Лабораторная работа

Самоорганизующиеся карты Кохонена

Цель работы – изучение самоорганизующихся карт Кохонена.

Содержание:

1.Изучить структуру сети Кохонена.

2. Изучить топологию самоорганизующихся карт Кохонена.

3. Изучить меры расстояний между нейронами при создании самоорганизующихся карт Кохонена.

4. Создать самоорганизующуюся карту Кохонена.

Самоорганизующиеся карты (Self-Organizing Maps, SOM) могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации.

Наиболее распространенное применение сетей Кохонена - решение задачи классификации без учителя, т.е. кластеризации.

Напомним, что при такой постановке задачи нам дан набор объектов, каждому из которых сопоставлена строка таблицы (вектор значений признаков). Требуется разбить исходное множество на классы, т.е. для каждого объекта найти класс, к которому он принадлежит.

В результате получения новой информации о классах возможна коррекция существующих правил классификации объектов.

Вот два из распространенных применений карт Кохонена: разведочный анализ данных и обнаружение новых явлений.

Разведочный анализ данных. Сеть Кохонена способна распознавать кластеры в данных, а также устанавливать близость классов. Таким образом, пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель. Если в данных распознаны классы, то

их можно обозначить, после чего сеть сможет решать задачи классификации. Сети Кохонена можно использовать и в тех задачах классификации, где классы уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.

Обнаружение новых явлений. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам. Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.

Таким образом, карты Кохонена (как и географические карты) можно отображать:

- в двухмерном виде, тогда карта раскрашивается в соответствии с уровнем выхода нейрона;

- в трехмерном виде.

В результате работы алгоритма получаем такие карты:

- карта входов нейронов;

- карта выходов нейронов;

- специальные карты.

Координаты каждой карты определяют положение одного нейрона. Так, координаты [15:30] определяют нейрон, который находится на пересечении 15-го столбца с 30-м рядом в матрице нейронов. Рассмотрим, что же представляют собой эти карты.

Карта входов нейронов. Веса нейронов подстраиваются под значения входных переменных и отображают их внутреннюю структуру. Для каждого входа рисуется своя карта, раскрашенная в соответствии со значением конкретного веса нейрона.

При анализе данных используют несколько карт входов.

На одной из карт выделяют область определенного цвета - это означает, что соответствующие входные примеры имеют приблизительно одинаковое значение соответствующего входа. Цветовое распределение нейронов из этой

области анализируется на других картах для определения схожих или отличительных характеристик. Пример рассмотренных *карт входов* будет приведен ниже.

Карта выходов нейронов.

На *карту выходов нейронов* проецируется взаимное расположение исследуемых входных данных. Нейроны с одинаковыми значениями выходов образуют кластеры - замкнутые области на карте, которые включают нейроны с одинаковыми значениями выходов.

Специальные карты. Это карта кластеров, матрица расстояний, матрица плотности попадания и другие карты, которые характеризуют кластеры, полученные в результате обучения сети Кохонена.

Важно понимать, что между всеми рассмотренными картами существует взаимосвязь - все они являются разными раскрасками одних и тех же нейронов. Каждый пример из обучающей выборки имеет одно и то же расположение на всех картах.

Сеть Кохонена обучается методом последовательных приближений. В процессе обучения таких сетей на входы подаются данные, но сеть при этом подстраивается не под эталонное значение выхода, а под закономерности во входных данных. Начинается обучение с выбранного случайным образом выходного расположения центров.

В процессе последовательной подачи на вход сети обучающих примеров определяется наиболее схожий нейрон (тот, у которого скалярное произведение весов и поданного на вход вектора минимально). Этот нейрон объявляется победителем и является центром при подстройке весов у соседних нейронов. Такое правило обучения предполагает «соревновательное» обучение с учетом расстояния нейронов от «нейрона-победителя».

Обучение при этом заключается не в минимизации ошибки, а в подстройке весов (*внутренних параметров* нейронной сети) для наибольшего совпадения с входными данными.

Основной *итерационный алгоритм* Кохонена последовательно проходит ряд эпох, на каждой из которых обрабатывается один пример из обучающей выборки. Входные сигналы последовательно предъявляются сети, при этом желаемые выходные сигналы не определяются. После предъявления достаточного числа входных векторов синаптические веса сети становятся способны определить кластеры. Веса организуются так, что топологически близкие узлы чувствительны к похожим входным сигналам.

В результате работы алгоритма *центр кластера* устанавливается в определенной позиции, удовлетворительным образом кластеризующей примеры, для которых данный нейрон является «победителем». В результате обучения сети необходимо определить меру соседства нейронов, т.е. *окрестность* нейрона-победителя.

Окрестность представляет собой несколько нейронов, которые окружают нейрон-победитель. Сначала к *окрестности* принадлежит большое число нейронов, далее ее размер постепенно уменьшается. Сеть формирует топологическую структуру, в которой похожие примеры образуют группы примеров, близко находящиеся на топологической карте.

Полученную карту можно использовать как средство визуализации при анализе данных. В результате обучения карта Кохонена классифицирует входные примеры на кластеры (группы схожих примеров) и визуально отображает многомерные входные данные на плоскости нейронов.

Уникальность метода *самоорганизующихся карт* состоит в преобразовании n-мерного пространства в двухмерное. Применение двухмерных сеток связано с тем, что существует проблема отображения пространственных структур большей размерности.

Имея такое представление данных, можно визуально определить наличие или отсутствие взаимосвязи во входных данных.

Нейроны карты Кохонена располагают в виде двухмерной матрицы, раскрашивают эту матрицу в зависимости от анализируемых параметров нейронов.

Порядок выполнения заданий

Задание 1. Изучить структуру сети Кохонена.

Для создания самоорганизующихся нейронных сетей, являющихся слоем или картой Кохонена, предназначены М-функции newc и newsom соответственно. По команде help selforg можно получить следующую информацию об М-функциях, входящих в состав ППП Neural Network Toolbox и относящихся к построению сетей Кохонена. При создании самоорганизующихся карт Кохонена используется специальные функции табл. 1-2.

Таблица 1 – М-функции, используемые при создании карт Кохонена

Функция	Содержание
Self-organizing networks	Самоорганизующиеся сети
New networks	Формирование сети
newc	Создание слоя Кохонена
newsom	Создание карты Кохонена
Using networks	Работа с сетью
sim	Моделирование
init	Инициализация
adapt	Адаптация
train	Обучение
Weight functions	Функции расстояния и взвешивания
negdist	Отрицательное евклидово расстояние
Net input functons	Функции накопления
netsum	Сумма взвешенных входов
Transfer functions	Функции активации
compet	Конкурирующая функция активации
Topology functions	Функции описания топологии сети
gridtop	Прямоугольная сетка
hextop	Гексагональная сетка
randtop	Сетка со случайно распределенными узлами
Distance functions	Функции расстояния

dist boxdist mandist linkdist	Евклидово расстояние Расстояние максимального координатного смещения Расстояние суммарного координатного смешения Расстояние связи
Initialization functions	Функции инициализации сети
initlay initwb initcon midpoint	Послойная инициализация Инициализация весов и смещений Инициализация смещений с учетом чувствительности нейронов Инициализация весов по правилу средней точки
Learning functions	функции настройки параметров
learnk learncon learnsom	Правило настройки весов для слоя Кохонена Правило настройки смещений для слоя Кохонена Правило настройки весов карты Кохонена
Adapt functions	Функции адаптации
adaptwb	Адаптация весов и смещений
Training functions	Функции обучения
trainwb1	Обучение
Demonstrations	Демонстрационные примеры
democ1 demosm1 demosm2	Настройка слоя Кохонена Одномерная карта Кохонена Двумерная карта Кохонена

Синтаксис функции selforgmap, приведен ниже:

selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn)

Таблица 2 – Параметры функции selforgmap

dimensions	Row vector of dimension sizes (по умолчанию задается в виде вектора [8 8])
coverSteps	Количество циклов обучения сети По умолчанию задается равным 100 (default = 100)
initNeighbor	Начальная размерность соседства меду нейронами По умолчанию задается равным 3 (default = 3)
topologyFcn	Функция для задания топологии нейронов По умолчанию задается в виде гексагональной решетки (default = 'hextop')
distanceFcn	Функция определяет расстояние между нейронами По умолчанию принимается евклидово расстояние (default = 'linkdist')

Пример 1. Использование стандартной функции для создания самоорганизующейся карты.

```
x = simplecluster_dataset;
net = selforgmap([8 8]);
net = train(net,x);
view(net)
y = net(x);
      classes = vec2ind(y);
```

Задание 2. Изучить топологию самоорганизующихся карт Кохонена.

При формировании самоорганизующейся карты можно задать определенную топологию расположения нейронов. Для этого используются М-функции gridtop, hextop, randtop. Для создания простой прямоугольной сетки, состоящей из шести нейронов размером 2x3 необходимо использовать функцию gridtop (рис. 1):

```
pos = gridtop(2,3)
pos =
    0    1    0    1    0    1
    0    0    1    1    2    2
plotsom(pos) %.
```

Соответствующая сетка показана на рис.1. Метки position(1, i) и position(2,i) вдоль координатных осей генерируются функцией plotsom и задают позиции расположения нейронов по первой, второй и т. д. размерностям карты.

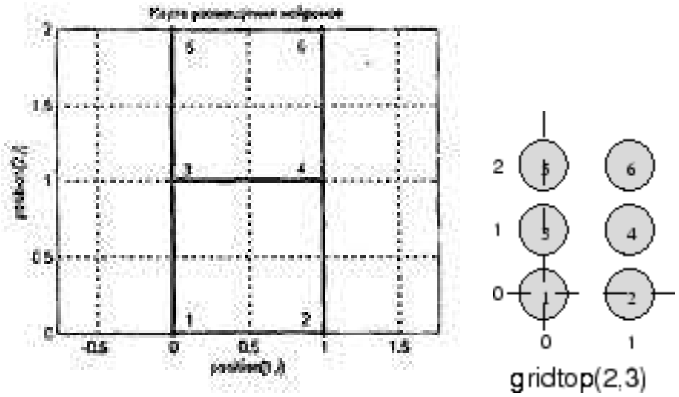


Рисунок 1 – Прямоугольная сетка

На прямоугольной сетке нейрон 1 расположен в точке с координатами (0,0), нейрон 2 - в точке (1,0), нейрон 3 - в точке (0,1) и т. д. Заметим, что, если применить команду gridtop, переставив аргументы местами, получим иное размещение нейронов;

```
pos = gridtop(3,2)
pos =
    0    1    2    0    1    2
    0    0    0    1    1    1
```

Если нужно создать прямоугольную сетку размером 8*10 с помощью функции gridtop необходимо задать программный код (рис.3):

```
pos = gridtop(8,10);
plotsom(pos)
```

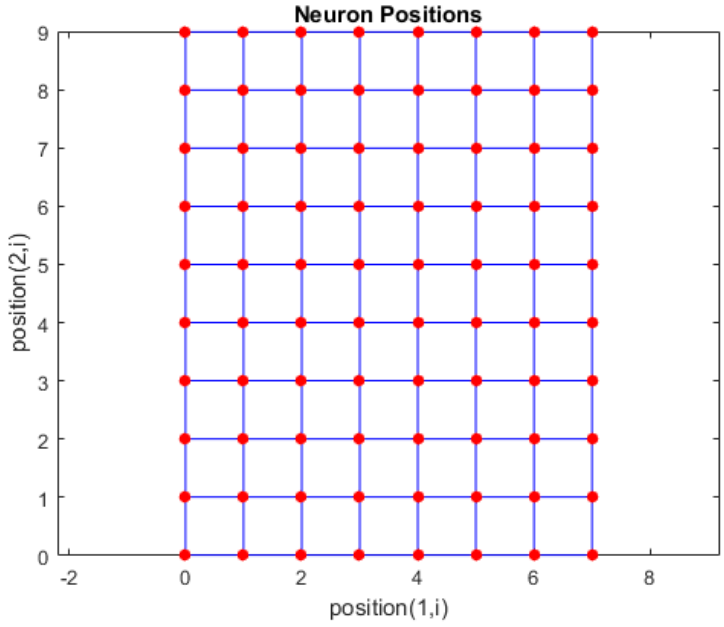


Рисунок 3 – Прямоугольная решетка, размеров 8*10

Гексагональную сетку можно сформировать с помощью функции hextop рис. 4:

pos = hextop(2,3)

pos =

0 1.0000 0.5000 1.5000 0 1.0000

0 0 0.8660 0.8660 1.7321 1.7321.

plotsom (pos) %

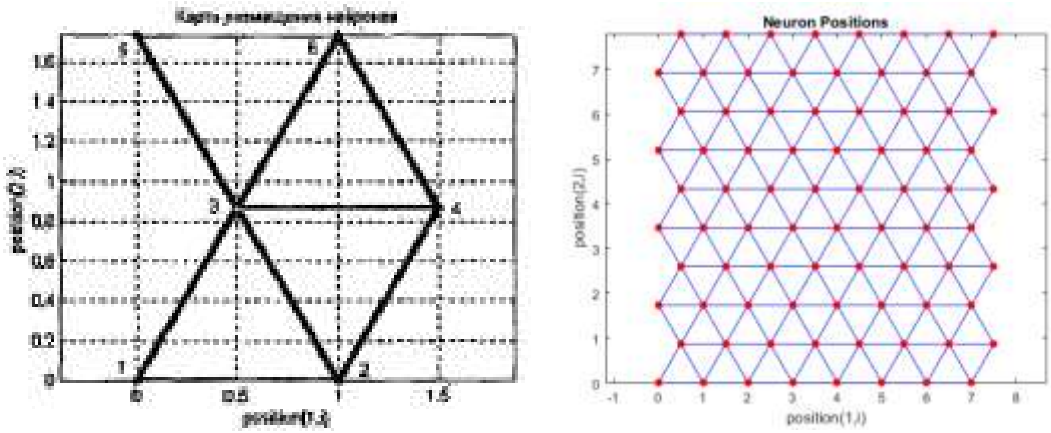


Рисунок 4 – Гексагональная сетка

М-функция hextop используется по умолчанию при создании карт Кохонена при применении функции newsom. Сетка со случайным расположением узлов может быть создана с помощью функции randtop:

pos = randtop(2,3)

pos =

0 0.7620 0.6268 1.4218 0.0663 0.7862

0.0925 0 0.4984 0.6007 1.1222 1.4228

plotsom(pos) % (рис.5).

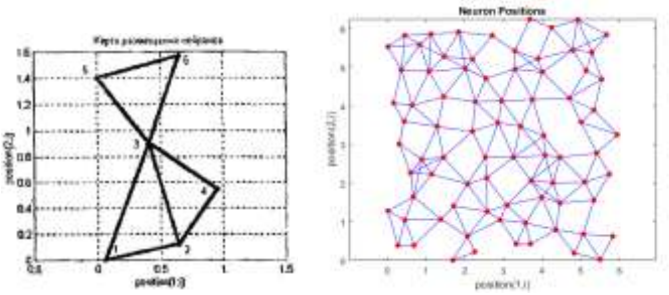


Рисунок 5 – Сетка со случайным расположением узлов

Задание 3. Изучить меры расстояний между нейронами при создании самоорганизующихся карт Кохонена.

В ПК MATLAB реализованы 4 функции для расчета расстояний между узлами сетки. Функция dist используется для вычисления евклидово расстояния между нейронами, размещенными в узлах сетки, в соответствии с формулой:

$$d = sqrt(sum((pos_i - pos_j).^2)), \quad (1)$$

где pos_i, pos_j - векторы положения нейронов с номерами *i* и j.

Обратимся к прямоугольной сетке из шести нейронов и вычислим соответствующий массив расстояний:

```
pos = gridtop(2,3)
d = dise(pos)
d =
0 1 1 1.4142 2 2.2361
1 0 1.4142 1 2.2361 2
1 1.4142 0 1 1 1.4142
1.4142 1 1 0 1.4142 1
2 2.2361 1 1.4142 0 1
2.2361 2 1.4142 1 1 0.
```

Данный массив размером 6x6 описывает расстояния между нейронами и содержит на диагонали нули, поскольку они определяют расстояние нейрона

до самого себя, а затем, двигаясь вдоль строки, до второго, третьего и т. д. На рис. 6 показано расположение нейронов в узлах прямоугольной сетки. Введем понятие окрестности для прямоугольной сетки. В этом случае окрестность размера 1, или просто окрестность 1, включает базовый нейрон и его непосредственных соседей; окрестность 2 включает нейроны из окрестности 1 и их соседей.

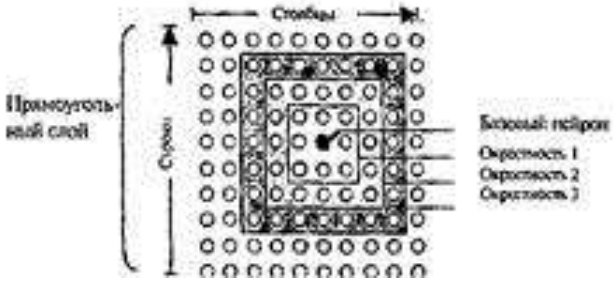


Рисунок 6 – Расположение нейронов в узлах прямоугольной сетки

Размер, а соответственно и номер окрестности, определяется максимальным значением координаты смещения нейрона от базового. Вводимое таким способом расстояние между нейронами называется расстоянием максимального координатного смещения и может быть вычислено по формуле:

$$d = \max(abs((pos_i - pos_j)), \quad (2)$$

Для вычисления этого расстояния в ППП NNT предназначена М-функция boxdist. Для конфигурации нейронов, показанной на рис.1, расстояния между нейронами равны:

```
pos = gridtop(2,3)
d = boxdist(pos)
d=
0 1 1 1 2 2
1 0 1 1 2 2
1 1 0 1 1 1
1 1 1 0 1 1
2 2 1 1 0 1
```

2 2 1 1 1 0.

Расстояние максимального координатного смещения между базовым нейроном 1 и нейронами 2, 3 и 4 равно 1, поскольку они находятся в окрестности 1, а расстояние между базовым нейроном и нейронами 5 и 6 равно 2, и они находятся в окрестности 2. Расстояние максимального координатного смещения от нейронов 3 и 4 до всех других нейронов равно 1.

Определим другое расстояние между нейронами, которое учитывает то количество связей, которое необходимо установить, чтобы задать путь движения от базового нейрона. Если задано S нейронов, положение которых определяется векторами p_i, i = 1,..., S, то расстояние связи между ними определяется соотношением:

$$d_{ij} = \begin{cases} 1, dist(pos_i - pos_j) \le 1; \\ 2, \forall k, d_{ik} = d_{kj} = 1; \\ 3, \forall (k_1, k_2), d_{ik} = d_{kj} = 1; \\ N, \forall (k_1, k_2, ..., k_N), d_{ik} = d_{kj} = d_{kN} = 1; \\ S, \quad \text{в остальных случаях} \end{cases}$$

Если евклидово расстояние между нейронами меньше или равно 1, то расстояние связи принимается равным 1; если между нейронами с номерами *i* и *j* имеется единственный промежуточный нейрон с номером *k*, то расстояние связи равно 2, и т. д.

Для вычисления расстояния связи используется функция linkdist. Для конфигурации нейронов, доказанной на рисунке 3.6, эти расстояния равны:

pos = gridtop{2,3)

d = linkdist(pos)

d =

0 1 1 2 2 3

1 0 2 1 3 2

1 2 0 1 1 2

2 1 1 0 2 1

2 3 1 2 0 1

3 2 2 1 1 0.

Расстояние связи между базовым нейроном 1 и нейронами 2, 3 равно 1, между базовым нейроном и нейронами 4 и 5 равно 2, между базовым нейроном и нейроном 6 равно 3. Наконец, определим расстояние максимального координатного смещении по формуле:

$$d = sum(abs((pos_i - pos_j)), \quad (4)$$

Для вычисления расстояния максимального координатного смещения предназначена функции mandist.

```
pos = gridtop(2,3)
d = mandist(pos)
d =
0 1 1 2 2 3
1 0 2 1 3 2
1 2 0 1 1 2
2 1 1 0 2 1
2 3 1 2 0 1
3 2 2 1 1 0.
```

В случае прямоугольной сетки оно совпадает с расстоянием связи.

Задание 4. Создать самоорганизующуюся карту Кохонена.

Для создания самоорганизующейся карты Кохонена в составе ППП MATLAB NNT предусмотрена M-функция newsom. Допустим, что требуется создать сеть для обработки двухэлементных векторов входа с диапазоном изменения элементов от 0 до 2 и от 0 до 1 соответственно. Предполагается использовать гексагональную сетку размера 2x3. Тогда для формирования такой нейронной сети достаточно воспользоваться оператором:

```
net = newsom([0 2; 0 1], [2 3])
net.layers{1}
ans =
dimensions:[2 3]
```

distanceFcn:'linkdist'

distances:[6x6 double]

initFcn:'initwb'

netInputFcn:'netsum'

positions:[2x6 double]

size:6

topologyFcn:'hextop'

transferFcn:'compet'

userdata:[1x1 struct].

Из анализа характеристик этой сети следует, что она использует по умолчанию гексагональную топологию hextop и функцию расстояния linkdist.

Для обучения сети используем следующие 12 двухэлементных векторов входа:

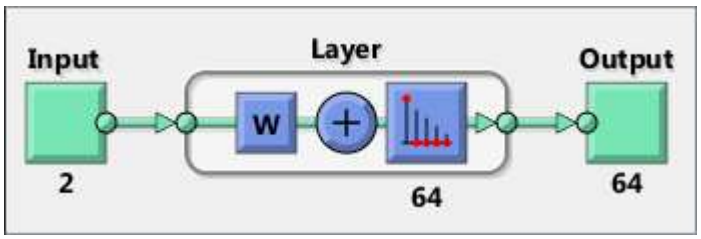
```
P = [0.1 0.3 1.2 1.1 1.8 1.7 0.1 0.3 1.2 1.1 1.8 1.7; ...
0.2 0.1 0.3 0.1 0.3 0.2 1.8 1.8 1.9 1.9 1.7 1.8]
net = selforgmap([2,3]);

%ЗАДАЕМ ВХОДНОЙ ВЕКТОР ДЛЯ ОБУЧЕНИЯ
P = [.1 .3 1.2 1.1 1.8 1.7 .1 .3 1.2 1.1 1.8 1.7;...
0.2 0.1 0.3 0.1 0.3 0.2 1.8 1.8 1.9 1.9 1.7 1.8];

% Задаем конфигурацию сети

net = configure(net,P);
plotsompos(net,P)

net.trainParam.epochs = 1000;
net = train(net,P);
plotsompos(net,P)
```


Input
Layer
Output
W
2
64
64