- Лекция 2

Классификация

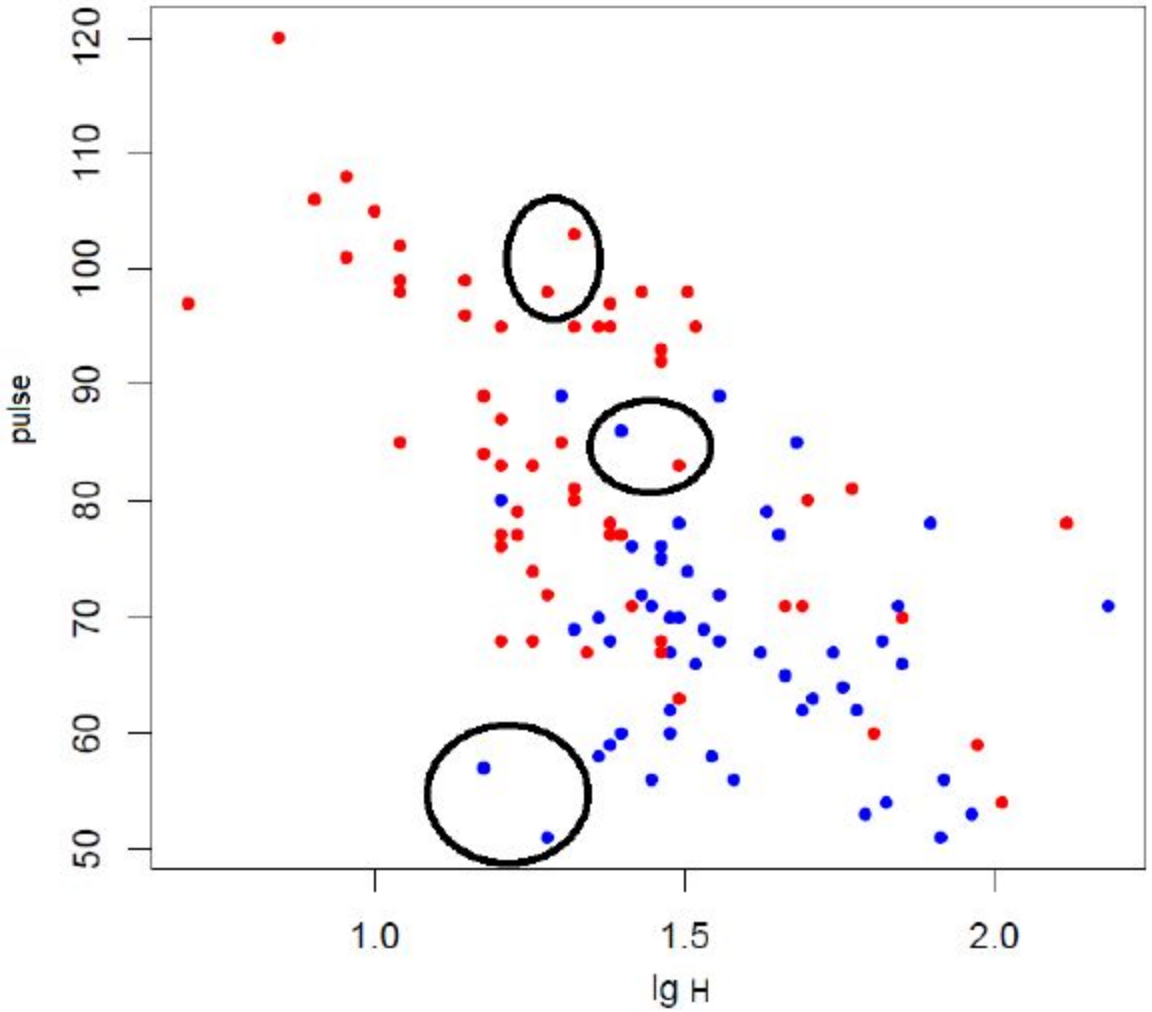
Гипотезы компактности и непрерывности

Задачи классификации и регрессии:
X - объекты, Y - ответы; $X^n = (x_i, y_i)_{i=1}^n$ - обучающая выборка.

Гипотеза компактности (для классификации): *Близкие объекты, как правило, лежат в одном классе.*

Гипотеза непрерывности (для регрессии): *Близким объектам соответствуют близкие ответы.*

Гипотеза компактности



Постановка задачи классификации

- Исторически возникла из задачи машинного зрения, поэтому часто употребляемый синоним – распознавание образов.
- В классической задаче классификации обучающая выборка представляет собой набор отдельных объектов

$$X = \{\boldsymbol{x}_i\}_{i=1}^{n},$$

характеризующихся вектором вещественнозначных признаков

$$\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d}) \ .$$

В качестве исхода объекта $\boldsymbol{x}$ фигурирует переменная, принимающая значения, обычно из множества

$$Y = \{1, \ldots, l\} \ .$$

Требуется постросить алгоритм (классификатор), который по вектору признаков $\boldsymbol{x}$ вернул бы метку класса или вектор оценок принадлежности (апостериорных вероятностей) к каждому из классов характеризующихся вектором вещественнозначных признаков

$$\{p(s|\boldsymbol{x})\}_{s=1}^{l} \ .$$

Метод k ближайших соседей

Метод k ближайших соседей (kNN — k nearest neighbours) метрический алгоритм для классификации объектов, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

Алгоритм:

1. Вычислить расстояние до каждого из объектов обучающей выборки
2. Отобрать k объектов обучающей выборки, расстояние до которых минимально
3. Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей

Меры близости

Что такое близкие объекты? Задана функция расстояния $\rho : X \times X \to [0, \infty)$.
Виды функций расстояния:

- Евклидово: $\rho(x_i, x_j) = \sqrt{\sum_{k=1}^{m} w_k \left(x_i^{(k)} - x_j^{(k)}\right)^2}$
- L_p-метрика: $\rho(x_i, x_j) = \left(\sum_{k=1}^{m} w_k \left|x_i^{(k)} - x_j^{(k)}\right|^p\right)^{1/p}$
- L_∞-метрика: $\rho(x_i, x_j) = \max_{k=1,\ldots,m} \left|x_i^{(k)} - x_j^{(k)}\right|$
- L_1-метрика: $\rho(x_i, x_j) = \sum_{k=1}^{m} \left|x_i^{(k)} - x_j^{(k)}\right|$

$x_i = (x_i^{(1)}, \ldots, x_i^{(m)})$ - вектор m признаков i-го объекта;
$x_i = (x_i^{(1)}, \ldots, x_i^{(m)})$ - вектор m признаков i-го объекта.

Метод k ближайших соседей

Достоинства:

- Простота реализации.
- Классификацию, проведенную алгоритмом, легко интерпретировать путем предъявления пользователю нескольких ближайших объектов.

Недостатки:

- Необходимость хранения обучающей выборки целиком.
- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки.

Выбор k

- Малые значения k приведут к тому, что "шум" (выбросы) будет существенно влиять на результаты.
- Большие значения усложняют вычисления и искажают логику ближайших соседей, в соответствии с которой ближайшие точки могут принадлежать одному классу (гипотеза компактности).
- **Эвристика**: $k = \sqrt{n}$

Пример

Анализ брака древесины: по признакам средняя длина трещины и средний диаметр сучка

длина трещины	диаметр сучка	класс
7	7	брак
7	4	брак
3	4	не брак
1	4	не брак

Новый объект (длина трещины=3, диаметр сучка=7), $k = 3$

длина трещины	диаметр сучка	ρ
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

длина трещины	диаметр сучка	ρ	ранк	входит в 3 соседа?
7	7	16	3	да
7	4	25	4	нет
3	4	9	1	да
1	4	13	2	да

длина трещины	диаметр сучка	ρ	ранк	класс объекта
7	7	16	3	брак
7	4	25	4	-
3	4	9	1	не брак
1	4	13	2	не брак

*Объект (3,7) принадлежит классу "**не брак**"*

1-правило

Алгоритм построения 1-правил

- Простейший алгоритм формирования элементарных правил для классификации объекта. Он строит правила по значению одной независимой переменной, поэтому в литературе его часто называют "1-правило" (1-rule) или кратко 1R-алгоритм.
- Идея алгоритма :

  Для любой независимой переменной формируется правило, которое классифицирует объекты из обучающей выборки. При этом указывается значение зависимой переменной, которое наиболее часто встречается у объектов с выбранным значением независимой переменной. В этом случае ошибкой правила является количество объектов, имеющих то же значение рассматриваемой переменной, но не относящихся к выбранному классу.
- Таким образом, для каждой переменной будет получен набор правил (для каждого значения). Оценив степень ошибки каждого набора, выбирается переменная, для которой построены правила с наименьшей ошибкой.

Example

- Let T be the following table

Office	Party	State	Vote
House	Dem	NY	Yes
House	Dem	NJ	No
House	Dem	CT	Yes
House	Rep	NJ	No
House	Rep	CT	Yes
Senate	Dem	NY	No
Senate	Dem	NJ	No
Senate	Rep	NY	No
Senate	Rep	NJ	Yes
Senate	Rep	CT	Yes

Then BestRule(Vote,Office,T) is "case (X.Office) { House: predict (X.Vote==Yes); Senate: predict(X.Vote==No); }" with an accuracy of 6/10.

BestRule(Vote,Party,T) is "case (X.Party) { Dem: predict (X.Vote==No); Rep: predict(X.Vote==Yes); }" with an accuracy of 6/10.

BestRule(Vote,State,T) is "case (X.State) { NY: predict (X.Vote==No); NJ: predict (X.Vote==No); CT: predict(X.Vote==Yes) }" with an accuracy of 8/10.

The 1R algorithm therefore returns the rule "case (X.State) { NY: predict (X.Vote==No); NJ: predict (X.Vote==No); CT: predict(X.Vote==Yes) }"

Vote - избирательный голос
House - палата

- Если переменная имеет вещественный тип, то количество возможных значений может быть бесконечно. Для решения этой проблемы всю область значений такой переменной разбивают на интервалы таким образом, чтобы каждый из них соответствовал определенному классу в обучающей выборке.
- Проблема 1R-алгоритма — это сверхчувствительность (overfitting). Дело в том, что алгоритм будет выбирать переменные, принимающие наибольшее количество возможных значений, т. к. для них ошибка будет наименьшей. Например, для переменной, являющейся ключом (т. е. для каждого объекта свое уникальное значение), ошибка будет равна нулю. Однако для таких переменных правила будут абсолютно бесполезны, поэтому при формировании обучающей выборки для данного алгоритма важно правильно выбрать набор независимых переменных.
  В заключение необходимо отметить, что 1R-алгоритм, несмотря на свою простоту, во многих случаях на практике оказывается достаточно эффективным. Это объясняется тем, что многие объекты действительно можно классифицировать лишь по одному атрибуту. Кроме того, немногочисленность формируемых правил позволяет легко понять и использовать полученные результаты.

- Томас Байес — английский математик, священник, член Лондонского королевского общества. Автор теоремы Байеса — одной из основных теорем элементарной теории вероятностей.

Биография

Байес родился в Лондоне в 1702 году. Его отец — Джошуа Байес — был пресвитерианским священником, представителем известного нонконформистского рода из Шеффилда. Умер в 1761 году.

Математические интересы Байеса относились к теории вероятностей. Он сформулировал и решил одну из основных задач этого раздела математики (теорема Байеса). Работа Байеса была опубликована уже после его смерти, в 1763 году.

За всю жизнь Томас Байес опубликовал только две работу — одну богословскую и одну математическую:

Divine Benevolence, or an Attempt to Prove That the Principal End of the Divine Providence and Government is the Happiness of His Creatures (1731г.)

- An Introduction to the Doctrine of Fluxions, and a Defence of the Mathematicians Against the Objections of the Author of The Analyst (опубликовано анонимно в 1736г.)

Теорема Байеса и классификация

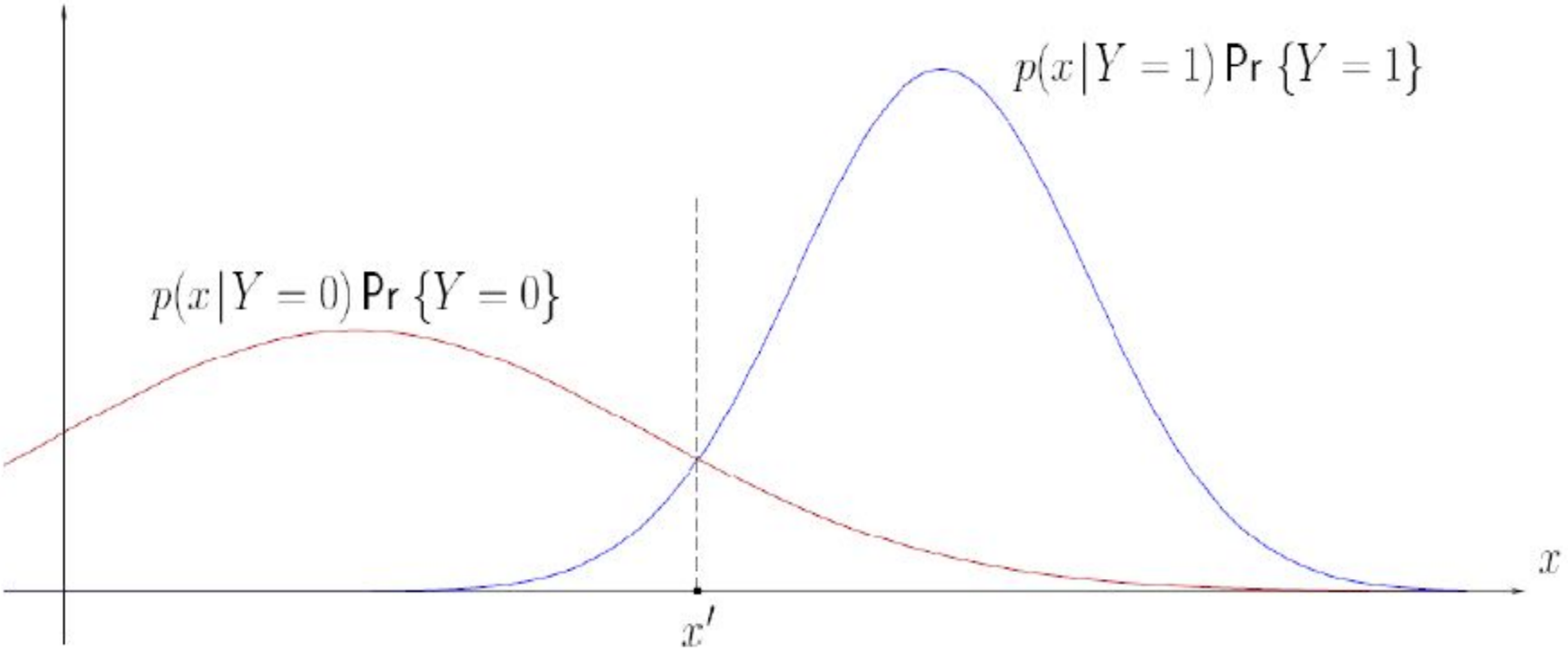
Цель классификации состоит в том чтобы понять к какому классу принадлежит объект x. Следовательно необходимо найти наиболее вероятный класс объекта x, т.е., необходимо из всех классов выбрать тот, который дает максимум вероятности $P(y = c|x)$:

$$c_{opt} = \arg\max_{c \in C} P(y = c|x) = \arg\max_{c \in C} \frac{P(x|y = c)P(y = c)}{P(x)}. \quad \textbf{(1)}$$

Для каждого класса c вычисляется $P(y = c|x)$ и выбирается класс, имеющий максимальную вероятность. Вероятность $P(x)$ не зависит от c и является константой:

$$c_{opt} = \arg\max_{c \in C} P(x|y = c)P(y = c).$$

Принцип максимума апостериорной вероятности



При $x < x'$ считаем $c_{opt} = 0$ иначе $c_{opt} = 1$

Байесовский классификатор

Выбор:

$$\begin{cases} \text{класс } c_1, & \text{если } P(y = c_1|x) > P(y = c_2|x) \\ \text{класс } c_2, & \text{иначе} \end{cases}$$

или

$$\begin{cases} \text{класс } c_1, & \text{если } \dfrac{P(x|y = c_1)}{P(x|y = c_2)} > \dfrac{P(y = c_2)}{P(y = c_1)} \\ \text{класс } c_2, & \text{иначе} \end{cases}$$

Байесовский классификатор минимизирует ошибку принятия решений

Наивный байесовский классификатор

Naive bayes

Байесовский классификатор представляет объект как набор признаков (атрибутов), вероятности которых условно не зависят друг от друга:

$$P(x|y=c) = P(f_1|y=c)P(f_2|y=c)\cdots P(f_m|y=c) = \prod_{i=1}^{m} P(f_i|y=c).$$

Наивный байесовский классификатор:

$$c_{opt} = \arg\max_{c\in C} \left(P(y=c)\prod_{i=1}^{m} P(f_i|y=c)\right).$$

или

$$c_{opt} = \arg\max_{c\in C} \left(\log P(y=c) + \sum_{i=1}^{m} \log P(f_i|y=c)\right).$$

Оценка вероятностей в наивном байесовском классификаторе

Вероятность класса $P(y = c)$ оценивается по обучающей выборке как:

$$P(y = c) = N_c/N$$

N_c – количество объектов, принадлежащих классу ,
N – общее количество объектов в обучающей выборке.

Вероятность $P(f_i|y = c)$ оценивается по обучающей выборке как:

$$P(f_i|y = c) = \frac{M_i(c) + \alpha}{\sum_{j=1}^{m} (M_j(c) + \alpha)}$$

$M_i(c)$ - общее количество элементов с заданным значением признака i в классе c.
$\alpha > 0$ - для избежания нулевых значений вероятности, например, $\alpha = 1$

Naive Bayes: как это считать

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

доля слов в документах с меткой C_k в обучающей выборке

доля слова x_i среди всех текстов с меткой C_k

1. Вычислить оценки вероятностей из формулы выше
2. Для каждого нового документа вычислять значение выражения для каждого класса
3. Выбрать класс с наибольшим произведением

Пример Naive bayes

x

y

Наблюдение	Температура	Влажность	Ветер	Игра
Солнце	Жарко	Высокая	Нет	Нет
Солнце	Жарко	Высокая	Есть	Нет
Облачность	Жарко	Высокая	Нет	Да
Дождь	Норма	Высокая	Нет	Да
Дождь	Холодно	Норма	Нет	Да
Дождь	Холодно	Норма	Есть	Нет
Облачность	Холодно	Норма	Есть	Да
Солнце	Норма	Высокая	Нет	Нет
Солнце	Холодно	Норма	Нет	Да
Дождь	Норма	Норма	Нет	Да
Солнце	Норма	Норма	Есть	Да
Облачность	Норма	Высокая	Есть	Да
Облачность	Жарко	Норма	Нет	Да
Дождь	Норма	Высокая	Есть	Нет

Метод Naive bayes.

Необходимо определить, состоится ли игра при следующих значениях независимых переменных (событие E):

наблюдение = солнечно;
температура = холодно;
влажность = высокая;
ветер = есть,

Определяем условные вероятности

$$P(\text{наблюдение} = \text{солнце} \mid \text{игра} = \text{да}) = 2/9;$$

$$P(\text{наблюдение} = \text{облачно} \mid \text{игра} = \text{да}) = 4/9;$$

$$P(\text{наблюдение} = \text{дождь} \mid \text{игра} = \text{да}) = 3/9;$$

$$P(\text{наблюдение} = \text{солнце} \mid \text{игра} = \text{нет}) = 3/5;$$

$$P(\text{наблюдение} = \text{облачно} \mid \text{игра} = \text{нет}) = 0/5;$$

$$P(\text{наблюдение} = \text{дождь} \mid \text{игра} = \text{нет}) = 2/5.$$

Метод Naive bayes.

Априорные Вероятности $P(y = c_r)$ есть отношение объектов из обучающей выборки, принадлежащих классу , к общему количеству объектов в выборке. В данном примере это:

$$P(\text{игра} = \text{да}) = 9/14;$$

$$P(\text{игра} = \text{нет}) = 5/14.$$

Метод Naive bayes.

Вычислим следующие апостериорные вероятности:

$$P(\text{игра} = \text{да} \mid E) = P(\text{наблюдение} = \text{солнечно} \mid \text{игра} = \text{да}) \times$$
$$\times P(\text{температура} = \text{холодно} \mid \text{игра} = \text{да}) \times$$
$$\times P(\text{влажность} = \text{высокая} \mid \text{игра} = \text{да}) \times$$
$$\times P(\text{ветер} = \text{есть} \mid \text{игра} = \text{да}) \times P(\text{игра} = \text{да}) / P(E);$$
$$P(\text{игра} = \text{нет} \mid E) = P(\text{наблюдение} = \text{солнечно} \mid \text{игра} = \text{нет}) \times$$
$$\times P(\text{температура} = \text{холодно} \mid \text{игра} = \text{нет}) \times$$
$$\times P(\text{влажность} = \text{высокая} \mid \text{игра} = \text{нет}) \times$$
$$\times P(\text{ветер} = \text{есть} \mid \text{игра} = \text{нет}) \times P(\text{игра} = \text{нет}) / P(E).$$

Метод Naive bayes.

Подставляя соответствующие вероятности получим следующие

$$P(\text{игра} = \text{да} \mid E) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 \,/\, P(E) = 0{,}0053/P(E);$$

$$P(\text{игра} = \text{нет} \mid E) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 \,/\, P(E) = 0{,}0206/P(E).$$

Вероятность $P(E)$ не учитывается, т.к. при нормализации вероятностей для каждого из возможных правил она исчезает. Нормализованная вероятность для правила вычисляется по формуле:

$$P'(y = c_r \mid E) = P(y = c_r \mid E) / \sum P(y = c_r \mid E)$$

Метод Naive bayes.

В данном случае можно утверждать, что при указанных условиях игра состоится с вероятностью:

$$P'(игра = да \mid E) = 0{,}0053 / (0{,}0053 + 0{,}0206) = 0{,}205$$

и не состоится с вероятностью:

$$P'(игра = нет \mid E) = 0{,}0206 / (0{,}0053 + 0{,}0206) = 0{,}795$$

Таким образом, при указанных условиях более вероятно, что игра не состоится.

Использование многомерного нормального распределения в задаче распознавания образов

- В статистической теории распознавания образов используется аппроксимация плотности с помощь многомерного нормального распределения.
- При решении задач распознавания с помощью формулы Байеса в форме (1) могут использоваться плотности вероятности $p_1(\boldsymbol{x}),\ldots,p_n(\boldsymbol{x})$, в которых переменные $X_1,\ldots,X_n$ не обязательно являются независимыми. Чаще всего используется многомерное нормальное распределения. Плотность данного распределения в общем виде представляется выражением

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} \mid \Sigma \mid^{\frac{1}{2}}} exp[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^t],$$

- где

$\boldsymbol{\mu}$ - математическое ожидание вектора признаков $\boldsymbol{x}$; Σ - матрица ковариаций признаков $X_1,\ldots,X_n$; $\mid \Sigma \mid$ -детерминант матрицы Σ.

Для построения распознающего алгоритма достаточно оценить вектора математических ожиданий $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L$ и матрицы ковариаций $\Sigma_1, \ldots, \Sigma_L$ для классов $K_1, \ldots, K_L$, соответственно.

Оценка вектора математических ожиданий $\boldsymbol{\mu}_i$ вычисляется как среднее значение векторов признаков по объектам обучающей выборки $\widetilde{S}_t$ из класса K_i :

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{m_i} \sum_{s_j \in \widetilde{S}_t \bigcap K_i} \boldsymbol{x}_j \quad ,$$

где m_i - число объектов класса K_i в обучающей выборке. Элемент матрицы ковариаций для класса K_i вычисляется по формуле

$$\hat{\sigma}^i_{kk'} = \frac{1}{m_i} \sum_{s_j \in \widetilde{S}_t \bigcap K_i} (x_{jk} - \mu_{ik})(x_{jk'} - \mu_{ik'}),$$

где $x_{jk} - \mu_{ik}$ - k-я компонента вектора $\boldsymbol{\mu}_i$. Матрицу ковариации, состоящую из элементов $\hat{\sigma}^i_{kk'}$ обозначим $\hat{\Sigma}_i$. Очевидно, что согласно формуле Байеса максимум $P(K_i \mid \boldsymbol{x})$ достигается для тех же самых классов для которых максимально произведение $P(K_i)p_i(\boldsymbol{x})$.

Очевидно, что для байесовской классификации может использоваться также натуральный логарифм $\ln[P(K_i)p_i(\boldsymbol{x})]$ который согласно вышеизложенному может быть оценён выражением

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}\hat{\Sigma}_i^{-1}\boldsymbol{x}^t + \boldsymbol{w}_i\boldsymbol{x}^t + g_i^0,$$

где $\boldsymbol{w}_i = \hat{\boldsymbol{\mu}}_i\hat{\Sigma}_i^{-1}$;

g_i^0 - не зависящее от $\boldsymbol{x}$ слагаемое:

ν_i - доля объектов класса K_i в обучающей выборке. Слагаемое g_i^0 имеет вид

$$g_i^0 = -\frac{1}{2}\hat{\boldsymbol{\mu}}_i\hat{\Sigma}_i^{-1}\hat{\boldsymbol{\mu}}_i^t - \frac{1}{2}\ln\left(\mid \hat{\Sigma}_i \mid\right) + \ln(\nu_i) - \frac{n}{2}\ln(2\pi).$$

В результате решающее правило распознавания имеет вид

$$j = \arg\max_i \{-\frac{1}{2} x \hat{\Sigma}_i^{-1} x^t + w_i x^t + g_i^0\}$$

где

$$g_i^0 = -\frac{1}{2} \hat{\mu}_i \hat{\Sigma}_i^{-1} \hat{\mu}_i^t - \frac{1}{2} \left| \hat{\Sigma}_i \right| + \ln(\nu_i)$$

Таким образом объект с признаковым описанием $\boldsymbol{x}$ будет отнесён построенной выше аппроксимацией байесовского классификатора к классу, для которого оценка $g_i(\boldsymbol{x})$ является максимальной. Следует отметить, что построенный классификатор в общем случае является квадратичным по признакам. Однако классификатор превращается в линейный, если оценки ковариационных матриц разных классов оказываются равными.

В случае линейного классификатора $\hat{\Sigma}_i = \hat{\Sigma}, i = 1,2,...$ и рещающее правило примет вид

$$j = \arg\max_i \{w_i x^t + g_i^0\}$$

где

$$g_i^0 = -\frac{1}{2} \hat{\mu}_i \hat{\Sigma}^{-1} \hat{\mu}_i^t + \ln(\nu_i)$$

- Сэр **Ро́налд Э́йлмер Фи́шер** (или **Рональд**, англ. *Sir Ronald Aylmer Fisher*, 17 февраля, 17 февраля 1890, 17 февраля 1890 — 29 июля, 17 февраля 1890 — 29 июля 1962, 17 февраля 1890 — 29 июля 1962) — английский, 17 февраля 1890 — 29 июля 1962) — английский статистик, 17 февраля 1890 — 29 июля 1962) — английский статистик, биолог-эволюционист, 17 февраля 1890 — 29 июля 1962) — английский статистик, биолог-эволюционист и генетик, 17 февраля 1890 — 29 июля 1962) — английский статистик, биолог-эволюционист и генетик. Член Лондонского королевского общества, 17 февраля 1890 — 29 июля 1962) — английский статистик, биолог-эволюционист и генетик. Член Лондонского королевского общества (1929) и Королевского статистического общества, почётный член многих академий и научных обществ.
- В математической статистикеВ математической статистике Фишер, разработал фундамент теории оценок параметровВ математической статистике Фишер, разработал фундамент теории оценок параметров, статистических решенийВ математической статистике Фишер, разработал фундамент теории оценок параметров, статистических решений, планирования экспериментаВ математической статистике Фишер, разработал фундамент теории оценок параметров, статистических решений, планирования эксперимента и проверки гипотез.
- Фишер перечислил требования к статистическим оценкам: состоятельностьФишер перечислил требования к статистическим оценкам: состоятельность, эффективностьФишер перечислил требования к

Линейный дискриминант Фишера

Рассмотрим вариант метода Линейный дискриминант Фишера (ЛДФ) для распознавания двух классов K_1 и K_2 . В основе метода лежит поиск в многомерном признаковом пространстве такого направления $\boldsymbol{w}$, чтобы средние значения проекции на него объектов обучающей выборки из классов K_1 и K_2 максимально различались. Проекцией произвольного вектора $\boldsymbol{x}$ на направление $\boldsymbol{w}$ является отношение

$$\frac{(\boldsymbol{w}\boldsymbol{x}^t)}{|\,\boldsymbol{w}\,|}.$$

В качестве меры различий проекций классов на используется функционал

$$\Phi(\boldsymbol{w}, \widetilde{S}_t) = \frac{(\hat{X}_{\boldsymbol{w}1} - \hat{X}_{\boldsymbol{w}2})^2}{\hat{d}_{\boldsymbol{w}1} + \hat{d}_{\boldsymbol{w}2}},$$

где

$$\hat{X}_{\boldsymbol{w}i} = \frac{1}{m_i} \sum_{s_j \in \widetilde{S}_t \bigcap K_i} \frac{(\boldsymbol{w}\boldsymbol{x}_j^t)}{\mid \boldsymbol{w} \mid}$$

- среднее значение проекции векторов переменных $X_1, \ldots, X_n$, описывающих объекты из класса K_i;

$$\hat{d}_{\boldsymbol{w}i} = \frac{1}{m_i} \sum_{s_j \in \widetilde{S}_t \bigcap K_i} [\frac{(\boldsymbol{w}\boldsymbol{x}_j^t)}{\mid \boldsymbol{w} \mid} - \hat{X}_{\boldsymbol{w}i}]^2$$

- дисперсия проекций векторов, описывающих объекты из класса $K_i, i \in \{1, 2\}$. Смысл функционала $\Phi(\boldsymbol{w}, \widetilde{S}_t)$ ясен из его структуры. Он является по сути квадратом отличия между средними значениями проекций классов на направление $\boldsymbol{w}$, нормированным на сумму внутриклассовых выборочных дисперсий.

Можно показать, что $\Phi(\boldsymbol{w}, \widetilde{S}_t)$ достигает максимума при

$$\boldsymbol{w} = \hat{\Sigma}_{12}^{-1}(\hat{\boldsymbol{\mu}}_1^t - \hat{\boldsymbol{\mu}}_2^t),$$

где $\hat{\Sigma}_{12} = \hat{\Sigma}_1 + \hat{\Sigma}_2$. Таким образом оценка направления, оптимального для распознавания K_1 и K_2 может быть записана в виде (5)
Распознавание нового объекта s_* по векторному описанию $\boldsymbol{x}_*$ производится по величине его проекции на направление $\boldsymbol{w}$:

$$\gamma(\boldsymbol{x}_*) = \frac{(\boldsymbol{w}, \boldsymbol{x}_*^t)}{|\,\boldsymbol{w}\,|}.$$

При этом используется простое пороговое правило: при $\gamma(\boldsymbol{x}_*) > b$ объект s_* относится к классу K_1 и s_* относится к классу K_2 в противном случае.

Граничный параметр b подбирается по обучающей выборке таким образом, чтобы проекции объектов разных классов на оптимальное направление $\boldsymbol{w}$ оказались бы максимально разделёнными. Простой, но эффективной, стратегией является выбор в качестве порогового параметра b средней проекции объектов обучающей выборки на $\boldsymbol{w}$.
Метод ЛДФ легко обобщается на случай с несколькими классами. При этом исходная задача распознавания классов $K_1, \ldots, K_L$ сводится к последовательности задач с двумя классами K_1' и K_2' :

Зад. 1. Класс $K_1' = K_1$, класс $K_2' = \Omega \setminus K_1$

. . . . . .

Зад. L. Класс $K_1' = K_L$, класс $K_2' = \Omega \setminus K_L$

Для каждой из L задач ищется оптимальное направление. В результате получается набор из L направлений $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$.

В результате получается набор из L направлений $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$. При распознавании нового объекта s_* по признаковому описанию $\boldsymbol{x}_*$ вычисляются проекции на $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$:

$$\gamma_1(\boldsymbol{x}_*) = \frac{(\boldsymbol{w}_1, \boldsymbol{x}_*^t)}{|\boldsymbol{w}_1|}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \gamma_L(\boldsymbol{x}_*) = \frac{(\boldsymbol{w}_L, \boldsymbol{x}_*^t)}{|\boldsymbol{w}_L|}.$$

Распознаваемый объект относится к тому классу, соответствующему максимальной величине проекции. Распознавание может производится также по величинам

$$[\gamma_1(\boldsymbol{x}_*) - b_1], \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, [\gamma_L(\boldsymbol{x}_*) - b_L].$$

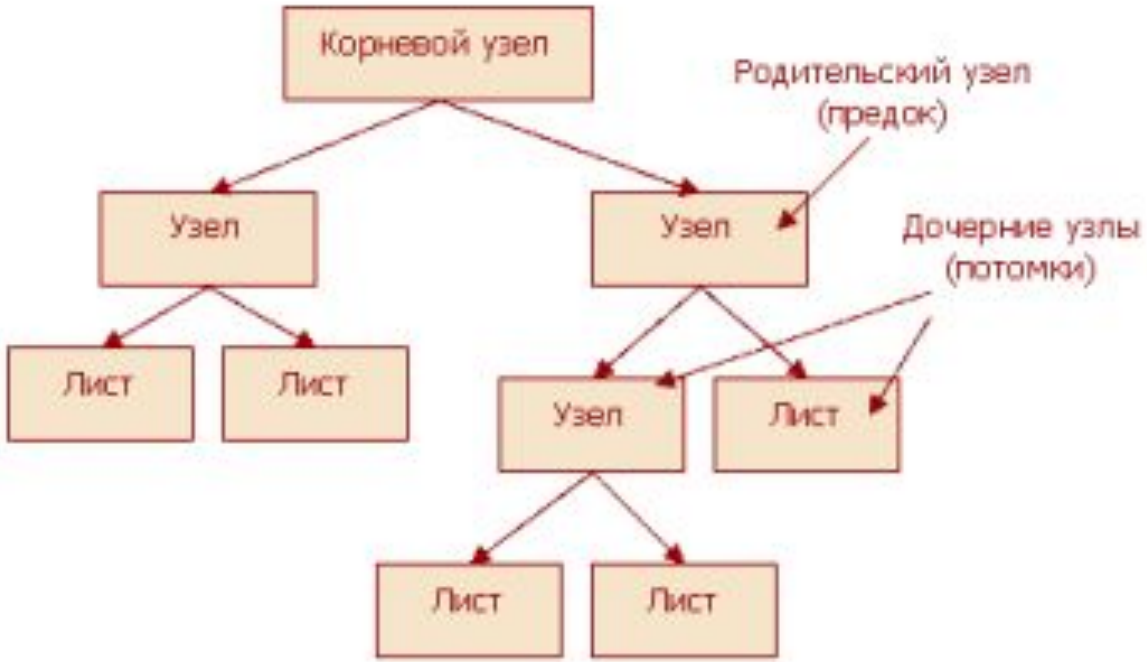
- Лекция 3

- **Классификация с использованием деревьев решений**

Деревья решений

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

Деревья решений – это логический алгоритм классификации, основанный на поиске конъюнктивных закономерностей.



Пример дерева классификации (Выдавать ли кредит?)

Деревья решений

- **Деревья решений** - это способ представления классификационных правил в иерархической, последовательной структуре.
- Области применения деревьев решений
- Деревья решений являются прекрасным инструментом в системах поддержки принятия решений, интеллектуального анализа данных (data mining). В состав многих пакетов, предназначенных для интеллектуального анализа данных, уже включены методы построения деревьев решений.
- Деревья решений успешно применяются для решения практических задач в следующих областях:
- Банковское дело. Оценка кредитоспособности клиентов банка при выдаче кредитов.
- Промышленность. Контроль за качеством продукции (выявление дефектов), испытания без разрушений (например проверка качества сварки) и т.д.
- Медицина. Диагностика различных заболеваний.
- Молекулярная биология. Анализ строения аминокислот.
- Телекоммуникации.

- Обычно каждый узел дерева решений включает проверку одной независимой переменной. Иногда в узле дерева две независимые переменные сравниваются друг с другом или определяется некоторая функция от одной или нескольких переменных.
  Если переменная, которая проверяется в узле, принимает категориальные значения, то каждому возможному значению соответствует ветвь, выходящая из узла дерева. Если значением переменной является число, то проверяется больше или меньше это значение некоторой константы. Иногда область числовых значений разбивают на интервалы. (Проверка попадания значения в один из интервалов).
- Листья деревьев соответствуют значениям зависимой переменной, т.е. классам

Методика "Разделяй и властвуй"

- Методика основана на рекурсивном разбиении множества объектов из обучающей выборки на подмножества, содержащие объекты, относящиеся к одинаковым классам.
  Сперва выбирается независимая переменная, которая помещается в корень дерева. Из вершины строятся ветви, соответствующие всем возможным значениям выбранной независимой переменной.
  Множество объектов из обучающей выборки разбивается на несколько подмножеств в соответствии со значением выбранной независимой переменной. Таким образом, в каждом подмножестве будут находиться объекты, у которых значение выбранной независимой переменной будет одно и то же.
  Относительно обучающей выборки T и множества классов C возможны три ситуации:
- множество T содержит один или более объектов, относящихся к одному классу c_r. Тогда дерево решений для T - это лист, определяющий класс c_r;
- множество T не содержит ни одного объекта (пустое множество). Тогда это снова лист, и класс, ассоциированный с листом, выбирается из другого множества, отличного от T, например из множества, ассоциированного с родителем;

- Множество T содержит объекты, относящиеся к разным классам. В этом случае следует разбить множество T на некоторые подмножества. Для этого выбирается одна из независимых переменных x_h, имеющая два и более отличных друг от друга значений

$$c_h^1, c_h^2 \ldots, c_h^n$$

- Множество T разбивается на подмножества $T_1, T_2, \ldots, T_n$, где каждое подмножество T_i содержит все объекты, у которых значение выбранной зависимой переменной равно c_h^i

- Далее процесс продолжается рекурсивно для каждого подмножества до тех пор, пока значение зависимой переменной во вновь образованном подмножестве не будет одинаковым (когда объекты принадлежат одному классу). В этом случае процесс для данной ветви дерева прекращается.
- При использовании данной методики построение дерева решений будет происходить сверху вниз. Большинство алгоритмов, которые её используют, являются "жадными алгоритмами". Это значит, что если один раз переменная была выбрана и по ней было произведено разбиение, то алгоритм не может вернуться назад и выбрать другую переменную, которая дала бы лучшее разбиение

- Вопрос в том, какую зависимую переменную выбрать для начального разбиения. От этого целиком зависит качество получившегося дерева. Общее правило для выбора переменной для разбиения: выбранная переменная должны разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т.е. чтобы количество объектов из других классов ("примесей") в каждом из этих множеств было минимальным. Другой проблемой при построении дерева является проблема остановки его разбиения. Методы её решения:
- Ранняя остановка. Использование статистических методов для оценки целесообразности дальнейшего разбиения. Экономит время обучения модели, но строит менее точные классификационные модели.
- Ограничение глубины дерева. Нужно остановить дальнейшее построение, если разбиение ведёт к дереву с глубиной, превышающей заданное значение.
- Разбиение должно быть нетривиальным, т.е. получившиеся в результате узлы должны содержать не менее заданного количества объектов.
- Отсечение ветвей (снизу вверх). Построить дерево, отсечь или заменить поддеревом те ветви, которые не приведут к возрастанию ошибки. Под ошибкой понимается количество неправильно классифицированных объектов, а точностью дерева решений отношение правильно классифицированных объектов при обучении к общему количеству объектов из обучающего множества.

Конструирование дерева решений

- **Процесс конструирования дерева решений**
- 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*.

Критерии 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, преимущественно являлись представителями одного класса. 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.

Критерии расщепления или меры неоднородности множества относительно его меток:

- мера энтропии (cross-entropy): $-\sum_{i=1}^{C} p_i \log(p_i)$
- индекс Gini: $1 - \sum_{i=1}^{n} p_i^2$,

p_i - частота или вероятность точек i-го класса в блоке;

Алгориты построения деревьв

- Есть различные способы выбирать очередной атрибут:
- Алгоритм ID3Алгоритм ID3, где выбор атрибута происходит на основании прироста информации (англ. *Gain*.
- Алгоритм C4.5Алгоритм C4.5 (улучшенная версия ID3), где выбор атрибута происходит на основании нормализованного прироста информации (англ. *Gain Ratio*).
- Алгоритм CARTАлгоритм CART и его модификации — на основании критерия Джини.
- На практике в результате работы этих алгоритмов часто получаются слишком детализированные деревья, которые при их дальнейшем применении дают много ошибок. Это связано с явлением переобучения. Для сокращения деревьев используется отсечение ветвей.

Алгоритмы ID3 и C4.5

- Алгоритм ID3 [32] был предложен Россом Куинланом в 1986 г. и основывался на алгоритме Ханта, учеником которого был Куинлан. Алгоритм ID3 был основан на использовании критерия информационного выигрыша для выбора вершины и переменной для ветвления. Синтез решающего дерева завершался либо в случае достижения его корректности относительно выборки, либо когда ветвление ни в одной некорректной вершине не приводило к увеличению информационного выигрыша.

- Алгоритм C4.5 [31]. Этот алгоритм явился развитием идей, реализованных в ID3, был разработан Р. Куинланом в 1993 г. и использовал отношение выигрыша (gain ratio) как критерий ветвления. Процесс синтеза (добавления вершин) в алгоритме C4.5 прекращался, когда число точек для разбиения становилось меньше некоторого порога.

Алгоритм ID3

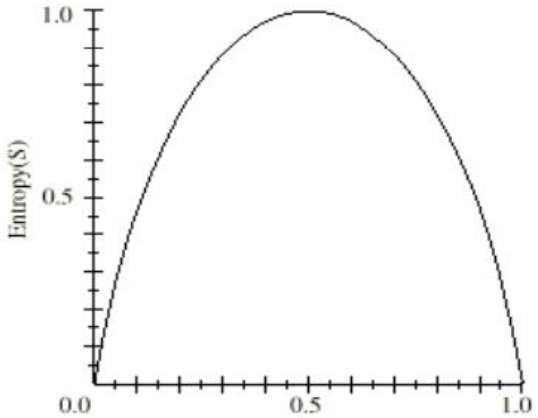
- **ID3** (**Iterative Dichotomiser 3**) is an algorithm) is an algorithm invented by Ross Quinlan
- Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81–106
- Рассмотрим критерий выбора независимой переменной, от которой будет строиться дерево.
  Полный набор вариантов разбиения |X| - количество независимых переменных.
  Рассмотрим проверку переменой x_h, которая принимает m значений $c_h1, c_h2, ..., c_hm$.
  Тогда разбиение множества всех объектов обучающей выборки N по проверке переменной x_h даст подмножества $T_1, T_2, ..., T_m$.
- Мы ожидаем, что при разбиении исходного множества, будем получать подмножества с меньшим числом объектом, но более упорядоченные. Так, чтобы в каждом из них были по-возможности объекты одного класса. Эта мера упорядоченности (неопределенности) характеризуется информацией. В контексте рассматриваемой задачи это количество информации, необходимое для того, чтобы отнести объект к тому или иному классу.
- При разделении исходного множества на более мелкие подмножества, используя в качестве критерия для разделения значения выбранной независимой переменной, неопределённость принадлежности объектов конкретным классам будет уменьшаться. Задача состоит в том, чтобы выбрать такие независимые переменные, чтобы максимально уменьшить эту неопределенность и в конечном итоге получить подмножества, содержащие объекты только одного класса. В последнем случае неопределенность равна нулю.

Критерий расщепления ID3

Тогда: *энтропия* оценивает количество примесей в S:

$E(S) = - p^+ \log_2 p^+ - p^- \log_2 p^-$

$$E(s) = -\sum_{i=1}^{N} p_i \log p_i$$



Критерием для выбора атрибута будет являться следующая формула:

$$\text{Gain}(x_h) = \text{Info}(T) - \text{Info}_{x_h}(T).$$

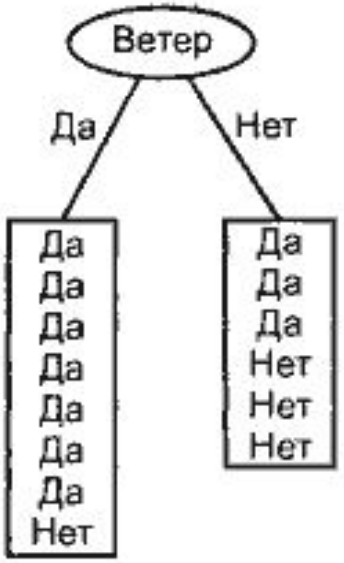
где

$$\text{Info}(T) = -\sum_{j=1}^{k} \frac{\text{freq}(c_r, T)}{|T|} \log_2 \left(\frac{\text{freq}(c_r, T)}{|T|} \right).$$

$$\text{Info}_{x_h}(T) = \sum_{i=1}^{m} T_i / |T| \, \text{Info}(T_i).$$

- Единственная доступная информация - каким образом классы распределены в множестве T и его подмножествах, получаемых при разбиении.
Именно она и используется при выборе переменной.

- Рассмотрим пример, в котором требуется построить дерево решений относительно того, состоится ли игра при заданных погодных условиях.
Исходя из прошлых наблюдений (накопленных исторических данных), возможны четыре варианта разбиения дерева.



- Пусть freq(c_r,I) - число объектов из обучающей выборки, относящихся к классу c_r. Тогда вероятность того, что случайно выбранный объект из обучающего множества I будет принадлежать классу c_r равняется:

$$P = \frac{freq(c_r, I)}{|I|}$$

Подсчитаем количество информации, основываясь на числе объектов того или иного класса, получившихся в узле дерева после разбиения исходного множества. Согласно теории информации оценку среднего количества информации, необходимого для определения класса объекта из множества T, даёт выражение:

$$H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

(**понятие информационной энтропии**)

Подставляя в эту формулу полученное значение для P, получим:

$$Info(I) = -\sum_{r=1}^{k} \frac{freq(c_r, I)}{|I|} \log_2 \left(\frac{freq(c_r, I)}{|I|} \right)$$

Пример

x

y

Наблюдение	Температура	Влажность	Ветер	Игра
Солнце	Жарко	Высокая	Нет	Нет
Солнце	Жарко	Высокая	Есть	Нет
Облачность	Жарко	Высокая	Нет	Да
Дождь	Норма	Высокая	Нет	Да
Дождь	Холодно	Норма	Нет	Да
Дождь	Холодно	Норма	Есть	Нет
Облачность	Холодно	Норма	Есть	Да
Солнце	Норма	Высокая	Нет	Нет
Солнце	Холодно	Норма	Нет	Да
Дождь	Норма	Норма	Нет	Да
Солнце	Норма	Норма	Есть	Да
Облачность	Норма	Высокая	Есть	Да
Облачность	Жарко	Норма	Нет	Да
Дождь	Норма	Высокая	Есть	Нет

- Поскольку используется логарифм с двоичным основанием, то это выражение даёт количественную оценку в битах. Для оценки количества информации справедливы следующие утверждения:
- Если число объектов того или иного класса в получившемся подмножестве равно нулю, то количество информации также равно нулю.
- Если число объектов одного класса равно числу объектов другого класса, то количество информации максимально.
- Посчитаем значение информационной энтропии для исходного множества до разбиения.

$$Info(I) = -\frac{9}{14} * log_2(\frac{9}{14}) - \frac{5}{14} * log_2(\frac{5}{14}) = 0.94$$

Ту же оценку, но уже после разбиения множества T по x_h даёт следующее выражение:

$$Info_{x_h}(T) = \sum_{i=1}^{m} \frac{T_i}{|T|} Info(T_i) \qquad Info_{x_h}(T) = \sum_{i=1}^{m} \frac{T_i}{|T|} (-\sum_{r=1}^{k} \frac{freq(c_r, T_i)}{|T_i|} \log_2(\frac{freq(c_r, T_i)}{|T_i|}))$$

- Например, для переменной "Наблюдение", оценка будет следующей:

$$Info_{sun} = -\frac{2}{5} * log_2(\frac{2}{5}) - \frac{3}{5} * log_2(\frac{3}{5}) = 0.971$$ **бит.**

$$Info_{clouds} = -\frac{4}{4} * log_2(\frac{4}{4}) - \frac{0}{4} * log_2(\frac{0}{4}) = 0$$ бит.

$$Info_{rain} = -\frac{3}{5} * log_2(\frac{3}{5}) - \frac{2}{5} * log_2(\frac{2}{5}) = 0.971$$ **бит.**

$$Info_{condition} = \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.693$$ **бит.**

Критерием для выбора атрибута (зависимой переменной) будет являться следующая формула:

$$Gain(x_h) = Info(I) - Info_{x_h}(T).$$

Критерий Gain рассчитывается для всех независимых переменных после чего выбирается переменная с максимальным значением Gain.
Необходимо выбрать такую переменную, чтобы при разбиении по ней один из классов имел наибольшую вероятность появления. Это возможно в том случае, когда энтропия $Info_x$ имеет минимальное значение и, соответственно, критерий Gain(X) достигает своего максимума.

- В нашем примере значение Gain для независимой переменной "Наблюдение" (перспектива) будет равно:

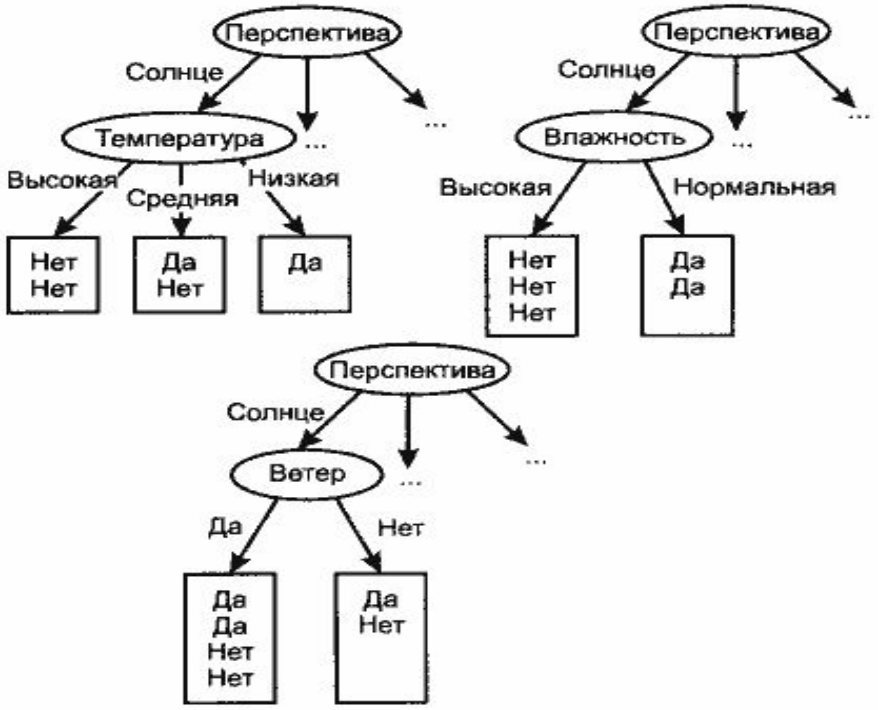
Gain(перспектива) = Info(I) - Info(перспектива) = 0.94 - 0.693 = 0.247 бит.

Аналогичные расчеты можно провести для других независимых переменных. В результате получаем:

Gain(наблюдение) = 0.247 бит.
Gain(температура) = 0.029 бит.
Gain(влажность) = 0.152 бит.
Gain(ветер) = 0.048 бит.

Таким образом, для первоначального разбиения лучше всего выбрать независимую переменную "Наблюдение". Далее требуется выбрать следующую переменную для разбиения. Варианты разбиения представлены на рисунке.

Gain (гейн) - прирост



Наблюдение	Температура	Влажность	Ветер	Игра
Солнце	Жарко	Высокая	Нет	Нет
Солнце	Жарко	Высокая	Есть	Нет
Солнце	Норма	Высокая	Нет	Нет
Солнце	Холодно	Норма	Нет	Да
Солнце	Норма	Норма	Есть	Да

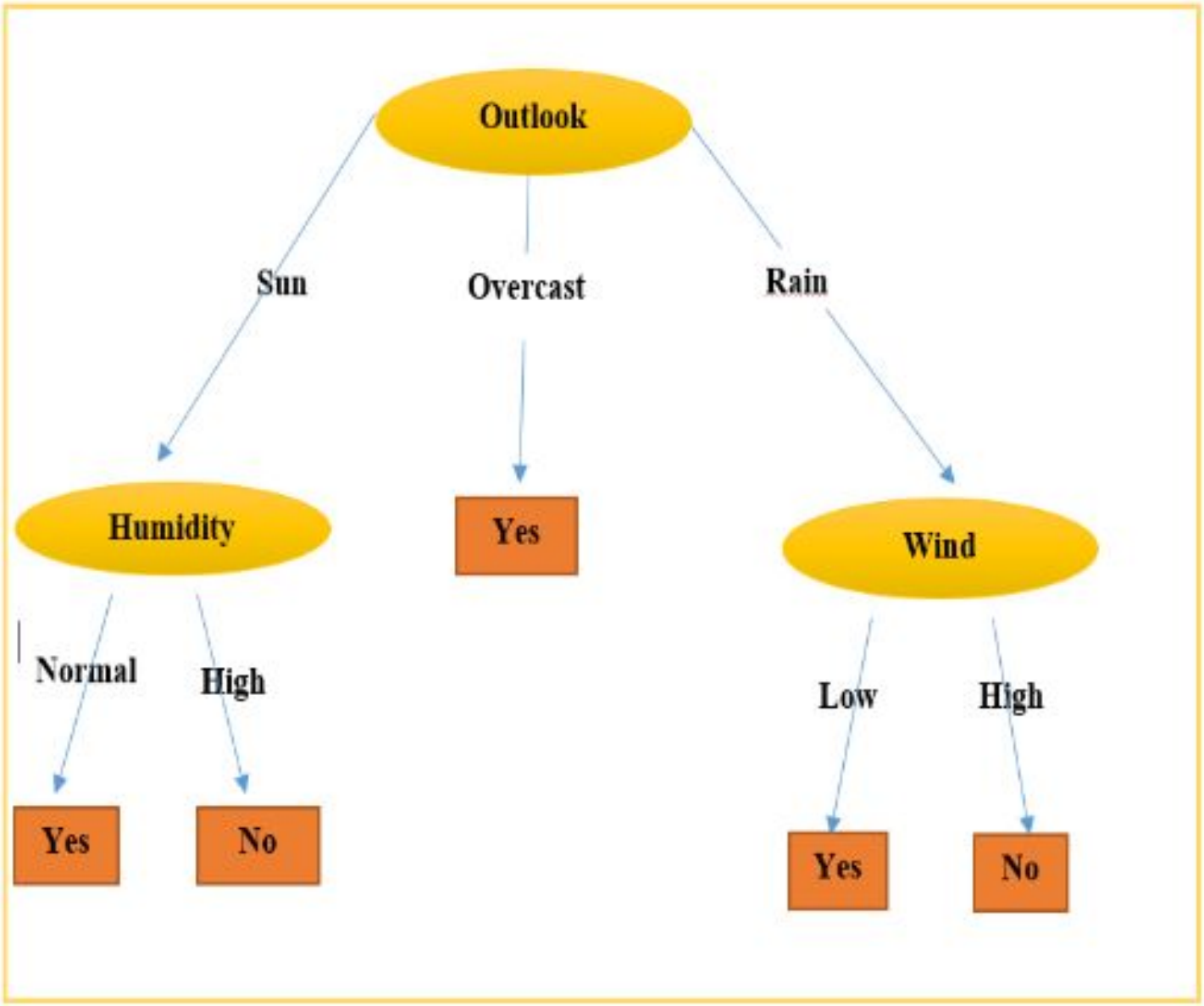
Аналогичным образом можно посчитать значение Gain для каждого разбиения:

Gain(температура) = 0.571 бит.
Gain(влажность) = 0.971 бит.
Gain(ветер) = 0.02 бит.

Видно, что следующей переменной, по которой будет разбиваться подмножество T (солнечно) будет "Влажность".

- Дальнейшее разбиение этой ветви уже не потребуется, т.к. в получившихся подмножествах все объекты относятся только к одному классу.

Если в процессе работы алгоритма получен узел, ассоциированный с пустым множеством (ни один объект не попал в данный узел), то он помечается как лист, и в качестве решения листа выбирается наиболее часто встречающийся класс у непосредственного предка данного листа.



ID3 Final tree

Алгоритм C4.5

- **C4.5** is an algorithm used to generate a decision tree is an algorithm used to generate a decision tree developed by Ross Quinlan.[1] C4.5 is an extension of Quinlan's earlier ID3 algorithm.
- Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
- Представляет собой усовершенствованный вариант алгоритма ID3. Среди улучшений стоит отметить следующие:
- Возможность работать не только с категориальными атрибутами, но также с числовыми. Для этого алгоритм разбивает область значений независимой переменной на несколько интервалов и делит исходное множество на подмножества в соответствии с тем интервалом, в который попадает значение зависимой переменной.
- После построения дерева происходит усечение его ветвей. Если получившееся дерево слишком велико, выполняется либо группировка нескольких узлов в один лист, либо замещение узла дерева нижележащим поддеревом. Перед операцией над деревом вычисляется ошибка правила классификации, содержащегося в рассматриваемом узле. Если после замещения (или группировки) ошибка не возрастает (и не сильно увеличивается энтропия), значит замену можно произвести без ущерба для построенной модели.

- Один из недостатков алгоритма ID3 является то, что он некорректно работает с атрибутами, имеющими уникальные значения для всех объектов из обучающей выборки. Для таких объектов информационная энтропия равна нулю и никаких новых данных от построенного дерева по данной зависимой переменной получить не удасться. Поскольку получаемые после разбиения подмножества буду содержать по одному объекту.
  Алгоритм C4.5 решает эту проблему путём введения нормализации. Оценивается не количество объектов того или иного класса после разбиения, а число подмножеств и их мощность (число элементов).
- Выражение

$$splitinfo(x_h) = -\sum_{i=1}^{m} \frac{T_i}{T} \log_2(\frac{T_i}{T})$$

оценивает потенциальную информацию, получаемую при разбиении множества T на m подмножеств.
Критерием выбора переменной для разбиения будет выражение:

$$gainratio(x_h) = \frac{Gain(x_h)}{splitinfo(x_h)}$$

Ratio (рейдже) - отношение

Split (сплет)- расщепление

- При условии, что имеется k классов и n - число объектов в обучающей выборке и одновременно количество значений переменных, тогда числитель максимально будет равен $log_2 k$, а знаменатель максимально равен $log_2 n$. Если предположить, что количество объектов заведомо больше количества классов, то знаменатель растёт быстрее, чем числитель и, соответственно, значение выражения будет небольшим.
  В обучающей выборке могут присутствовать объекты с пропущенными значениями атрибутов. В этом случае их либо отбрасывают (что влечёт за собой риск потерять часть данных), либо применить подход, предполагающий, что пропущенные значения по переменной вероятностно распределены пропорционально частоте появления существующих значений.

Алгоритм CART

Алгоритм CART (Classification and Regression Tree) разработан в 1974-1984 годах L.Breiman (Berkeley), J.Friedman (Stanford), C.Stone (Berkeley) и R.Olshen (Stanford).

Алгоритм CART предназначен для построения *бинарного* дерева решений.

Особенности алгоритма CART:

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

- Обучение дерева решений относится к классу обучения с учителем, то есть обучающая и тестовая выборки содержат *классифицированный* набор примеров. Оценочная функция, используемая алгоритмом CART, базируется на интуитивной идее уменьшения нечистоты (неопределённости) в узле. Рассмотрим задачу с двумя классами и узлом, имеющим по 50 примеров одного класса. Узел имеет максимальную "нечистоту". Если будет найдено разбиение, которое разбивает данные на две подгруппы 40:5 примеров в одной и 10:45 в другой, то интуитивно "нечистота" уменьшится. Она полностью исчезнет, когда будет найдено разбиение, которое создаст подгруппы 50:0 и 0:50. В алгоритме CART идея "нечистоты" формализована в индексе *Gini*. Если набор данных *T* содержит данные n классов, тогда индекс *Gini* определяется как:

$$Gini\,(T) = 1 - \sum_{i=1}^{n} p_i^2 \ ,$$

- где p_i – вероятность (относительная частота) класса *i* в *T*.

- Если набор T разбивается на две части T_1 и T_2 с числом примеров в каждом N_1 и N_2 соответственно, тогда показатель качества разбиения будет равен:

$$Gini_{split}(T) = \frac{N_1}{N} \cdot Gini(T_1) + \frac{N_2}{N} \cdot Gini(T_2) \quad .$$

Наилучшим считается то разбиение, для которого $Gini_{split}(T)$ минимально. Обозначим N – число примеров в узле – предке, L, R – число примеров соответственно в левом и правом потомке, l_i и r_i – число экземпляров i-го класса в левом/правом потомке. Тогда качество разбиения оценивается по следующей формуле:

$$Gini_{split} = \frac{L}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{r_i}{R}\right)^2\right) \to \min$$

или, что эквивалентно,

$$\tilde{G}_{split} = \frac{1}{L} \cdot \sum_{i-1}^{n} l_i^2 + \frac{1}{R} \cdot \sum_{i-1}^{n} r_i^2 \to \max \quad .$$

Процедура разбиения CART

- Вектор предикторных переменных, подаваемый на вход дерева может содержать как числовые (порядковые) так и категориальные переменные. В любом случае в каждом узле разбиение идет только по *одной переменной*. Если переменная числового типа, то в узле формируется правило вида $x_i <= c$. Где c – некоторый порог, который чаще всего выбирается как среднее арифметическое двух соседних *упорядоченных* значений переменной x_i обучающей выборки. Если переменная категориального типа, то в узле формируется правило $x_i\ V(x_i)$, где $V(x_i)$ – некоторое непустое подмножество множества значений переменной x_i в обучающей выборке. Следовательно, для n значений числового атрибута алгоритм сравнивает $n-1$ разбиений, а для категориального $(2^{n-1} – 1)$. На каждом шаге построения дерева алгоритм последовательно сравнивает все возможные разбиения для всех атрибутов и выбирает наилучший атрибут и наилучшее разбиение для него.

Процедура разбиения CART

1. Выбирается k-ый признак f_k с множеством значений $X^{(k)}$.
2. Определяется такое значение $x_0^{(k)} \in X^{(k)}$ для всех признаков f_k, $k = 1, \ldots, m$, чтобы мера неоднородности $\mathrm{Gini}_{\mathrm{split}}(T)$ была минимальной, т.е.

$$x_0^{(k)} = \arg \min_{f_k, x^{(k)} \in X^{(k)}} \mathrm{Gini}_{\mathrm{split}}(T, x^{(k)})$$

3. Данная процедура выполняется рекурсивно для каждого полученного подмножества до тех пор, пока не будут достигнуты критерии остановки.

Если набор данных разбивается на две части $_1$ и $_2$ с числом примеров в каждом N_1 и N_2 соответственно, тогда показатель качества разбиения будет равен:

$$\mathrm{Gini}_{\mathrm{split}}(T) = \frac{N_1}{N} \cdot \mathrm{Gini}(T_1) + \frac{N_2}{N} \cdot \mathrm{Gini}(T_2)$$

Механизм отсечения дерева

- Механизм отсечения дерева, оригинальное название minimal cost-complexity tree pruning, – наиболее серьезное отличие алгоритма CART от других алгоритмов построения дерева. CART рассматривает отсечение как получение компромисса между двумя проблемами: получение дерева оптимального размера и получение точной оценки вероятности ошибочной классификации.
- Основная проблема отсечения – большое количество всех возможных отсеченных поддеревьев для одного дерева.
- Базовая идея метода – не рассматривать все возможные поддеревья, ограничившись только "лучшими представителями" согласно приведённой ниже оценке.
- Обозначим $|T|$ – число листов дерева, $R(T)$ – ошибка классификации дерева, равная отношению числа неправильно классифицированных примеров к числу примеров в обучающей выборке.

- Определим полную стоимость (оценку/показатель затраты-сложность) дерева T как:

$$C_\alpha(T) = R(T) + \alpha * |T|,$$

- где $|T|$ – число листов (терминальных узлов) дерева, – некоторый параметр, изменяющийся от 0 до бесконечности. Полная стоимость дерева состоит из двух компонент – ошибки классификации дерева и штрафа за его сложность. Если ошибка классификации дерева неизменна, тогда с увеличением полная стоимость дерева будет увеличиваться. Тогда в зависимости от менее ветвистое дерево, дающее большую ошибку классификации может стоить меньше, чем дающее меньшую ошибку, но более ветвистое.

- Определим T_{max} – максимальное по размеру дерево, которое предстоит обрезать. Если мы зафиксируем значение α тогда существует наименьшее минимизируемое поддерево, которое выполняет следующие условия:

1. $C_\alpha\left(T(\alpha)\right) = min_{\,T \leq T_{max}} C_\alpha(T)$
2. $if\ C_\alpha(T) = C_\alpha\left(T(\alpha)\right)\ then\ T(\alpha) \leq T$

- Первое условие говорит, что не существует такого поддерева дерева T_{max} , которое имело бы меньшую стоимость, чем T(α) при этом значении α. Второе условие говорит, что если существует более одного поддерева, имеющего данную полную стоимость, тогда мы выбираем наименьшее дерево.

Выбор финального дерева

- Итак, мы имеем последовательность деревьев для последовательно возрастающих значений α, нам необходимо выбрать лучшее дерево из неё. То, которое мы и будем использовать в дальнейшем. Наиболее очевидным является выбор финального дерева через тестирование на тестовой выборке. Дерево, давшее минимальную ошибку классификации, и будет лучшим.