Методы машинного обучения

Лекция 8

Деревья решений

Деревья решений

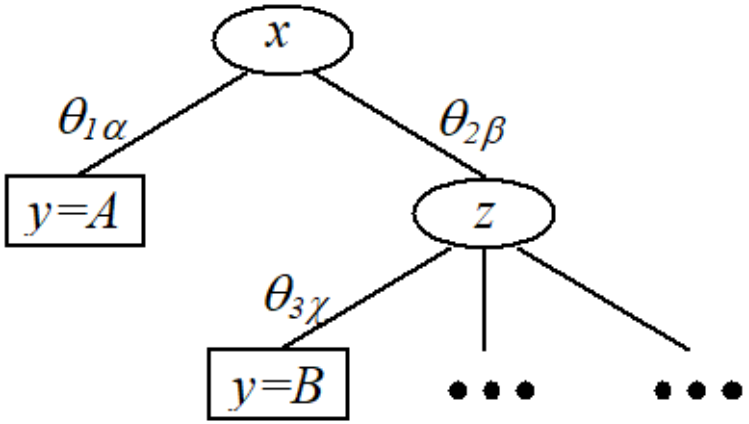
Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукций «если… то…».

Если (условие 1) $\wedge$ (условие 2) $\wedge$ … $\wedge$ (условие N) то (значение вершины вывода)



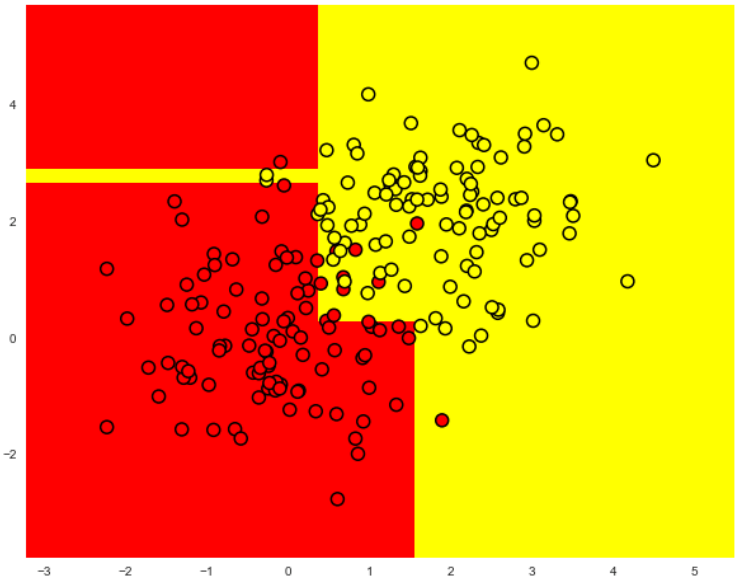
$$\theta_1(x,\alpha) \rightarrow (y = A)$$

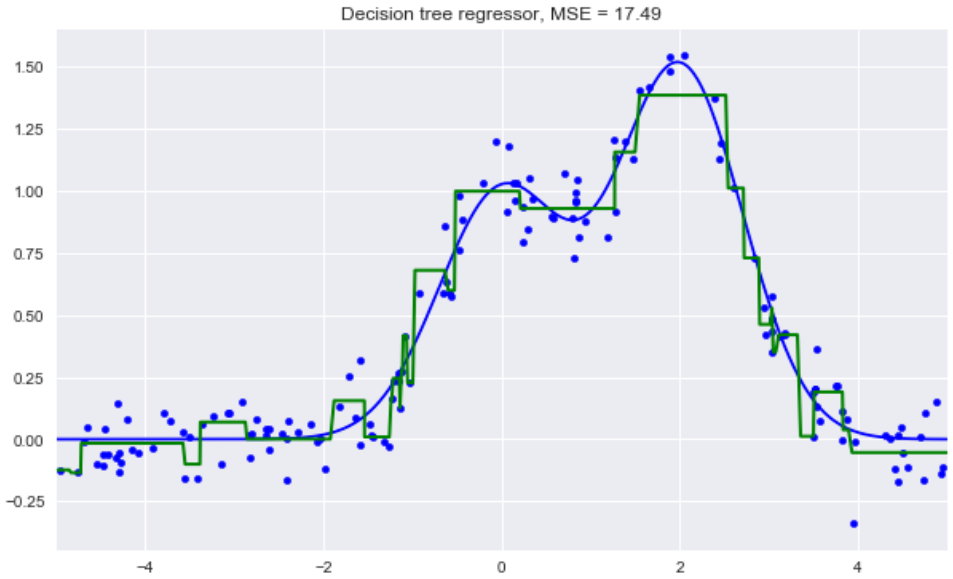
$$\theta_2(x,\beta) \cap \theta_3(z,\chi) \rightarrow (y = B)$$

Задачи

Сфера применения — поддержка процессов принятия управленческих решений, используемая в статистике, анализе данных и машинном обучении. Задачами, решаемыми с помощью данного аппарата, являются:

- **Классификация** — отнесение объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
- **Регрессия (численное предсказание)** — предсказание числового значения независимой переменной для заданного входного вектора.
- **Описание объектов** — набор правил в дереве решений позволяет компактно описывать объекты. Поэтому вместо сложных структур, описывающих объекты, можно хранить деревья решений.





Основные понятия

Название	Описание
Объект	Пример, шаблон, наблюдение
Атрибут	Признак, независимая переменная, свойство
Целевая переменная	Зависимая переменная, метка класса
Узел	Внутренний узел дерева, узел проверки. В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.
Корневой узел	Начальный узел дерева решений
Лист	Конечный узел дерева, узел решения, терминальный узел - определяет решение для каждого попавшего в него примера.
Решающее правило	Условие в узле, проверка

Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.

Алгоритмы обучения деревья решений

В настоящее время разработано значительное число алгоритмов обучения деревья решений: CLS, ID3, C4.5, CART, NewId, ITrule, CHAID, CN2 и т.д. Наибольшее распространение и популярность получили следующие:

- **ID3 (Iterative Dichotomizer 3)** — позволяет работать только с дискретной целевой переменной, т.е. строит только классифицирующие деревья решений.
- **C4.5** — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов (по версии издания Springer Science в 2008 году алгоритм занял 1-е место в топ-10 наиболее популярных алгоритмов Data Mining).
- **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений способный использовать как дискретную, так и непрерывную целевую переменную, т.е. решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

Процесс построения

Деревья решений являются моделями, строящимися на основе обучения с учителем, следовательно, в обучающем множестве для примеров должно быть задано целевое значение.

- Целевая переменная дискретная -> дерево классификации.
- Целевая переменная непрерывная -> дерево регрессии.

Процесс построения деревьев решений заключается в рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

Объявление узла листом может произойти:

- содержит единственный объект, или объекты только одного класса;
- достижение некоторого условия остановки (минимально допустимое число примеров в узле или максимальная глубина дерева).

Алгоритмы построения деревьев решений относят к категории **«жадных алгоритмов»**. Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к оптимальному итоговому решению. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

Принцип «разделяй и властвуй»

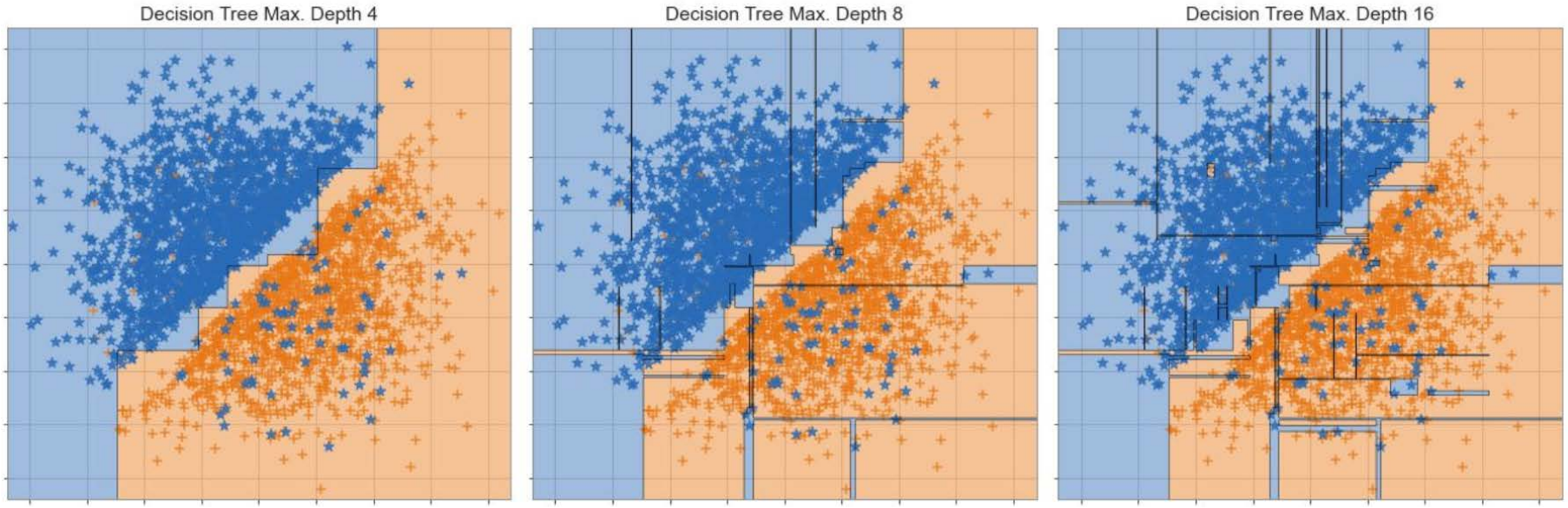
Пусть задано обучающее множество S, содержащее n примеров, для каждого из которых задана метка класса $C_i (i = 1..k)$, и m атрибутов $A_j (j = 1..m)$, которые, как предполагается, определяют принадлежность объекта к тому или иному классу. Тогда возможны три случая:

- Все примеры множества S имеют одинаковую метку класса C_i (т.е. все обучающие примеры относятся только к одному классу). Дерево решений в этом случае будет представлять собой лист, ассоциированный с классом C_i.
- Множество S не содержит примеров, т.е. является пустым. Для него тоже будет создан лист. Применять правило, чтобы создать узел, к пустому множеству бессмысленно.
- Множество S содержит обучающие примеры всех классов C_k. Требуется разбить множество S на подмножества, ассоциированные с классами. Для этого:
  - выбирается один из атрибутов A_j множества S, который содержит два и более уникальных значения $(a_1, a_2, \dots, a_p)$, где p — число уникальных значений признака.
  - множество S разбивается на p подмножеств $(S_1, S_2, \dots, S_p)$, каждое из которых включает примеры, содержащие соответствующее значение атрибута.
  - выбирается следующий атрибут и разбиение повторяется.
  - процедура рекурсивно повторяется до тех пор, пока все примеры в результирующих подмножествах не окажутся одного класса.

При использовании данной методики, построение дерева решений будет происходить сверху вниз (от корневого узла к листьям).

Проблемы основных этапов построения

- Выбор атрибута, по которому будет производиться разбиение в данном узле (атрибут разбиения).
- Выбор критерия остановки обучения.
- Выбор метода отсечения ветвей (упрощения).
- Оценка точности построенного дерева.



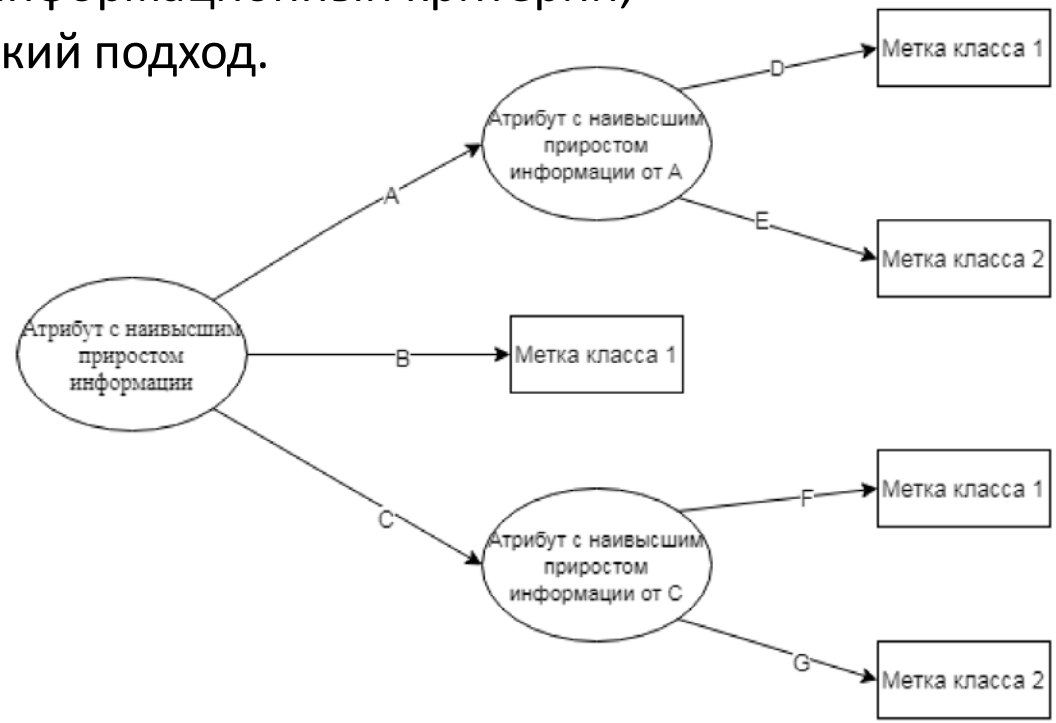
Выбор атрибута разбиения

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут.

Общее правило: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Наиболее популярные критерии:

- Теоретико-информационный критерий;
- Статистический подход.



Теоретико-информационный критерий

Критерий основан на понятиях теории информации (информационной энтропии):

$$H(S,C) = -\sum_{i=1}^{n} \frac{N_i}{N}\ log\left(\frac{N_i}{N}\right)$$

где *n* — число классов в исходном подмножестве, N_i — число примеров *i*-го класса, *N* — общее число примеров в подмножестве.

Энтропия может рассматриваться как мера неоднородности подмножества по представленным в нём классам. Если классы представлены в равных долях, то неопределённость классификации и энтропия максимальны. Если все примеры в узле относятся к одному классу, т.е. $N = N_i$, логарифм от единицы обращает энтропию в ноль.

Лучшим атрибутом разбиения A_j будет тот, который обеспечит максимальное снижение энтропии результирующего подмножества относительно родительского. На практике говорят не об энтропии, а об обратной величине, которая называется информацией. Тогда лучшим атрибутом разбиения будет тот, который обеспечит максимальный прирост информации результирующего узла относительно исходного:

$$Gain(S,A) = Info(S) - Info(S_A)$$

где $Info(S)$ — информация, связанная с подмножеством S до разбиения, $Info(S_A)$ — информация, связанная с подмножеством, полученным при разбиении по атрибуту A.

$$Gain(S,A) = H(S,C) - \sum_{j=1}^{p} \frac{|S_j|}{|S|}\ H(S_j,C)$$

где S_j-множество элементов S, на которых атрибут A имеет значение a_j.

Таким образом, задача выбора атрибута разбиения в узле заключается в максимизации величины $Gain(S,A)$, называемой приростом информации (gain — прирост, увеличение).

Этот критерий впервые был применён в алгоритме ID3, а затем в C4.5 и других алгоритмах.

Непрерывный атрибут

Если атрибут, по которому производится разбиение, непрерывный, то его сначала преобразуют в дискретный вид, например, с помощью операции квантования. Затем значения ранжируются и ищется среднее, которое используется для выбора порога. Все примеры, имеющие значения атрибута выше порогового, помещаются в один узел-потомок, а те, которые ниже — в другой.

Выбор порога. Пусть числовой атрибут X принимает конечное множество значений $(x_1, x_2, \dots, x_p)$. Упорядочив примеры по возрастанию значений атрибута, получим, что любое значение, лежащее между x_i и x_{i+1} делит все примеры на два подмножества. Первое подмножество будет содержать значения атрибута $(x_1, x_2, \dots, x_i)$, а второе — $(x_{i+1}, x_{i+2}, \dots, x_p)$. Тогда в качестве порога можно выбрать среднее: $T_i = \frac{x_i + x_{i+1}}{2}$

Задача нахождения порога сводится к рассмотрению $n - 1$ потенциальных пороговых значений $T_1, T_2, \dots, T_{n-1}$.

Последовательно применяя формулы для расчета теоретико-информационного критерия (информационной энтропии) ко всем потенциальным порогам, выбираем то, которое даёт максимальное значение по критерию:

$$Gain(S, A) = Info(S) - Info(S_A).$$

Затем, полученное значение сравнивается со значениями, рассчитанными для других атрибутов. Если это значение окажется наибольшим из всех атрибутов, то оно выбирается в качестве порога для проверки.

Следует отметить, что все числовые тесты являются бинарными, т.е. делят узел дерева на две ветви (два потомка).

Статистический подход

В основе статистического подхода лежит использование индекса Джини (назван в честь итальянского статистика и экономиста Коррадо Джини). Статистический смысл данного показателя в том, что он показывает — насколько часто случайно выбранный пример обучающего множества будет распознан неправильно, при условии, что целевые значения в этом множестве были взяты из определённого статистического распределения.

Таким образом индекс Джини фактически показывает расстояние между двумя распределениями — распределением целевых значений, и распределением предсказаний модели. Очевидно, что чем меньше данное расстояние, тем лучше работает модель.

Индекс Джини может быть рассчитан по формуле:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

где S — результирующее множество, k — число классов в нём, p_i^2 — вероятность i-го класса (относительная частота примеров соответствующего класса). Данный показатель меняется от 0 до 1 (равен 0, если все примеры S относятся к одному классу; равен 1, когда классы представлены в равных пропорциях). Лучшим будет то разбиение, для которого значение индекса Джини будет минимальным.

Соответственно, для выборки S, атрибута A (m значений) и целевого свойства C (k значений), индекс для выбора разделяющего атрибута вычисляется следующим образом:

$$Gini(S, A, C) = Gini(S, A) - \sum_{j=1}^{p} \frac{|S_i|}{|S|} Gini(S_j, A) \rightarrow max$$

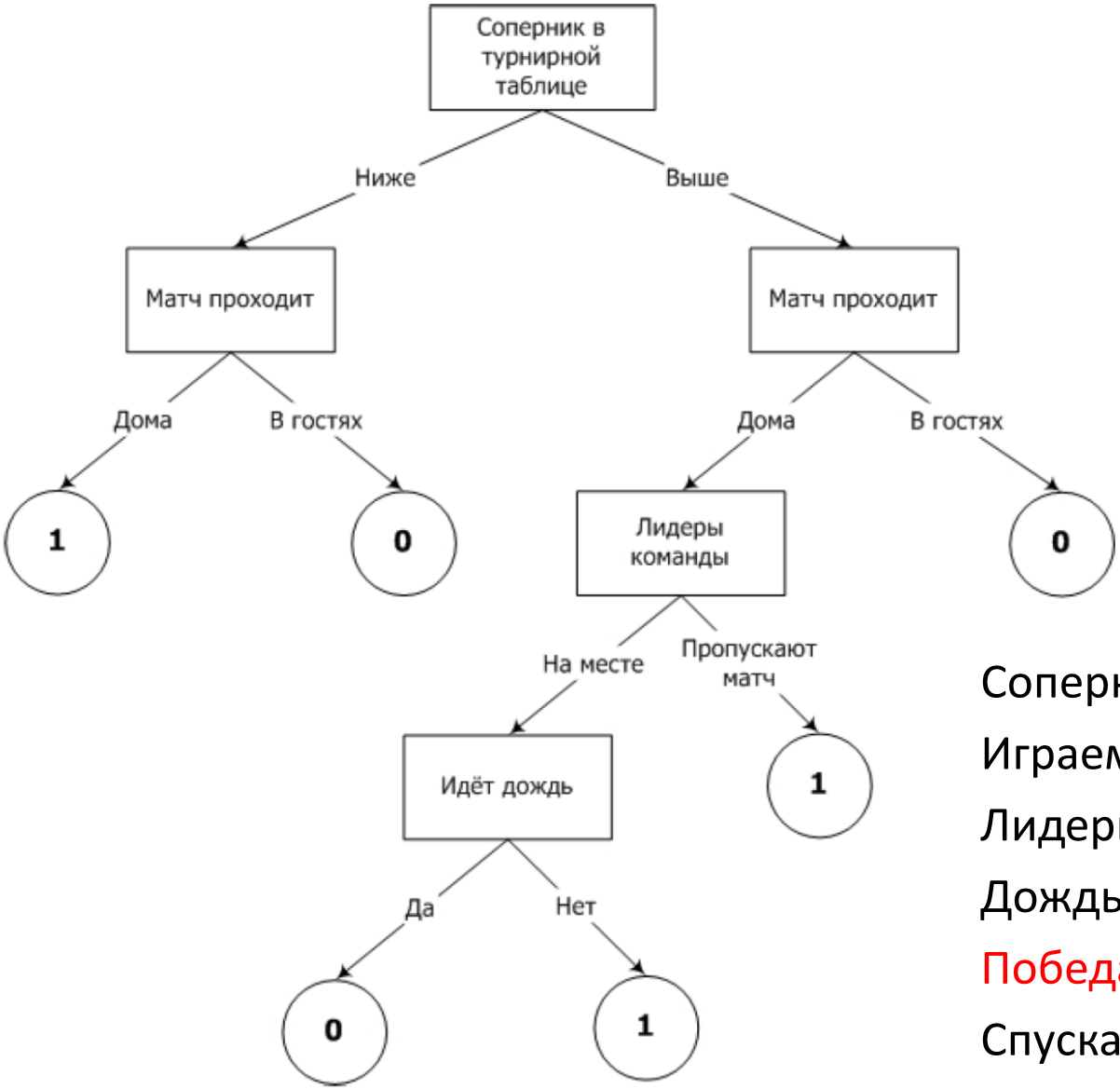
Пример: Футбольная команда

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	*???*

Атрибуты: Соперник (выше/ниже по турнирной таблице), Играем (домашняя или выездная игра), Лидеры (Лидеры команды: играют/не играют), Дождь (наличие дождя: Да/Нет).

Целевой показатель: Победа (Да/Нет).

Пример: построенное дерево



Соперник = Ниже

Играем = В гостях

Лидеры = На месте

Дождь = Нет

Победа = ?

Спускаясь по дереву получаем, что команда должна проиграть.

Пример: Строим дерево

Воспользуемся Теоретико-информационным критерием (энтропией и приростом информации).

В нашем примере из 7 матчей команда три проиграла и четыре выиграла. Поэтому исходная энтропия:

$$H(S, \text{Победа}) = -\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7} \approx 0.9852.$$

Прирост информации:

$$\text{Gain}(S, \text{Соперник}) =$$

$$= H(S, \text{Победа}) - \frac{4}{7}H(S_{\text{выше}}, \text{Победа}) - \frac{3}{7}H(S_{\text{ниже}}, \text{Победа}) \approx$$

$$\approx 0.9852 - \frac{4}{7}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{3}{7}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.0202.$$

Явно не самый удачный критерий для корня дерева.

$$\text{Gain}(S, \text{Играем}) \approx 0.4696.$$

$$\text{Gain}(S, \text{Лидеры}) \approx 0.1281.$$

$$\text{Gain}(S, \text{Дождь}) \approx 0.1281.$$

Проблема критерия прироста информации

Можно заметить, что критерий разбиения на основе прироста информации отдаёт предпочтение атрибутам, которые содержат большее число уникальных значений. Это связано с тем, что при этом создается большое количество узлов-потомков, дающее в итоге более равномерное распределение классов и, соответственно, меньшую энтропию разбиения. В предельном случае, если все значения атрибута будут уникальными, то для каждого примера будет создано отдельное правило и отдельный лист с единственным классом.

Как следствие, разбиение даст нулевую энтропию:

$$Info(S, A) = 0$$

и максимальный прирост информации.

Это оптимально с «точки зрения» алгоритма, но абсолютно бессмысленно с практический точки зрения, поскольку:

- значимость правил будет чрезвычайно низкая, т.к. правило действует только для конкретного объекта;
- дерево решений окажется очень сложным для понимания;
- дерево решений получится переобученным, т.е. не будет обладать обобщающей способностью.

Пример: Проблема критерия прироста информации

Пусть в таблице игр были дополнительно записаны даты матчей. Тогда прирост информации:

$$\text{Gain}(S, \text{Дата}) = H(S, \text{Победа}) -$$
$$- \sum_{i=1}^{n} \frac{1}{n} H(S_{\text{Дата}=i}, \text{Победа}) = H(S, \text{Победа})$$

И равен энтропии победы в исходном множестве. Это происходит потому, что в каждой из веток только один случай, а энтропия в каждой ветке равна нулю.

Прирост информации - максимальный из возможных но полученное дерево абсолютно бесполезно.

Решение данной проблемы достаточно очевидно — чтобы алгоритм при выборе разбиения не отдавал предпочтение атрибутам с большим числом уникальных значений, следует ввести некоторый штраф, накладываемый на атрибут за количество созданных с его помощью узлов-потомков.

Split-Info и Gain-Ratio

Штраф может быть представлен в виде некоторого коэффициента для значения прироста информации.

Введём в рассмотрение показатель:

$$SplitInfo(S, A) = -\sum_{j=1}^{p} \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}$$

который представляет собой оценку потенциальной информации, созданной при разбиении множества S на p подмножеств S_j. Тогда критерий прироста информации с учётом показателя $SplitInfo(S, A)$ будет иметь вид:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInfo(S, A)}$$

Новый критерий позволяет оценить долю информации, полученной при разбиении, которая является «полезной», то есть способствует улучшению классификации. Применение данного показателя, как правило, приводит к выбору более удачного атрибута, чем использование обычного критерия прироста информации.

Смысл этой модификации достаточно прост. $|S_j|/|S|$ — отношение числа примеров в подмножестве, полученном в результате разбиения, к числу примеров в родительском множестве $|S|$. Если в результате разбиения получается большое число подмножеств с небольшим числом примеров, что характерно для переобучения, то показатель $SplitInfo$ растет. Поскольку он стоит в знаменателе выражения, то $GainRatio$ есть обычный прирост информации, «оштрафованный» с помощью $SplitInfo$.

Пример: Split-Info и Gain-Ratio

У атрибута «Дата»:

$$\mathrm{SplitInfo}(S, \text{Дата}) = -\sum_{i=1}^{7} \frac{1}{7} \log_2 \frac{1}{7} \approx 2.80735\ldots$$

$$\mathrm{GainRatio}(S, \text{Дата}) = \frac{\mathrm{Gain}(S, \text{Дата})}{\mathrm{SplitInfo}(S, \text{Дата})} \approx 0.350935\ldots$$

А для атрибута «Играем», показывающего, где проходит матч:

$$\mathrm{SplitInfo}(S, \text{Играем}) = -\frac{5}{7} \log_2 \frac{5}{7} - \frac{2}{7} \log_2 \frac{2}{7} \approx 0.86312\ldots$$

$$\mathrm{GainRatio}(S, \text{Играем}) = \frac{\mathrm{Gain}(S, \text{Играем})}{\mathrm{SplitInfo}(S, \text{Играем})} \approx 0.5452\ldots$$

Обработка пропусков в данных - Критерий

Пусть S — множество обучающих примеров. Обозначим U - количество примеров, у которых не определено значение атрибута A. Изменим слагаемые из критерия $Gain(S, A) = Info(S) - Info(S_A)$ таким образом, чтобы учитывать только те примеры, для которых значения атрибута существуют.

$$Info(S) = H(S, C) = -\sum_{i=1}^{k} \frac{N(C_i, S)}{N(S) - U} \log\left(\frac{N(C_i, S)}{N(S) - U}\right)$$

$$Info(S_A) = \sum_{j=1}^{p} \frac{N(S_j)}{N(S) - U} H(S_j, C)$$

где k — число классов в исходном подмножестве, $N(C_i, S)$ — число примеров i-го класса в множестве S (учитываются примеры, не содержащие пропусков в значениях атрибута A), $N(S)$ — общее число примеров в множестве S, p — число уникальных значений признака атрибута A.

Критерий $Gain(S, A)$ может быть преобразован к виду:

$$Gain(S, A) = \frac{N(S) - U}{N(S)}\big(Info(S) - Info(S_A)\big).$$

Обработка пропусков в данных - Разбиение

Пусть разбиение, формирующее p выходов $(O_1, O_2, \dots, O_p)$, использует атрибут A, выбранный по критерию $Gain(S, A)$. Каждый пример из множества S, порождающий выход O_j, может быть ассоциирован с подмножеством S_j с некоторой вероятностью, выраженной через веса:

- Если пример содержит значение атрибута A, то вес примера равен 1.
- В противном случае пример ассоциируется со всеми подмножествами $S_1, S_2, \dots, S_p$ с соответствующими весами:

$$\frac{N(S_1)}{N(S) - U}, \frac{N(S_2)}{N(S) - U}, \dots, \frac{N(S_p)}{N(S) - U}$$

Несложно убедиться, что

$$\sum_{j=1}^{p} \frac{N(S_j)}{N(S) - U} = 1$$

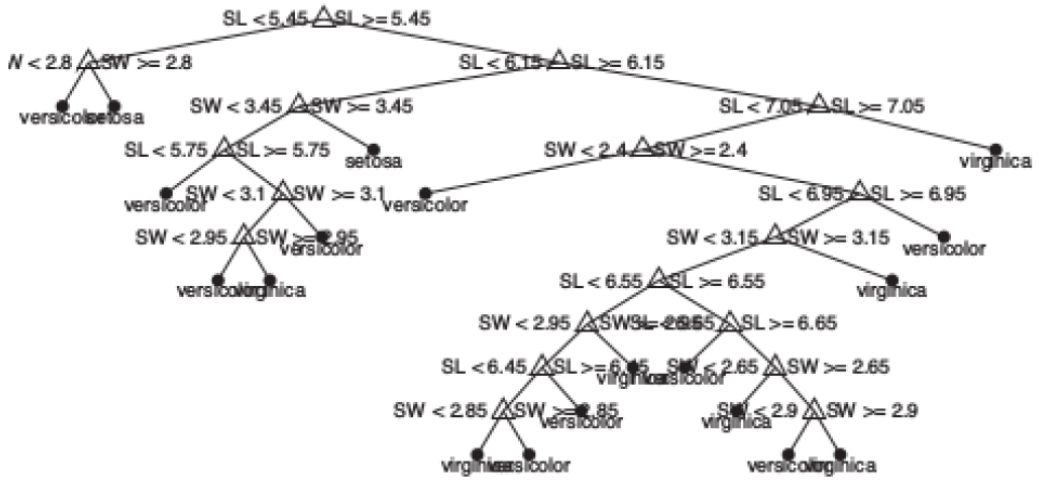
Таким образом, этот подход можно сформулировать так: предполагается, что пропущенные значения по атрибуту распределены пропорционально частоте появления существующих значений.

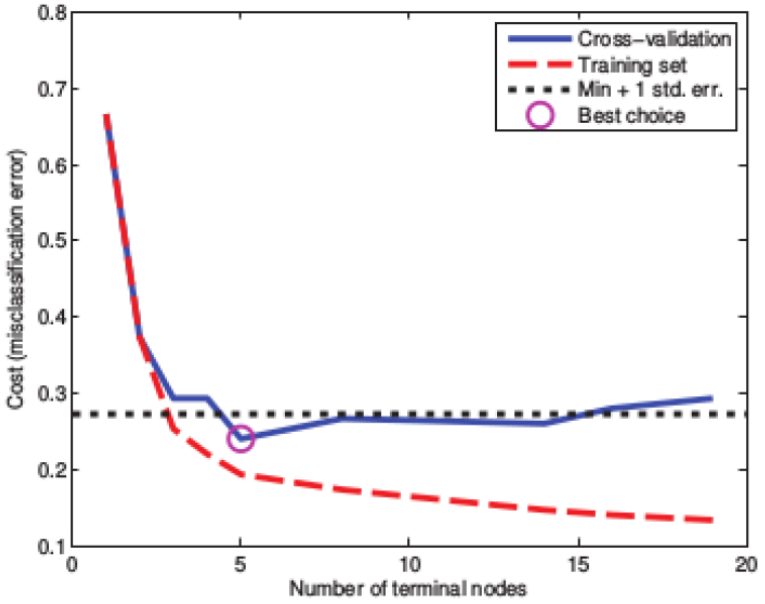
Если при классификации новых объектов на каком-то узле выясняется, что значение соответствующего атрибута пропущено, то алгоритм исследует все возможные пути вниз по дереву и определяет с какой вероятностью пример относится к различным классам.

Эффект переобучения

Теоретически, алгоритм обучения дерева решений будет работать до тех пор, пока в результате не будут получены абсолютно «чистые» подмножества, в каждом из которых будут примеры одного класса. Возможна ситуация, что для каждого примера будет создан отдельный лист. Такое дерево окажется переобученным, т.к. каждому примеру будет соответствовать свой уникальный путь в дереве, а следовательно, и набор правил.

Переобучение — точное распознавание примеров, участвующих в обучении и полная несостоятельность на новых данных. Переобученные деревья имеют очень сложную структуру, и их сложно интерпретировать.





Критерий остановки алгоритма

Подходы для борьбы с переобучением:

- **Ранняя остановка** — алгоритм будет остановлен, как только будет достигнуто заданное значение некоторого критерия (например, процентной доли правильно распознанных примеров). Преимущество - снижение времени обучения. Главный недостаток – снижение точности.
- **Ограничение глубины дерева** — задание максимального числа разбиений в ветвях. Данный метод также ведёт к снижению точности дерева.
- **Задание минимально допустимого** числа примеров в узле — запрет алгоритму создавать узлы с числом примеров меньше заданного значения (например, 5).

Все перечисленные подходы являются эвристическими, т.е. не гарантируют лучшего результата или вообще работают только в каких-то частных случаях.

Борьба с переобучением
Отсечение ветвей - pruning

Представляет интерес подход, альтернативный ранней остановке — построить все возможные деревья и выбрать то из них, которое при разумной глубине обеспечивает приемлемый уровень ошибки распознавания, т.е. найти наиболее выгодный баланс между сложностью и точностью дерева.

К сожалению, это задача относится к классу NP-полных задач, что было показано Л. Хайфилем (L. Hyafill) и Р. Ривестом (R. Rivest), и, как известно, этот класс задач не имеет эффективных методов решения.

Альтернативным подходом является так называемое **отсечение ветвей (pruning)**. Он содержит следующие шаги:

- Построить полное дерево (чтобы все листья содержали примеры одного класса).
- Определить два показателя: относительную точность модели — отношение числа правильно распознанных примеров к общему числу примеров, и абсолютную ошибку — число неправильно классифицированных примеров.
- Удалить из дерева листья и узлы, отсечение которых не приведёт к значимому уменьшению точности модели или увеличению ошибки.

Отсечение ветвей, очевидно, производится в направлении, противоположном направлению роста дерева, т.е. снизу вверх, путём последовательного преобразования узлов в листья. Преимуществом отсечения ветвей по сравнению с ранней остановкой является возможность поиска оптимального соотношения между точностью и понятностью дерева. Недостатком является большее время обучения из-за необходимости сначала построить полное дерево.

Извлечение правил

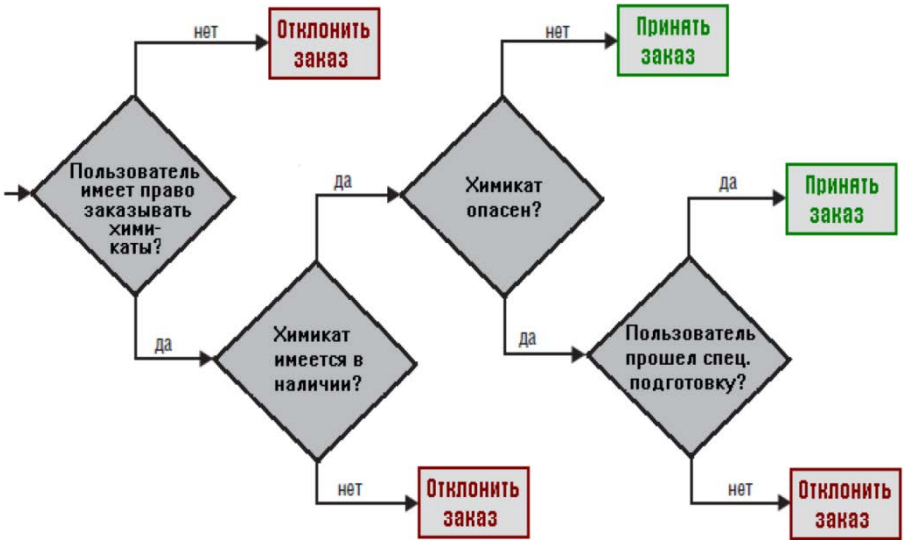
Для упрощения интерпретации результата может оказаться полезным извлечь из дерева **решающие правила** и организовать их в наборы, описывающие классов.

Для извлечения правил нужно отследить все пути от корневого узла к листьям дерева. Каждый такой путь даст правило, состоящее из множества условий, представляющих собой проверку в каждом узле пути.

Визуализация сложных деревьев решений в виде решающих правил вместо иерархической структуры из узлов и листьев может оказаться более удобной для визуального восприятия.

Если (условие 1) ∧ (условие 2) ∧ … ∧ (условие N)
то (значение вершины вывода)

Представление решающих правил в **табличной форме** с возможностью фильтрации по значениям атрибутов и целевой переменной.



Номер требования					
Условие	1	2	3	4	5
Пользователь имеет право заказывать химикаты?	нет	да	да	да	да
Химикат в наличии?	—	нет	да	да	да
Химикат опасен?	—	—	нет	да	да
Пользователь прошел спец.подготовку?	—	—	—	нет	да
Действие					
Принять заказ			X		X
Отклонить заказ	X	X		X	

Алгоритм CART (CART algorithm)

Может работать как с дискретной, так и с непрерывной выходной переменной, т.е. решать задачи и классификации, и регрессии. Строит бинарные деревья решений, которые содержат только два потомка в каждом узле.

Алгоритм реализует обучение с учителем и использует в качестве критерия для выбора разбиений в узлах индекс Джини (*Gini impurity index*). В процессе роста дерева алгоритм CART проводит для каждого узла полный перебор всех атрибутов, на основе которых может быть построено разбиение, и выбирает тот, который максимизирует значение индекса Джини. Полученные дочерние узлы должны быть максимально однородными.

Если атрибут X, по которому производится разбиение, является номинальным с i категориями, то для него существует $2(i-1)$ возможных разбиения, а если порядковым или непрерывным с K различными значениями, существует $K-1$ различных разбиений по X.

Дерево строится, начиная с корневого узла, путем итеративного использования следующих шагов в каждом узле:

- Для каждого атрибута ищется лучшее разбиение (в смысле однородности результирующих подмножеств).
- Среди всех разбиений, найденных на предыдущем шаге, выбирается то, для которого критерий разбиения наибольший.
- Узел разбивается с использованием лучшего разбиения, найденного на шаге 2, если не выполнено условие остановки.

Основные достоинства деревьев решений

- быстрый процесс обучения;
- генерация правил в областях, где эксперту трудно формализовать свои знания;
- извлечение правил на естественном языке;
- интуитивно понятная классификационная модель;
- высокая точность предсказания, сопоставимая с другими методами анализа данных (статистика, нейронные сети);
- построение непараметрических моделей.

Основные недостатки деревьев решений

- Деревья очень чувствительны к шумам во входных данных, вся модель может кардинально измениться, если немного изменится обучающая выборка (например, если убрать один из признаков или добавить несколько объектов), поэтому и правила классификации могут сильно изменяться;
- Нестабильность. Небольшие изменения в данных могут существенно изменять построенное дерево решений. С этой проблемой борются с помощью ансамблей деревьев решений (рассмотрим далее);
- Разделяющая граница, построенная деревом решений, имеет свои ограничения (состоит из гиперплоскостей, перпендикулярных какой-то из координатной оси), и на практике дерево решений по качеству классификации уступает некоторым другим методам;
- Необходимость отсекать ветви дерева (pruning) или устанавливать минимальное число элементов в листьях дерева или максимальную глубину дерева для борьбы с переобучением;
- Проблема поиска оптимального дерева решений (минимального по размеру и способного без ошибок классифицировать выборку) NP-полна, поэтому на практике используются эвристики типа жадного поиска признака с максимальным приростом информации, которые не гарантируют нахождения глобально оптимального дерева;
- Сложно поддерживаются пропуски в данных. Friedman оценил, что на поддержку пропусков в данных ушло около 50% кода CART (Classification And Regression Trees);
- Модель умеет только интерполировать, но не экстраполировать (это же верно и для леса и бустинга на деревьях).

Спасибо за внимание