Лекция

Дерево решений – это дерево, внутренние узлы которого представляют собой проверки для входных наблюдений (примеров, объектов) из обучающей выборки (обучающего множества), а вершины–листы являются категориями, классами (примеров, объектов)

Описание признаков (атрибутов). Вся информация об объектах (примерах) из предметной области должна описываться конечным набором признаков (атрибутов). Каждый признак должен иметь качественное или количественное (числовое) значение. Набор признаков не должен меняться от примера к примеру, и количество признаков должно быть фиксированным для всех примеров.

•**Принадлежность классу.** Каждый пример (объект) в обучающей выборке должен быть ассоциирован с конкретным классом, т.е. один из признаков должен быть выбран в качестве имени или номера класса.

•**Объем классов.** Классы должны иметь конечное число примеров. Каждый пример должен однозначно относиться к конкретному классу. Случаи, когда примеры принадлежат к классу с вероятностными оценками, исключаются. Количество классов должно быть значительно меньше количества примеров.

Существует несколько характеристик, по которым различаются деревья решений:

1.Проверки могут быть многопризнаковыми (выполняется проверка нескольких признаков входного примера за один раз) или однопризнаковыми (выполняется проверка только одного признака).

2.Проверки могут приводить к двум или более результатам. Если все проверки приводят к двум результатам, то мы получаем двоичное дерево решений.

3.Признаки, которые используются в узлах дерева, могут быть качественными или количественными. Бинарные признаки могут рассматриваться как любые из них.

4.Классов может быть два или более.

Дерево решений – это способ представления правил в иерархической, последовательной структуре, где каждому объекту соответствует единственный узел, дающий решение.

Под решающим правилом понимается логическая конструкция в виде продукции. Продукционные правила представляются в виде

если <посылка>, **то** <заключение >.

В системах извлечения знаний в качестве посылки выступает описание объекта через его признаки, а заключением будет вывод о принадлежности объекта к определенному классу. В экспертных системах часто используются правила, в которых посылкой является описание ситуации, а заключением – действия, которые необходимо выполнить в данной ситуации.

Любое решающее дерево может быть преобразовано в набор продукционных правил: каждому пути от корня дерева до терминальной вершины соответствует одно продукционное правило. Его посылкой является конъюнкция условий «признак – значение», соответствующих пройденным вершинам и ребрам дерева, а заключением – имя или номер класса, соответствующего терминальной вершине.

Область применения деревья решений. Область применения деревья решений в настоящее время широка, но все задачи, решаемые этим аппаратом могут быть объединены в следующие три класса:

Описание данных: Деревья решений позволяют хранить информацию о данных в компактной форме, вместо них мы можем хранить дерево решений, которое содержит точное описание объектов.

Классификация: Деревья решений отлично справляются с задачами классификации, т.е. отнесения объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.

Регрессия: Если целевая переменная имеет непрерывные значения, деревья решений позволяют установить зависимость целевой переменной от независимых(входных) переменных. Например, к этому классу относятся задачи численного прогнозирования(предсказания значений целевой переменной).

Идею построения деревьев решений на основе примеров рассмотрим по Р. Куинлану
(R. Quinlan). Им разработан известный алгоритм ID3 (Induction of Decision trees) [3]. Пусть задано некоторое обучающее множество *T*, содержащее объекты (примеры), каждый из которых характеризуется *n* атрибутами (признаками), причем один из них указывает на принадлежность объекта к определенному классу. Назовем признаки, которые задают свойства каждого примера обучающей выборки, предсказывающими (предикторными) атрибутами. Такие признаки могут быть бинарными, количественными или качественными. Признак, который для каждого примера задает принадлежность к формируемому понятию, называется предсказываемым атрибутом. Этот признак также входит в обучающую выборку.

Пусть через ($i = 1,\ldots, m$) обозначены классы. Тогда существуют три ситуации:

1)множество *T* содержит один или более примеров, относящихся к одному классу *Ck*. Тогда дерево решений для *T* – это лист (терминальный узел), определяющий класс *Ck*;

2)множество *T* не содержит ни одного примера, т.е. пустое множество. Тогда это снова лист, и класс, ассоциированный с листом, выбирается из другого множества отличного от *T*, скажем, из множества, ассоциированного с родителем;

3)множество *T* содержит примеры, относящиеся к разным классам. В этом случае следует разбить множество *T* на некоторые подмножества. Для этого выбирается один из

признаков, имеющий два и более отличных друг от друга значений $O1$, $O2$, …, Os. Множество T разбивается на подмножества $T1$, $T2$, … Ts, где каждое подмножество Ti содержит все примеры, имеющие значение Oi для выбранного признака. Это процедура будет рекурсивно продолжаться до тех пор, пока конечное множество не будет состоять из примеров, относящихся к одному и тому же классу.

Вышеописанная процедура лежит в основе многих современных алгоритмов построения деревьев решений. Очевидно, что при использовании данной методики, построение дерева решений будет происходить сверху вниз.

Для построения дерева на каждом внутреннем узле необходимо найти такое условие (проверку), которое бы разбивало множество, ассоциированное с этим узлом на подмножества. В условие должен быть включен один из атрибутов (признаков). Общее правило для выбора атрибута можно сформулировать следующим образом: выбранный
атрибут должен разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше. Такой атрибут считается наиболее информативным среди всех атрибутов, еще не рассмотренных на пути от корня дерева. В качестве меры информативности обычно используется теоретико-информационное понятие энтропии. Возможны и другие критерии. Например, при построении древовидных классификаторов (см. Лекц. 9) применяются статистические критерии, на основании которых производится выбор признака для разбиения множества объектов в узле.

Рассмотрим информационный критерий выбора [3]. Если имеется n равновероятных значений признака, то вероятность p каждого из них равна $1/n$ и информация, связанная со значением признака, равна $-\log p = \log n$ ($\log$ обозначает логарифм по основанию 2). В общем случае, если мы имеем дискретное распределение

$$P = (p1, p2, \ldots, pn), \qquad (10.1)$$

то передаваемая информация вычисляется по формуле

$$I(P) = -\sum_{i=1}^{n} pi \log pi . \qquad (10.2)$$

Значение I(P) дает оценку среднего количества информации, необходимого для определения класса примера из множества S.

Чем ближе распределение к равномерному, тем больше его энтропия.

Если множество S примеров (объектов) разбито на попарно непересекающиеся классы $C1$, $C2$,…, Ck, то информация, необходимая для того, чтобы установить класс примера, равна $Info\,(S) = I(P)$, где P – дискретное распределение вероятностей появления соответствующего примера при

условии его принадлежности классу $C1$, $C2$,…, Ck. Каждая из оценок вероятностей pi (10.1) того, что случайно выбранный пример из множества S будет принадлежать к классу Ci , вычисляется как
$pi = ||CSi||$,

где $|Ci|$, $|S|$ – мощности как отдельных классов, так и всей обучающей выборки

соответственно.

Разбив множество примеров на основе значений некоторого признака X на подмножества $S1$, $S2$,…, Sn, мы можем вычислить $Info(S)$ как взвешенное среднее информации, необходимой для установления принадлежности примера определенному классу в каждом подмножестве:

$$Info(X, S) = \sum_{i=1}^{n} \frac{|Si|}{|S|} Info(Si) . \qquad (10.3)$$

Величина

$$Gain(X, S) = Info(S) – Info(X, S) \qquad (10.4)$$

показывает количество информации, которое мы получаем благодаря признаку X. Алгоритм ID3 использует эту величину для оценки информативности признака при построении решающих деревьев, Это позволяет получать деревья минимального размера. Критерий (10.4) считается для всех признаков. Выбирается признак, максимизирующий данное выражение. Этот признак будет являться условием разбиения в текущем узле дерева.

Если в процессе работы алгоритма получен узел, ассоциированный с пустым множеством (т.е. ни один пример не попал в данный узел), то он помечается как лист

(терминальный узел), и в качестве решения для листа выбирается наиболее часто встречающийся класс у непосредственного предка данного листа.

Поясним, почему критерий (10.4) должен максимизироваться. Из свойств энтропии известно, что максимально возможное значение энтропии достигается в том случае, когда все сообщения равновероятны. В нашем случае, энтропия (10.3) достигает своего максимума, когда частота появления классов в примерах множества S равновероятна. Нам же необходимо выбрать такой признак , чтобы при разбиении по нему один из классов имел наибольшую вероятность появления. Это возможно в том случае, когда энтропия (10.3) будет иметь минимальное значение и, соответственно, критерий (10.4) достигнет своего максимума.

Если признаки являются количественными (числовыми), то следует выбрать некий порог, с которым должны сравниваться все значения признака. Пусть количественный
признак имеет конечное число значений. Обозначим их {*v*1, *v*2,…, *vn*}. Предварительно отсортируем все значения. Тогда любое значение, лежащее между *vi* и *vi*+1, делит все примеры на два множества: те, которые лежат слева от этого значения {*v*1, *v*2,…, *vi*}, и те, что справа {*vi*+1, *vi*+2 … *vn*}. В качестве порога можно выбрать среднее между значениями *vi* и
vi+1:

$$THi = (\underline{vi + vi+1})/ 2$$

Таким образом, мы существенно упростили задачу нахождения порога, и привели к рассмотрению всего *n* – 1 потенциальных пороговых значений *TH*1, *TH*2 … *THn*-1. Формулы (10.2) – (10.4) последовательно применяются ко всем потенциальным пороговым значениям. Затем среди них выбирается то, которое дает максимальное значение по критерию (10.4). Далее это значение сравнивается со значениями критерия (10.4), подсчитанными для остальных признаков. Если выяснится, что среди всех признаков данный числовой признак имеет максимальное значение по критерию (10.4), то в качестве условия разбиения выбирается именно он.

Алгоритм ID3 основан на следующей рекурсивной процедуре.

1.Выбирается признак для корневого узла дерева, и формируются ветви для каждого из возможных значений этого признака.

2.Дерево используется для классификации обучающего множества. Если все примеры на некотором листе принадлежат одному классу, то этот лист помечается именем этого класса.

3.Если все листья помечены именами классов, алгоритм заканчивает работу. В противном случае узел помечается именем очередного признака, и создаются ветви для каждого из возможных значений этого признака, после чего алгоритм снова выполняет шаг
2.

Метод деревьев решений (*decision trees*) является одним из наиболее популярных методов решения задач классификации и прогнозирования. Иногда этот метод *Data Mining* также называют деревьями решающих *правил*, деревьями классификации и регрессии.

Как видно из последнего названия, при помощи данного метода решаются задачи классификации и прогнозирования.

Если зависимая, т.е. целевая *переменная* принимает дискретные значения, при помощи метода дерева решений решается задача классификации.

Если же зависимая *переменная* принимает непрерывные значения, то *дерево* решений устанавливает зависимость этой переменной от независимых переменных, т.е. решает задачу численного прогнозирования.

Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов прошлого века. Самая ранняя и известная работа Ханта и др., в которой излагается суть деревьев решений - "Эксперименты в индукции" ("Experiments in *Induction*") - была опубликована в 1966 году.

В наиболее простом виде *дерево* решений - это способ представления *правил* в иерархической, последовательной структуре. Основа такой структуры - ответы "Да" или "Нет" на ряд вопросов.

Целью построения дерева решения является *определение* значения категориальной зависимой переменной.

Итак, для нашей задачи основными элементами дерева решений являются:

Корень дерева: "Солнечно?"

Внутренний узел дерева или узел проверки: "Температура воздуха высокая?", "Идет ли дождь?"

Лист, конечный узел дерева, узел решения или вершина: "Играть", "Не играть"

Ветвь дерева (случаи ответа): "Да", "Нет".

В рассмотренном примере решается задача *бинарной классификации*, т.е. создается дихотомическая классификационная модель. Пример демонстрирует работу так называемых бинарных деревьев.

В узлах бинарных деревьев *ветвление* может вестись только в двух направлениях, т.е. существует возможность только двух ответов на поставленный вопрос ("да" и "нет").

Бинарные деревья являются самым простым, частным случаем деревьев решений. В остальных случаях, ответов и, соответственно, *ветвей* дерева, выходящих из его *внутреннего узла*, может быть больше двух.

Рассмотрим более сложный пример. *База данных*, на основе которой должно осуществляться прогнозирование, содержит следующие ретроспективные данные о клиентах банка, являющиеся ее атрибутами: возраст, наличие недвижимости, образование, среднемесячный доход, вернул ли клиент вовремя *кредит*. Задача состоит в том, чтобы на основании перечисленных выше данных (кроме последнего атрибута) определить, стоит ли выдавать *кредит* новому клиенту.

Как мы уже рассматривали в лекции, посвященной задаче классификации, такая задача решается в два этапа: построение классификационной модели и ее использование.

На этапе построения модели, собственно, и строится *дерево* классификации или создается набор неких *правил*. На этапе использования модели построенное *дерево*, или *путь* от его корня к одной из вершин, являющийся набором *правил* для конкретного клиента, используется для ответа на поставленный вопрос "Выдавать ли *кредит*?"

***Правилом* является логическая конструкция, представленная в виде "если : то :".**

Как мы видим, ***внутренние узлы* дерева (возраст, наличие недвижимости, доход и образование) являются атрибутами описанной выше *базы данных*. Эти атрибуты называют прогнозирующими, или *атрибутами расщепления* (splitting *attribute*). *Конечные* узлы дерева, или листы, именуются метками класса, являющимися значениями зависимой категориальной переменной "выдавать" или "не выдавать" *кредит*.**

Каждая *ветвь* дерева, идущая от *внутреннего узла*, отмечена *предикатом расщепления*. Последний может относиться лишь к одному *атрибуту расщепления* данного узла. Характерная особенность *предикатов расщепления*: каждая *запись* использует уникальный *путь* от корня дерева только к одному узлу-решению. Объединенная *информация* об *атрибутах*

расщепления и *предикатах расщепления* в узле называется ***критерием расщепления*** (splitting *criterion*) [33].

На рис. 9.2. изображено одно из возможных деревьев решений для рассматриваемой *базы данных*. Например, *критерий расщепления* "Какое образование?", мог бы иметь два *предиката расщепления* и выглядеть иначе: образование "высшее" и "не высшее". Тогда *дерево* решений имело бы другой вид.

Таким образом, для данной задачи (как и для любой другой) может быть построено множество деревьев решений различного качества, с различной прогнозирующей точностью.

Качество построенного дерева решения весьма зависит от правильного выбора *критерия расщепления*. Над разработкой и усовершенствованием критериев работают многие исследователи.

Метод деревьев решений часто называют "наивным" подходом [34]. Но благодаря целому ряду преимуществ, данный метод является одним из наиболее популярных для решения задач классификации.

Преимущества деревьев решений

Интуитивность деревьев решений. Классификационная модель, представленная в виде дерева решений, является интуитивной и упрощает понимание решаемой задачи. Результат работы алгоритмов конструирования деревьев решений, в отличие, например, от нейронных сетей, представляющих собой "черные ящики", легко интерпретируется пользователем. Это свойство деревьев решений не только важно при отнесении к определенному классу нового объекта, но и полезно при интерпретации модели классификации в целом. *Дерево* решений позволяет понять и объяснить, почему конкретный *объект* относится к тому или иному классу.

Деревья решений дают возможность извлекать *правила* из *базы данных* на **естественном языке**. Пример *правила*: Если Возраст > 35 и Доход > 200, то выдать *кредит*.

Деревья решений позволяют создавать классификационные модели в тех областях, где аналитику достаточно сложно формализовать знания.

Алгоритм конструирования дерева решений **не требует от пользователя выбора входных атрибутов** (независимых переменных). На *вход алгоритма* можно подавать все существующие атрибуты, *алгоритм* сам выберет наиболее значимые среди них, и только они будут использованы для

построения дерева. В сравнении, например, с нейронными сетями, это значительно облегчает пользователю работу, поскольку в нейронных сетях выбор количества входных атрибутов существенно влияет на время обучения.

Точность моделей, созданных при помощи деревьев решений, сопоставима с другими методами построения классификационных моделей (*статистические методы*, нейронные сети).

Разработан ряд **масштабируемых алгоритмов**, которые могут быть использованы для построения деревьев решения на сверхбольших базах данных; *масштабируемость* здесь означает, что с ростом числа примеров или записей *базы данных* время, затрачиваемое на обучение, т.е. построение деревьев решений, растет линейно. Примеры таких алгоритмов: SLIQ, SPRINT.

Быстрый процесс обучения. На построение классификационных моделей при помощи алгоритмов конструирования деревьев решений требуется значительно меньше времени, чем, например, на обучение нейронных сетей.

Большинство алгоритмов конструирования деревьев решений имеют возможность специальной обработки **пропущенных значений**.

Многие классические *статистические методы*, при помощи которых решаются задачи классификации, могут работать только с числовыми данными, в то время как деревья решений работают и с числовыми, и с **категориальными** типами данных.

Многие *статистические методы* являются параметрическими, и *пользователь* должен заранее владеть определенной информацией, например, знать вид модели, иметь гипотезу о виде зависимости между переменными, предполагать, какой вид распределения имеют данные. Деревья решений, в отличие от таких методов, строят непараметрические модели. Таким образом, деревья решений способны решать такие задачи *Data Mining*, в которых отсутствует априорная *информация* о виде зависимости между исследуемыми данными.

Процесс конструирования дерева решений

Напомним, что рассматриваемая нами задача классификации относится к стратегии *обучения с учителем*, иногда называемого индуктивным обучением. В этих случаях все объекты тренировочного набора данных заранее отнесены к одному из предопределенных классов.

Алгоритмы конструирования деревьев решений состоят из этапов "построение" или " *создание* " дерева (*tree* building) и " *сокращение* " дерева (*tree pruning*). В ходе *создания* дерева решаются вопросы выбора *критерия*

расщепления и остановки обучения (если это предусмотрено алгоритмом). В ходе этапа *сокращения* дерева решается вопрос отсечения некоторых его *ветвей*.

Рассмотрим эти вопросы подробней.

Критерий расщепления

Процесс *создания* дерева происходит сверху вниз, т.е. является нисходящим. В ходе процесса алгоритм должен найти такой *критерий расщепления*, иногда также называемый критерием разбиения, чтобы разбить множество на подмножества, которые бы ассоциировались с данным *узлом проверки*. Каждый *узел проверки* должен быть помечен определенным атрибутом. Существует *правило* выбора атрибута: он должен разбивать исходное множество данных таким образом, чтобы объекты подмножеств, получаемых в результате этого разбиения, являлись представителями одного класса или же были максимально приближены к такому разбиению. Последняя фраза означает, что количество объектов из других классов, так называемых "примесей", в каждом классе должно стремиться к минимуму.

Существуют различные *критерии расщепления*.

Наиболее известные - мера энтропии и индекс Gini.

В некоторых методах для выбора *атрибута расщепления* используется так называемая мера информативности подпространств атрибутов, которая основывается на энтропийном подходе и известна под названием "мера информационного выигрыша" (information *gain* measure) или мера энтропии.

Другой *критерий расщепления*, предложенный Брейманом (Breiman) и др., реализован в алгоритме *CART* и называется индексом Gini. При помощи этого индекса атрибут выбирается на основании расстояний между распределениями классов.

Если дано множество T, включающее примеры из n классов, индекс Gini, т.е. gini(T), определяется по формуле:

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

где T - текущий узел, pj - вероятность класса j в узле T, n - количество классов.

Большое дерево не означает, что оно "подходящее"

Чем больше частных случаев описано в дереве решений, тем меньшее количество объектов попадает в каждый частный случай. Такие деревья называют "ветвистыми" или "кустистыми", они состоят из неоправданно большого числа узлов и *ветвей*, исходное множество разбивается на большое число подмножеств, состоящих из очень малого числа объектов. В результате "переполнения" таких деревьев их способность к обобщению уменьшается, и построенные модели не могут давать верные ответы.

В процессе построения дерева, чтобы его размеры не стали чрезмерно большими, используют специальные процедуры, которые позволяют создавать оптимальные деревья, так называемые деревья "подходящих размеров" (Breiman,1984).

Какой размер дерева может считаться оптимальным? Дерево должно быть достаточно сложным, чтобы учитывать информацию из исследуемого набора данных, но одновременно оно должно быть достаточно простым [39]. Другими словами, дерево должно использовать информацию, улучшающую качество модели, и игнорировать ту информацию, которая ее не улучшает.

Тут существует две возможные стратегии. Первая состоит в наращивании дерева до определенного размера в соответствии с параметрами, заданными пользователем. Определение этих параметров может основываться на опыте и интуиции аналитика, а также на некоторых "диагностических сообщениях" системы, конструирующей дерево решений.

Вторая стратегия состоит в использовании набора процедур, определяющих "подходящий размер" дерева, они разработаны Бриманом, Куилендом и др. в 1984 году. Однако, как отмечают авторы, нельзя сказать, что эти процедуры доступны начинающему пользователю.

Процедуры, которые используют для предотвращения *создания* чрезмерно больших деревьев, включают: *сокращение* дерева путем отсечения *ветвей* ; использование *правил* остановки обучения.

Следует отметить, что не все алгоритмы при конструировании дерева работают по одной схеме. Некоторые алгоритмы включают два отдельных последовательных этапа: построение дерева и его *сокращение* ; другие чередуют эти этапы в процессе своей работы для предотвращения наращивания *внутренних узлов*.

Остановка построения дерева

Рассмотрим *правило* остановки. Оно должно определить, является ли рассматриваемый узел *внутренним узлом*, при этом он будет разбиваться дальше, или же он является *конечным узлом*, т.е. *узлом решением*.

Остановка - такой момент в процессе построения дерева, когда следует прекратить дальнейшие ветвления.

Один из вариантов *правил* остановки - "ранняя остановка" (prepruning), она определяет целесообразность разбиения узла. Преимущество использования такого варианта - уменьшение времени на обучение модели. Однако здесь возникает риск снижения точности классификации. Поэтому рекомендуется "вместо остановки использовать отсечение" (Breiman, 1984).

Второй вариант остановки обучения - ограничение глубины дерева. В этом случае построение заканчивается, если достигнута заданная глубина.

Еще один вариант остановки - задание минимального количества примеров, которые будут содержаться в *конечных узлах* дерева. При этом варианте ветвления продолжаются до того момента, пока все *конечные узлы* дерева не будут чистыми или будут содержать не более чем заданное число объектов.

Существует еще ряд *правил*, но следует отметить, что ни одно из них не имеет большой практической ценности, а некоторые применимы лишь в отдельных случаях [35].

Сокращение дерева или отсечение ветвей

Решением проблемы слишком ветвистого дерева является его *сокращение* путем отсечения (*pruning*) некоторых *ветвей*.

Качество классификационной модели, построенной при помощи дерева решений, характеризуется двумя основными признаками: точностью распознавания и ошибкой.

Точность распознавания рассчитывается как отношение объектов, правильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении.

Ошибка рассчитывается как отношение объектов, неправильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении.

Отсечение *ветвей* или замену некоторых *ветвей* поддеревом следует проводить там, где эта процедура не приводит к возрастанию ошибки. Процесс проходит снизу вверх, т.е. является восходящим. Это более популярная

процедура, чем использование *правил* остановки. Деревья, получаемые после отсечения некоторых *ветвей*, называют усеченными.

Если такое усеченное дерево все еще не является интуитивным и сложно для понимания, используют извлечение *правил*, которые объединяют в наборы для описания классов. Каждый путь от корня дерева до его вершины или листа дает одно *правило*. Условиями *правила* являются проверки на *внутренних узлах* дерева.

Алгоритмы

На сегодняшний день существует большое число алгоритмов, реализующих деревья решений: *CART*, C4.5, CHAID, CN2, NewId, ITrule и другие.

Алгоритм CART

Алгоритм *CART* (*Classification* and *Regression* Tree), как видно из названия, решает задачи классификации и регрессии. Он разработан в 1974-1984 годах четырьмя профессорами статистики - *Leo* Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley) и Richard Olshen (Stanford).

Атрибуты набора данных могут иметь как дискретное, так и числовое значение.

Алгоритм *CART* предназначен для построения *бинарного дерева решений*. Бинарные деревья также называют двоичными. Пример такого дерева рассматривался в начале лекции.

Другие особенности алгоритма *CART*:

- функция оценки качества разбиения;
- механизм отсечения дерева;
- алгоритм обработки пропущенных значений;
- построение деревьев регрессии.

Каждый узел бинарного дерева при разбиении имеет только двух потомков, называемых дочерними ветвями. Дальнейшее разделение ветви зависит от того, много ли исходных данных описывает данная *ветвь*. На каждом шаге построения дерева *правило*, формируемое в узле, делит заданное множество примеров на две части. Правая его часть (*ветвь* right) - это та часть множества, в которой *правило* выполняется; левая (*ветвь* left) - та, для которой *правило* не выполняется.

Функция оценки качества разбиения, которая используется для выбора оптимального *правила*, - индекс Gini - был описан выше. Отметим, что данная оценочная функция основана на идее уменьшения неопределенности в узле.

Допустим, есть узел, и он разбит на два класса. Максимальная неопределенность в узле будет достигнута при разбиении его на два подмножества по 50 примеров, а максимальная определенность - при разбиении на 100 и 0 примеров.

Правила разбиения. Напомним, что алгоритм *CART* работает с числовыми и категориальными атрибутами. В каждом узле разбиение может идти только по одному атрибуту. Если атрибут является числовым, то во *внутреннем узле* формируется *правило* вида xi <= c, Значение c в большинстве случаев выбирается как среднее арифметическое двух соседних упорядоченных значений переменной xi обучающего набора данных. Если же атрибут относится к категориальному типу, то во *внутреннем узле* формируется *правило* xi V(xi), где V(xi) - некоторое непустое подмножество множества значений переменной xi в обучающем наборе данных.

Механизм отсечения. Этим механизмом, имеющим название *minimal* cost-complexity tree *pruning*, алгоритм *CART* принципиально отличается от других алгоритмов конструирования деревьев решений. В рассматриваемом алгоритме отсечение - это некий компромисс между получением дерева "подходящего размера" и получением наиболее точной оценки классификации. Метод заключается в получении последовательности уменьшающихся деревьев, но деревья рассматриваются не все, а только "лучшие представители".

Перекрестная проверка (V-*fold* cross-validation) является наиболее сложной и одновременно оригинальной частью алгоритма *CART*. Она представляет собой путь выбора окончательного дерева, при условии, что набор данных имеет небольшой объем или же записи набора данных настолько специфические, что разделить набор на обучающую и тестовую выборку не представляется возможным.

Итак, основные характеристики алгоритма *CART*: бинарное расщепление, *критерий расщепления* - индекс Gini, алгоритмы *minimal* cost-complexity tree *pruning* и V-*fold* cross-validation, принцип "вырастить дерево, а затем сократить", высокая скорость построения, обработка пропущенных значений.

Алгоритм C4.5

Алгоритм C4.5 строит дерево решений с неограниченным количеством *ветвей* у узла. Данный алгоритм может работать только с дискретным зависимым атрибутом и поэтому может решать только задачи классификации. C4.5 считается одним из самых известных и широко используемых алгоритмов построения деревьев классификации.

Для работы алгоритма C4.5 необходимо соблюдение следующих требований:

- Каждая запись набора данных должна быть ассоциирована с одним из предопределенных классов, т.е. один из атрибутов набора данных должен являться меткой класса.
- Классы должны быть дискретными. Каждый пример должен однозначно относиться к одному из классов.
- Количество классов должно быть значительно меньше количества записей в исследуемом наборе данных.

Последняя версия алгоритма - алгоритм C4.8 - реализована в инструменте Weka как J4.8 (Java). Коммерческая реализация метода: C5.0, разработчик RuleQuest, Австралия.

Алгоритм C4.5 медленно работает на сверхбольших и зашумленных наборах данных.

Мы рассмотрели два известных алгоритма построения деревьев решений *CART* и C4.5. Оба алгоритма являются робастными, т.е. устойчивыми к шумам и выбросам данных.

Алгоритмы построения деревьев решений различаются следующими характеристиками:

- вид расщепления - бинарное (binary), множественное (multi-way)
- *критерии расщепления* - энтропия, Gini, другие
- возможность обработки пропущенных значений
- процедура сокращения *ветвей* или отсечения
- возможности извлечения *правил* из деревьев.

Ни один алгоритм построения дерева нельзя априори считать наилучшим или совершенным, подтверждение целесообразности использования конкретного алгоритма должно быть проверено и подтверждено экспериментом.

Разработка новых масштабируемых алгоритмов

Наиболее серьезное требование, которое сейчас предъявляется к алгоритмам конструирования деревьев решений - это масштабируемость, т.е. алгоритм должен обладать масштабируемым методом доступа к данным.

Разработан ряд новых масштабируемых алгоритмов, среди них - алгоритм Sprint, предложенный Джоном Шафером и его коллегами [36]. Sprint, являющийся масштабируемым вариантом рассмотренного в лекции алгоритма *CART*, предъявляет минимальные требования к объему оперативной памяти.

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^{C} f_i(1-f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^{C} -f_i log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Variance	Regression	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$.

The *information gain* is the difference between the parent node impurity and the weighted sum of the two child node impurities. Assuming that a split s partitions the dataset D of size N into two datasets D_{left} and D_{right} of sizes N_{left} and N_{right}, respectively, the information gain is:

$$IG(D,s) = Impurity(D) - \frac{N_{left}}{N} Impurity(D_{left}) - \frac{N_{right}}{N} Impurity(D_{right})$$