10.46}$$

Матрица $W \in R^{n\times n}$ в этом выражении является полной. При таком решении однонаправленная сеть равнозначна сети с обратными связями, если матрица весов W удовлетворяет условию

$$W = (\hat{W}+1)^{-1}, \tag{10.47}$$

где $\hat{W}$ обозначена матрица весов рекуррентной сети. В результате простого математического преобразования получаем:

$$\hat{W} = W^{-1} - 1. \tag{10.48}$$

Алгоритм обучения весов W можно получить непосредственно из обучающих зависимостей для рекуррентных сетей, если принять во внимание, что

$$\frac{d\hat{W}}{dt} = \frac{dW^{-1}}{dt}. \tag{10.49}$$

С учетом матричного тождества $\frac{d(WW^{-1})}{dt} = 0$ получаем $\frac{d(WW^{-1})}{dt} = W\frac{dW^{-1}}{dt} + \frac{dW}{dt}W^{-1} = 0$, откуда

$$\frac{dW}{dt} = -W\frac{dW^{-1}}{dt}W = -W\frac{d\hat{W}}{dt}W. \tag{10.50}$$

Если выбрать для дальнейшей реализации одно из правил обучения, сформулированных для рекуррентных сетей, то можно получить его аналог для однонаправленной сети. Например, принимая во внимание модифицированное правило Чихотского [17], строится адаптивная зависимость, представляемая в матричной форме

$$\frac{dW}{dt} = \eta(t)W[1 - f(y(t))[g(y(t))]^T]W \tag{10.51}$$

с начальным условием W(0) = 1, имеющая свойства, идентичные свойствам алгоритма обучеиия рекуррентной сети, на базе которого она была создана. В отличие от формулы обучения рекуррентной сети в выражении (10.51) изменения весов обусловлены их фактическими значениями. К настоящему времени известно большое количество вариантов обучающей формулы (10.51), характеризующихся особенно выдающимися качествами при плохой обусловленности матрицы A либо при большой разнице амплитуд исходных сигналов $s_i(t)$. Среди них можно выделить [12]:

• алгоритм Чихотского–Амари–Янга

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t}=\eta(t)[1-f(y(t))g^T(y(t))]\,\mathbf{W}^{-1}, \tag{10.52}$$

• алгоритм, основанный на естественном градиенте,

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t}=\eta(t)[1-f(y(t))g^T(y(t))]\,\mathbf{W}, \tag{10.53}$$

• алгоритмы Кардоссо [3, 12]:

$$\frac{d\mathbf{W}}{dt}=\eta(t)\,[1-y\,(t)\,y^T(t)\,-\alpha f(y(t)\,g^T(y(t))+\beta g(y(t)f^T(y\,(t))]\,\mathbf{W}, \tag{10.54}$$

$$\frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t}=\eta(t)\,[1-f(y(t)\,y^T(t))]\,\mathbf{W}, \tag{10.55}$$

где α и β – это численные коэффициенты из интервала [0, 1].

Особенно хорошими качествами характеризуется алгоритм, основанный на естественном градиенте, при реализации которого, как показано в работе [18], процесс сепарации практически не зависит от соотношения амплитуд сигналов $s_i(t)$ и от степени обусловленности смешивающей матрицы **A**.

Так же как и в рекуррентных сетях, коэффициент обучения $\eta(t)$ представляется функцией, значение которой уменьшается с течением времени до нуля. Обычно это показательная функция вида $\eta\,(t)=Ae^{-t/\tau}$ со значением амплитуды **A** и постоянной времени, индивидуально подбираемой в каждом конкретном случае.

Экспериментальные исследования однонаправленной сети показали высокую эффективность разделения исходных сигналов с большими относительными различиями (доходящими до 1010) амплитуд и плохой обусловленностью смешивающей матрицы **A**. На рис. 10.9 представлены результаты сепарации семи сигналов со значительно отличающимися уровнями амплитуд:

$$s_1 = 0{,}00001\ \mathrm{sgn}(\cos 157t);$$

$$s_2 = 0{,}001\ \sin(310t+\cos 57t);$$

$$s_3 = 0{,}01\ \sin(90t);$$

$$s_4 = \mathrm{rand}(0{,}0001,\ t);$$

$$s_5 = \mathbf{randw}(10,\ 0,\ 000001,\ t);$$

$$s_6 = 0{,}0000000001\ \cos(155t);$$

$$s_7 = 0{,}00001\ \sin(800t)\cdot\sin(60t).$$

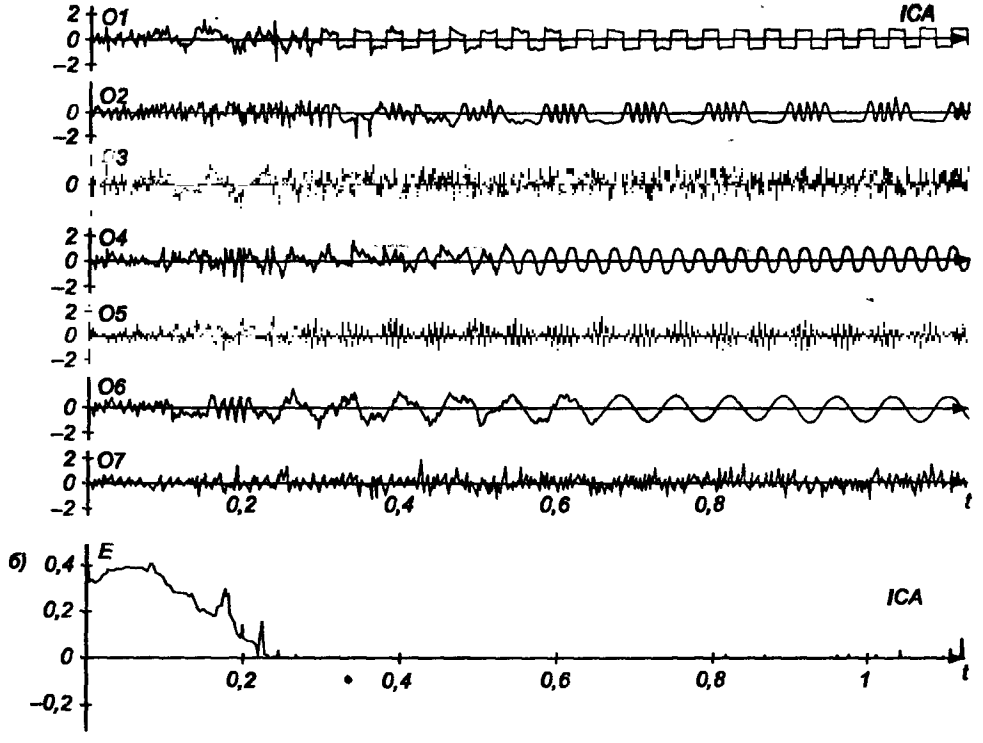

Рис. 10.9. Процесс разделения семи сигналов, смешанных посредством матрицы A: *а*) графики изменения сигналов; *б*) график изменения эвклидовой нормы погрешности

Использовалась смешивающая матрица вида

$$\mathbf{A} = \begin{bmatrix} 0{,}70 & 0{,}15 & -0{,}20 & 0{,}12 & -0{,}48 & 0{,}13 & 0{,}11 \\ -0{,}91 & -0{,}90 & 0{,}20 & -0{,}90 & -0{,}69 & 0{,}06 & 0{,}08 \\ 0{,}39 & -0{,}90 & 1{,}00 & -0{,}80 & 0{,}50 & 0{,}00 & 0{,}20 \\ -0{,}86 & 0{,}99 & 0{,}10 & 0{,}10 & 0{,}29 & 0{,}10 & 0{,}00 \\ 0{,}84 & -0{,}33 & 0{,}02 & -0{,}60 & -0{,}51 & 0{,}13 & 0{,}14 \\ 0{,}12 & -0{,}05 & 0{,}20 & 0{,}21 & 0{,}30 & 1{,}00 & 0{,}12 \\ 0{,}30 & 0{,}40 & 0{,}02 & 0{,}10 & 0{,}22 & 0{,}20 & 1{,}00 \end{bmatrix}$$

Применялись типовые нелинейные функции $f_i(y_i(t)) = y_i^2 \operatorname{sgn}(y_i)$, $g_i(y_i(t)) = 3\tanh(10y_i)$ для i = 1, 2, 3, 4, 5, 6, 7. Коэффициент обучения $\eta(t)$ изменялся в соответствии с выражением

$$\eta(t) = \begin{cases} 200 & \text{для } t < 0{,}25s \\ 200\ \exp^{-6(t-t_0)} & \text{для } t \geq 0{,}25 \end{cases}. \qquad (10.56)$$

Веса подбирались путем решения методом Рунге–Кутта системы дифференциальных уравнений, соответствующих алгоритму (10.53). Как видно

на рис. 10.9, разделение сигналов произошло менее чем за двадцать итераций. Процесс сепарации сигналов протекал равномерно и практически не зависел от уровня амплитуды исходных сигналов. Все выделенные сетью сигналы были нормализованы и имели амплитуду, близкую к единице. Вызывает интерес динамика изменения адаптировавшихся весов сети с учетом значительных различий амплитуд исходных сигналов. На рис. 10.10 для примера

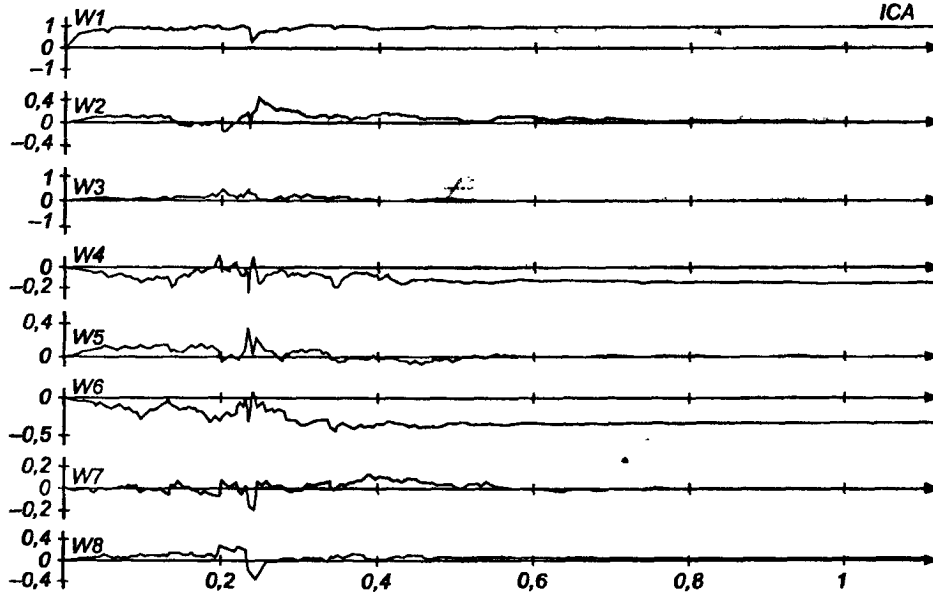

Рис. 10.10. Процесс адаптации некоторых весов нейронной сети

продемонстрирован процесс адаптации восьми весов одного из нейронов описанной выше сети, разделяющей семь сигналов. Веса нейронов, соответствующих самым слабым сигналам, принимают большие значения. Благодаря этому выравнивается влияние каждого исходного сигнала на окончательную форму выделенных сигналов $y_i(t)$.

Раздел 11

МАТЕМАТИЧЕСКИЕ ОСНОВЫ НЕЧЕТКИХ СИСТЕМ

Понятие нечетких множеств (англ.: *fuzzy sets*) как обобщение обычных (четких) множеств было введено Л. Заде в 1965 г. [177, 178]. Традиционный способ представления элемента множества A состоит в применении характеристической функции $\mu_A(x)$, которая равна 1, если этот элемент принадлежит к множеству A, или равна 0 в противном случае. В нечетких системах элемент может частично принадлежать к любому множеству. Степень принадлежности к множеству A, представляющая собой обобщение характеристической функции, называется функцией принадлежности $\mu_A(x)$, причем $\mu_A(x) \in [0,1]$. Значения функции принадлежности являются рациональными числами из интервала $[0, 1]$, где 0 означает отсутствие принадлежности к множеству, а 1 – полную принадлежность. Конкретное значение функции принадлежности называется *степенью или коэффициентом принадлежности*. Эта степень может быть определена явным образом в виде функциональной зависимости (например, $\mu_A(x) = \exp\left(-\left(\frac{x-3}{0,2}\right)^2\right)$ либо дискретно – путем задания конечной последовательности значений $x \in \{x_n\}$ в виде

$$A(x) = \left\{\frac{\mu(x_1)}{x_1}, \frac{\mu(x_2)}{x_2}, \ldots, \frac{\mu(x_N)}{x_N}\right\}. \qquad (11.1)$$

Например, для последовательности дискретных значений переменной x, равных $x_1 = 7$, $x_2 = 8$, $x_3 = 9$, $x_4 = 10$, $x_5 = 11$, $x_6 = 12$, $x_7 = 13$, их коэффициент принадлежности к числам, близким 10, может быть определен в виде

$$A(x) = \left\{\frac{0,1}{7}, \frac{0,3}{8}, \frac{0,8}{9}, \frac{1,0}{10}, \frac{0,8}{11}, \frac{0,3}{12}, \frac{0,1}{13}\right\}.$$

В теории нечетких множеств, помимо переменных цифрового типа, существуют лингвистические переменные с приписываемыми им значениями. Пусть переменная x обозначает температуру (x = "температура"). Можно определить нечеткие множества "отрицательная", "близкая к нулю", "положительная", характеризуемые функциями принадлежности $\mu_{отриц}(x)$, $\mu_{бнул}(x)$, $\mu_{полож}(x)$. Так же как обычная переменная может принимать различные значения, лингвистическая переменная "температура" может принимать различные

лингвистические значения. В нашем примере это: "отрицательная", "близкая к нулю" и "положительная". Следовательно, лингвистическое выражение может иметь вид: "температура отрицательная", "температура, близкая к нулю", "температура положительная".

На рис. 11.1 приведена графическая иллюстрация функции принадлежности переменной $x = T$ (где T означает температуру) для трех названных множеств значений температуры. Непрерывными линиями обозначена классическая (точная) принадлежность, а пунктирными линиями – нечеткая принадлежность. Можно отметить, что функция нечеткой принадлежности является непрерывным приближением пороговой функции точной принадлежности.

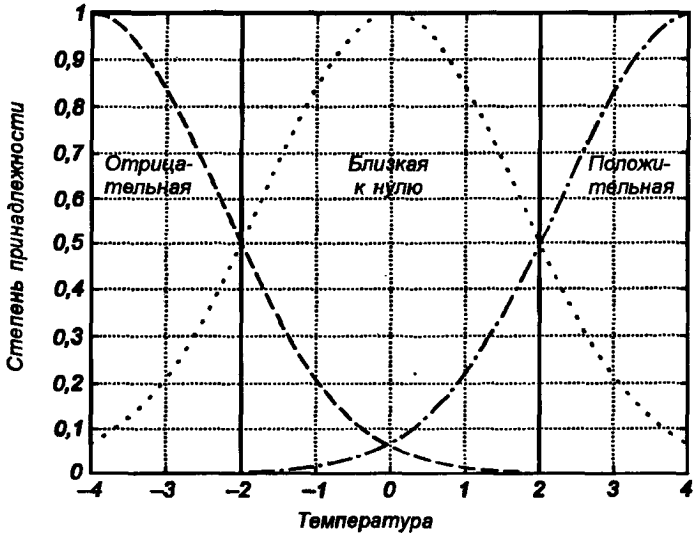

Рис. 11.1. Иллюстрация понятия принадлежности температуры к области отрицательной, близкой к нулю либо положительной (пунктирные линии – нечеткая система, сплошные линии – точная система)

Каждое нечеткое множество имеет определенный *носитель* (англ.: *support*). Носителем множества Supp(A) является подмножество тех элементов A, для которых коэффициент принадлежности к A не равен нулю, т.е. Supp(A) = $=\{x, \mu_A(x) > 0\}$. В приведенном выше примере на рис. 11.1 носителем множества "близкая к нулю" является множество температур в интервале от -4°С до +4°С.

Два множества $A(x)$ и $B(x)$ равны между собой, когда $\mu_A(x) = \mu_B(x)$ для каждого элемента обоих множеств. *Кардинальное число* нечеткого множества A равно сумме коэффициентов принадлежности всех элементов к этому множеству, $M(A) = \sum\mu_A(x)$. Это обобщение аналогичного понятия, относящегося к обычным множествам, для которых кардинальное число равно сумме элементов множества. Нечеткое множество является *нормальным*, если хотя бы один элемент этого множества имеет коэффициент принадлежности, равный 1. *Сечение* α нечеткого множества A образуется подмножеством A_α,

содержащим те элементы множества A, для которых $\mu_A(x) > \alpha$ (слабое сечение) или $\mu_A(x) \geq \alpha$ (сильное сечение), причем $\alpha \in [0,1]$.

11.1. Операции на нечетких множествах

На нечетких множествах, рассматриваемых как обобщение обычных множеств, можно определить ряд математических операций, являющихся обобщением аналогичных операций, выполняемых на "четких" множествах. К ним среди прочих относятся:

1. *Логическая сумма множеств* $A \cup B$

$$\mu_{A \cup B}(x) = \mu_A(x) \cup \mu_B(x) = Max[A(x), B(x)], \tag{11.2}$$

где знак $\cup$ обозначает оператор Max.

ПРИМЕР 11.1

Пусть даны два нечетких множества A и B, определенные следующим образом:

$$A = \left\{ \frac{1,0}{x_1}, \frac{0,7}{x_2}, \frac{0,5}{x_3}, \frac{0,1}{x_4} \right\},$$

$$B = \left\{ \frac{0,2}{x_1}, \frac{0,5}{x_2}, \frac{0,6}{x_3}, \frac{0,7}{x_4} \right\}.$$

Логическая сумма этих множеств $C = A \cup B$ равна:

$$C = \left\{ \frac{1,0}{x_1}, \frac{0,7}{x_2}, \frac{0,6}{x_3}, \frac{0,7}{x_4} \right\}.$$

2. *Логическое произведение множеств* $A \cap B$

$$\mu_{A \cap B}(x) = \mu_A(x) \cap \mu_B(x) = Min\left[A(x), B(x)\right], \tag{11.3}$$

где знак $\cap$ обозначает оператор Min. Для данных из примера 11.1 множество C, являющееся логическим произведением множеств A и B, будет иметь вид

$$C = \left\{ \frac{0,2}{x_1}, \frac{0,5}{x_2}, \frac{0,5}{x_3}, \frac{0,1}{x_4} \right\}.$$

3. *Отрицание множества* $\overline{A}$

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \ . \tag{11.4}$$

В отличие от обычных (четких) множеств, где отрицание элементов, принадлежащих к множеству, дает пустое множество, отрицание нечеткого множества определяет непустое множество, состоящее из элементов, функции принадлежности которых также определены на интервале [0,1].

4. *Равенство множеств A и B*

Нечеткие множества $A(x)$ и $B(x)$ равны между собой, когда для всех элементов x_i обоих множеств выполняется условие $\mu_A(x_i) = \mu_B(x_i)$.

5. *Операция концентрации* CON (A)

$$\mu_{CON}(x) = [\mu_A(x)]^2 . \tag{11.5}$$

Эта операция весьма часто выполняется при действиях с лингвистической переменной, в которых она отождествляется с интенсификатором "очень".

6. *Операция растяжения* DIL(A)

$$\mu_{DIL}(x) = [\mu_A(x)]^{0,5}. \tag{11.6}$$

Лингвистическое значение этой операции формулируется как "примерно" либо "приблизительно".

7. *Алгебраическое произведение двух множеств A * B*

$$\mu_{A \cdot B}(x) = \mu_A(x) * \mu_B(x) . \tag{11.7}$$

8. *Ограниченная сумма двух нечетких множеств A | + | B*

$$\mu_{A \,|+|\, B}(x) = \min \{1, \mu_A(x) + \mu_B(x)\}. \tag{11.8}$$

9. *Ограниченная разница двух нечетких множеств A | − | B*

$$\mu_{A \,|-|\, B}(x) = \max \{0, \mu_A(x) - \mu_B(x)\}. \tag{11.9}$$

10. *Ограниченное произведение двух нечетких множеств A | · | B*

$$\mu_{A \,|\cdot|\, B}(x) = \max \{0, \mu_A(x) + \mu_B(x) - 1\} \tag{11.10}$$

11. *Нормализация множества* NORM (A)

$$\mu_{NORM}(x) = \frac{\mu_A(x)}{\max\{\mu_A(x)\}}, \tag{11.11}$$

Следует отметить, что множество A считается подмножеством множества B, т.е. $A \subset B$, когда для всех элементов выполняется неравенство $\mu_A(x_i) \le \mu_B(x_i)$. Например, если $A = \{0{,}5 / x_1, 0{,}3/ x_2, 0{,}1/ x_3\}$ и $B = \{0{,}6/ x_1, 0{,}5/ x_2, 0{,}4/ x_3\}$, то $A \subset B$.

Определенные на нечетких множествах операции обладают свойствами ассоциативности, коммутативности и дистрибутивности, причем эти свойства понимаются следующим образом:

- ассоциативность: $(A * B) * C = A * (B * C)$;
- коммутативность: $A * B = B * A$ (за исключением ограниченной разности);
- дистрибутивность: $A * (B \circ C) = (A * B) \circ (A * C)$,

где операторы $*$ и $\circ$ обозначают любую определенную выше операцию на нечетких множествах. Из свойств нечетких множеств следует, что в отличие от произведения обычных множеств логическое произведение множества и его

отрицание не обязательно образуют пустое множество, что можно записать в виде

$$A \cap \overline{A} \neq \varnothing \ . \tag{11.12}$$

Точно так же и логическая сумма нечеткого множества A и его отрицание не образуют полное множество U, что можно записать в виде

$$A \cup \overline{A} \neq U \ . \tag{11.13}$$

11.2. Меры нечеткости нечетких множеств

Для определения степени нечеткости множества введено понятие меры нечеткости, отвечающей так называемым условиям А..ДеЛуки и С.Термини [173, 181], сводящейся к измерению уровня различия между множеством A и его отрицанием B.

Наиболее популярна мера Р.Егера, в соответствии с которой степень нечеткости множества A в метрике p, обозначаемая $FUZ_p(A)$, определяется выражением

$$FUZ_p(A) = 1 - \frac{D_p(A,\overline{A})}{n^{1/p}} \ , \tag{11.14}$$

где $D_p(A, \overline{A})$ – это мера расстояния между множествами A и $\overline{A}$, содержащими n элементов. Значение $p = 1$ соответствует метрике Хемминга, в которой

$$D_1(A,\overline{A}) = \sum_{i=1}^{n} | 2\mu_A(x_i) - 1 | \ , \tag{11.15}$$

а значение $p = 2$ соответствует метрике Евклида, в которой

$$D_2(A,\overline{A}) = \sqrt{\sum_{i=1}^{n} (2\mu_A(x_i) - 1)^2} \ . \tag{11.16}$$

ПРИМЕР 11.2

Если нечеткое множество A определяется дискретным способом как

$$A = \left\{ \frac{0{,}1}{x_1}, \frac{0{,}5}{x_2}, \frac{0{,}8}{x_3}, \frac{1{,}0}{x_4}, \frac{0{,}8}{x_5}, \frac{0{,}5}{x_6}, \frac{0{,}1}{x_7} \right\},$$

то, принимая во внимание, что

$$\overline{A} = \left\{ \frac{0{,}9}{x_1}, \frac{0{,}5}{x_2}, \frac{0{,}2}{x_3}, \frac{0}{x_4}, \frac{0{,}2}{x_5}, \frac{0{,}5}{x_6}, \frac{0{,}9}{x_7} \right\},$$

в соответствии с мерой Егера получаем:

$$FUZ_1(A) = 1 - \frac{1}{7}(0{,}8 + 0 + 0{,}6 + 1 + 0{,}6 + 0 + 0{,}8) = 0{,}457 \ ,$$

$$FUZ_2(A) = 1 - \frac{1}{\sqrt{7}}(0{,}64 + 0 + 0{,}36 + 1 + 0{,}36 + 0 + 0{,}64) = 0{,}347 \ .$$

Другую меру нечеткости (энтропийную) предложил Б.Коско [173]. Она основана на понятии кардинального числа множества. В соответствии с этой мерой

$$FUZ(A) = \frac{M(A \cap \overline{A})}{M(A \cup \overline{A})} \ , \qquad (11.17)$$

где $M(F)$ обозначает кардинальное число множества F. Для множества A из примера 11.2 получаем меру Коско, равную

$$FUZ(A) = \frac{0{,}1 + 0{,}5 + 0{,}2 + 0 + 0{,}2 + 0{,}5 + 0{,}1}{0{,}9 + 0{,}5 + 0{,}8 + 1 + 0{,}8 + 0{,}5 + 0{,}9} = \frac{1{,}6}{5{,}4} = 0{,}296 \ .$$

Следует обратить внимание, что обе меры – Егера и Коско – для четких множеств дают один и тот же нулевой результат, поскольку в мере Коско $M(A, \overline{A}) = 0$, а $D_p(A, \overline{A}) = n^{1/p}$, что вследствие зависимости (11.14) дает в результате также $FUZ_p(A) = 0$.

11.3. Нечеткость и вероятность

В теории вероятности событие $u \in U$ либо происходит, либо нет, а вероятность $p(u)$ представляет меру того, что оно состоится или что случайная переменная x примет значение u. Оценка вероятности $p(u)$ может быть рассчитана как отношение количества экспериментов, в которых указанное событие свершилось, к общему количеству экспериментов. Например, если день 21 ноября в течение последних 100 лет был дождливым 82 раза, то вероятность дождя в этот день в настоящем году оценивается как 0,82. Следует подчеркнуть, что вероятность события относится исключительно к будущему. Когда соответствующий день наступит, данное событие либо произойдет, либо нет, и в этот момент понятие вероятности его свершения утрачивает смысл (например, в момент начала дня 21 ноября все еще не ясно, будет этот день дождливым или нет, однако когда он завершится, мы будет говорить о том, произошло или нет событие, а не о его вероятности).

Понятие нечеткости оценивается совершенно по-другому. Оно измеряется степенью, с которой событие $x = u$ (например, только что происшедшее) принадлежит к некоторому множеству событий A. Фактически измеряется степень, в которой универсальное множество U содержится в подмножестве A. Например, если 21 ноября дождь шел на протяжении 15 ч., то степень его принадлежности к множеству дождливых дней можно определить как 15/24. С этой точки зрения понятие нечеткости относится не только к прошлому, как это имеет место в случае вероятности, но также к настоящему и к будущему.

Следует отметить, что вероятность может быть определена как нечеткое значение, особенно тогда, когда оно оценивается приближенно, а не точным способом. Поэтому можно сказать, что вероятность наступления определенного события составляет, например, "около 0,7", поскольку переменная "около 0,7" является лингвистической. Если же нечеткое понятие относится к будущему, ему можно приписать некоторую вероятность.

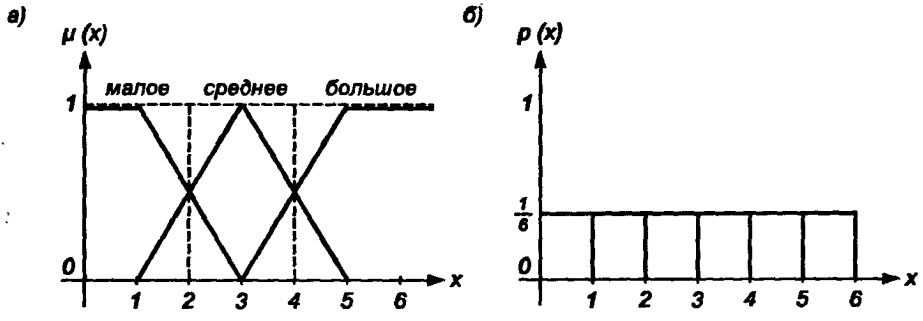

Рис. 11.2. Графическая иллюстрация данных из примера выбрасывания кости: *а*) функции принадлежности; *б*) вероятность выпадения соответствующего номера

ПРИМЕР 11.3

Рассмотрим вероятность выпадения кости с определенным номером из интервала [1 – 6]. Допустим, что имеются три нечетких множества чисел: "малое", "среднее" и "большое", функции принадлежности к которым представлены на рис. 11.2*а*. Вероятность $p(x)$ наступления нечеткого события, что x – это малое, среднее или большое число, определяется по формуле [67, 173]

$$p(x) = \sum_{i=1}^{6} \mu(x_i) p(x_i),$$

где $p(x_i)$ обозначает вероятность выпадения определенного числа из [1 – 6]. Если допустить равномерное распределение вероятности выпадения каждого числа ($p(x_i) = 1/6$) так, как это представлено на рис. 11.2*б*, получим:

$$p(x = \text{"}малое\text{"}) = \frac{1}{6}(1 + 0{,}5) = 0{,}25,$$

$$p(x = \text{"}среднее\text{"}) = \frac{1}{6}(0{,}5 + 1 + 0{,}5) = 0{,}333,$$

$$p(x = \text{"}большое\text{"}) = \frac{1}{6}(0{,}5 + 1 + 1) = 0{,}418.$$

В приведенном результате учитывается понятие иечеткости лингвистической переменной: "малое число", "среднее число", "большое число", а каждому событию приписывается вероятность его наступления.

11.4. Нечеткие правила вывода

Базовое правило вывода типа "если - то" (англ.: *if – then rule*) называется также *нечеткой импликацией*, принимающей форму

$$\textit{если } x \textit{ это } A, \textit{ то } y \textit{ это } B\ , \tag{11.18}$$

где A и B – это лингвистические значения, идентифицированные нечетким способом через соответствующие функции принадлежности для переменных x и y. Часть "x это A" называется *условием* (*предпосылкой*), а "y это B" – следствием (заключением). Импликацию (11.18) можно записать в сокращенном виде $A \rightarrow B$.

Нечеткое рассуждение – это процедура, которая позволяет определить заключение, вытекающее из множества правил "если – то". Такое множество при N переменных x_i может принять вид

$$\textit{если } x_1 \textit{ это } A_1 \textit{ и } x_2 \textit{ это } A_2 \textit{ и } \ldots \textit{ и } x_N \textit{ это } A_N, \textit{ то } y \textit{ это } B\ . \tag{11.19}$$

Переменные x_1, x_2, ..., x_N образуют N-мерный входной вектор $\boldsymbol{x}$, составляющий аргумент условия, в котором A_1, A_2, ..., A_N и B обозначают величины соответствующего коэффициента принадлежности $\mu_A(x_i)$ и $\mu_B(y)$. Необходимо обратить внимание, что здесь присутствуют индивидуальные функции принадлежности для каждой переменной x_i и отдельно для y. Случайное значение функции принадлежности $\mu_A(\boldsymbol{x})$, где $\boldsymbol{x}$ – это вектор $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]$, относящееся к условию импликации (уровень активации правила), должно в последующем интерпретироваться с использованием введенных ранее нечетких операций. Возможна интерпретация в форме логического произведения множеств либо в форме алгебраического произведения:

- интерпретация в форме логического произведения

$$\mu_A(\boldsymbol{x}) = \min_{i=1,\ldots,N}(\mu_A(x_i)), \tag{11.20}$$

- интерпретация в форме алгебраического произведения

$$\mu_A(\boldsymbol{x}) = \prod_{i=1}^{N}\mu_A(x_i)\,. \tag{11.21}$$

Приписывание единственного значения функции принадлежности, описывающей многомерное условие, будем называть *агрегированием предпосылки*. Каждой импликации A → B, определенной выражением (11.19), можно приписать также единственное значение функции принадлежности $\mu_{A\rightarrow B}(\boldsymbol{x}, y)$. Наиболее популярные интерпретации этой функции также имеют форму логического или алгебраического произведения:

- форма логического произведения

$$\mu_{A\rightarrow B} = \min\{\mu_A(\boldsymbol{x}), \mu_B(y)\}, \tag{11.22}$$

- форма алгебраического произведения

$$\mu_{A\rightarrow B} = \mu_A(\boldsymbol{x})\mu_B(y)\,. \tag{11.23}$$

Приписывание единственного значения функции принадлежности всей импликации будем называть процедурой агрегирования на уровне импликации.

11.5. Системы нечеткого вывода Мамдани-Заде

Элементы теории нечетких множеств, правила импликации и нечетких рассуждений образуют систему нечеткого вывода. В ней можно выделить множество используемых в системе нечетких правил, базу данных, содержащую описания функций принадлежности, а также механизм вывода и агрегирования, который формируется применяемыми правилами импликации. Следует упомянуть, что в случае технической реализации в качестве входных и выходных сигналов выступают измеряемые величины, однозначно сопоставляющие входным значениям соответствующие выходные значения. Для обеспечения взаимодействия множеств этих двух видов вводится нечеткая система с так называемыми *фуззификатором* (преобразователем множества входных данных в нечеткое множество) на входе и *дефуззификатором* (преобразователем нечетких множеств в конкретное значение выходной переменной) на выходе [160, 173]. Структура такой системы представлена на рис. 11.3. Фуззификатор преобразует точное

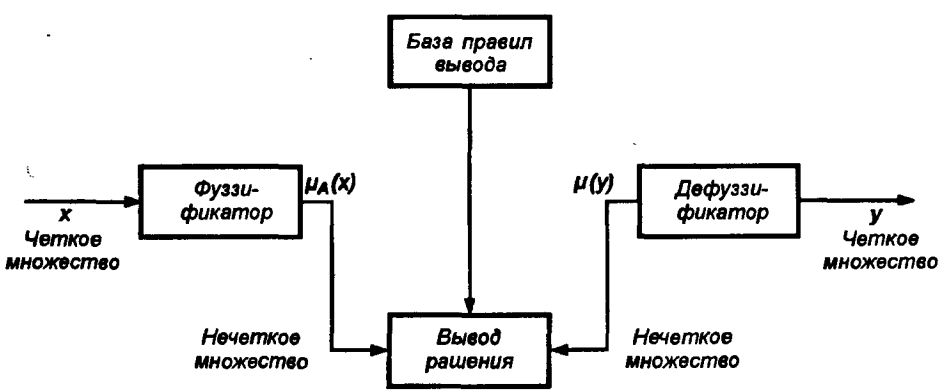

Рис. 11.3. Структура нечеткой системы с фуззификатором и дефуззификатором

множество входных данных в нечеткое множество, определяемое с помощью значений функций принадлежности, тогда как дефуззификатор решает обратную задачу – он формирует однозначное решение относительно значения выходной переменной на основании многих нечетких выводов, вырабатываемых исполнительным модулем нечеткой системы. Выходной сигнал этого модуля может иметь вид M нечетких множеств, определяющих диапазон

изменения выходной переменной. Дефуззификатор преобразует этот диапазон в одно конкретное значение, принимаемое в качестве выходного сигнала всей системы.

Необходимо отметить, что также существуют системы нечеткого вывода, в которых исполнительный механизм непосредственно генерирует четкие значения, которые уже не требуется подвергать дефуззификации. В качестве примера назовем систему Такаги–Сугено–Канга, которая будет подробно описана в следующем подразделе.

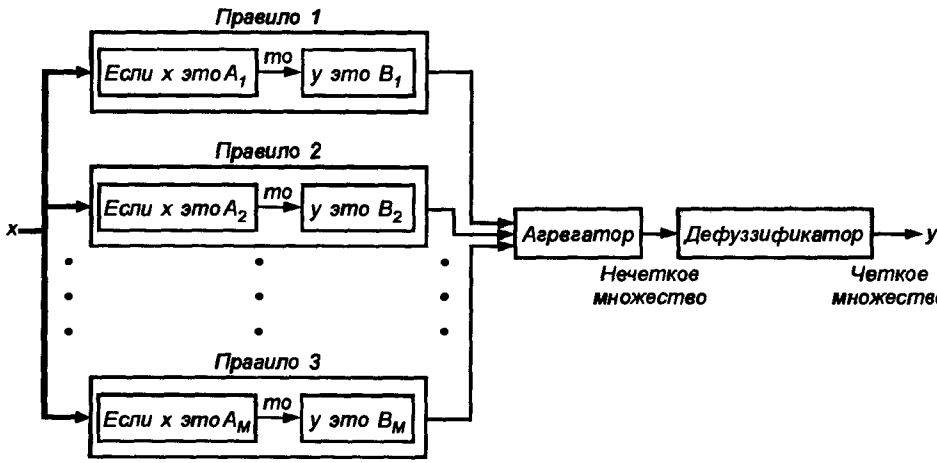

Рис. 11.4. Организация вывода в нечеткой системе при наличии M правил вывода

Обобщенная функциональная структура системы, приведенная на рис. 11.3, может быть представлена в расширенной форме, которая в явном виде демонстрирует правила нечеткого вывода так, как это изображено на рис. 11.4. Поскольку допускается применение множества нечетких правил, в ней также предусмотрен блок агрегирования, чаще всего реализуемый в виде логического сумматора (оператор Max). Описываемая система вывода называется системой Мамдани-Заде. Она очень популярна в обычных (неадаптивных) нечетких системах. Как правило, в модели Мамдани-Заде присутствуют следующие операторы:

- оператор логического или арифметического произведения для определения результирующего уровня активации, в котором учитываются все компоненты вектора x условия;
- оператор логического или арифметического произведения для определения значения функции принадлежности для всей импликации $A \rightarrow B$;
- оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;

- оператор дефуззификации, трансформирующий нечеткий результат $\mu(y)$ в четкое значение выходной переменной y.

ПРИМЕР 11.4

На рис. 11.5 представлен способ агрегирования двух правил нечеткого вывода при существовании двух значений переменных x_1 и x_2. Логическое произведение (оператор Min) используется как для агрегирования нечетких правил относительно конкретных переменных x_i (i = 1, 2), образующих вектор x, так и на уровне импликации $A \rightarrow B$ для одиночных правил вывода. Агрегирование импликаций, касающихся правил 1 и 2, проводится с использованием логической суммы (оператор Max). В правой нижней части рисунка представлен нечеткий результат в виде функции принадлежности переменной y. Получение четкого значения y, соответствующего также четким значениям входных переменных x_1 и x_2, требовало бы в этом случае применения процедуры дефуззификации.

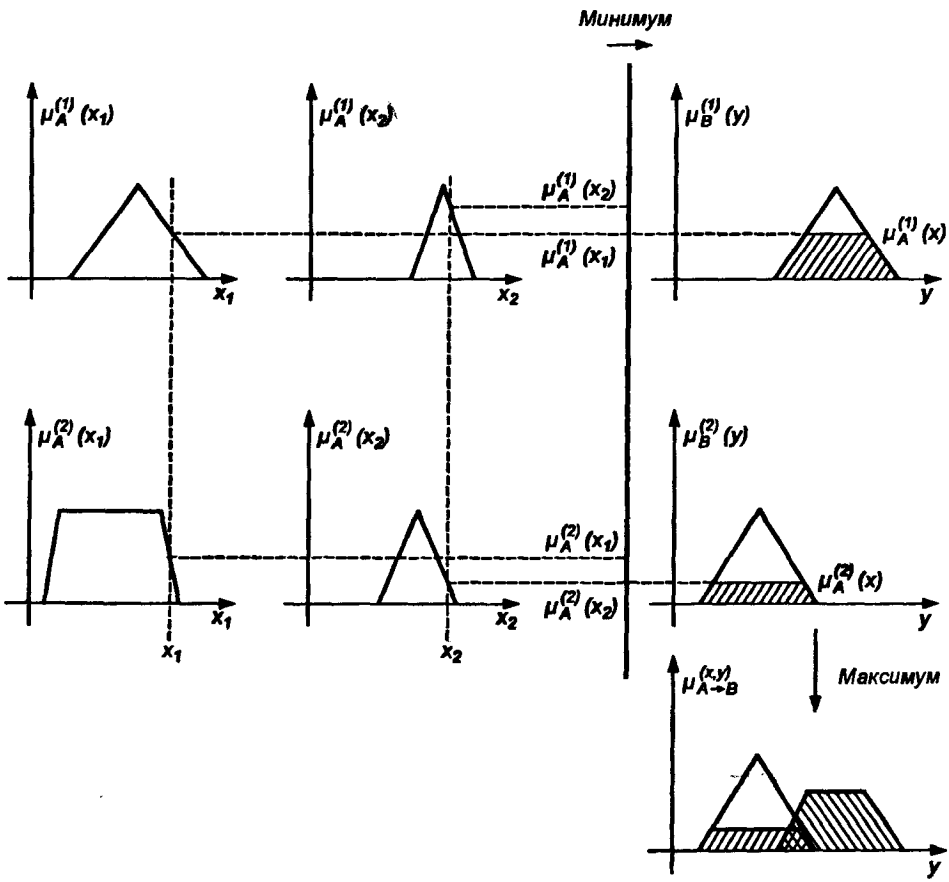

Рис. 11.5. Иллюстрация примера системы вывода Мамдани-Заде

11.5.1. Фуззификатор

Фуззификатор преобразует N-мерный входной вектор $\boldsymbol{x} = [x_1, x_2, ..., x_N]^T$ в нечеткое множество A, характеризуемое функцией принадлежности $\mu_A(\boldsymbol{x})$ с четкими переменными. Несмотря на то, что нечеткие системы могут иметь функции принадлежности произвольной структуры, с практической точки зрения

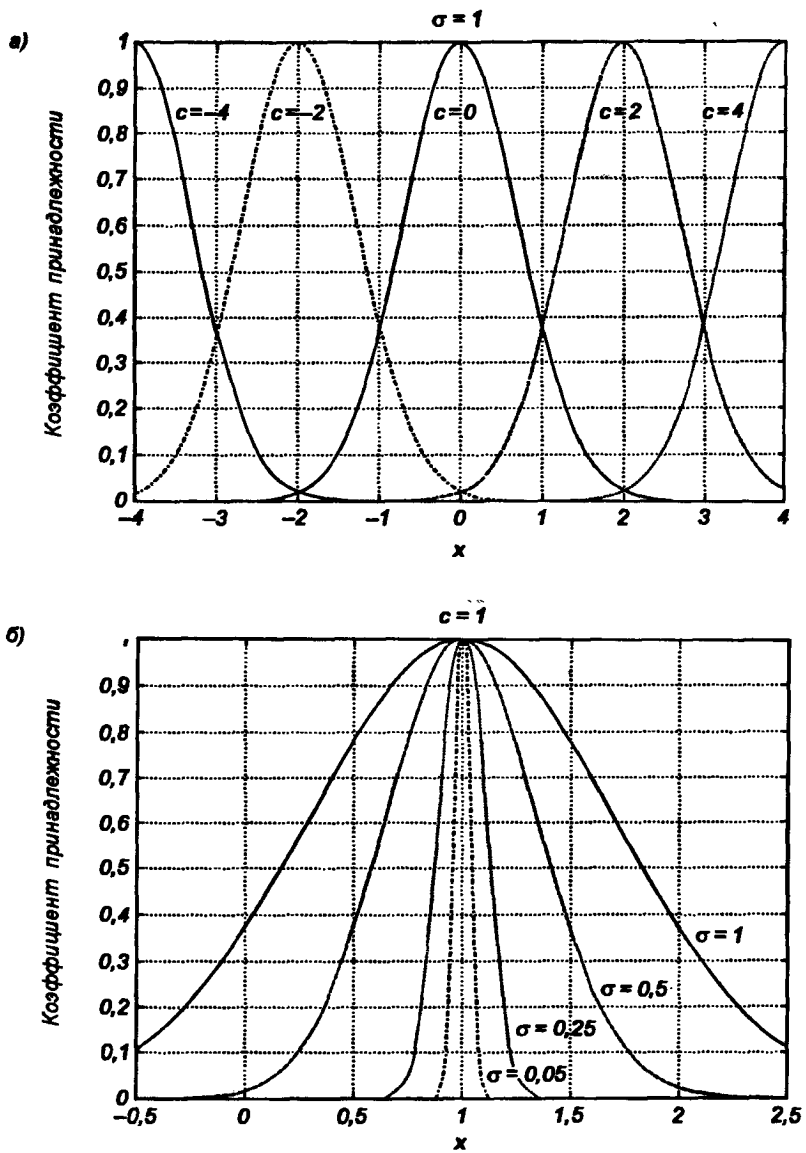

Рис. 11.6. Иллюстрация влияния параметров гауссовской функции на ее форму:
а) влияние размещения центра c при $\sigma = 1$;
б) влияние значения σ при постоянном значении $c = 1$

наибольшей популярностью пользуются функции гауссовского типа, а также треугольные и трапецеидальные функции.

Общая форма гауссовской функции для переменной x с центром c и вариацией σ для множества F имеет вид:

$$\mu_A(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right]. \qquad (11.24)$$

На рис. 11.6 представлена форма типовых гауссовских функций при различных параметрах c и σ, причем на рис. 11.6*а* показано влияние размещения центра c при неизменном значении σ, а на рис. 11.6*б* – влияние значения σ при фиксированном положении c. Параметр c обозначает центр нечеткого множества, а его изменение соответствует смещению функции принадлежности по горизонтальной оси. Праметр σ, иногда называемый коэффициентом широты, отвечает за форму функции. Чем меньше его значение, тем больше крутизна функции. Следует отметить, что при соответствующем смещении центра гауссовская функция может реализовать и сигмоидальную функцию (чаще всего при смещении вправо с $c=4$).

В нечетких сетях также применяется обобщенная гауссовская функция, которая определяется формулой

$$\mu_A(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^{2b}\right]. \qquad (11.25)$$

Она оперирует тремя параметрами: c, σ и b. Значение параметра b существенным образом влияет на форму кривой, что демонстрируется на рис. 11.7.

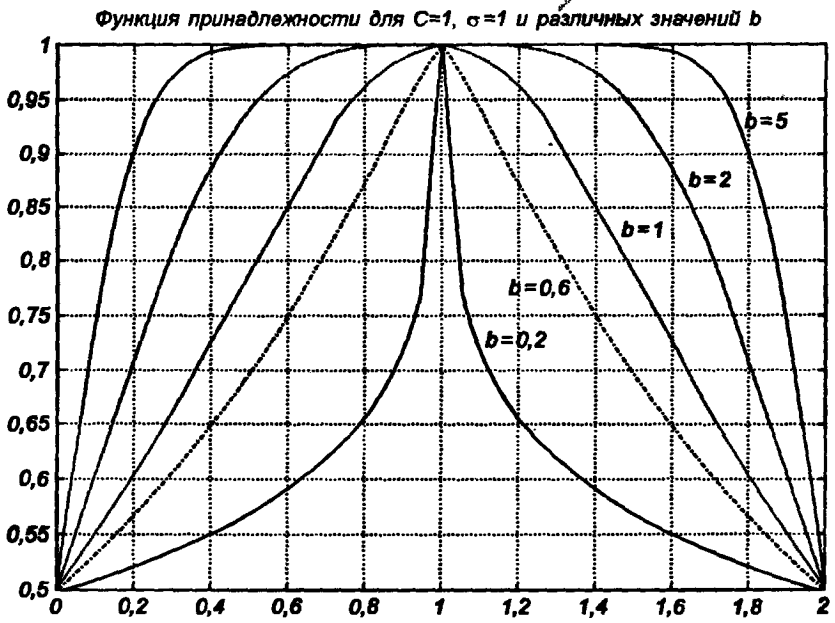

Рис. 11.7. Иллюстрация влияния параметра b на форму гауссовской функции

На нем видно, что при соответствующем подборе показателя степени b (зависимость (11.25)) она может определять как функцию Гаусса, так и треугольную или трапецеидальную функцию. Значение $b = 1$, очевидно, соответствует стандартной гауссовской функции. Обобщенная гауссовская функция также может быть представлена в рациональной форме:

$$\mu_A(x) = \frac{1}{1+\left(\frac{x-c}{\sigma}\right)^{2b}}, \tag{11.26}$$

которая аналогична описываемой выражением (11.25).

Помимо гауссовской функции принадлежности, на практике часто применяется симметричная треугольная функция, которую можно записать в виде

$$\mu_A(x) = \begin{cases} 1-\frac{|x-c|}{d} & \text{для } x \in [c-d, c+d] \\ 0 & \text{для остальных} \end{cases}. \tag{11.27}$$

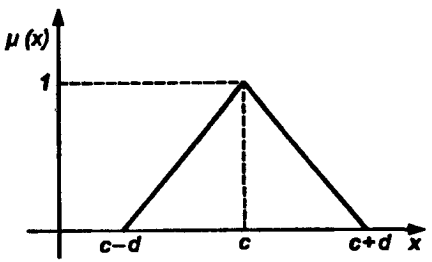

Рис. 11.8. Треугольная форма функции принадлежности

Интерпретация центральной точки c и ширины d для треугольной функции представлена на рис. 11.8. Эта функция тоже нормирована и принимает единичное значение в центральной точке c.

Обобщением треугольной функции является трапецеидальная функция принадлежности, форма и обозначения которой показаны на рис. 11.9.

Если определить $y = c - \frac{t}{2} - \frac{1}{s}$, $z = c + \frac{t}{2} + \frac{1}{s}$, где s обозначает угол наклона, то трапецеидальная функция описывается зависимостью

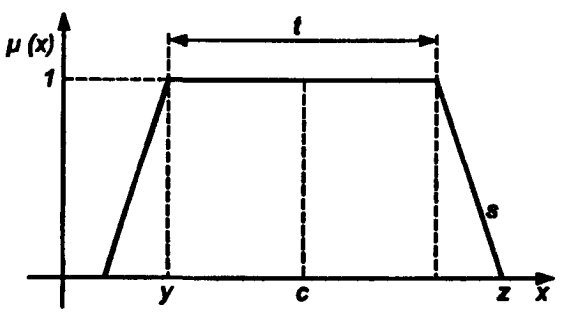

Рис. 11.9. Трапецеидальная форма функции принадлежности

$$\mu_{A^{(i)}}(x_i)=\begin{cases}0 & \text{для } x>z \text{ или } x<y\\ 1 & \text{для } c-\frac{t}{2}\le x\le c+\frac{t}{2}\\ s(z-x) & \text{для } c+\frac{t}{2}\le x\le z\\ s(z-y) & \text{для } y\le x\le c-\frac{t}{2}\end{cases}. \qquad (11.28)$$

Выбор значения $t = 0$ редуцирует трапецеидальную функцию до треугольной формы.

11.5.2. Дефуззификатор

Дефуззификатор трансформирует нечеткое множество в полностью детерминированное точечное решение y. Нечеткое множество представляет зависимость $\mu(y) = \mu_{A\to B}(y)$ как функцию от выходной переменной y. Преобразование этого множества в единственное точечное решение возможно многими способами. Наиболее известны среди них:

- дефуззификация относительно центра области (англ.: *Center of Area*)

$$y_c=\frac{\int_y \mu(y)y\,dy}{\int_y \mu(y)\,dy} \qquad (11.29a)$$

либо в дискретной форме

$$y_c=\frac{\sum_i \mu(y_i)y_i}{\sum_i \mu(y_i)}; \qquad (11.29б)$$

- дефуззификация относительно среднего центра (англ.: *Center Average*)

$$y_c=\frac{\sum_i^M \mu(y_{ci})y_{ci}}{\sum_i \mu(y_{ci})}, \qquad (11.30)$$

где y_{ci} обозначает центр i-го нечеткого правила, а $\mu(y_{ci})$ – значение функции принадлежности, соответствующей этому правилу;

- дефуззификация относительно среднего максимума (англ.: *Mean of Maxima*)

$$y_M=\frac{\sum_{i=1}^m y_i}{m}, \qquad (11.31)$$

где m обозначает количество точек переменной y, в которых $\mu(y_{ci})$ достигает максимального значения. Если функция $\mu(y)$ имеет максимальное значение только в одной точке y_{max}, то $y_M = y_{max}$. Если $\mu(y)$ достигает своих максимальных значений между y_l и y_p, то $y_M = \frac{1}{2}(y_l + y_p)$;

- в форме выбора минимального из максимальных значений y

$$y_s - \textit{наименьшее значение } y, \textit{ для которого } \{ \mu(y) = \max \}; \qquad (11.32)$$

- дефуззификация в форме выбора максимального из максимальных значений y

$$y_l - \textit{наибольшее значение } y, \textit{ для которого } \{ \mu(y) = \max \}. \qquad (11.33)$$

На практике чаще всего применяется дефуззификация относительно среднего центра.

ПРИМЕР 11.5

На рис. 11.10 представлен нечеткий сигнал, полученный после агрегирования двух правил вывода из примера 11.4.

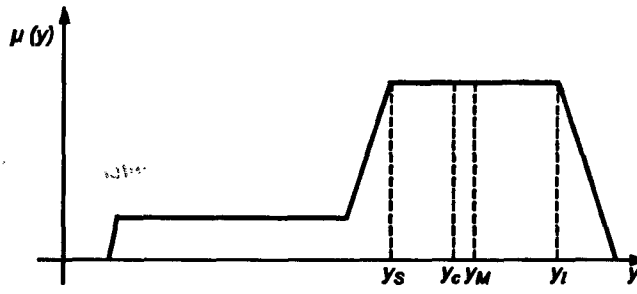

Рис. 11.10. Иллюстрация влияния различных способов дефуззификации на итоговое решение

Применение перечисленных выше способов дефуззификации приводит к получению результатов, соответствующих точкам y_c, y_M, y_s и y_l на этом рисунке.

11.5.3. Модель Мамдани-Заде как универсальный аппроксиматор

Результат, интересный с точки зрения его применения в нечетких сетях, может быть получен при использовании в качестве агрегатора оператора алгебраического произведения с последующей дефуззификацией относительно среднего центра. Следует отметить, что $\mu(y)$ состоит из суммы нечетких функций для импликаций всех M правил, образующих систему нечеткого вывода. В модели Мамдани-Заде каждое из этих M правил определяется уровнем активации условия, $\mu(y_i) = \prod_{j=1}^{N} \mu_A(x_j)$, тогда как y_i – это значение y, при котором величина $\mu(y)$ становится максимальной (либо принимает среднее из максимальных значений). Пусть величина y_i обозначает центр c_i нечеткого множества заключения i-го

правила вывода. Тогда дефуззификация относительно среднего центра ведет к модели Менделя–Ванга [173], в соответствии с которой

$$y = \frac{\sum_{i=1}^{M} c_i \left[\prod_{j=1}^{N} \mu_{A_i}(x_j) \right]}{\sum_{i=1}^{M} \left[\prod_{j=1}^{N} \mu_{A_i}(x_j) \right]} \tag{11.34}$$

Допустим, что существует нечеткая система, описываемая зависимостью (11.34), на вход которой подается последовательность векторов $\boldsymbol{x} = [x_1, x_2, ..., x_N]^T$. При использовании фуззификатора в виде обобщенной гауссовской функции $\mu(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^{2b}\right]$ выходной сигнал у этой системы определяется по формуле

$$y = f(x) = \frac{\sum_{i=1}^{M} c_i \left[\prod_{j=1}^{N} \exp\left[-\left(\frac{x_j - c_j^{(i)}}{\sigma_j^{(i)}} \right)^{2b_j^{(i)}} \right] \right]}{\sum_{i=1}^{M} \left[\prod_{j=1}^{N} \exp\left[-\left(\frac{x_j - c_j^{(i)}}{\sigma_j^{(i)}} \right)^{2b_j^{(i)}} \right] \right]} \tag{11.35}$$

в которой $c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}$ обозначают параметры центра, ширины и формы (условия) j-го компонента вектора $\boldsymbol{x}$ для i-го нечеткого правила вывода. Выражение (11.35) определяет непрерывную функцию, которая может использоваться для аппроксимации произвольно заданной непрерывной функции $g(\boldsymbol{x})$ от многих переменных x_i, образующих вектор $\boldsymbol{x}$. В [160, 173] было доказано, что при соответствующем подборе параметров условия ($c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}$) и заключения ($c_i$) функция (11.35) может аппроксимировать заданную функцию $g(\boldsymbol{x})$ с произвольной точностью ε. Способность нечеткой системы, характеризующейся рядом нелинейных функций от одной переменной, к аппроксимации нелинейной функции от многих переменных свидетельствует о возможностях практического применения нечетких систем. В следующем разделе мы рассмотрим представление формулы (11.35) в виде многослойной сети со структурой, подобной структуре нейронной сети, которая в дальнейшем будет называться *нечеткой нейронной сетью*.

11.6. Модель вывода Такаги–Сугено–Канга

Наибольшую популярность среди нечетких систем адаптивного типа приобрела модель вывода Такаги–Сугено–Канга (TSK) [153]. В этой модели функция заключения определяется нечетким, но точечным образом. Благодаря

этому дефуззификатор на выходе системы не требуется, а модель вывода значительно упрощается. Общая форма модели TSK может быть представлена в виде

$$\textit{если } x_1 \textit{ это } A_1 \, I \, x_2 \textit{ это } A_2 \, I \cdots I \, x_N \textit{ это } A_N, \textit{ то } y = f(x_1, x_2, \ldots, x_N). \quad (11.36)$$

В векторной записи ее можно записать еще проще:

$$\textit{если } \boldsymbol{x} \textit{ это } A, \textit{ то } y = f(\boldsymbol{x}) \quad (11.37)$$

где $f(\boldsymbol{x}) = f(x_1, x_2, \ldots, x_N)$ – четкая функция. В этой зависимости часть, относящаяся к условию, точно такая же, как и в модели Мамдани-Заде. Принципиальное отличие касается заключения, которое представляется в форме функциональной зависимости, чаще всего – в виде полиномиальной функции нескольких переменных. Классическое представление этой функции, чаще всего используемое на практике, – это полином первого порядка

$$y = f(x) = p_0 + \sum_{i=1}^{N} p_i x_i \, . \quad (11.38)$$

в котором коэффициенты $p_0, p_1, \ldots, p_N$ – это цифровые веса, подбираемые в процессе адаптации (обучения). Еще более простая модель вывода TSK получается, если применять функцию $f(x)$ в виде полинома первого порядка, в котором

$$y = f(\boldsymbol{x}) = p_0 \, . \quad (11.39)$$

В этом случае значение p_0 можно отождествить с центром заключения c_i модели Мамдани-Заде, присутствующим в формуле (11.35).

Если в модели вывода TSK используется несколько (M) правил, то выход системы определяется как их средневзвешенное. Приписывая каждому правилу вес w_i, получим выходной сигнал, представленный в виде

$$y = \frac{\sum_{i=1}^{M} w_i y_i}{\sum_{j=1}^{M} w_j} \, , \quad (11.40)$$

или

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{M} w_j} y_i = \sum_{i=1}^{M} w_i' y_i \, . \quad (11.41)$$

Необходимо отметить, что в выражении (11.41) веса w_i отвечают условию нормализации: $\sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{M} w_j} = 1$. Если для каждого i-го правила (где $i = 1, 2, \ldots, M$) реализуется функция TSK первого порядка

$$y_i = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \, , \quad (11.42)$$

то можно получить описание выходной функции модели TSK в форме

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{M} w_j} \left(p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right), \qquad (11.43)$$

которая линейна относительно всех входных переменных системы x_j для $j = 1, 2, ..., N$.

Веса w_i, присутствующие в формуле (11.43), являются нелинейными параметрами функции y. В адаптивных системах они подвергаются обучению для достижения наилучшей приспособленности модели к заданным значениям, тогда как в неадаптивных системах они уточняются для определения уровня активации условия в правиле вывода непосредственно в процессе анализа данных. Подбор этих уровней – это результат агрегирования правил, соответствующих конкретным компонентам вектора x условия; он выполняется с использованием логического или алгебраического произведения так же, как это имело место в модели Мамдани-Заде.

ПРИМЕР 11.6

Этот пример (рис. 11.11) иллюстрирует модель TSK с двумя правилами вывода для системы с двумя входными переменными x_1 и x_2 (аналогично примеру 11.3

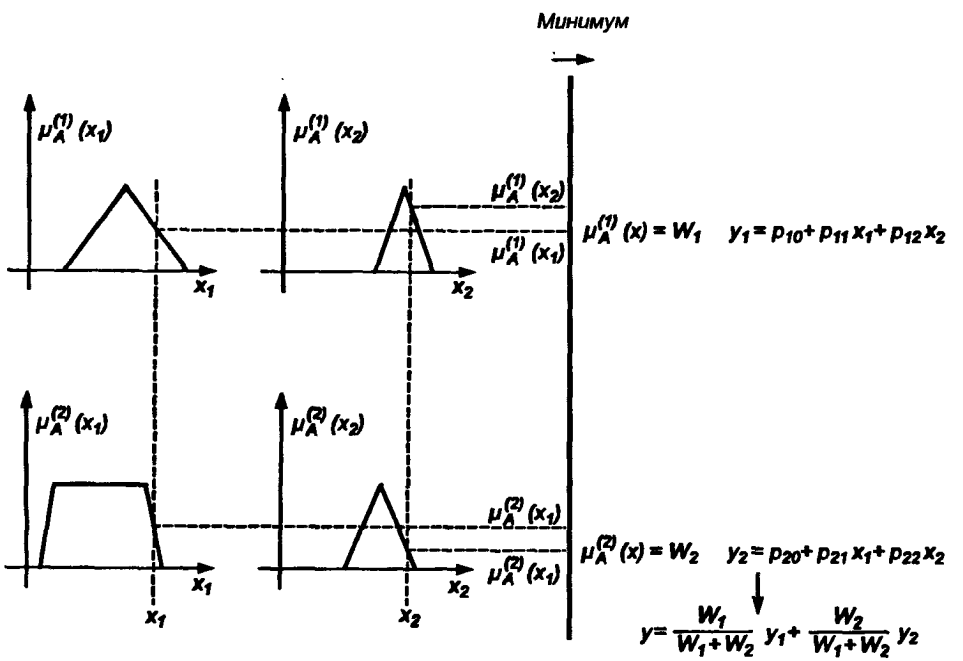

Рис. 11.11. Иллюстрация системы вывода TSK

для модели Мамдани-Заде). Левая часть рисунка относится к условию. Уровень активации $\mu_A(x)$ в этом примере определяется как логическое произведение (минимум) от $\{\mu_A(x_1), \mu_A(x_2)\}$. Веса w_1 и w_2, которые учитываются при агрегировании обоих правил вывода в модели TSK, соответственно равны: $w_1 = \mu_A^{(1)}(x)$, $w_2 = \mu_A^{(2)}(x)$. В правой части рисунка представлены соответствующие формы линейной функциональной зависимости TSK, описывающей заключения обоих правил вывода. Таким образом, окончательный (агрегированный) результат вывода по этой модели TSK в соответствии с (11.43) можно представить в виде

$$y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} = \frac{w_1}{w_1 + w_2} y_1 + \frac{w_2}{w_1 + w_2} y_2,$$

где $y_1 = p_{10} + p_{11} x_1 + p_{12} x_2$, $y_2 = p_{20} + p_{21} x_1 + p_{22} x_2$. Следует отметить, что в отличие от модели Мамдани-Заде выражение, описывающее выходной сигнал, является четким и соответственно отсутствует необходимость дефуззификации.

Раздел 12

НЕЧЕТКИЕ НЕЙРОННЫЕ СЕТИ

Представленные в предыдущем разделе модели вывода Мамдани-Заде и TSK позволяют описать выходной сигнал многомерного процесса как нелинейную функцию входных переменных x_i (i = 1, 2, ..., N) и параметров нечеткой системы (формула (11.34) в модели Мамдани-Заде и (11.43) в модели TSK). В литературе [67, 160] отмечается, что эти выражения позволяют аппроксимировать с произвольной точностью любую нелинейную функцию многих переменных суммой нечетких функций одной переменной. Формулы (11.34) и (11.43) имеют модульную структуру, идеально подходящую для системного представления в виде равномерной многослойной структуры, напоминающей структуру классических нейронных сетей. В дальнейшем мы будем называть их *нечеткими нейронными сетями* (англ.: *neurofuzzy networks*). Характерной особенностью этих сетей является возможность использования нечетких правил вывода для расчета выходного сигнала. В отличие от классических нечетких систем (англ.: *fuzzy logic*) в них вместо непосредственного расчета уровня активации конкретных правил вывода выполняется адаптивный подбор параметров функции фуззификации. Мы обсудим структуру таких сетей, а также алгоритмы обучения, способные адаптировать и линейные веса (аналогично тому, как это производилось в классических сетях), и параметры нечетких функций фуззификатора (аналогично параметрам гауссовской функции в сетях RBF).

12.1. Структура нечеткой сети TSK

Обобщенную схему вывода в модели TSK при использовании M правил и N переменных x_j можно представить в виде

$$\begin{aligned} &IF\ (x_1.IS.\ A_1^{(1)}).AND.(x_2.IS.\ A_2^{(1)}).AND.\ ....\ .AND.(x_N.IS.\ A_N^{(1)}), \\ &\qquad\qquad\qquad\qquad\qquad\qquad THEN\ \ y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j \\ &\ldots \qquad\qquad\qquad \ldots \qquad\qquad\qquad\qquad \ldots \qquad\qquad (12.1) \\ &IF\ (x_1.IS.\ A_1^{(M)}).AND.(x_2.IS.\ A_2^{(M)}).AND.\ ....\ .AND.(x_N.IS.\ A_N^{(M)}), \\ &\qquad\qquad\qquad\qquad\qquad\qquad THEN\ \ y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} x_j\ . \end{aligned}$$

Условие $IF(x_i.\ IS.\ A_i)$ реализуется функцией фуззификации, которая представляется обобщенной функцией Гаусса отдельно для каждой переменной x_i:

$$\mu_A(x_i)=\frac{1}{1+\left(\frac{x_i-c_i}{\sigma_i}\right)^{2b_i}}, \tag{12.2}$$

где $\mu_A(x_i)$ представляет оператор A_i. В нечетких сетях целесообразно задавать это условие в форме алгебраического произведения, из которой следует, что для k-го правила вывода

$$\mu_A^{(k)}(x)=\prod_{j=1}^{N}\left[\frac{1}{1+\left(\frac{x_i-c_j^{(k)}}{\sigma_j^{(k)}}\right)^{2b_j^{(k)}}}\right]. \tag{12.3}$$

При M правилах вывода агрегирование выходного результата сети производится по формуле (11.43), которую можно представить в виде

$$y(x)=\frac{1}{\sum_{k=1}^{N} w_k}\sum_{k=1}^{M} w_k y_k(x), \tag{12.4}$$

где $y_k(x)=p_{k0}+\sum_{j=1}^{N} p_{kj}x_j$. Присутствующие в этом выражении веса w_k интерпретируются как значимость компонентов $\mu_A^{(k)}(x)$, определенных формулой (12.3). При этом условии формуле (12.4) можно сопоставить многослойную структуру сети, изображенную на рис. 12.1. В такой сети выделяется пять слоев.

- Первый слой выполняет раздельную фуззификацию каждой переменной $x_i (i = 1, 2, ..., N)$, определяя для каждого k-го правила вывода значение коэффициента принадлежности $\mu_A^{(k)}(x_i)$ в соответствии с применяемой функцией фуззификации (например, (12.2)). Это параметрический слой с параметрами $c_j^{(k)}, \sigma_j^{(k)}, b_j^{(k)}$, подлежащими адаптации в процессе обучения.
- Второй слой выполняет агрегирование отдельных переменных x_i, определяя результирующее значение коэффициента принадлежности $w_k = \mu_A^{(k)}(x)$ для вектора x (уровень активации правила вывода) в соответствии с формулой (12.3). Это слой непараметрический.
- Третий слой представляет собой генератор функции TSK, рассчитывающий значения $y_k(x)=p_{k0}+\sum_{j=1}^{N} p_{kj}x_j$. В этом слое также производится умножение сигналов $y_k(x)$ на значения w_k, сформированные в предыдущем слое. Это параметрический слой, в котором адаптации подлежат линейные веса p_{kj} для $k = 1, 2, ..., M$ и $j = 1, 2, ..., N$, определяющие функцию следствия модели TSK.

- Четвертый слой составляют два нейрона-сумматора, один из которых рассчитывает взвешенную сумму сигналов $y_k(x)$, а второй определяет сумму весов $\sum_{k=1}^{M} w_k$. Это непараметрический слой.

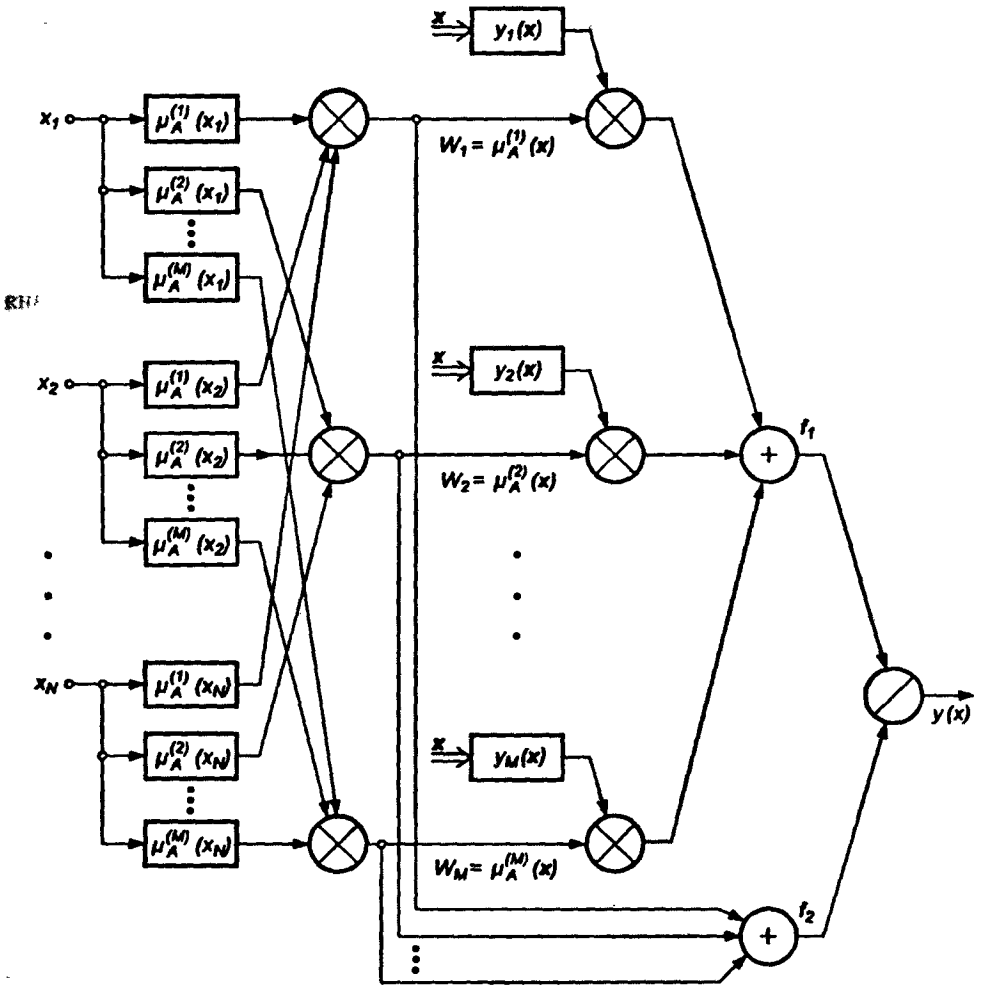

Рис. 12.1. Структура нечеткой нейронной сети TSK

- Последний, пятый слой, состоящий из единственного выходного нейрона, – это нормализующий слой, в котором веса подвергаются нормализации в соответствии с формулой (12.4). Выходной сигнал $y(x)$ определяется выражением, соответствующим зависимости (11.43),

$$y(x) = f(x) = \frac{f_1}{f_2} \cdot \tag{12.5}$$

Это также непараметрический слой.

Из приведенного описания следует, что нечеткая сеть TSK содержит только два параметрических слоя (первый и третий), параметры которых уточняются в процессе обучения. Параметры первого слоя будем называть нелинейными параметрами, поскольку они относятся к нелинейной функции (12.2), а параметры третьего слоя – линейными весами, так как они относятся к параметрам p_{kj} линейной функции TSK.

При уточнении функциональной зависимости (12.4) для сети TSK получаем:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]} \sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]\left[p_{k0} + \sum_{j=1}^{N} p_{kj}x_j\right]. \qquad (12.6)$$

Если принять, что в конкретный момент времени параметры условия зафиксированы, то функция $y(x)$ является линейной относительно переменных x_i ($i = 1, 2, ..., N$).

При наличии N входных переменных каждое правило формирует $N+1$ переменных $p_j^{(k)}$ линейной зависимости TSK. При M правилах вывода это дает $M(N+1)$ линейных параметров сети. В свою очередь, каждая функция принадлежности использует три параметра (c, s, b), подлежащих адаптации. Если принять, что каждая переменная x_i характеризуется собственной функцией принадлежности, то при M правилах вывода мы получим $3MN$ нелинейных параметров. В сумме это дает $M(4N+1)$ линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети.

На практике для уменьшения количества адаптируемых параметров оперируют меньшим количеством независимых функций принадлежности для отдельных переменных, руководствуясь правилами, в которых комбинируются функции принадлежности различных переменных. Если принять, что каждая переменная x_i имеет m различных функций принадлежности, то максимальное количество правил, которые можно создать при их комбинировании, составит: $M = m^N$ (при трех функциях принадлежности, распространяющихся на две переменные, это $3^2 = 9$ правил вывода). Таким образом суммарное количество нелинейных параметров сети при M правилах вывода уменьшается с $3MN$ в общем случае до $3NM^{1/N}$. Количество линейных параметров при подобной модификации остается без изменений, т.е. $M(N+1)$.

12.2. Структура сети Ванга–Менделя

Если использовать в качестве основы дальнейших рассуждений выражение (11.35), вытекающее из модели вывода Мамдани-Заде, можно получить структуру нечеткой сети (рис. 12.2), определенную Л.Вангом и Дж.Менделем [160].

Это четырехслойная структура, в которой первый слой выполняет фуззификацию входных переменных, второй – агрегирование значений активации условия, третий (линейный) – агрегирование M правил вывода (первый нейрон) и генерацию нормализующего сигнала (второй нейрон), тогда как состоящий из одного нейрона выходной слой осуществляет нормализацию, формируя выходной сигнал $y(x)$. Только первый и третий слои являются параметрическими. В первом слое это параметры функции фуззификации ($c_j^{(k)}, \sigma_j^{(k)}, b_j^{(k)}$), а в третьем слое – веса v_1, v_2, ..., v_M, интерпретируемые как центр c_k функции принадлежности следствия k-го нечеткого правила вывода. Представленная на рис. 12.2 сетевая структура реализует функцию аппроксимации (11.34), которую с учетом введенных обозначений можно записать в виде

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]} \sum_{k=1}^{M} v_k \left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]. \qquad (12.7)$$

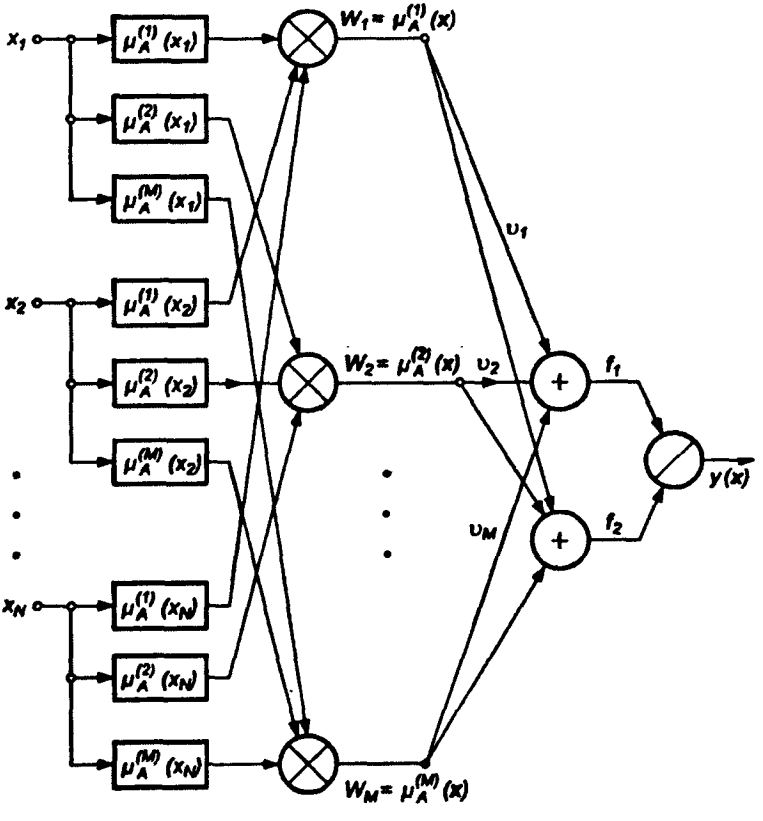

Рис. 12.2. Структура нечеткой нейронной сети Ванга–Менделя

Следует отметить большое сходство структур обеих нечетких сетей. Части, определяющие условие – первый и второй слои – у них идентичны, поскольку они соответствуют компонентам правил вывода *"IF ... "*, одинаково представляемым и в модели Мамдани-Заде, и в модели TSK. Различия наблюдаются в представлении компонентов *"THEN ..."*. В сети TSK результат представляется полиномом первого порядка. В сети Ванга–Менделя результат представляется константой ($v_k = c_k$), которую можно рассматривать как полином нулевого порядка, определяющий центр функции принадлежности следствия. Таким образом, с функциональной точки зрения сеть Ванга–Менделя подобна сети TSK, а точнее – является ее частным случаем.

Задача обеих сетей (TSK и Ванга–Менделя) состоит в таком отображении пар данных (x, d), чтобы ожидаемое значение, соответствующее входному вектору x, формировалось выходной функцией сети $y(x)$. Несмотря на то, что приведенные рассуждения касаются сетей с одним выходным нейроном, они могут быть обобщены на случай систем с несколькими выходами.

Обучение нечетких сетей, так же как и классических сетей, может проводиться либо по алгоритму с учителем, основанному на минимизации целевой функции, задаваемой, как правило, с использованием эвклидовой нормы как

$$E = \frac{1}{2}\sum_{i=1}^{p}(y(x^{(i)}) - d^{(i)})^2 , \qquad (12.8)$$

где p обозначено количество обучающих пар (x, d), либо по алгоритму самоорганизации, согласно которому выполняется группирование (кластеризация) данных.

12.3. Гибридный алгоритм обучения нечетких сетей

Гибридный алгоритм применяется к обеим описанным выше сетевым структурам. Сеть Ванга–Менделя может при этом трактоваться как сеть TSK, в которой все параметры (кроме подлежащего уточнению $v_k = p_{k0}$) p_{kj} ($k = 1, 2, ..., M, j = 1, 2, ..., N$) тождественно равны нулю. Поэтому дальнейшие рассуждения будут касаться сети TSK как более общей, чем сеть Ванга–Менделя.

В гибридном алгоритме подлежащие адаптации параметры разделяются на две группы. Первая из них состоит из линейных параметров p_{kj} третьего слоя, а вторая группа – из параметров нелинейной функции принадлежности первого слоя. Уточнение параметров проводится в два этапа.

- На первом этапе при фиксации определенных значений параметров функции принадлежности (в первом цикле – это значения, полученные в результате инициализации) путем решения системы линейных уравнений

рассчитываются линейные параметры p_{kj} полинома TSK. При известных значениях функции принадлежности зависимость (12.6) можно представить в линейной форме

$$y(x) = \sum_{k=1}^{M} w'_k \left(p_{k0} + \sum_{j=1}^{N} p_{kj} x_j \right), \tag{12.9}$$

где

$$w'_k = \frac{\prod_{j=1}^{N} \mu_A^{(k)}(x_j)}{\sum_{r=1}^{N} \left[\prod_{j=1}^{N} \mu_A^{(r)}(x_j) \right]} = const \tag{12.10}$$

для $k = 1, 2, ..., M$. При p обучающих выборках $(x^{(l)}, d^{(l)})$ $(l = 1, 2, ..., p)$ и замене выходного сигнала сети ожидаемым значением $d^{(l)}$ получим систему из p линейных уравнений вида

$$\begin{bmatrix} w'_{11} & w'_{11}x_1^{(1)} & \dots & w'_{11}x_N^{(1)} \dots & w'_{1M} & w'_{1M}x_1^{(1)} & \dots w'_{1M}x_N^{(1)} \\ w'_{21} & w'_{21}x_1^{(2)} & \dots & w'_{21}x_N^{(2)} \dots & w'_{2M} & w'_{2M}x_1^{(2)} & \dots w'_{2M}x_N^{(2)} \\ & \dots & & \dots & & \dots & \\ w'_{p1} & w'_{p1}x_1^{(p)} & \dots & w'_{p1}x_N^{(p)} \dots & w'_{pM} & w'_{pM}x_1^{(p)} & \dots w'_{pM}x_N^{(p)} \end{bmatrix} \begin{bmatrix} p_{10} \\ \dots \\ p_{1N} \\ \dots \\ p_{M0} \\ \dots \\ p_{MN} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \dots \\ d^{(p)} \end{bmatrix}, \tag{12.11}$$

где w'_{k_i} обозначает уровень активации (вес) условия i-го правила при предъявлении k-го входного вектора x. Это выражение можно записать в сокращенной матричной форме

$$\mathbf{A}\,\boldsymbol{p} = \boldsymbol{d}. \tag{12.12}$$

Размерность матрицы A равна $p \times (N+1)M$, при этом обычно количество строк значительно больше количества столбцов $(N + 1)M$. Решение этой системы уравнений можно получить за один шаг при помощи псевдоинверсии матрицы A

$$\boldsymbol{p} = \mathbf{A}^{+}\boldsymbol{d}. \tag{12.13}$$

Псевдоинверсия матрицы заключается в проведении декомпозиции SVD с последующим сокращением ее размерности так, как это представлено в разделе 5. Подробности вычислительной реализации алгоритма SVD можно найти, например, в работе Г. Голуба и К. ВанЛоана [42].

На втором этапе после фиксации значений линейных параметров p_{kj} рассчитываются фактические выходные сигналы $y(i)$ сети для $i = 1, 2, ..., p$,

для чего используется линейная зависимость

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(p)} \end{bmatrix} = \boldsymbol{Ap} \tag{12.14}$$

и следом за ними – вектор ошибки $\boldsymbol{\varepsilon} = \boldsymbol{y} - \boldsymbol{d}$. Сигналы ошибок направляются через подключенную сеть по направлению ко входу сети (обратное распространение) вплоть до первого слоя, где могут быть рассчитаны компоненты градиента целевой функции относительно конкретных параметров $c_j^{(k)}, \sigma_j^{(k)}, b_j^{(k)}$. После формирования вектора градиента параметры уточняются с использованием одного из градиентных методов обучения. Если применяется простейший метод наискорейшего спуска, то соответствующие формулы адаптации принимают форму:

$$c_j^{(k)}(n+1) = c_j^{(k)}(n) - \eta_c \frac{\partial E(n)}{\partial c_j^{(k)}}, \tag{12.15}$$

$$\sigma_j^{(k)}(n+1) = \sigma_j^{(k)}(n) - \eta_\sigma \frac{\partial E(n)}{\partial \sigma_j^{(k)}}, \tag{12.16}$$

$$b_j^{(k)}(n+1) = b_j^{(k)}(n) - \eta_b \frac{\partial E(n)}{\partial b_j^{(k)}}, \tag{12.17}$$

где n обозначает номер очередной итерации.

После уточнения нелинейных параметров вновь запускается процесс адаптации линейных параметров функции TSK (первый этап) и нелинейных параметров (второй этап). Этот цикл повторяется вплоть до стабилизации всех параметров процесса. Формулы (12.15) – (12.17) требуют расчета градиента целевой функции относительно параметров функции принадлежности. Окончательный вид этих формул зависит как от используемого определения функции погрешности на выходе сети, так и от формы функции принадлежности. Например, при использовании обобщенной функции Гаусса

$$\mu_A(x) = \frac{1}{1 + \left(\frac{x - c}{\sigma} \right)^{2b}} \tag{12.18}$$

соответствующие формулы градиента целевой функции (12.8) для одной пары обучающих данных (x, d) принимают вид

$$\frac{\partial E}{\partial c_j^{(k)}} = (y(x) - d) \sum_{r=1}^{M} \left[p_{r0} + \sum_{j=1}^{N} p_{rj} x_j \right] \frac{\partial w_r'}{\partial c_j^{(k)}}, \tag{12.19}$$

$$\frac{\partial E}{\partial \sigma_j^{(k)}} = (y(x) - d) \sum_{r=1}^{M} \left[p_{r0} + \sum_{j=1}^{N} p_{rj} x_j \right] \frac{\partial w_r'}{\partial \sigma_j^{(k)}} , \qquad (12.20)$$

$$\frac{\partial E}{\partial b_j^{(k)}} = (y(x) - d) \sum_{r=1}^{M} \left[p_{r0} + \sum_{j=1}^{N} p_{rj} x_j \right] \frac{\partial w_r'}{\partial b_j^{(k)}} . \qquad (12.21)$$

Производные $\frac{\partial w_r'}{\partial c_j^{(k)}}$, $\frac{\partial w_r'}{\partial \sigma_j^{(k)}}$ и $\frac{\partial w_r'}{\partial b_j^{(k)}}$, определенные на основе зависимости (12.10) и (12.18), можно записать как

$$\frac{\partial w_r'}{\partial c_j^{(k)}} = \frac{\delta_{rk} m(x_j) - l(x_j)}{[m(x_j)]^2} \prod_{i=1, i \neq j}^{N} [\mu_A^{(k)}(x_i)] \frac{\left[\frac{2b_j^{(k)}}{\sigma_j^{(k)}} \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)} - 1} \right]}{\left[1 + \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}} \right]^2}, \qquad (12.22)$$

$$\frac{\partial w_r'}{\partial \sigma_j^{(k)}} = \frac{\delta_{rk} m(x_j) - l(x_j)}{[m(x_j)]^2} \prod_{i=1, i \neq j}^{N} [\mu_A^{(k)}(x_i)] \frac{\left[\frac{2b_j^{(k)}}{\sigma_j^{(k)}} \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}} \right]}{\left[1 + \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}} \right]^2}, \qquad (12.23)$$

$$\frac{\partial w_r'}{\partial b_j^{(k)}} = \frac{\delta_{rk} m(x_j) - l(x_j)}{[m(x_j)]^2} \times$$

$$\times \prod_{i=1, i \neq j}^{N} [\mu_A^{(k)}(x_i)] \frac{\left[-2 \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}} \ln \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right) \right]}{\left[1 + \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}} \right]^2} , \qquad (12.24)$$

для $r = 1, 2, ..., M$, где δ_{rk} обозначает дельту Кронекера, $l(x_j) = \prod_{j=1}^{N} \mu_A^{(k)}(x_i)$, $m(x_j) = \sum_{k=1}^{M} \left[\prod_{j=1}^{N} \mu_A^{(k)}(x_i) \right]$. Несмотря на сложную структуру приведенных формул, выражающих компоненты вектора градиента, они позволяют аналитически определить величины, необходимые для уточнения параметров нечеткой сети.

При практической реализации гибридного метода обучения нечетких сетей доминирующим фактором их адаптации считается первый этап, на котором веса p_{kj} подбираются с использованием псевдоинверсии за один шаг. Для

уравновешивания его влияния второй этап (подбор нелинейных параметров градиентным методом) многократно повторяется в каждом цикле.

Представленный гибридный алгоритм – один из наиболее эффективных способов обучения нечетких сетей. Его главная черта состоит в разделении процесса обучения на два обособленных во времени этапа. На каждом этапе уточняется только часть параметров сети. Если принять во внимание, что вычислительная сложность каждого алгоритма оптимизации пропорциональна (нелинейно) количеству параметров, то уменьшение размерности задачи оптимизации существенным образом сокращает количество математических операций и увеличивает скорость выполнения алгоритма. Благодаря этому гибридный алгоритм значительно более эффективен, чем обычный градиентный алгоритм фронтального типа, согласно которому уточнение всех параметров сети производится параллельно и одновременно.

12.4. Применение алгоритма самоорганизации для обучения нечеткой сети

Алгоритм самоорганизации приписывает вектор x к соответствующей группе данных, представляемых центром c_i, с использованием обучения конкурентного типа подобно тому, как это имело место в сетях с самоорганизацией Кохонена. При обучении этого типа процесс самоорганизации становится возможным при предъявлении вектора x. Базовая форма алгоритма самоорганизации позволяет точно определять положение центров c_i соответствующих групп данных (кластеров), на которые подразделяется многомерное пространство. Эти центры в последующем могут использоваться в гибридном алгоритме обучения нечеткой сети в качестве начальных значений, что существенно ускоряет процесс обучения и гарантирует сходимость решения к глобальному минимуму.

12.4.1. Алгоритм нечеткой самоорганизации C-means

Допустим, что в сети существует K нечетких нейронов с центрами в точках c_i (i = 1, 2, ..., K). Начальные значения этих центров могут быть выбраны случайным образом из областей допустимых значений соответствующих компонентов векторов x_j (j = 1, 2, ..., p), использованных для обучения. Пусть функция фуззификации задана в форме обобщенной функции Гаусса, выраженной формулой (12.2).

Подаваемый на вход сети вектор x_j будет принадлежать к различным группам, представляемым центрами c_i, в степени u_{ij}, причем $0 \le u_{ij} \le 1$, а суммарная степень принадлежности ко всем группам, очевидно, равна 1. Поэтому

$$\sum_{i=1}^{K} u_{ij} = 1 \qquad (12.25)$$

для j = 1, 2, ..., p. Функцию погрешности, соответствующую такому

представлению, можно определить как сумму частных погрешностей принадлежности к центрам c_i с учетом степени принадлежности m_{ij}. Следовательно,

$$E = \sum_{i=1}^{K} \sum_{j=1}^{p} u_{ij}^{m} \| c_i - x_j \|^2 , \tag{12.26}$$

где m – это весовой коэффициент, который принимает значения из интервала $[1, \infty)$. Цель обучения с самоорганизацией состоит в таком подборе центров c_i, чтобы для заданного множества обучающих векторов x_j обеспечить достижение минимума функции (12.26) при одновременном соблюдении условий ограничения (12.25). Таким образом возникает задача минимизации нелинейной функции (12.26) с p ограничениями типа (12.25). Решение этой задачи можно свести к минимизации функции Лагранжа, определенной в виде [60]

$$LE = \sum_{i=1}^{K} \sum_{j=1}^{N} u_{ij}^{m} \| c_i - x_j \|^2 + \sum_{j=1}^{p} \lambda_j \left(\sum_{i=1}^{K} u_{ij} - 1 \right), \tag{12.27}$$

где λ_j ($j = 1, 2, ..., p$) – это множители Лагранжа. В [60] доказано, что решение задачи (12.27) можно представить в виде

$$c_i = \frac{\sum_{j=1}^{p} u_{ij}^{m} x_j}{\sum_{j=1}^{p} u_{ij}^{m}} \tag{12.28}$$

и

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d_{ij}^2}{d_{kj}^2} \right)^{\frac{1}{m-1}}} , \tag{12.29}$$

где d_{ij} – это эвклидово расстояние между центром c_i и вектором x_j, $d_{ij} = \| c_i - x_j \|$. Поскольку точные значения центров c_i в начале процесса не известны, алгоритм обучения должен быть итерационным. Он может быть сформулирован в следующем виде.

1. Выполнить случайную инициализацию коэффициентов u_{ij}, выбирая их значения из интервала [0, 1] таким образом, чтобы соблюдалось условие (12.25).
2. Определить K центров c_i в соответствии с (12.28).
3. Рассчитать значение функции погрешности согласно выражению (12.26). Если ее значение окажется ниже установленного порога либо если уменьшение этой погрешности относительно предыдущей итерации пренебрежимо мало, то завершить вычисления. Последние значения центров составляют искомое решение. В противном случае перейти к п. 4.
4. Рассчитать новые значения u_{ij} по формуле (12.29) и перейти к п. 2.

Такую процедуру нечеткой самоорганизации будем называть алгоритмом *C-means*.

Многократное повторение итерационной процедуры ведет к достижению минимума функции E, который необязательно будет глобальным минимумом. Качество находимых центров, оцениваемое значением функции погрешности E, существенным образом зависит от предварительного подбора как значений u_{ij}, так и центров c_i. Наилучшим может быть признано такое размещение центров, при котором они располагаются в областях, содержащих наибольшее количество предъявленных векторов x_j. При таком подборе центров они будут представлять векторы данных x_j с наименьшей суммарной погрешностью.

Поэтому начало итерационной процедуры расчета оптимальных значений центров должно предваряться процедурой их инициализации. К наиболее известным алгоритмам инициализации относятся алгоритмы пикового группирования и разностного группирования данных.

12.4.2. Алгоритм пикового группирования

Алгоритм пикового группирования был предложен Р. Егером и Д. Филевым [60, 173]. В качестве меры плотности размещения векторов x_j в нем генерируются так называемые пиковые функции. При использовании p входных векторов создается сетка, равномерно накрывающая пространство векторов x_j. Узлы этой сетки рассматриваются как потенциальные центры v, и для каждого из них рассчитывается пиковая функция $m(v)$

$$m(v) = \sum_{j=1}^{p} \exp\left(-\frac{\| v - x_j \|^{2b}}{2\sigma^2} \right). \qquad (12.30)$$

Коэффициент σ – это константа, индивидуально подбираемая для каждой конкретной задачи, а b – показатель степени обобщенной функции Гаусса, которая применяется в нечеткой сети.

Величина функции $m(v)$ рассматривается как оценка высоты пиковой функции. Она пропорциональна количеству векторов x_j, находящихся в окрестности потенциального центра v. Малое значение $m(v)$ свидетельствует о том, что центр v располагается в области, в которой сосредоточено небольшое количество векторов x_j. Следует обратить внимание, что коэффициент σ оказывает незначительное влияние на итоговые пропорции между $m(v)$ для различных значений v, поэтому подбор его величины не является критичным.

После расчета значений $m(v)$ для всех потенциальных центров среди них отбираются первые (c_1), имеющие наибольшее значение $m(v)$. Для выбора следующих центров необходимо прежде всего исключить c_1 и узлы, расположенные в непосредственной близости от c_1. Это можно сделать путем переопределения пиковой функции за счет отсечения от нее функции Гаусса с

центром в точке c_1. Если эту вновь определяемую функцию обозначить $m_{new}(v)$, то получим

$$m_{new}(v) = m(v) - m(c_1)\exp\left(-\frac{\| v - x_j \|^{2b}}{2\sigma^2}\right). \qquad (12.31)$$

Необходимо обратить внимание, что новая функция $m_{new}(v)$ имеет нулевое значение в точке c_1. Рис. 12.3 иллюстрирует типичный процесс пикового группирования в двумерном пространстве [60]. На рис. 12.3*а* показан график исходной функции $m(v)$, на рис. 12.3*б* – график функции $m_{new}(v)$ после отсечения первого центра в точке c_1, на рис. 12.3*в* – график после отсечения второго центра в точке c_2, а на рис. 12.3*г* – после отсечения третьего центра в точке c_3. Заметно, что последовательное отсечение центров (с максимальными значениями пиковой функции) позволяет обнаруживать и устранять очередные центры.

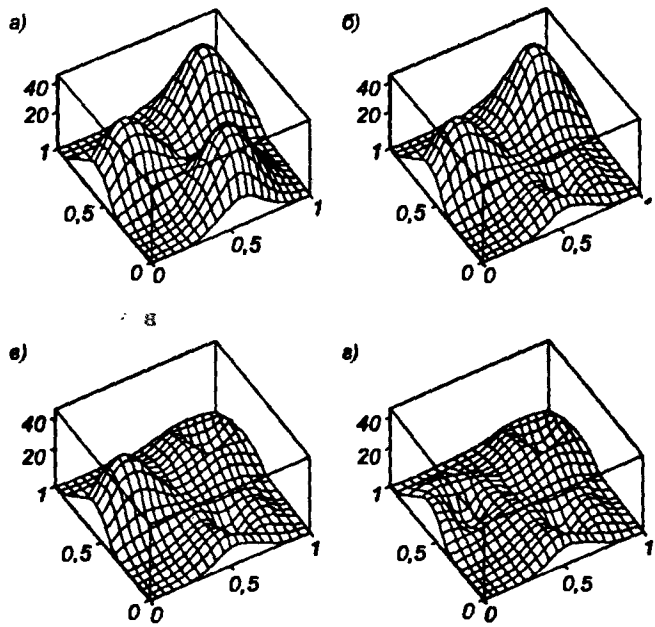

Рис. 12.3. Иллюстрация функционирования алгоритма пикового группирования

Процесс нахождения следующих центров c_2, c_3, ... осуществляется последовательно на модифицированных значениях функции $m_{new}(v)$, получаемых при исключении ближайшего окружения центра, обнаруженного на предыдущем этапе. Он завершается в момент локализации всех центров, используемых в модели нечеткой сети. Метод пикового группирования эффективен, если размерность вектора x не слишком велика. В противном случае (при большом количестве компонентов x) число потенциальных

центров нарастает лавинообразно, и процесс расчета очередных пиковых функций становится слишком длительным, а процедура малоэффективной.

12.4.3. Алгоритм разностного группирования

Алгоритм разностного группирования данных – это модификация предыдущего алгоритма, в которой обучающие векторы x_i рассматриваются в качестве потенциальных центров v. Пиковая функция $D(x_i)$ в этом алгоритме задается в виде

$$D(x_i) = \sum_{j=1}^{p} \exp\left(- \frac{\| x_i - x_j \|^{2b}}{\left(\frac{r_a}{2}\right)^2} \right). \qquad (12.32)$$

Значение коэффициента r_a определяет сферу соседства. На значение $D(x_i)$ существенным образом влияют только те векторы x_j, которые расположены в пределах этой сферы. При большой плотности точек вокруг x_i (потенциального центра) значение функции $D(x_i)$ велико. Напротив, малое ее значение свидетельствует о том, что в окрестности x_i находится незначительное количество данных. Такая точка считается "неудачным" кандидатом в центры. После расчета значений пиковой функции для каждой точки x_j отбирается вектор x, для которого мера плотности $D(x)$ оказалась наибольшей. Именно эта точка становится первым отобранным центром c_1. Выбор следующего центра возможен после исключения предыдущего и всех точек, лежащих в его окрестности. Так же, как и в методе пикового группирования, пиковая функция переопределяется в виде

$$D_{new}(x_i) = D(x_i) - D(c_1) \exp\left(- \frac{\| x_i - c_1 \|^{2b}}{\left(\frac{r_b}{2}\right)^2} \right). \qquad (12.33)$$

При новом определении функции D коэффициент r_b обозначает новое значение константы, задающей сферу соседства очередного центра. Обычно соблюдается условие $r_b \geq r_a$. Пиковая функция $D_{new}(x_i)$ принимает нулевое значение при $x_i = c_1$ и близка к нулю в ближайшей окрестности этой точки.

После модификации значений пиковой функции отыскивается следующая точка x, для которой величина $D_{new}(x)$ оказывается максимальной. Эта точка становится следующим центром c_2. Процесс поиска очередного центра возобновляется после исключения компонентов, соответствующих уже отобранным точкам. Инициализация завершается в момент фиксации всех центров, предусмотренных начальными условиями.

В соответствии с описанным алгоритмом происходит самоорганизация множества векторов x, состоящая в нахождении оптимальных значений цент-

ров, представляющих множество данных с минимальной погрешностью. Если мы имеем дело с множеством обучающих данных в виде пар векторов (x_i, d_i) так, как это происходит при обучении с учителем, то для нахождения центров, соответствующих множеству векторов d_i, достаточно сформировать расширенную версию векторов x в форме

$$x_i \leftarrow [x_i, d_i]. \tag{12.32}$$

Процесс группирования, проводимый с предъявлением расширенных векторов x_i, позволяет определить также расширенные версии центров c_i. Если принять во внимание, что размерность каждого нового центра равна сумме размерностей векторов x и d, то в описании этого центра можно легко выделить часть p, соответствующую векторам x (первые N компонентов), и остаток q, соответствующий вектору d. Таким образом можно получить центры как входных переменных, так ожидаемых выходных значений

$$c_i = [p_i, q_i] \tag{12.35}$$

для $i = 1, 2, ..., K$. В случае применения нечетких правил с одним выходом векторы d и q сводятся к скалярным величинам d и q соответственно. Таким образом, при использовании правила вывода Ванга–Менделя процесс самоорганизации позволяет восстановить функцию $f(x)$, аппроксимирующую множество данных (x_i, d_i) для $i = 1, 2, ..., p$. В частности, при введенных выше обозначениях формула (12.7) принимает вид:

$$f(x) = \frac{\sum_{l=1}^{K} q_l \exp\left(-\frac{\| x - p_l \|^{2b_l}}{\sigma_l^{2b_l}}\right)}{\sum_{l=1}^{K} \exp\left(-\frac{\| x - p_l \|^{2b_l}}{\sigma_l^{2b_l}}\right)}, \tag{12.36}$$

согласно которому все центры подбираются оптимальным образом. При этом остальные параметры (b_l, σ_l), менее критичные для сходимости алгоритма, могут эффективно подбираться гибридным методом при небольшом количестве итераций. Конечно, итерационный процесс при реализации гибридного метода охватывает также и расчеты координат центров, однако с учетом их удачного начального размещения изменения, вносимые в процессе обучения, обычно оказываются очень незначительными.

12.4.4. Алгоритм нечеткой самоорганизации Густафсона–Кесселя

В классическом алгоритме *C-means*, представленном в подразделе 12.4.1, нейрон-победитель выбирается на основании обычного эвклидового расстояния между вектором x и центром c кластера, т.е.

$$d(x, c) = \| x - c \| = \sqrt{(x - c)^T (x - c)}\ . \tag{12.37}$$

Определенное таким образом расстояние учитывалось в формуле (12.26), характеризующей значение функции погрешности. При подобном задании меры расстояния между двумя векторами множество точек, равноудаленных от центрального нейрона (победителя), принимает форму окружности с одинаковым масштабом по всем координатам. Если входные данные образуют группы, форма которых отличается от окружности, либо если шкалы значений отдельных координат вектора сильно различаются, рассчитанные значения будут отражать принадлежность векторов x конкретным кластерам. В такой ситуации качество группирования можно существенно повысить за счет применения усовершенствованной версии алгоритма самоорганизации, называемой алгоритмом Густафсона–Кесселя (ГК) [185].

По отношению к обычному алгоритму *C-means* главное изменение состоит во введении в формулу расчета расстояния между векторами масштабирующей матрицы **A**. При этом масштабированное расстояние между вектором x и центром c определяется по формуле

$$d(x,c) = \| x-c \| = \sqrt{(x-c)^T A(x-c)} \ . \qquad (12.38)$$

Легко показать, что определенная таким образом мера расстояния масштабирует единичный вектор e_i, где $e_i = [0, 0, ..., 0, 1, 0, ..., 0]^T$, пропорционально квадратному корню i-го диагонального значения матрицы **A**. Вследствие этого

$$\| e_i \| = \sqrt{e_i^T A e_i} = \sqrt{A_{ii}} \ . \qquad (12.39)$$

В качестве масштабирующей обычно применяется симметричная положительно определенная матрица, т.е. матрица, у которой все собственные значения являются действительными и положительными. Множество собственных векторов, удовлетворяющих таким собственным значениям, образует в этом случае ортогональную базу многомерного пространства. В новом масштабированном пространстве длины нормированных собственных векторов матрицы **A** трансформируются согласно формуле

$$\sqrt{\| v_i^2 \|} = \sqrt{v_i^T A v_i} = \sqrt{v_i^T \lambda_i v_i} = \sqrt{\lambda_i v_i^T v_i} = \sqrt{\lambda_i} \ . \qquad (12.40)$$

Таким образом, длины собственных векторов масштабируются с коэффициентом $\sqrt{\lambda_i}$.

Так же как и при использовании алгоритма *C-means*, цель обучения сети с применением алгоритма Густафсона–Кесселя состоит в таком размещении центров, чтобы минимизировать функцию погрешности, определяемую в несколько более общем виде, а именно

$$E = \sum_{i=1}^{K} \sum_{j=1}^{p} u_{ij}^m d^2(x_j, c_i) \ , \qquad (12.41)$$

где расстояние между вектором x_j и центром c_i определяется с учетом масштабирования как

$$d(x_j, c_i) = \sqrt{(x_j - c_i)^T A_i (x_j - c_i)} \quad . \tag{12.42}$$

Решение задачи оптимального размещения центров по алгоритму Густафсона–Кесселя происходит так же, как и по алгоритму *C-means* путем многократного пересчета коэффициентов принадлежности u_{ij} по формуле (12.29) и координат центров c_i по формуле (12.28), но с учетом масштабирования при расчете расстояний. Алгоритм Густафсона–Кесселя может быть сформулирован в следующем виде.

1. Произвести начальное размещение центров в пространстве данных. Эта инициализация может быть случайной или основанной на результатах пикового или разностного группирования данных. Создать элементарную форму масштабирующей матрицы $\mathbf{A}_i$.
2. Сформировать матрицу коэффициентов принадлежности всех векторов $x_j (i = 1, 2, ..., p)$ к центрам $c_i (i = 1, 2, ..., K)$ путем расчета значений u_{ij} по формуле

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d^2(x_j, c_i)}{d^2(x_j, c_k)} \right)^{\frac{1}{m-1}}} , \tag{12.43}$$

   где $d^2(x_j, c_i)$ определяется из выражения (12.42). Если для некоторого $j = I$ $d_{ij} = 0$, то принять $u_{il} = 1$ и $u_{ij} = 0$ для всех j, отличных от I.
3. Рассчитать новое размещение центров в соответствии с формулой

$$c_i = \frac{\sum_{j=1}^{p} u_{ij}^m x_j}{\sum_{j=1}^{p} u_{ij}^m} . \tag{12.44}$$

4. Сгенерировать для каждого центра матрицу ковариации $\mathbf{S}_i$

$$\mathbf{S}_i = \sum_{j=1}^{p} u_{ij}^m (x_j - c_i)(x_j - c_i)^T . \tag{12.45}$$

5. Рассчитать новую масштабирующую матрицу для каждого i-го центра $(i = 1, 2, ..., K)$ по формуле

$$\mathbf{A}_i = \sqrt[N]{\det(\mathbf{S}_i)}\,\mathbf{S}_i^{-1} , \tag{12.46}$$

   где N обозначает размерность входного вектора x.
6. Если последние изменения положений центров и матрицы ковариации пренебрежимо малы по отношению к предыдущим значениям и не превышают изначально заданной пороговой величины ε, то завершить итерационный процесс; в противном случае перейти к п. 2.

Функционирующий таким образом алгоритм обучения параллельно генерирует все центры самоорганизующихся нечетких нейронов и связанные с ними масштабирующие матрицы, используемые при расчете расстояний. По завершении процесса обучения как положения центров, так и значения элементов масштабирующих матриц фиксируются и могут использоваться в режиме эксплуатации сети.

Работа алгоритма будет проиллюстрирована на примере множества данных, изображенных на рис. 12.4.

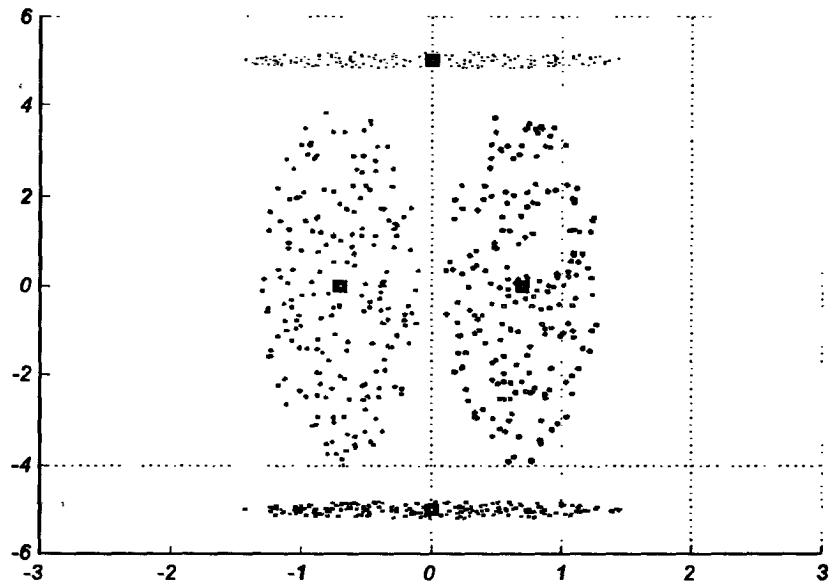

Рис. 12.4. Распределение данных на четыре группы с неравномерной структурой

Это множество образовано четырьмя группами данных, сгенерированных случайным образом и размещенных в окрестностях центров со следующими координатами (x, y):

(0, 5),
(0, −5),
(0,7, 0),
(–0,7, 0).

Разбросы данных по осям 0-x и 0-y составляют: (1.5, 0,2) – для первого кластера, (1.5, 0,2) – для второго кластера, (0,6, 4) – для третьего кластера и (0,6, 4) – для четвертого кластера. Главная проблема группирования этих данных состоит в том, что граничные данные вытянутых кластеров лежат ближе к центру соседнего кластера, чем своего собственного. Поэтому применение обычного алгоритма *C-means* привело бы к некорректной классификации данных.

С помощью нечеткой самоорганизации Густафсона–Кесселя уточнено размещение 4 центров и рассчитаны соответствующие им матрицы ковариации.

Окончательные координаты центров, найденные алгоритмом ГК, равны:

первый центр: (0,0197, – 4,9914),
второй центр: (0,7085, – 0,1310),
третий центр: (–0,7080, – 0,0951),
четвертый центр: (–0,0895, 4,9894).

Заметно, что воссозданные алгоритмом позиции центров лишь незначительно отличаются от принятых начальных координат центральных точек областей, в которых случайным образом генерировались данные. Матрицы трансформации, соответствующие конкретным нейронам, имеют вид:

$$\mathbf{A}_1 = \begin{bmatrix} 0{,}2130 & 0{,}0224 \\ 0{,}0224 & 4{,}6999 \end{bmatrix}, \qquad \mathbf{A}_2 = \begin{bmatrix} 5{,}7251 & -0{,}0787 \\ -0{,}0787 & 0{,}1760 \end{bmatrix},$$

$$\mathbf{A}_3 = \begin{bmatrix} 5{,}5401 & 0{,}0261 \\ 0{,}0261 & 0{,}1812 \end{bmatrix}, \qquad \mathbf{A}_4 = \begin{bmatrix} 0{,}1681 & 0{,}0041 \\ 0{,}0041 & 5{,}9460 \end{bmatrix}.$$

На рис. 12.5 представлены результаты самоорганизации тестовых данных из примера в режиме эксплуатации. Вокруг каждого центра размещены равноудаленные линии тестовых данных, образующие эллипсовидную структуру. Все точки расположены внутри области, ограниченной внешним эллипсом. Это свидетельствует о корректном упорядочении данных по конкретным областям.

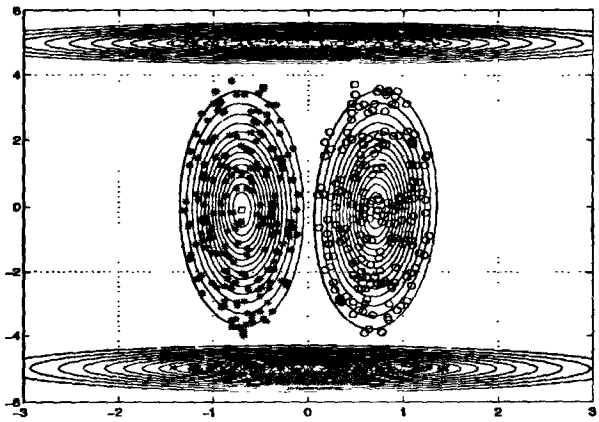

Рис. 12.5. Распределение линий равной удаленности данных от центров соответствующих кластеров, полученное с помощью алгоритма ГК

Обратим внимание, что независимо от способа реализации алгоритма обучения сеть с нечеткой самоорганизацией выполняет нечеткое группирование данных путем приписывания их к различным центрам на основе коэффициентов принадлежности, значения которых изменяются от нуля до единицы. Это означает, что каждый вектор x представляется множеством центров, причем влияние каждого из них на значение вектора различно и зависит от величины

коэффициента принадлежности. Если принять, что вектор x_j представляется К центрами c_i (i = 1, 2, ..., K), а принадлежность вектора к каждому центру задана коэффициентом u_{ij} (формула (12.43)), то реконструкция исходного вектора x_j происходит согласно выражению

$$x_j = \sum_{i=1}^{K} u_{ij} c_i . \qquad (12.47)$$

Заметно, что влияние каждого центра на окончательное значение реконструированного вектора различно и зависит от расстояния между этим центром и исходным вектором x. В этом существенное отличие нечеткой самоорганизации от классической самоорганизации Кохонена, при которой реконструкция вектора выполняется исключительно на базе одного центра, ближайшего данному вектору, путем простого приписывания ему значения этого центра.

12.4.5. Сеть с нечеткой самоорганизацией в гибридной структуре

При практической реализации систем с нечеткой самоорганизацией возникает необходимость преобразовать коэффициенты принадлежности в требуемую форму представления выходного вектора. При реконструкции только вектора x достаточно простого взвешенного суммирования центров в соответствии с формулой (12.47). При более сложных операциях преобразования сигналов сеть с нечеткой самоорганизацией используется в качестве одного из компонентов более общей сетевой структуры. Наиболее известный пример – это гибридная сеть, обобщенная структура которой приведена на рис. 9.13. Уточненная структура гибридной нечеткой сети изображена на рис. 12.6.

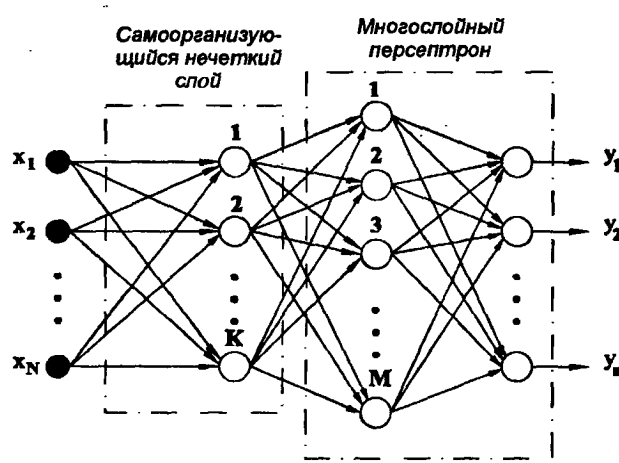

Рис. 12.6. Структура гибридной нечеткой сети

Гибридная сеть объединяет в себе сеть с нечеткой самоорганизацией, выполняющей функции препроцессора, и многослойный (обычно двухслойный) персептрон (MLP) в качестве постпроцессора.

Выходы самоорганизующегося слоя используются в качестве входов многослойного персептрона. Если на вход сети подается вектор $\boldsymbol{x} = [x_1, x_2, ..., x_N]^T$, то на выходе слоя с самоорганизацией формируется вектор $\boldsymbol{u}$, состоящий из коэффициентов принадлежности $\boldsymbol{x}$ к конкретным центрам: $\boldsymbol{u} = [u_{i1}(x), u_{i2}(x), ..., u_{iK}(x)]^T$. Конкретные компоненты u_{ij} рассчитываются в соответствии с универсальной формулой (12.43). Они удовлетворяют условию нормализации $\sum_{i=1}^{K} u_{ij} = 1$ для каждого вектора x_j.

Количество входов персептронной компоненты гибридной сети равно количеству самоорганизующихся нейронов. Количество скрытых слоев и число нейронов в этих слоях может быть, в принципе, произвольным, хотя обычно для восстановления данных с требуемой точностью достаточно одного слоя. Размерность выходного слоя MLP (т.е. количество сигналов y_i, составляющих фактический выходной вектор $\boldsymbol{y}$) зависит от размерности заданного вектора $\boldsymbol{d}$, сопряженного с входным вектором $\boldsymbol{x}$.

На практике гибридная сеть, как правило, более эффективна, чем одиночная сеть с нечеткой самоорганизацией и чем самостоятельный многослойный персептрон. Этот вывод следует из факта, что при использовании гибридной сети задача разделяется на два независимых этапа, реализуемых отдельно друг от друга. На этапе самоорганизации пространство входных данных разделяется на кластеры, при этом количество кластеров (самоорганизующихся нейронов) может быть произвольным и определяться условиями решаемой задачи. Многослойный персептрон приписывает каждой группе кластеров соответствующий ей ожидаемый результат. Например, при решении задачи классификации это может выглядеть как отнесение к одному конкретному классу нескольких кластеров данных.

С учетом ярко выраженной двухкомпонентной структуры гибридной сети для ее обучения применяется алгоритм, состоящий из двух этапов. На первом из них проводится обучение самоорганизующегося слоя, состоящее в подборе позиций центров, представляющих данные. Для этого можно применять как алгоритм *C-means*, так и алгоритм Густафсона–Кесселя. По завершении первого этапа, когда стабилизировались значения коэффициентов принадлежности всех векторов, представляющих входные сигналы для многослойного персептрона, начинается второй этап обучения. На нем значения параметров самоорганизующейся части сети остаются неизменными, а уточняются только веса нейронов персептронной компоненты. Это обычное обучение многослойного персептрона, для которого входом является множество коэффициентов принадлежности вектора $\boldsymbol{x}$ к центрам самоорганизующегося слоя. В зависимости от типа решаемой задачи выходом сети может быть код класса, к которому принадлежит входной вектор $\boldsymbol{x}$, либо ожидаемое значение $\boldsymbol{d}$ выходного вектора, соответствующего вектору $\boldsymbol{x}$. По завершении второго

этапа обучения веса замораживаются, и сеть становится готовой к функционированию в режиме эксплуатации (в котором на нее подаются только входные векторы x без соответствующих им векторов d).

Функционирование гибридной нечеткой сети проиллюстрируем на примере задачи классификации трехмерных данных, принадлежащих к трем частично пересекающимся классам. Распределение этих данных представлено на рис. 12.7.

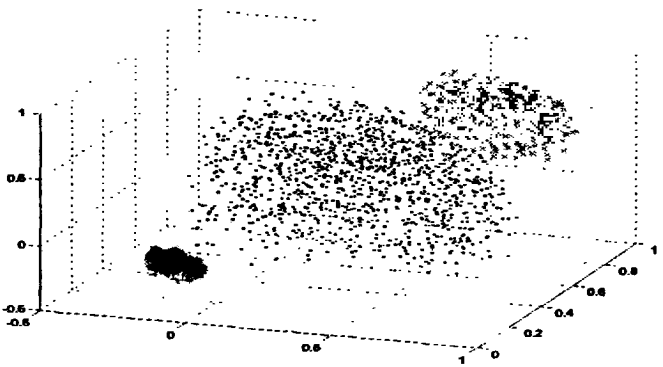

Рис. 12.7. Распределение тестовых трехмерных данных для решения задачи классификации

Количество данных, относящихся к разным классам, различно. В первом классе содержится 600 векторов x, обозначенных точками. Второй класс составляют 100 векторов, обозначенных звездочками, а в третий класс входят 300 векторов, обозначенных знаком "+". В ходе вычислительного эксперимента классы кодировались в двоичной системе (1 означала принадлежность к классу, а 0 – отсутствие принадлежности). По количеству классов размерность выходного вектора равна 3. Решение задачи при использовании обычного классификатора Кохонена либо сети с нечеткой самоорганизацией малоэффективно, поскольку количество выходных нейронов должно быть равно числу классов, т.е. трем, что существенно обедняет архитектуру сети, а вследствие частичного пересечения классов будет просто нецелесообразным.

Применение гибридной нечеткой сети со структурой 10-10-8-3 (10 самоорганизующихся нейронов и многослойный персептрон со структурой 10-8-3) дало хорошие результаты классификации. На 1000 тестовых данных, не участвовавших в обучении, получено только 6 ошибочных решений об отнесении к конкретным классам (эффективность классификации составила 99,4%). Для сравнения самостоятельный многослойный персептрон при классификации допустил 11 ошибок (эффективность 98,9%), а одиночная сеть с нечеткой самоорганизацией – 51 ошибку (эффективность 94,9%).

12.4.6. Примеры реализации нечетких сетей

Нечеткие нейронные сети как на основе самоорганизации, так и обучаемые с учителем, находят применение в тех же практических областях, что и классические сети соответствующих типов. По сравнению с традиционными решениями они демонстрируют качества, связанные с их способностью гладкой аппроксимации пороговых функций (см. графики на рис. 11.1). В настоящем подразделе будут приведены результаты аппроксимации нелинейной кривой с помощью сети TSK (сеть обучалась с учителем), представления большого количества многомерных данных ограниченным числом нечетких нейронов (сеть с самоорганизацией), а также примеры использования сети TSK для решения задачи распознавания газовых смесей.

В качестве первого примера рассмотрим аппроксимацию нелинейной функции от трех переменных ($x = [x_1, x_2, x_3]$), описываемой зависимостью

$$y = (1 + x_1^{0,5} + x_2^{-1} + x_3^{1,5})^2$$

при диапазоне изменения входных переменных x_i (i = 1, 2, 3) от 1 до 6. Для восстановления этой функции применялась нечеткая сеть TSK с функцией принадлежности

$$\mu_A(x) = \frac{1}{1 + \left(\frac{x - c}{\sigma}\right)^{2b}}.$$

В сети использовались восемь нечетких правил, а начальные значения всех параметров выбирались случайным образом. Общее количество параметров сети было равно 50, из которых 18 – это параметры нелинейной части условий, а остальные 32 – линейные веса p_{ij}. Сеть обучалась по гибридному алгоритму, основаниому на декомпозиции SVD. После обработки 200 обучающих выборок, равномерно распределенных в пространстве параметров, погрешность MSE уменьшилась с 57,19 в начале обучения до 1,61. Это соответствует примерно 0,7% относительной погрешности, приходящейся на одну выборку. Наиболее интенсивно погрешность уменьшалась в начальной фазе обучения. На рис. 12.8 иллюстрируется процесс последовательного итерационного уточнения нелинейных параметров c_i, σ_i и b_i функции принадлежности $\mu(x)$. По графикам видно, что изменения происходили в широком диапазоне значений.

Способность нечеткой сети к самоорганизации можно продемонстрировать на примере двумерных данных с распределением, представленным в разделе 9 на рис. 9.2. Нечеткие сети состояли из такого же количества нейронов, что и сети Кохонена, т.е. 40 и 200. На рис. 12.9 и 12.10 в графическом виде показаны результаты отображения обучающих данных векторами весов нечетких нейронов сетями из 40 (рис. 12.9) и 200 (рис. 12.10) нейронов соответственно. Ситуации, проиллюстрированные на рис. 12.9 и

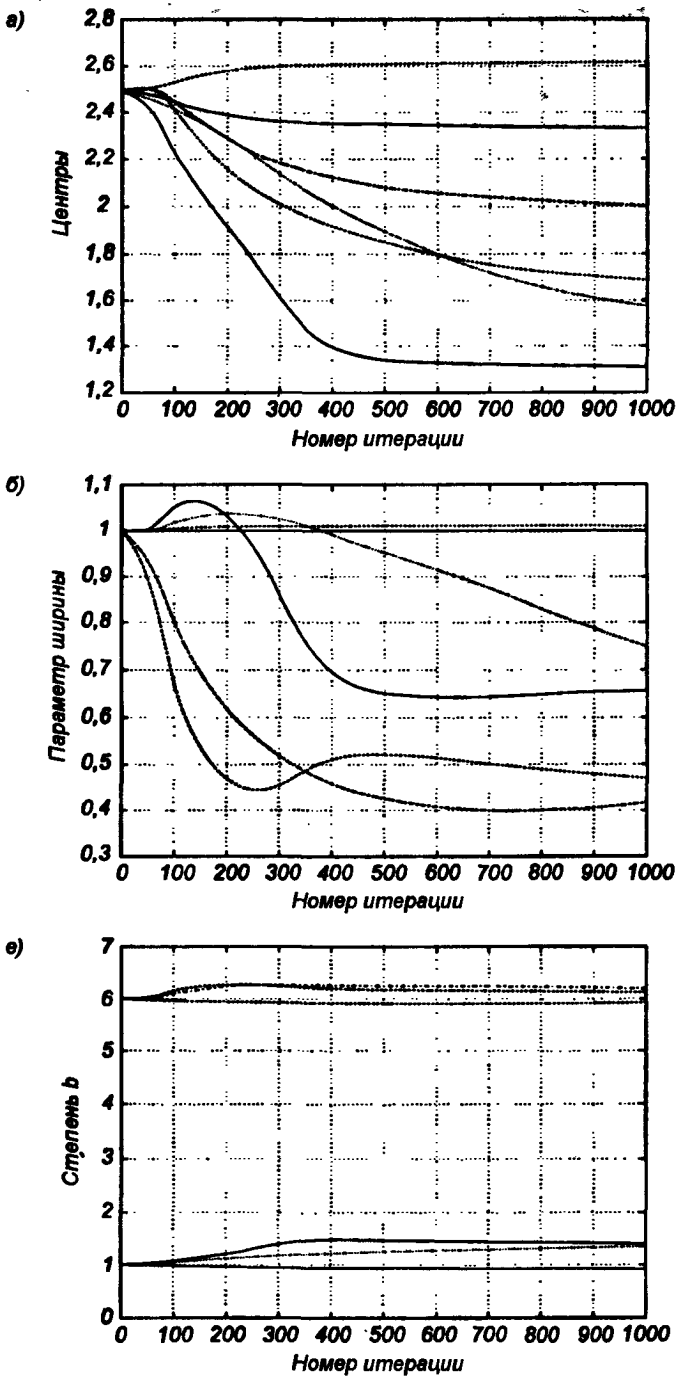

Рис. 12.8. Процесс уточнения параметров нечеткой сети:
а) центры c_i функции принадлежности; *б*) параметры σ_i; *в*) показатели степени b_i

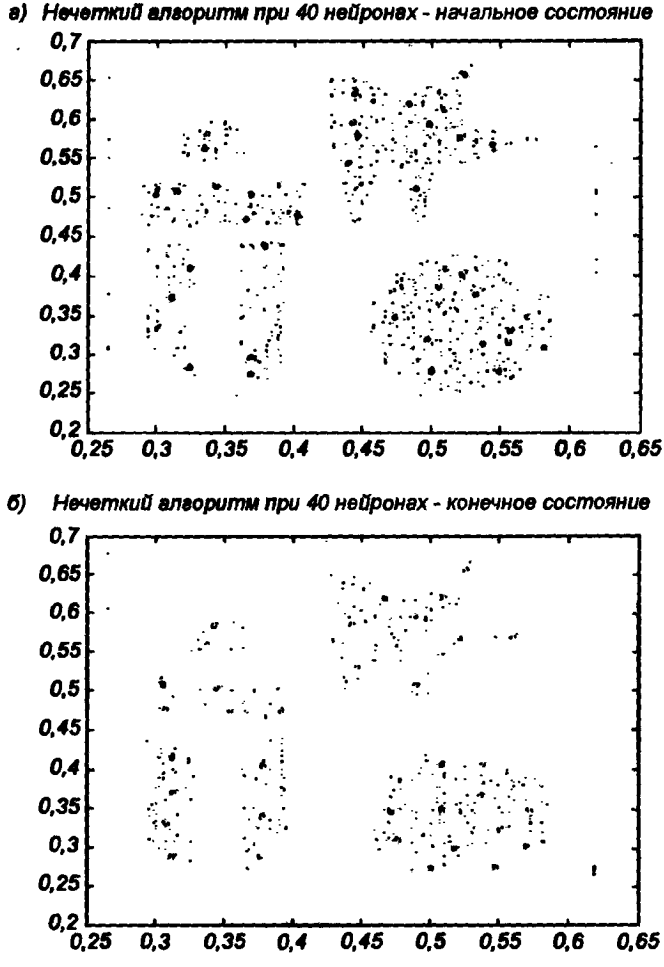

Рис. 12.9. Результаты восстановления обучающих данных с рис. 9.2 нечеткой нейронной сетью с самоорганизацией, состоящей из 40 нейронов:
а) исходное состояние после разностного группирования; *б*) окончательное размещение упорядоченных нейронов

12.10, относятся к начальному состоянию процесса обучения при использовании алгоритма разностного группирования (рис. 12.10*а*) и результирующего состояния (рис. 12.10*б*). Анализ исходного расположения нейронов показывает, что алгоритм разностного группирования обеспечивает весьма точное позиционирование центров нечетких функций. Алгоритм самоорганизации очень быстро приводит к их оптимальному размещению. Результаты, достигнутые с применением нечеткой сети, оказываются конкурентоспособными по отношению к наилучшим результатам, полученным на сети Кохонена. При использовании 40 нейронов погрешность квантования в нечеткой сети составила 0,00035 (наилучший результат в сети Кохонена был равен 0,0174), а при 200 нейронах

погрешность квантования не превысила 0,000172 (в сети Кохонена – 0,00705). Различие в уровне погрешности достаточно велико; оно обусловлено разными способами представления данных в классической системе и в нечеткой сети. В системе Кохонена каждую точку данных в многомерном пространстве

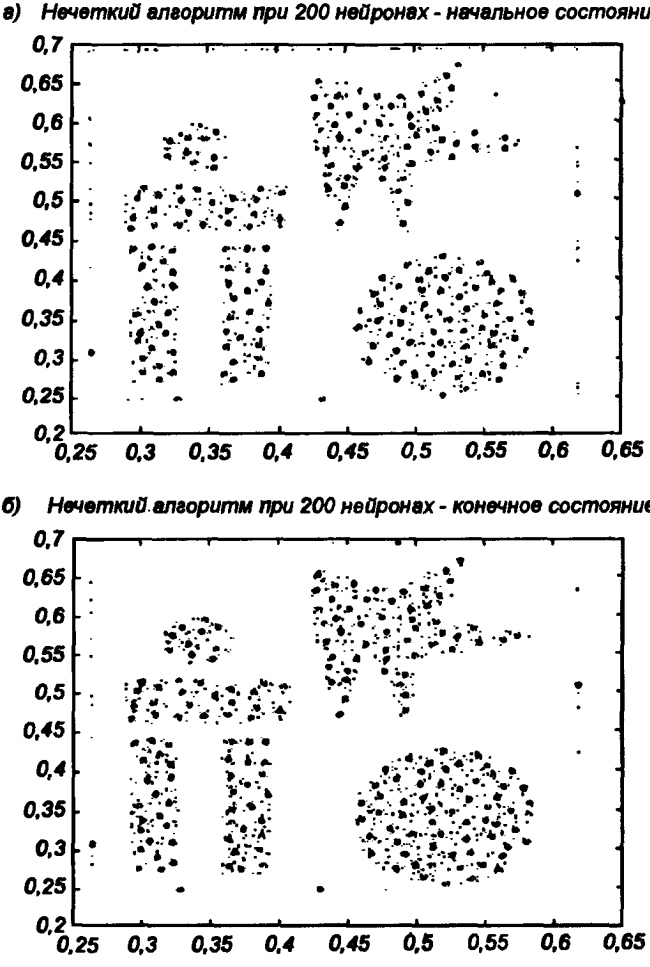

Рис. 12.10. Результаты восстановления обучающих данных с рис. 9.2 нечеткой нейронной сетью с самоорганизацией, состоящей из 200 нейронов:
а) исходное состояние после разностного группирования; *б*) окончательное размещение упорядоченных нейронов

представляют веса только одного нейрона, тогда как в нечеткой сети с самоорганизацией – множество нейронов, расположенных поблизости друг от друга. Благодаря этому становится возможным лучшее и более точное отображение использованных при обучении исходных данных.

Эта разница наиболее отчетливо заметна при представлении данных, соответствующих простой одномерной функции, например, синуса.

На рис. 12.11 демонстрируется аппроксимация этой функции с использованием 20 нейронов. Рис. 12.11*а* отображает работу нечеткой сети, а рис. 12.11*б* – сети Кохоиена. Классическая сеть Кохонена аппроксимирует функцию отрезками прямой, а нечеткая сеть – непрерывной кривой линией. Уровни погрешности восстановления различаются при этом в среднем в 2,5 раза.

Очень хорошие результаты получены при использовании нечеткой сети TSK для распознавания компонентов газовых смесей. Исследования проводились на тех же обучающих и тестовых выборках, которые применялись в экспериментах с гибридной сетью, описанных в разделе 9. Применялась

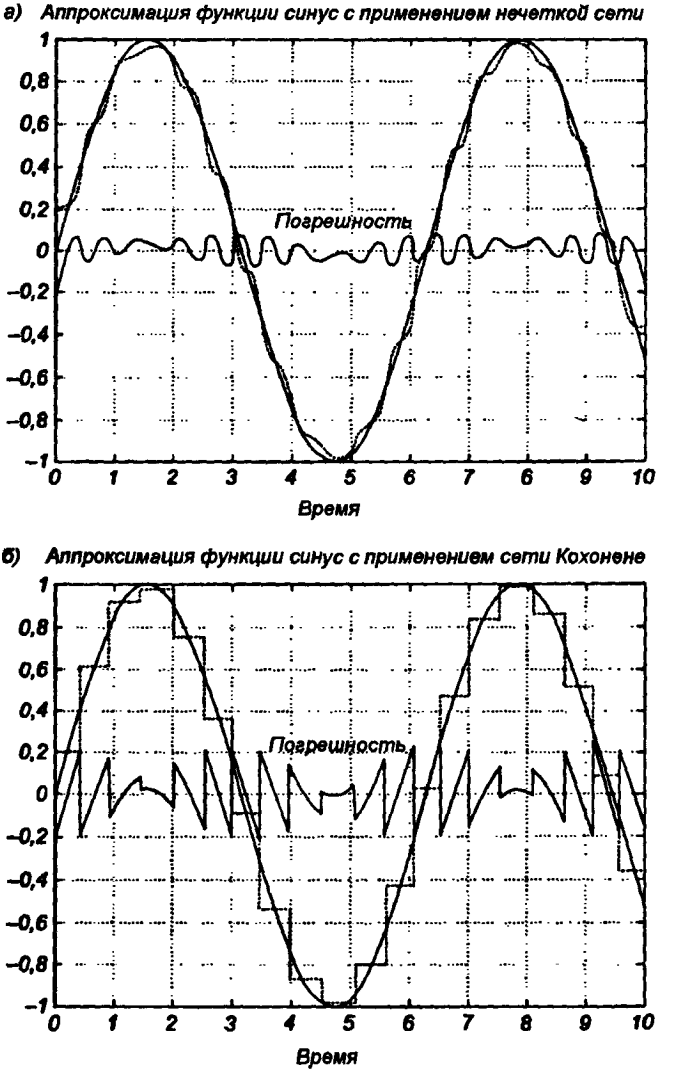

Рис. 12.11. Иллюстрация способа аппроксимации синусоидальной функции *а*) сетью с нечеткой самоорганизацией; *б*) сетью Кохонена

пятивходовая сеть TSK (для учета показаний пяти полупроводниковых датчиков) с пятью нечеткими правилами. Количество выходных нейронов было равно четырем (смесь состояла из четырех газовых компонентов). Обучение проводилось по гибридному алгоритму подбора параметров сети, объединенному с предварительной инициализацией центров нечетких функций, основанной на механизме самоорганизации по принципу разностного группирования. Этот принцип позволил получить практически идеальное начальное размещение центров, благодаря чему адаптированный к сети TSK гибридный алгоритм уже после первого прохода (выполнение декомпозиции SVD) перевел сеть в обученное состояние. На рис. 12.12 представлены результаты тестирования натренированной сети TSK на тех же данных, которые применялись в экспериментах с гибридной сетью. Эти результаты представляют собой графики относительной погрешности определения четырех газовых компонентов (1 – двуокись углерода, 2 – метан, 3 – метанол, 4 – пропан/бутан). Среднее значение относительной погрешности по результатам экспериментов составило 0,23%, что свидетельствует о существенном прогрессе по отношению к показателям, достигнутым с помощью гибридной сети.

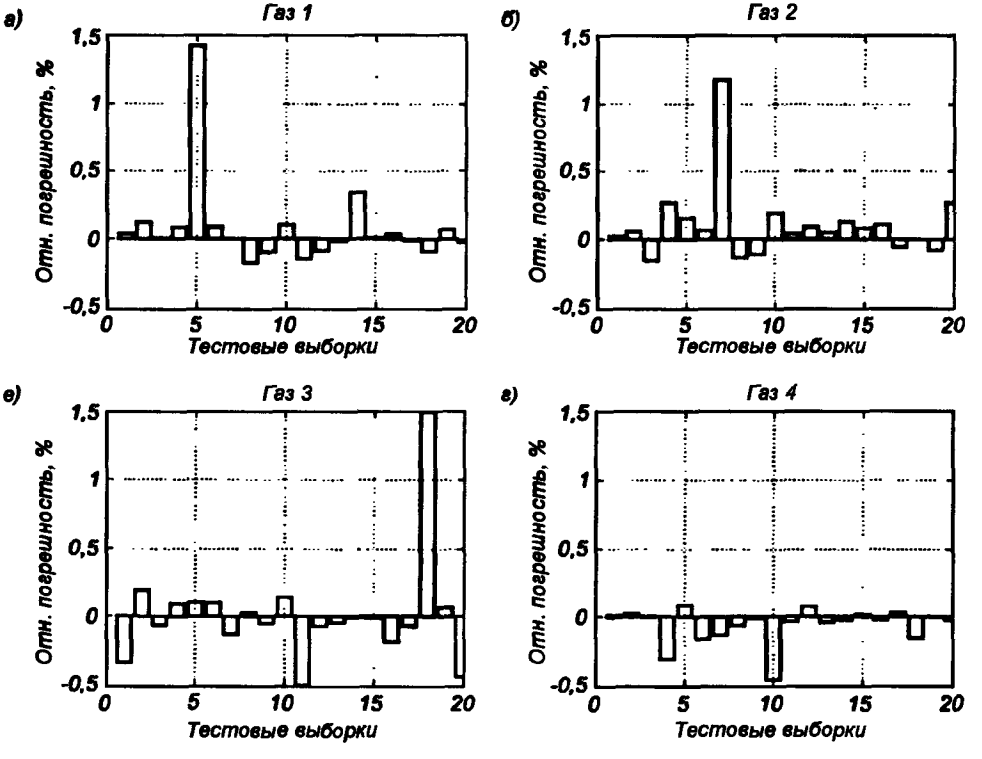

Рис. 12.12. Распределение относительных погрешностей определения четырех компонентов газовой смеси, полученных с использованием нечеткой сети TSK.

12.5. Адаптивный алгоритм самоорганизации нечеткой сети

Представленный в предыдущем подразделе алгоритм самоорганизации требует априорного знания количества центров (нечетких нейронов), которые будут предоставлять данные. С этой точки зрения наиболее универсальным представляется адаптивный алгоритм, автоматически добавляющий новые центры в режиме онлайн в зависимости от распределения входных данных x.

Адаптивный алгоритм был сформулирован только для гауссовской функции ($b = 1$) с использованием обобщенной модели Ванга–Менделя [160]. В результате его реализации определяются: количество центров и их расположение в части, соответствующей условиям (множество векторов x_i) и заключениям (множество скалярных ожидаемых значений d_i). Этот алгоритм можно описать следующим образом.

1. При старте с первой пары данных (x_1, d_1) создается первый кластер с центром $c_1 = x_1$. Принимается, что $w_1 = d_1$ и что мощность множества $L_1 = 1$. Пусть r обозначает предельное эвклидово расстояние между вектором x и центром, при котором данные будут трактоваться как принадлежащие к созданному кластеру. Для сохранения общности решения принимается, что в момент начала обучения существует M кластеров с центрами c_1, c_2, ..., c_M и соответствующие им значения w_i и L_i ($i = 1, 2, ..., M$).
2. После считывания k-й обучающей пары (x_k, d_k) рассчитываются расстояния между вектором x_k и всеми существующими центрами $\|x_k - c_l\|$ для $l = 1, 2, ..., M$. Допустим, что ближайший центр – это c_{l_k}. В таком случае в зависимости от значения $\|x_k - c_{l_k}\|$ может возникнуть одна из двух ситуаций:
   - если $\|x_k - c_{l_k}\| > r$, то создается новый кластер $c_{M+1} = x_k$, причем $w_{M+1}(k) = d_k$, $L_{M+1}(k) = 1$. Параметры созданных до этого кластеров не изменяются, т.е. $w_l(k) = w_l(k-1)$, $L_l(k) = L_l(k-1)$ для $l = 1, 2, ..., M$. Количество кластеров M увеличивается на 1 ($M \leftarrow M + 1$);
   - если $\|x_k - c_{l_k}\| \le r$, то данные включаются в l_k-й кластер, параметры которого следует уточнить в соответствии с формулами [60]:

$$w_{l_k}(k) = w_{l_k}(k-1) + d_k \ , \tag{12.48}$$

$$L_{l_k}(k) = L_{l_k}(k-1) + 1 \ , \tag{12.49}$$

$$c_{l_k}(k) = \frac{c_{l_k}(k-1)L_{l_k}(k-1) + x_k}{L_{l_k}(k)} \ . \tag{12.50}$$

В другой версии алгоритма фиксируется положение центров c_{l_k} после

инициализации, и их координаты уже не изменяются. Во многих случаях такой прием улучшает результаты адаптации.

3. После уточнения параметров нечеткой системы функция, аппроксимирующая входные данные системы, определяется в виде

$$\hat{f}(x)=\frac{\sum_{l=1}^{M} w_l(k)\exp\left(-\frac{\|x-c_l(k)\|^2}{\sigma^2}\right)}{\sum_{l=1}^{M} L_l(k)\exp\left(-\frac{\|x-c_l(k)\|^2}{\sigma^2}\right)}, \qquad (12.51)$$

тогда как остальные кластеры не изменяются, т.е. при $l\neq l_k$ $w_l(k)=w_l(k-1)$, $L_l(k)=L_l(k-1)$ и $c_l(k)=c_l(k-1)$, для $l=1, 2, ..., M$.

При повторении перечисленных этапов алгоритма до $k=p$ с уточнением каждый раз значения M пространство данных разделяется на M кластеров, при этом мощность каждого из них определяется как $L_l = L_l(k)$, центр – как $c_l = c_l(k)$, а значение приписанной ему накоплениой функции d – как $w_l = w_l(k)$.

Этот алгоритм называется самоорганизующимся, поскольку разделение пространства данных на кластеры происходит самостоятельно и без участия человека, в соответствии с заданным значением порога r. При малом значении r количество кластеров возрастает, в результате чего аппроксимация данных становится более точной, однако это достигается за счет более сложной функции и увеличения объема необходимых вычислений при одновременном ухудшении обобщающих свойств сети. Если значение r слишком велико, то вычислительная сложность уменьшается, однако возрастает погрешность аппроксимации. При подборе оптимальной величины порога r должен соблюдаться компромисс между точностью отображения и вычислительной сложностью. Как правило, оптимальное значение r подбирается методом проб и ошибок с использованием вычислительных экспериментов.

На рис. 12.13 представлены результаты аппроксимации кривой

$$f(x)=0{,}1\sin(0{,}2\pi x)+0{,}2\sin(0{,}3\pi x)+0{,}6\sin(0{,}9\pi x)+$$
$$+1{,}1\sin(1{,}9\pi x)+2{,}3\sin(3{,}7\pi x)$$

нечеткой сетью с самоорганизацией при использовании адаптивного алгоритма обучения. Рис. 12.13*а* иллюстрирует результаты, полученные при величине порога r = 0,2, а рис. 12.13*б* соответствует порогу r = 0,05. Пунктирная линия обозначает ожидаемые значения, а непрерывная линия – фактические значения, сгенерированные нейронной моделью. Алгоритм сам подбирал количество нейронов, отвечающее установленному значению порога r. В первом случае зафиксированное алгоритмом количество нейронов было равно 12, а во втором – 19. Для обучения использовались только 500 первых реализаций сигнала. Оставшиеся 500 значений применялись исключительно для тестирования.

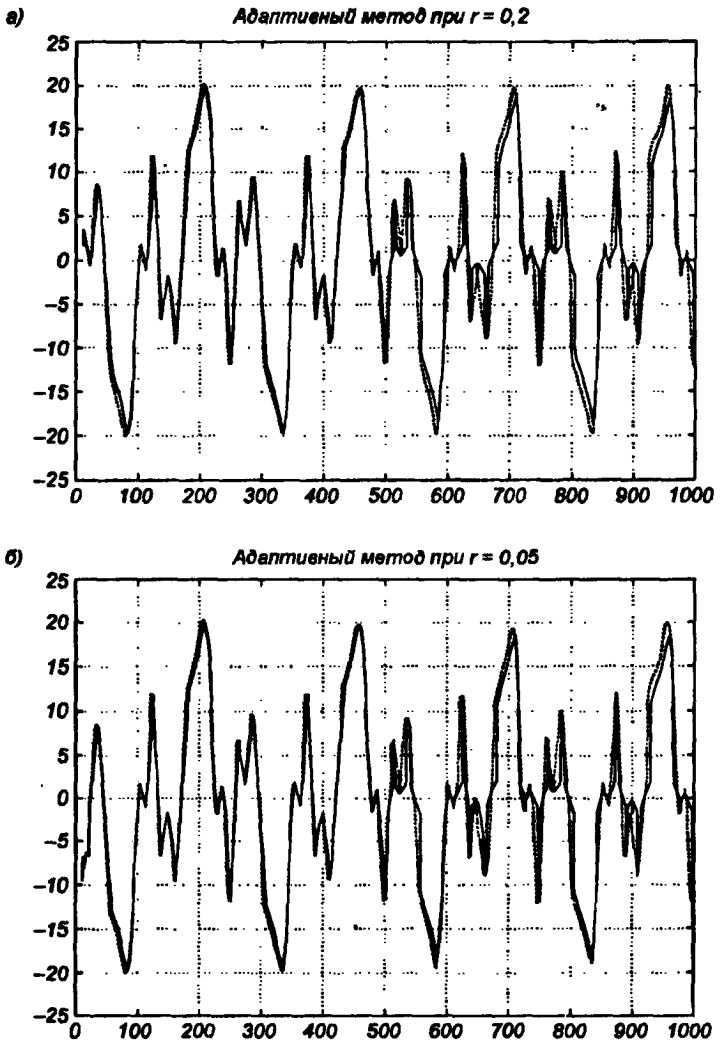

Рис. 12.13. Пример отображения данных нечеткой сетью с самоорганизацией при использовании адаптивного алгоритма обучения: *а*) порог $r = 0{,}2$; *б*) порог $r = 0{,}05$

Следует обратить внимание, что алгоритм самоорганизации нечеткой сети позволяет одновременно определять как параметры сети, так и ее структуру (количество нейронов скрытого слоя). Его реализация подобна модели Ванга–Менделя, описываемой формулой (12.7), в которой можно выделить центры c_l, соответствующие множеству векторов x, и коэффициенты w_l, связанные с положением центров через последовательность заданных функций $\{d\}$. В связи с накопительным характером формирования параметров w_l (формула (12.48)), в знаменателе выражения (12.51) суммирование производится с весами L_l, отражающими количество уточнений параметров конкретных групп данных (размерность кластера).

Литература

1. *Barron A. R.* Approximation and estimation bounds for artificial neural networks. Machine learning. - Vol. 14, 1994. - Pp. 115–133.
2. *Brudzewski K., Osowski S.* Gas analysis system composed of a solid state sensor array and hybrid neural network structure. Sensors and Actuators-B55, 1999. - Pp. 38–46.
3. *Cardosso J. F., Belouchrani A., Laheld B.* A new composite criterion for adaptive and iterative blind source separation // Proc. ICASSP-94. - Adalaide. - Vol. IV. - Pp. 273–276.
4. *Chui C. K.* An introduction to wavelets. N. Y.: Academic Press, 1992.
5. *Cottrell M., Girard B., Girard Y., Muller C., Rousset P.* Daily electrical power curve: classification and forecasting using a Kohonen map // From Natural to Artificial Neural Computation / J. Mira, F. Sandoval, eds. – Malaga: IWANN, 1995. - Pp. 1107–1113.
6. *Cottrell G., Munro P., Zipser D.* Image compression by back propagation: an example of extensional programming. - Technical Report ICS report 8702, ICS–UCSD San Diego, California, USA, February 1987.
7. *Chauvin Y.* A back propagation algorithm with optimal use of hidden units // Advances in NIPS2 / D. Touretzky, Ed. – San Mateo: Morgan Kaufmann, 1989. - Pp. 519–526.
8. *Chen S., Cowan C.F., Grant P.M.* Orthogonal least squares learning algorithm for radial basis function networks // IEEE Trans. Neural Networks, 1991. - Vol. 2. - Pp. 302–309.
9. *Chen D., Jain C.* A robust back propagation learning algorithm for function approximation // IEEE Trans. Neural Networks, 1994. – Vol. 5. – Pp. 467–479.
10. *Cheng Y.H., Lin C.S.* Learning algorithm for radial basis function network with the capability of adding and pruning neurons: Proc. 1994 Conf. ICNN. Orlando: 1994. - Pp. 797–801.
11. *Chinrungrueng C., Sequin C.H.* Optimal adaptive K-means algorithm with dynamic adjustment of learning rate // IEEE Trans. Neural Networks, 1995. - Vol. 6. - Pp. 157–169.
12. *Cichocki A., Bogner R., Moszczynski L.,Pope K.* Modified Herault-Jutten algorithms for blind separation of sources // Digital signal Processing, 1997. - Vol. 7. - Pp. 80–93.
13. *Cichocki A., Moszczynski L.* BS-program of blind separation of sources. – Warszawa: Politechnika Warszawska, 1994.
14. *Cichocki A., Unbehauen R.* SC neural networks for differential optimization // Int. J. C. T. Appl., 1991. - Vol. 19. - Pp. 161–187.

15. *Cichocki A., Unbehauen R.* Neural networks for solving systems of linear equations and related problems // IEEE Trans. CAS, 1992. - Vol. 39. - Pp. 124–138.

16. *Cichocki A., Unbehauen R.* Neural networks for optimization and signal processing. N. Y.: Wiley, 1993.

17. *Cichocki A., Unbehauen R., Moszczynski L., Rummert E. A* new on-line adaptive learning algorithm for blind separation of source signals. ISANN. – Taiwan, 1994. - Pp. 421–427.

18. *Cichocki A., Amari S.* Adaptive blind signal processing - neural network approaches // Proceedings of IEEE, 1998. - Vol. 86. - Pp. 2026–2048.

19. *Cohen M.A., Grossberg S.* Absolute stability of global pattern formation and parallel memory storage by competitive neural networks // IEEE Trans. SMC, 1983. - Vol. 13. - Pp. 815–826.

20. *Cover T.* Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition // IEEE Trans. Electronic Computers, 1965. - Vol. 14. - Pp. 326–334.

21. *Czogala E., Pedrycz W.* Elementy i metody teorii zbioryw rozmytych. – Warszawa: PWN, 1985.

22. *Darken C., Moody J.* Learning schedules for stochastic optimization // Proc. 1990 IEEE Conf. NIPS // Natural and Syntetic, 1990. - Pp. 518–523.

23. *Daubechies I.* Ten lectures on wavelets. CBMS- NSF Regional Conf. Series in Applied Mathematics, Montpelier: Capital City Press, 1992. - Vol. 61.

24. *Demartines P.* Analyse de donnees par reseaux de neurones auto-organises, Diss. de l'Institut National Polytechnique de Grenoble. - Grenoble: 1994.

25. *Demartines P., Blayo F.* Kohonen self organizing maps: is the normalization necessary? // Complex Systems, 1992. - Vol. 6. - Pp. 105–123.

26. *Demartines P., Herault J.* Representation of nonlinear data structures through fast VQP neural network // Proc. Neuronimes. - Nimes, France, 1993. - Pp. 1–18.

27. *Demuth H., Beale M.* Neural Network Toolbox for use with Matlab. - Natick: The MathWorks, Inc., 1992.

28. *Denoeux J., Lengalle R.* Initializing back propagation networks with prototypes // Neural Networks, 1993. - Vol 6. - Pp. 351–363.

29. *Diamantaras K., Kung S.* Principal component neural networks, theory and applications. - N. Y.: Wiley, 1996.

30. *Dinh Nghia Do, Osowski S.* Shape recognition using FFT preprocessing and neural network // Compel, 1998. - Vol. 17, No 5/6. - Pp. 658–666.

31. *Dinh Nghia Do.* Sieci neuronowe w zastosowaniu do rozpoznawania wzorcow dwuwymiarowych,: Rozprawa doktorska. – Warszawa: Politechnika Warszawska, 1999.

32. *Dubois D., Prade H.* Fuzzy sets and systems. – N. Y.: Academic Press, 1980.

33. *Fahlman S.E.* Faster learning variatious on backpropagation: en empirical study // Proc. 1988 Connectionist Models Summer School. – Los Altos, USA: Morgan Kaufmann, 1988. - Pp. 38–51.

34. *Fahlman S.E., Lebiere C.* The cascade-correlation learning architecture // Advances in NIPS2 / Ed. D. Touretzky. - Morgan Kaufmann, 1990. - Pp. 524–532.

35. *Floreen P.* The convergence of Hamming memory networks // IEEE Trans. – Neural Networks, 1991. - Vol. 2. - Pp. 449–457.

36. *Fukushima K., Wake N.* Handwritten alphanumeric character recognition by the neocognitron // IEEE Trans. N.N., 1991. - Vol. 2, - Pp. 355–365.

37. *Furth B.* A survey of multimedia technique and standards. - Part I: JPEG standard. Real Time Imaging, 1995. - Pp. 49–67.

38. *Geva S., Sitte J.* Progress in supervised neural networks // IEEE Trans. N. N., 1992. - Vol. 3. - Pp. 621–625.

39. *Gill P., Murray W.*, Wright M. Practical Optimization. – N.Y.: Academic Press, 1981.

40. *Girosi F., Jones M.. - Poggio T.* Regularization theory and neural network architecture // Neural Computation, 1995. - Vol. 7. - Pp. 219–270.

41. *Goldberg D.* Algorytmy genetyczne. – Warszawa: WNT, 1995.

42. *Golub G., Van Loan C.* Matrix computations. – N.Y.: Academic Press, 1991.

43. *Hampel F. R., Rousseeuw P. J., Ronchetti E. M., Stahel W.* Robust statistics - the approach based on influence function. – N.Y.: Wiley, 1986.

44. *Han I-S., Ahn K-W.* Implementation of million connections neural hardware with URAN-1 // Proc. ICANN-93. – Amsterdam: 1993. - Pp. 1030–1033.

45. *Hassibi B., Stork D.* Second order derivaties for network prunning: Optimal brain surgeon // Advances in NIPS5 / Ed. D. Touretzky, San Mateo: Morgan Kaufmann, 1993. - Pp. 164–171.

46. *Haykin S.* Neural networks, a comprehensive foundation. - N.Y.: Macmillan College Publishing Company, 1994.

47. *He Y., Ciringiroglu U.* A charge based on-chip adaptation Kohonen neural network // Trans. Neural Networks, 1993. - Vol. 4. - Pp. 462–469.

48. *He Y., Cilingiroglu Y., Sanchez-Sinencio E.* A high density and low power charge based Hamming network // IEEE Trans. VLSI Systems, 1993. - Vol. 1. - Pp. 56–62.

49. *Hebb D.* Organization of behaviour. – N.Y.: J. Wiley, 1949.

50. *Hecht-Nielsen R.* Neurocomputing. – Amsterdam: Addison Wesley, 1991.

51. *Hertz J., Krogh A., Palmer R.* Wstep do teorii obliczen neuronowych. Wyd. II. – Warszawa: WNT, 1995.

52. *Hinton G., Sejnowski T.* Learning and relearning in Boltzman machines // Parallel Distributed Processing: Exploration in Microstructure of Cognition / Eds. D. Rummelhart, J. McClelland. – Cambridge: Mass. Press, 1986.

53. *Hopfield J.* Neural networks and physical systems with emergent collective computational abilities // Proc. National Academy of Science USA, 1982. - Vol. 79. - Pp. 2554–2558.

54. *Hopfield J., Tank D.* Neural computations of decisions in optimization problems // Biological Cybernetics, 1985. - Vol. 52. - Pp. 141–152.

55. *Hopfield J., Tank D.* Computing with neural circuts: a model // Science, 1986. - Vol. 233. - Pp. 625–633.

56. *Hornik K., Stinchcombe M., White H.* Multilayer feedforward networks are universal approximators // Neural Networks, 1989. - Vol. 2. - Pp. 359–366.

57. *Hush D., Horne B.* Progress in supervised neural networks // IEEE Signal Processing Magazine, 1993, January. - Pp. 8–39.

58. Intel 80170NX Electrically trainable analog neural network. Product Description and Data Sheet. - Intel Corp., 1991.

59. *Jacobs R. A.* Increased rates of convergence through learning rate adaptation // Neural Networks, 1988. - Vol. 1. - Pp. 295–307.

60. *Jang J. S., Sun C. T., Mizutani E.* Neuro-fuzzy and soft computing. - N.Y.: Prentice Hall, 1997.

61. *Johnson D., Aragon C., Schevon C.* Optimization by simulated annealing: an experimental evaluation. Part I: graph partitioning // Operations Reasearch, 1989. - Vol. 37. - Pp. 865–892.

62. *Jutten C., Herault J.* Blind separation of sources part I: an adaptive algorithm based on neuromimetic architecture // Signal Processing, 1991. - Vol. 24. - Pp. 1-10.

63. *Jutten C., Nguyen T., Dijkstra E., Vittoz E., Caelen J.* Blind separation of sources: an algorithm for separation of convolutive mixtures. – Chamrouse: Workshop of Higher Order Statistics, 1991. - Pp. 273–276.

64. *Kamen E., Heck B.* Fundamentals of signals and systems using Matlab. – N.Y.: Prentice Hall, 1997.

65. *Karayiannis N.* Accelerating training of feedforward neural network using generalized hebbian rules for initializing the internal representation // IEEE Proc. ICNN, Orlando, 1994. - Pp. 1242–1247.

66. *Karnin E.D.* A simple procedure for pruning back-propagation trained neural networks // IEEE Trans. Neural Networks, 1990. - Vol. 1. - Pp. 239–242.

67. *Kasabov N.* Foundations of neural networks, fuzzy systems and knowledge engineering. - London: Bradford Book MIT Press, 1996.

68. *Khorasani K., Cuffaro A., Grigorin T.* A new learning algorithm for bidirectional associative memory neural networks // IEEE Proc. ICNN, Orlando, 1994. - Pp.1115–1120.

69. *Khotanzad A., Lu J.* Classification of invariant image reconstruction using neural networks // IEEE Trans. ASSP, 1990. - Vol. 38. - Pp. 1028–1038.

70. *Kirby M., Sirovich L.* Application of Karhunen-Loeve procedures for the characterization of human faces // IEEE Trans. Pattern Analysis, 1990. - Vol. 12. - Pp. 103–108.

71. *Kirkpatrick S., Gelatt C. D. - Vecchi M. P.* Optimization by simulated annealing // Science, 1983. - Vol. 220. - Pp. 671 - 680.

72. *Klimauskas G.* Neural Ware - User manual. Natick, USA: Neural Ware Inc., 1992.

73. *Kohonen T.* The self organising map // Proc. of IEEE, 1990. - Vol. 78. - Pp. 1464–1479.

74. *Kohonen T.* Self-organizing maps. – Berlin: Springer Verlag, 1995.

75. *Kohonen T., Kangas J., Laakson J.* SOMPAK, the selforganizing map program package. Technical Report. - Espoo, Finland, Helsinki: University of Technology, 1992.

76. *Kolmogorov A. N.* On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition // Dokl. Akad. Nauk USSR, 1957. - Vol. 114. - Pp. 953–956.

77. *Korbicz J., Obuchowicz A., Ucinski D.* Sztuczne sieci neuronowe - podstawy i zastosowania. – Warszawa: Akademicka Oficyna Wydawnicza PLJ, 1994.

78. *Kosko B.* Bidirectional associative memories // IEEE Trans. Systems, Man and Cybernetics, 1988. - Vol. 18. - Pp. 49–60.

79. *Krzyzak A.* Metody nieparametryczne w przetwarzaniu sygnalow i w sieciach neuronowych. Prace Naukowe 1998, z. 106, Warszawa: Oficyna Wydawnicza PW, 1998.

80. *Krzyzak A., Xu L., Suen C.Y.* Methods of combining multiple classifiers and their applications to handwritten recognition // IEEE Trans. SMC. - Vol. 22. - Pp. 418–435.

81. *Kuhl I., Giardina C.* Elliptic Fourier features of a closed contours // Computer Graphics and Image Processing, 1982. - Vol. 18. - Pp. 236–258.

82. *Kung S.Y.* Digital neural networks. - New Jersey: Prentice Hall, Englewood Cliffs, 1993.

83. *Lansner J., Lehmann T.* An analog CMOS chip set neural networks with arbitrary topologies // Trans. Neural Networks, 1993. - Vol. 4. - Pp. 441–444.

84. *LeCun Y., Denker J., Solla S.* Optimal brain damage //Advances in NIPS2 / Ed. D. Touretzky, San Mateo: Morgan Kaufmann, 1990. - Pp. 598–605.

85. *Leonard J.A., Kramer M.A.* Radial basis function networks for classifying process faults // IEEE Control System Magazine, 1991, April. - Pp. 31–38.

86. *Li Q., Tufts D.* Synthesizing neural networks by sequencial addition of hidden modes // IEEE Proc. ICNN, Orlando, 1994. - Pp. 708–713.

87. *Liano K.* A robust approach to supervised learning in neural network // IEEE Proc. ICNN, Orlando, 1994. - Pp. 513–516.

88. *Lin Y., Cunningham G.* A new approach to fuzzy neural system modeling // IEEE Trans. Fuzzy Systems, 1995. - Vol. 3. - Pp. 190–198.

89. *Linde Y., Buzo A., Gray R.* An algorithm for vector quantizer design // IEEE Trans. Comm., 1980. - Vol. 28. - Pp. 84–95.

90. *Linsker R.* From basic network principles to neural architecture // Proc. of National Academy of Sciences of the USA, 1986. - Vol. 83. - Pp. 7508–7512, 8390–8394, 8779–8783.

91. *Lippmann R.* An introduction to computing with neural nets // IEEE ASSP Magazine, 1987, April. - Pp. 4–22.

92. *Majkowski A.* Zastosowanie transformacji falkowej i obliczen neuronowych w kompresji sygnalow, Rozprawa doktorska. – Warszawa: PolitechnikaWarszawska, 1999.

93. *Mallat S.* A theory for multiresolution signal decomposition: the wavelet representation // IEEE Trans. PAMI, 1989. - Vol. 11. - Pp. 674–693.

94. *Martinetz M., Berkovich S., Schulten K.* "Neural-gas" network for vector quantization and its application to time series prediction // Trans. Neural Networks, 1993.- Vol. 4. - Pp. 558–569.

95. *Marquardt D.* An algorithm for least squares estimation of nonlinear parameters, SIAM, 1963. - Pp. 431–442.

96. *Masters T.* Practical neural network recipes in C++. – Boston: Academic Press, 1993.

97. *Matsuoka K.* Noise injection into inputs in back-propagation learning // IEEE Trans. on Systems, Man, and Cybernetics, 1992. - Vol.22. - Pp. 436–440.

98. *McCulloch W. S., Pitts W. H.* A logical calculus of ideas immanent in nervous activity // Bull. Math. Biophysics, 1943. - Vol. 5. - Pp. 115–119.

99. *Mehrotra K., Mohan C., Ranka S.* Bounds on the number of samples needed for neural learning // IEEE Trans. Neural Networks, 1991. - Vol. 2. - Pp. 548–558.

100. *Michel A., Farrel J.* Associative memories via artificial neural networks // IEEE Control System Magazine, 14990, April. - Pp. 6–16.

101. *Minsky M., Papert S.* Perceptrons: an introduction to computational geometry. - Cambridge, MA, 1988.

102. *Moller M. F.* A scaled conjugate gradient algorithm for fast supervised learning // Neural Networks, 1993. - Vol. 6. - Pp. 525–533.

103. *Moody T. J., Darken C.J.* Fast learning in networks of locally tuned processing units // Neural Computation, 1989. - Vol. 1. - Pp. 151–160.

104. *Moszczynski L.* Uklady adaptacyjne w zastosowaniu do separacji sygnaiyw elektrycznych: Praca doktorska. – Warszawa: Politechnika Warszawska, 1995.

105. *Mougeot M., Azencott R.*, Angeniol B. Image compression with backpropagation: improvement of the visual restoration using different cost functions // Neural Networks, 1991. - Vol. 4. - Pp. 467–476.

106. *Mozer M., Smolensky P.* Skeletonization - a technique for trimming the fat from a network via relevance assessment // Advances in NIPS1 / Ed. D. Touretzky. – San Mateo: Morgan Kaufmann, 1989. - Pp. 107–115.

107. *Narendra K. S., Parthasarathy K.* Identification and control of dynamical systems using neural networks // IEEE Trans. Neural Networks, 1990. - Vol. 1. - Pp. 4–27.

108. NI1000 recognition accelerator and data sheet (2pp), RI09906. Providence: Nestor Inc., 1994.

109. NI1000 development system, data sheet (2pp), RI09906. - Providence: Nestor Inc., 1994.

110. *Oja E.* Simplified neuron model as a principal component analyzer // J. Math. Biology, 1982. - Vol. 15. - Pp. 267–273.

111. *Oja E.* Principal components, minor components and linear neural networks // Neural Networks, 1992. - Vol 5. - Pp. 927–935.

112. *Oja E., Ogawa H., Wangviwattana J.* Learning in nonlinear constrained hebbian networks // Proc. ICANN-91, Espoo, Finland, 1991. - Pp. 385–390.

113. *Osowski S.* Sieci neuronowe. – Warszawa: Oficyna Wydawnicza PW, 1994.

114. *Osowski S.* Sieci neuronowe w ujeciu algorytmicznym. – Warszawa: WNT, 1996.

115. *Osowski S.* Signal flow graphs and neural networks // Biological Cybernetics, 1994. Vol. 70. - Pp. 387–395.

116. *Osowski S.* Neural network approach to the solution of linear complementarity problems // Int. Journal of Numerical Modelling, Electronic Networks, Devices and Fields, 1995. - Vol. 8. - Pp. 431 - 445

117. *Osowski S.* Multilayer Volterra filter and its applications // Neural Computing and Applications, 1996. - Vol. 4. - Pp. 228–236.

118. *Osowski S., Brudzewski K.* The hybrid neural network for gas analysis measuring system // IMTC99, Venice, 1999. - Pp. 440–445

119. *Osowski S., Cichocki A.* Learning in dynamic neural networks using signal flow graphs // Int. Journal of Circuit Theory and Applications,1999. - Vol. 27. - Pp. 209–228.

120. *Osowski S., Herault J.* Signal flow graphs as an efficient tool for gradient and hessian determination // Complex Systems, 1995. - Vol. 9 . - Pp. 29–45.

121. *Osowski S., Herault J., Demartines P.* Fault location in analogue circuits using Kohonen neural network // Bulletin Academie Polonaise, 1995. - Vol. 43, No 1. - Pp. 111–123.

122. *Osowski S., Majkowski A., Cichocki A.* Robust PCA neural networks for random noise reduction of the data // 1977 Int. Conf. on Acoustic, Speech and Signal Processing. - Munich, 1997. - Pp. 3397–3400.

123. *Osowski S., Siwek K.* Study of generalization ability of neural networks // III Konferencja "Sieci Neuronowe i ich zastosowania". Kule, 1997.

124. *Osowski S., Siwek K.* Selforganizing neural networks for short term load forecasting in power system // Engineering Applications of Neural Networks (EANN), Gibraltar, 1998. - Pp. 253–256.

125. *Osowski S., Siwek K.* Selforganizing neural network for fault location in electrical circuits // ICECS, Lissabon, 1998. - Pp. 265–268.

126. *Osowski S., Stodolski M., Bojarczak P.* Fast second order learning algorithm for feedforward multilayer neural networks and its applications // Neural Networks, 1996. - Vol. 9. - Pp. 1583–1596

127. *Osowski S., Tran Hoai L.* Rekurencyjna siec neuronowa Elmana w zastosowaniu do problemow predykcji // XX SPETO, Ustron, 1997. Pp. 505–510.

128. *Osowski S., Tran Hoai L.* The neurofuzzy network for approximation // KKTOiUE, Poznan, 1998. Pp. 607–612.

129. *Pao Y. H.* Adaptive pattern recognition and neural networks – Reading: Adison Wesley, 1989.

130. *Platt J.* A resource-allocating network for function interpolation // Neural Computation, 1991. - Vol. 3. - Pp 213–225.

131. *Powell M. J.* The Theory of radial basis functions approximation //Advances in Numerical Analysis. Vol. II, Wavelts, Subdivision algorithms and radial basis functions // Ed. W. A. Light / Oxford: Oxford University Press, 1992. - Pp. 105–210.

132. *Reed R.* Pruning algorithms - a survey // IEEE Trans. Neural Networks, 1993. - Vol. 4. - Pp. 740–747.

133. *Riedmiller M., Braun H.* RPROP – a fast adaptive learning algorithm. Technical Report, Karlsruhe: University Karlsruhe, 1992.

134. *Ritter H., Schulten K.* On the stationary state of the Kohonen self-organizing sensory mapping // Biological Cybernetics. 1986. - Vol. 54. - Pp. 234–249.

135. *Rosenblatt F.* Principle of neurodynamics. – N.Y.: Spartan, 1992.

136. *Rumelhart D. E., Hinton G. E., Williams R. J.* Learning internal representations by error propagation //Parallel distributed processing: Exploratians in the Microstructures of Cognition / D. E. Rumelhart, J. L. McClelland, (Eds.). Cambridge: MIT Press, 1986. - Vol. 1.

137. *Rugh W.* Nonlinear Systems: a Volterra approach. – N.Y.: J. Hopkins Press, 1981.

138. *Rutkowska D., Pilinski M., Rutkowski L.* Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. – Warszawa: PWN, 1997.

139. *Rzesiowski D.* Neuronowe metody kwantyzacji wektorowej w zastosowaniu do kompresji obrazyw: Rozprawa doktorska. – Warszawa: Politechnika Warszawska, 1995.

140. *Sammon J.* A nonlinear mapping for data structure analysis // IEEE Trans. Computers, 1969. - Vol. 18. - Pp. 401–409.

141. *Sanger T. D.* Optimal unsupervised learning in single layer linear feedformed neural network // Neural Networks, 1989. - Vol. 2. - Pp. 459–473

142. *Sejnowski T.* Strong covariance with nonlinerly interacting neurons // Jour. Math. Biology, 1977. - Vol. 4. - Pp. 385–389.

143. *DeSieno D.* Adding a conscience to competitive learning // Neural Networks, 1988. - Vol. 1. - Pp. 117–124.

144. *Siwek K.* Implementacja komputerowa wybranych algorytmow uczacych i aplikacyjnych sieci samoorganizujacych sie Kohonena. Praca dyplomowa. – Warszawa: Politechnika Warszawska, 1995.

145. *Skarbek W.* Metody reprezentacji obrazow cyfrowych. – Warszawa: Akademicka Oficyna Wydawnicza, 1993.

146 *Skoneczny S.* Classical and neural methods of image filtering: Praca doktorska. Warszawa: Politechnika Warszawska, 1994.

147. *Speckmann H., Thole P., Bagdan M., Rosentiel W.* Coprocessors for special neural networks: KOKOS and KOBOLD // IEEE Proc. ICNN, Orlando, 1994.

148. *Speckmann H., Thole P., Rosenthiel W.* COKOS: a coprocessor for Kohonen self organizing map // ICANN-93, Amsterdam, 1993. - Pp. 1040–1044.

149. *Stepniewski S.* Zastosowanie algorytmow genetycznych do generacji struktur sieci neuronowych jednokierunkowych: Rozprawa doktorska. – Warszawa: Politechnika Warszawska, 1997.

150. *Sutton R. S.* Temporal credit assignment in reinforcement learning. Ph. D. dissert. - MA: Univ. Mass., 1984.

151. *Svarer C.* Neural networks for signal processing: Ph.D.dissert. – Lyngby, TUD: Electronics Institute, 1994.

152. *Tadeusiewicz R.* Sieci neuronowe. – Warszawa: Akademicka Oficyna Wydawnicza, 1993.

153. *Takagi T., Sugeno M.* Fuzzy identification of systems and its application to modeling and control // IEEE Trans. SMC, 1985. - Pp. 116–132.

154. *Tarassenko L., Roberts S.* Supervised and unsupervised learning in radial basis function classifiers // IEEE Proc. Vis. Image Signal Process., 1994. - Vol. 141. - Pp. 210–216.

155. *Thimm G., Fiesler E.* Neural network initialization // From Natural to Artificial Neural Computation / Eds. J. Mira, F. Sandoval. - Malaga: IWANN, 1995. - Pp. 533–542.

156. *Tran Hoai L.* Sieci neuronowe rekurencyjne - gradientowe algorytmy uczace i zastosowania w identyfikacji nieliniowych obiektow dynamicznych. Praca dyplomowa. - Warszawa: Politechnika Warszawska, 1997.

157. *Vapnik V. N.* Principle of risk minimization for learning theory // Advances in NIPS4 / Eds. J. Moody, S. Hanson, R. Lippmann. – San Mateo: Morgan Kaufmann, 1992. - Pp. 831–838.

158. *Vapnik V. N., Chervonenkis A.* On the uniform convergence of relative frequencies of events to their probabilities // Theory of Probability and its Applications, 1971. - Vol. 16. - Pp. 264–280.

159. *Veith A.C., Holmes G.* A modified quickprop algorithm // Neural Computation, 1991. - Vol. 3. - Pp. 310–311.

160. *Verbruggen H.B., Babuska R.* Constructing fuzzy models by product space clustering // Fuzzy model identification / Eds. H. Hellendorn, D. Driankov. – Berlin: Springer, 1998. - Pp. 53–90.

161. *Wang L. X.* Adaptive fuzzy systems and control. Design and stability analysis. New Jersey: Prentice Hall, 1994.

162. *Wang Y. F., Cruz J., Mulligan J.* On multiple training for BAM // IEEE Trans. Neural Networks, 1990. - Vol. 1. - Pp. 275–276.

163. *Wang Y. F., Cruz J., Mulligan J.* Two coding strategies for bidirectional associative memory // IEEE Trans. Neural Networks, 1990. - Vol. 1. - Pp. 81–92.

164. *Wang W. J., Lee D. L.* A modified bidirectional decoding strategy based on BAM structure // IEEE Trans. Neural Networks, 1993. - Vol. 4. - Pp. 710–717.

165. *Wessel L. F., Barnard E.* Avoiding false local minima by proper initialization of connections // IEEE Trans. Neural Networks, 1992. - Vol. 3. - Pp. 899–905.

166. *Weymaere N., Martens J.P.* On the initialization and optimization of multilayer perception // IEEE Trans. Neural Networks, 1994. - Vol. 5. - Pp. 738–751.

167. *Widrow B., Hoff M.* Adaptive switching circuits // Proc. IRE WESCON Convention Record, 1960. - Pp. 107–115.

168. *Widrow B., Winter R., Baxter R.* Layered neural nets for pattern recognition // IEEE Trans. ASSP, 1988. - Vol. 36. - Pp. 1109–1118.

169. *Widrow B., Bilello M.* Nonlinear adaptive signal processing for inverse control // Proc. World Congress on Neural Networks, San Diego 1994.

170. *Widrow B., Stearns S.* Adaptive signal processing. – N.Y.: Prentice Hall, 1985. 171. Wierzbicki A., Findeisen W., Szymanowski J. Teoria i metody obliczeniowe optymalizacji. – Warszawa: WNT, 1977.

171. *Williams R., Zipser D.* A learning algorithm for continually running fully recurent neural networks. - Neural Computers, 1989. - Vol. 1. - Pp. 270–280.

172. *Williams R., Zipser D.* A learning algorithm for continually running fully recurent neural networks // Neural Computers, 1989. - Vol. 1. - Pp. 270–280.

173. *Xue Q., Hu Y., Tompkins W.* Analysis of hidden units of back propagation model by SVD // Proc. IJCNN, 1990, Washington. - Pp. 1739–742.

174. *Yager R., Filev D.* Podstawy modelowania i sterowania rozmytego. – Warszawa: WNT, 1995.

175. *Yair E., Zeger K., Gersho A.* Competitive learning and soft competition for vector quantizer design // IEEE Trans. Signal Processing, 1992. - Vol. 40. - Pp. 294–309.

176. *Yamakawa T.* A fuzzy interference engine in nonlinear analog made and its application to fuzzy logic control // IEEE Trans. Neural Networks, 1993. - Vol. 4. - Pp. 496–522.

177. *Yuceer C., Oflazer K.* A rotation, scaling and translation invariant pattern classification system // Pattern Recognition, 1993. - Vol. 26. - Pp. 687–710.

178. *Zadeh L.A.* The concept of linguistic variable and its application to approximate reasoning. Part 1-3 // Information Sciences, 1975. - Pp. 199–249.

179. *Zadeh L.A.* Fuzzy sets // Information and Control, 1965. - Pp. 338–353.

180. *Zell A.* Simulation Neuronal Netze. – Amsterdam: Addison Wesley, 1994.

181. *Zell A.* SNNS - Stuttgart Neural Network Simulator. User manual, Stuttgart, 1993.

182. *Zimmermann H. J.* Fuzzy set theory and its applications. – Boston: Kluwer, 1985.

183. *Zuben F., Netto M.A.* Exploring the nonlinear dynamic behaviour of artificial neural networks // IEEE Proc. ICNN, Orlando, 1994. - Pp. 1000–1005.

184. *Zak S., Upatising V., Hui S.* Solving linear programming problems with neural networks: a comparative study // IEEE Trans. Neural networks, 1995. - Vol. 6. - Pp. 94–104.

185. *Zurada J., Barski M., Jedruch W.* Sztuczne sieci neuronowe. – Warszawa: PWN, 1997.

Предметный указатель

Агрегирование 288

Аксон 17

Алгоритм
– Видроу 33
– Вильямса–Зипсера 221
– Гаусса–Ньютона 67
– генетический 81
– гибридный 144, 304
– градиентный 60
– Густафсона–Кесселя 313
– Даркена–Муди 141
– естественного градиента 276
– имитации отжига 78
– Кардоссо 276
– Кохонена 233
– Левенберга–Марквардта 65
– Линде–Бузо–Грея 139
– наискорейшего спуска 62
– нейронного газа 233
– обратного распространения ошибки 51, 146
– ортогонализации Грэма-Шмидта 152
– переменной метрики 63
– пикового группирования 310
– разностного группирования 312
– самоорганизации 308
– сопряженных градиентов 67
– Хольта-Джуттена 270
– Чихотского 272, 276
– Эльмана 212
– BFGS 64
– C-means 308
– CWTA 232
– DFP 64
– K-means 139
– Quickprop 71, 160
– RPROP 73
– WTA 184, 232

Аппроксимация
– глобальная 129
– локальная 129

Ассоциативность 282

Ассоциация 23

Вектор направления 61

Восстановление Сэммона 241

Выбор центров
– вероятностный 142
– на основе самоорганизации 139
– случайный 139

Гессиан 61, 64

Гиперплоскость 47

Граф
– потоковый сигналов 55
– сопряженный 55, 147

Группирование данных 310, 312

Дендриты 17

Дефуззифнкатор 28, 293

Дефуззифнкация
– относительно среднего центра 293
– относительно центра области 293

Диафония 180

Дистрибутивность 282

Емкость ассоциативной памяти 180

Идентификация объекта 121, 221

Инвариантность относительно
– количества выборок 112
– угла поворота 112
– смещения 111
– масштаба 112

Инициализация весов 86

Кардинальное число 280

Квантование векторное 231

Классификатор
– векторный 35
– нейронный 110, 114

Кластер 139, 327

Кодирование двоичное 82

Колатералы 17

Компрессия данных 116, 242, 267

Коммутативность 282

Контекстный слой 210

Коэффициент
– асимметрии 99
– компрессии 119, 243
– обучения 52
– – адаптивный 68
– – постоянный 68
– – с направленной минимизацией 69
– принадлежности 279
– сопряжения 67
– PSNR 243, 267

Лингвистическая переменная 279

Мадалайн 33

Матрица
– автокорреляции 259
– корреляции 43

Мера
– Егера 283
– Коско 284
– нечеткости 283
– расстояния 228
– энтропийная 284
– VCdim 94

Метод
– проекций 181
– Δ-проекций 181
– Хебба 179
– OBD 99
– OBS 100

Методы эвристические 71

Механизм утомления 230

Модель
– адалайн 32
– инстар 34
– МакКаллока–Питса 20
– Мамдани-Заде 294
– оутстар 37
– персептронная 22
– сигмоидальная 27
– стохастическая 44
– Хебба 21
– TSK 295
– WTA 37

Мутация 82, 84

MAXNET 184

Независимость статистическая 268

Нейромедиатор 17

Нейроны мертвые 40

Нечеткое множество 279

Нормализация
– векторов 229
– множества 282

Носитель 280

Область Вороного 139

Обобщение 24, 92

Обучение
– без учителя 25
– онлайн 51
– оффлайн 52
– с учетом момента 30, 63
– с учителем 25
– с забыванием 41

Окрестность 227

Операция
– концентрации 282

– отрицания 281
– произведения 282
– растяжения 282
– суммирования 281

Оптимизация глобальная 75

Ортогонализация Грэма–Шмидта 152

Отрицание множества 281

Ошибочное решение 182

Память
– автоассоциативная 178
– ассоциативная 177
– гетероассоциативная 177, 185

Переобучение сети 97

Персептрон 26

Погрешность
– квантования 231
– обобщения 94
– обучения 94
– MAPE 125, 255
– MSE 118

Популяция 82

Правила вывода 286

Правило
– Видроу–Хоффа 27
– Гроссберга 34, 37
– Коско 190
– – модифицированное 192
– Кохонена 227
– Ойя 264
– персептрона 26
– Сенгера 265
– Хебба 41

Преобразование
– Карьюнена–Леве 259
– Фурье 111
– ICA 267
– PCA 117, 259

Прогнозирование 23
– нагрузок 124, 249
– числовой последовательности 73

Программа
– *BS* 274
– *Cascor* 162
– *Hfnet* 182
– *Kohon* 232
– *Netteach* 71
– *RMLP* 205
– *RTRN* 221

Произведение логическое 281

Равенство множеств 282

Расстояние Хемминга 177

Режим
– обучения 179, 180
– распознавания 182

Ряд Тейлора 60

Самоорганизация нейронов 266

Селекция 82

Сеть нейронная 46
– Вольтерри 163
– Ванга–Менделя 302
– гибридная 252
– корреляционная 257
– многослойная 50
– нечеткая 299
– однонаправленная 46
– однослойная 47
– персептронная 50
– радиальная HRBF 137
– радиальная RBF 129, 132
– рекуррентная 176
– Фальмана 159
– Хемминга 184
– Херольта–Джуттена 267, 269
– Хехта–Нильсена 254
– Хопфилда 178
– Эльмана 210
– BAM 189
– BAM модифицированная 192
– ICA 267
– PCA 259
– RMLP 200
– RTRN 219
– TSK 299

Синапс 17

Система
– вывода 287, 295
– – Мамдани-Заде 287
– – TSK 295

Скрещивание 82, 83

Сома 17

Стабильность ассоциативной памяти 180

Сумма логическая 281

Температура 79

Теорема Ковера 130

Теория Колмогорова 90

Универсальный аппроксиматор 22

Условия ДеЛуки–Термини 283

Фуззификатор 287 290

Функция
– активации 26
– Гаусса 134
– Гаусса обобщенная 291
– Лагранжа 309
– пиковая 310
– принадлежности 279
– приспособленности 82
– радиальная 129, 134
– сигмоидальная 27
– соседства 233
– трапецеидальная 292
– треугольная 292
– целевая 27, 47
– энергетическая сети 190, 257

Хромосома 82

Чувствительность 55

Шаг обучения 61

Ядро Вольтерри 165, 169

SVD-разложение 263

Учебно-справочное издание

Станислав Осовский

НЕЙРОННЫЕ СЕТИ ДЛЯ ОБРАБОТКИ ИНФОРМАЦИИ

Научный редактор *И.Д. Рудинский*

Заведующая редакцией *Л.А. Табакова*
Ведущий редактор *Л.Д. Григорьева*
Младший редактор *Н.А. Федорова*
Художественный редактор *Ю.И. Артюхов*
Технический редактор *И.В. Завгородняя*
Корректоры *Т.М. Колпакова, Г.В. Хлопцева*
Компьютерная верстка *Т.С. Левыкина*
Художественное оформление *О.В. Толмачева*

ИБ № 4464

Подписано в печать 20.06.2002. Формат 70×100/16
Печать офсетная. Гарнитура "Times"
Усл.п.л. 27,95. Уч.-изд.л. 24,36
Тираж 3000 экз. Заказ 2162. "С" 150

Издательство "Финансы и статистика"
101000, Москва, ул. Покровка, 7
Телефон (095) 925-35-02, факс (095) 925-09-57
E-mail: mail@finstat.ru http://www/finstat.ru

ГУП "Великолукская городская типография"
Комитета по средствам массовой информации
Псковской области,
182100, Великие Луки, ул. Полиграфистов, 78/12
Тел./факс: (811-53) 3-62-95
E-mail: VTL@MART.RU