Лекции по логическим алгоритмам классификации

К. В. Воронцов

24 июня 2010 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `vokov@forecsys.ru`, либо высказанные в обсуждении страницы «Машинное обучение (курс лекций, К.В.Воронцов)» вики-ресурса `www.MachineLearning.ru`.

Содержание

1 Логические алгоритмы классификации

Мы переходим к изучению ещё одного обширного класса алгоритмов классификации, в основе которых лежит принцип *индуктивного вывода логических закономерностей* или *индукции правил* (rule induction, rule learning).

Пусть $\varphi\colon X \to \{0,1\}$ — некоторый предикат, определённый на множестве объектов X. Говорят, что предикат φ *выделяет* или *покрывает* (cover) объект x, если $\varphi(x) = 1$. Предикат называют *закономерностью*, если он выделяет достаточно много объектов какого-то одного класса c, и практически не выделяет объекты других классов (более строгое определение будет дано ниже).

Особую ценность представляют закономерности, которые описываются простой логической формулой. Их называют *правилами* (rules). Процесс поиска правил по выборке называют *извлечением знаний из данных* (knowledge discovery). К знаниям предъявляется особое требование — они должны быть *интерпретируемы*, то есть понятны людям. На практике логические закономерности часто ищут в виде конъюнкций небольшого числа элементарных высказываний. Именно в такой форме люди привыкли выражать свой житейский и профессиональный опыт.

Пример 1.1. (из области медицины). Решается вопрос о целесообразности хирургической операции. Закономерность: если возраст пациента выше 60 лет и ранее он перенёс инфаркт, то операцию не делать — риск отрицательного исхода велик.

Пример 1.2. (из области банковской деятельности). Решается вопрос о выдаче кредита. Закономерность: если заёмщик указал в анкете свой домашний телефон, и его зарплата превышает \$1000 в месяц, и сумма кредита не превышает \$10 000, то кредит можно выдать — риск невозврата мал.

Всякая закономерность классифицирует лишь некоторую часть объектов. Объединив определённое количество закономерностей в композицию, можно получить алгоритм, способный классифицировать любые объекты. *Логическими алгоритмами классификации* будем называть композиции легко интерпретируемых закономерностей. При построении логических алгоритмов возникают три основных вопроса:

- Каков критерий информативности, позволяющий называть предикаты закономерностями? Различные критерии рассматриваются в §1.1.
- Как строить закономерности? Различные методы порождения бинарных предикатов из разнотипной исходной информации рассматриваются в §1.2.
- Как строить алгоритмы классификации на основе закономерностей? Рассматриваются наиболее распространённые типы логических алгоритмов: решающие списки (§1.3), решающие деревья и леса (§1.4), голосование правил (§1.5), алгоритмы вычисления оценок (§1.6).

Напомним основные обозначения. Имеется пространство объектов X и конечное множество имён классов $Y = \{1, \ldots, M\}$. Целевая зависимость $y^*\colon X \to Y$ известна только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется построить алгоритм классификации $a\colon X \to Y$, аппроксимирующий y^* на всём X.

§1.1 Понятие информативности

1.1.1 Эвристическое определение информативности

Интуитивно предикат φ тем более информативен, чем больше он выделяет объектов «своего» класса $c \in Y$ по сравнению с объектами всех остальных «чужих» классов. Свои объекты называют также *позитивными* (positive), а чужие — *негативными* (negative). Введём следующие обозначения:

P_c — число объектов класса c в выборке X^ℓ;

$p_c(\varphi)$ — из них число объектов, для которых выполняется условие $\varphi(x) = 1$;

N_c — число объектов всех остальных классов $Y \setminus \{c\}$ в выборке X^ℓ;

$n_c(\varphi)$ — из них число объектов, для которых выполняется условие $\varphi(x) = 1$.

Для краткости индекс c и аргумент (φ) будем иногда опускать. Предполагается, что $P \geqslant 1$, $N \geqslant 1$ и $P + N = \ell$.

На Рис. 1 показаны четыре части выборки, на которые она разбивается условиями $y_i = c$ и $\varphi(x_i) = 1$.

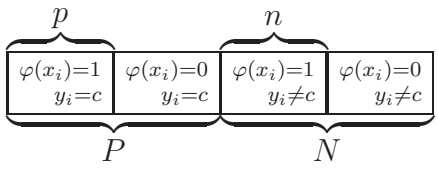


Рис. 1. К определению информативности.

Информативность предиката φ тем выше, чем больше объектов он выделяет, и чем меньше среди них негативных. Выделение негативного объекта является, по сути дела, ошибкой предиката. Не-выделение позитивного объекта также можно считать ошибкой, однако менее серьёзной, поскольку от закономерностей не требуется выделять все объекты. Объект, не выделенный одной закономерностью, может быть выделен другой.

Итак, задача построения информативного предиката φ сводится к оптимизации по двум критериям: $p_c(\varphi) \to \max$ и $n_c(\varphi) \to \min$. Наименее интересны те предикаты, которые либо выделяют слишком мало объектов, либо выделяют позитивные и негативные объекты примерно в той же пропорции, в которой они были представлены во всей выборке, $n : p \approx N : P$.

Введём обозначение E_c для доли негативных среди всех выделяемых объектов, и D_c для доли выделяемых позитивных объектов:

$$E_c(\varphi, X^\ell) = \frac{n_c(\varphi)}{p_c(\varphi) + n_c(\varphi)}, \qquad D_c(\varphi, X^\ell) = \frac{p_c(\varphi)}{\ell}.$$

Опр. 1.1. *Предикат $\varphi(x)$ будем называть логической ε, δ-закономерностью для класса $c \in Y$, если $E_c(\varphi, X^\ell) \leqslant \varepsilon$ и $D_c(\varphi, X^\ell) \geqslant \delta$ при заданных достаточно малом ε и достаточно большом δ из отрезка $[0, 1]$.*

Если $n_c(\varphi) = 0$, то закономерность φ называется чистой или непротиворечивой. Если $n_c(\varphi) > 0$, то закономерность φ называется частичной.

В некоторых случаях имеет смысл ограничиваться поиском только чистых закономерностей. В частности, когда длина выборки мала или заранее известно, что данные практически не содержат шума. Такие прикладные задачи нередко встречаются на практике, например, классификация месторождений редких полезных ископаемых по данным геологоразведки (стр. **??**), где данные стоят дорого, и потому, как правило, тщательно проверяются.

Но чаще данные оказываются неполными и неточными. Таковы многие медицинские и экономические задачи (в частности, задача о выдаче кредитов, стр. **??**).

p	n	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	$\frac{p}{n+p}$	I_c	IGain_c	$\sqrt{p}-\sqrt{n}$
50	0	**50**	50	0.25	50	1	22.65	23.70	7.07
100	50	**50**	−150	0	1.96	0.67	2.33	1.98	2.93
50	9	41	**5**	0.16	**5**	**0.85**	7.87	7.94	4.07
5	0	5	**5**	0.03	**5**	**1**	2.04	3.04	2.24
100	0	**100**	100	**0.5**	100	1	52.18	53.32	10.0
140	20	**120**	40	**0.5**	6.67	0.88	37.09	37.03	7.36

Таблица 1. Критерии информативности предикатов при $P = 200$, $N = 100$ и различных p и n. Левые колонки соответствуют «наивным» свёрткам двух характеристик p и n в один критерий. Три правые колонки соответствуют более адекватным свёрткам, которые рассматриваются далее.

Когда некоторая доля ошибок неизбежна, вполне допустимо пользоваться частичными закономерностями. Кроме того, существуют способы построения корректных алгоритмов на основе частичных закономерностей, например, взвешенное голосование. Во всех этих случаях приходится отбирать предикаты φ по двум критериям $p_c(\varphi)$ и $n_c(\varphi)$ одновременно. Что лучше: уменьшение n на 1 или увеличение p на 10? Для целенаправленного поиска лучших закономерностей удобно иметь критерий информативности, способный дать обоснованный ответ на этот вопрос.

Оказывается, предложить адекватную свёртку критериев n и p не так просто. Например, в статье [26] приведено около двух десятков различных критериев. На первый взгляд, закономерности можно сравнивать по разности $p-n$ или отношению p/n. В таблице 1 собраны контрпримеры, показывающие неадекватность «наивных» критериев информативности. Жирным шрифтом выделены пары примеров, в которых верхняя закономерность с очевидностью ценнее нижней, тем не менее, критерий принимает на них равные значения, или даже большее значение на нижней закономерности.

1.1.2 Статистическое определение информативности

Адекватную скалярную характеристику информативности даёт техника проверки статистических гипотез.

Пусть X — вероятностное пространство, выборка X^ℓ — *простая*, то есть случайная, независимая, одинаково распределённая (independent, identically distributed — i.i.d.), $y^*(x)$ и $\varphi(x)$ — случайные величины. Допустим, справедлива гипотеза о независимости событий $\{x\colon y^*(x) = c\}$ и $\{x\colon \varphi(x) = 1\}$. Тогда вероятность реализации пары (p, n) подчиняется гипергеометрическому распределению[1] [23]:

$$h_{P,N}(p,n) = \frac{C_P^p C_N^n}{C_{P+N}^{p+n}}, \quad 0 \leqslant p \leqslant P, \quad 0 \leqslant n \leqslant N, \tag{1.1}$$

где $C_m^k = \frac{m!}{k!(m-k)!}$ — биномиальные коэффициенты, $0 \leqslant k \leqslant m$.

Если вероятность (1.1) мала, и тем не менее пара (p, n) реализовалась, то гипотеза о независимости должна быть отвергнута. Чем меньше значение вероятности,

[1] Строго говоря, функция $h_{P,N}(p,n)$ является одномерным распределением отдельно по p и отдельно по n при каждом фиксированном значении $p+n$.

тем более значимой является связь между y^* и φ. Можно сказать и так: если реализовалось маловероятное событие, то, скорее всего, оно не случайно, а закономерно.

Опр. 1.2. *Информативность предиката $\varphi(x)$ относительно класса $c \in Y$ по выборке X^ℓ есть*[2]

$$I_c(\varphi, X^\ell) = -\ln h_{P_c,N_c}(p_c(\varphi), n_c(\varphi)).$$

Предикат $\varphi(x)$ будем называть статистической закономерностью для класса c, если $I_c(\varphi, X^\ell) \geqslant I_0$ при заданном достаточно большом I_0.

Порог информативности I_0 выбирается так, чтобы ему соответствовало достаточно малое значение вероятности, называемое *уровнем значимости*. Для каждой задачи он должен выбираться индивидуально.

Описанный критерий применяется в статистике для проверки гипотезы о независимости событий и называется *точным тестом Фишера* (Fisher's exact test, FET). Точным — потому что известны и другие критерии независимости, но они являются лишь асимптотическими приближениями гипергеометрического распределения. Преимущество асимптотических критериев в том, что они вычисляются по более простым формулам. Но при этом они допускают значительную погрешность на малых выборках (до нескольких десятков объектов). Этот недостаток может оказаться весьма ощутимым, когда информативность предикатов оценивается на малых выборках или на частях выборки. В частности, это происходит при синтезе решающих списков и деревьев.

Эффективное вычисление информативности. Заметим, что вычисление логарифма $h_{P,N}(p,n)$ сводится к сложению 9 чисел вида $\ln(k!)$. Когда длина выборки ℓ фиксирована, можно заранее составить таблицу логарифмов факториалов $L_k = \ln k!$ по рекуррентной формуле $L_1 = 0$; $L_k = L_{k-1} + \ln k$ для всех $k = 2, \ldots, \ell$.

Другой способ вычисления $\ln k!$ связан с применением формулы Стирлинга:

$$\ln k! \approx k \ln k - k + \frac{1}{2} \ln 2k\pi + \frac{1}{12k} - \frac{1}{360k^2}. \tag{1.2}$$

Эта формула даёт достаточно точное приближение. При $k = 2$ различие появляется только в пятой значащей цифре, при $k = 10$ — в десятой, и с ростом n точность возрастает[3]. Более точное приближение даёт формула Рамануджана:

$$\log k! \approx k \log k - k + \frac{1}{6} \ln(k(1 + 4k(1 + 2k))) + \frac{1}{2} \ln \pi.$$

Обе формулы можно применять даже когда P, N, p, n не являются целыми числами. Это позволяет обобщить понятие информативности на тот случай, когда объекты не равнозначны.

Сопоставление двух критериев информативности. Какой из критериев предпочтительнее — эвристический или статистический?

[2] Будем также пользоваться сокращёнными формами записи $I_c \equiv I_c(\varphi) \equiv I_c(\varphi, X^\ell)$.

[3] Тем не менее, пользоваться аппроксимацией Стирлинга следует с осторожностью. Экспонента от (1.2) может намного отличаться от $k!$; сумма членов гипергеометрического распределения (1.1), вычисленных по формуле Стирлинга, может существенно отличаться от единицы.

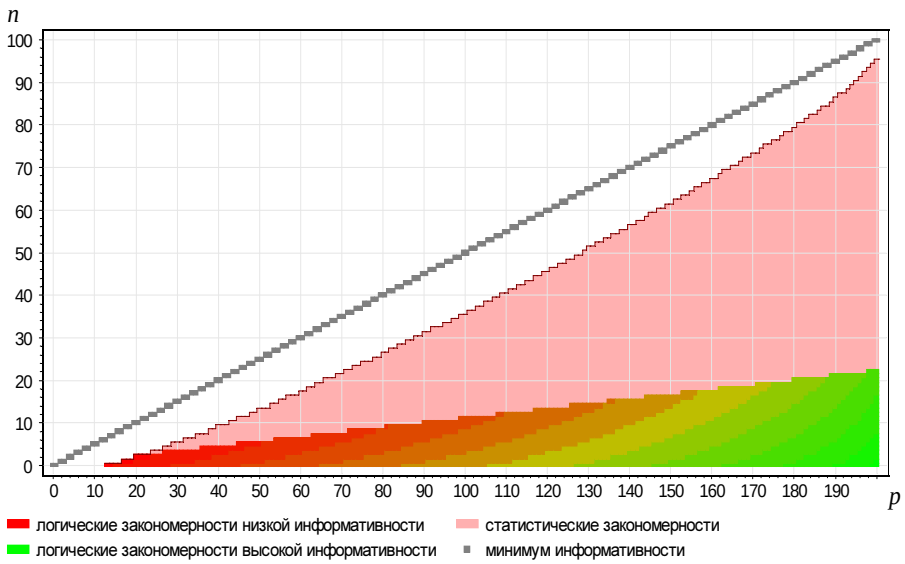


Рис. 2. Области логических ε, δ-закономерностей (при $\varepsilon = 0.1$) и статистических закономерностей (при $I_0 = 5$) в координатах (p, n) при $P = 200$, $N = 100$.

При разумных сочетаниях параметров ε и I_0 эвристический критерий практически всегда оказывается строже статистического, см. Рис. 2. Имеется довольно обширная область статистических закономерностей, для которых вероятность случайной реализации крайне низка, в то же время, они допускают слишком много ошибок и не являются логическими закономерностями в смысле ε, δ-критерия.

На самом деле полезны оба определения. В дальнейшем мы увидим, что в зависимости от ситуации оказывается более целесообразным применять то эвристический, то статистический критерий информативности.

1.1.3 Энтропийное определение информативности

Ещё один способ определения информативности вытекает из теории информации. Напомним, что если имеются два исхода ω_0, ω_1 с вероятностями q_0 и $q_1 = 1 - q_0$, то *количество информации*, связанное с исходом ω_i, по определению равно $-\log_2 q_i$. Это математическое ожидание числа бит, необходимых для записи информации о реализации исходов ω_i при использовании оптимального (наиболее экономного) кодирования. *Энтропия* определяется как матожидание количества информации:

$$H(q_0, q_1) = -q_0 \log_2 q_0 - q_1 \log_2 q_1.$$

Будем считать появление объекта класса c исходом ω_0, а появление объекта любого другого класса исходом ω_1. Тогда, подставляя вместо вероятностей частоты, можно оценить *энтропию выборки* X^ℓ:

$$\hat{H}(P, N) = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right).$$

Допустим, стало известно, что предикат φ выделил p объектов из P, принадлежащих классу c, и n объектов из N, не принадлежащих классу c. Тогда энтропия выборки $\left\{x \in X^\ell \;\middle|\; \varphi(x) = 1\right\}$ есть $\hat{H}(p, n)$. Вероятность появления объекта из этой

выборки оценивается как $\frac{p+n}{P+N}$. Аналогично, энтропия выборки $\left\{x \in X^\ell \;\middle|\; \varphi(x) = 0\right\}$ есть $\hat{H}(P-p, N-n)$, а вероятность появления объекта из неё оценивается как $\frac{P-p+N-n}{P+N}$. Таким образом, энтропия всей выборки после получения информации φ становится равна

$$\hat{H}_\varphi(P, N, p, n) = \frac{p+n}{P+N}\hat{H}(p, n) + \frac{P+N-p-n}{P+N}\hat{H}(P-p, N-n).$$

В итоге уменьшение энтропии составляет

$$\mathrm{IGain}_c(\varphi, X^\ell) = \hat{H}(P, N) - \hat{H}_\varphi(P, N, p, n).$$

Это и есть *информационный выигрыш* (information gain) — количество информации об исходном делении выборки на два класса «c» и «не c», которое содержится в предикате φ. Таким образом, появляется ещё одно, альтернативное, определение закономерности.

Опр. 1.3. *Предикат φ является закономерностью по энтропийному критерию информативности, если* $\mathrm{IGain}_c(\varphi, X^\ell) > G_0$ *при некотором достаточно большом* G_0.

Теорема 1.1. *Энтропийный критерий* IGain_c *асимптотически эквивалентен статистическому* I_c*:*

$$\mathrm{IGain}_c(\varphi, X^\ell) \to \frac{1}{\ell \ln 2} I_c(\varphi, X^\ell) \quad \textit{при } \ell \to \infty.$$

Для доказательства достаточно применить к статистическому определению (1.1) формулу Стирлинга, отбросив в ней члены порядка $O(\ell^{-1})$.

Несмотря на асимптотическую эквивалентность, значения I_c и IGain_c могут заметно отличаться при малых n или p. Согласно таблице 1, критерий IGain_c полагает, что «маломощная» закономерность $(n, p) = (0, 5)$ лучше, чем «полное отсутствие закономерности» $(50, 100)$, тогда как точный тест I_c показывает противоположное. Иными словами, критерий IGain завышает информативность маломощных закономерностей. Ситуации малых n или p вовсе не экзотичны, они регулярно возникают при построении решающих списков и деревьев. Точный критерий может давать в этих ситуациях ощутимые преимущества [32]. Однако на практике чаще используется энтропийный критерий, поскольку он проще вычисляется.

1.1.4 Многоклассовая информативность

Когда число классов превышает 3, приходится оценивать информативность не только таких предикатов, которые отделяют один класс от остальных, но и таких, которые отделяют некоторую группу классов от остальных.

Статистический критерий обобщает статистическое определение информативности 1.2 на случай произвольного числа классов $Y = \{1, \ldots, M\}$:

$$I(\varphi, X^\ell) = -\ln \frac{C_{P_1}^{p_1} \cdots C_{P_M}^{p_M}}{C_\ell^p},$$

где P_c — число объектов класса c в выборке X^ℓ, из них p_c объектов выделяются предикатом φ, $p = p_1 + \cdots + p_M$.

Энтропийный критерий для случая большого числа классов строится из тех же соображений, что в разделе 1.1.3, и также является асимптотическим приближением статистического:

$$I(\varphi, X^\ell) = \sum_{c\in Y} h\left(\frac{P_c}{\ell}\right) - \frac{p}{\ell}\sum_{c\in Y} h\left(\frac{p_c}{p}\right) - \frac{\ell-p}{\ell}\sum_{c\in Y} h\left(\frac{P_c-p_c}{\ell-p}\right),$$

где введена функция $h(z) \equiv -z\log_2 z$.

1.1.5 Взвешенная информативность

Цена ошибки на разных объектах может быть различной. Например, она может отличаться для разных классов: при выдаче кредитов «пропуск цели» (т. е. ненадёжного заёмщика) обходится банку существенно дороже, чем «ложная тревога» (отказ хорошему заёмщику). Обучающие объекты из класса «плохие заёмщики» должны учитываться с бóльшим весом при поиске логических закономерностей.

Пусть задан вектор неотрицательных весов объектов $w = (w_i)_{i=1}^\ell$ с условием нормировки $\sum_{i=1}^\ell w_i = \ell$. Определим величины, аналогичные P, N, $p(\varphi)$, $n(\varphi)$:

$$\begin{aligned} P_c^w &= \sum_{i=1}^{\ell} w_i[y_i = c]; \qquad & p_c^w(\varphi) &= \sum_{i=1}^{\ell} w_i[y_i = c][\varphi(x_i) = 1]; \\ N_c^w &= \sum_{i=1}^{\ell} w_i[y_i \neq c]; \qquad & n_c^w(\varphi) &= \sum_{i=1}^{\ell} w_i[y_i \neq c][\varphi(x_i) = 1]; \end{aligned} \tag{1.3}$$

Определение логической ε,δ-закономерности, статистическая и энтропийная информативность легко обобщаются на этот случай. Для вычисления гипергеометрического распределения (1.1) можно воспользоваться обобщённым определением факториала через гамма-функцию [4], однако это сопряжено с трудоёмкими вычислениями. Другой подход — округлять значения P_c^w, N_c^w, p_c^w, n_c^w до ближайших целых, но при этом снижается точность оценивания информативности, особенно, при малых p или n. Разумный компромисс между точностью приближения и скоростью вычислений даёт линейная аппроксимация:

$$\ln z! \approx (\lfloor z+1\rfloor - z)\ln\lfloor z\rfloor! + (z - \lfloor z\rfloor)\ln\lfloor z+1\rfloor!, \quad z \in \mathbb{R}.$$

§1.2 Методы поиска информативных закономерностей

Итак, информативные закономерности служат исходным сырьём для построения логических алгоритмов классификации. Возникает вопрос: в каком множестве предикатов следует искать информативные закономерности? Это множество называют ещё *пространством поиска* (search space).

Наиболее прост тот случай, когда все исходные признаки являются бинарными, $f_j\colon X \to \{0,1\}$, $j = 1,\ldots,n$. Тогда пространство поиска образуется самими признаками и всевозможными булевыми функциями, которые из этих признаков можно построить. При этом достаточно строить только дизъюнктивные нормальные формы, поскольку любую булеву функцию можно записать в виде ДНФ [17]. Более того,

в качестве закономерностей можно брать только конъюнкции признаков и их отрицаний, а дизъюнкцию реализовать как корректирующую операцию, например, как голосование по большинству или старшинству (см. **??**).

Несколько сложнее дело обстоит в тех случаях, когда объекты описываются разнотипными признаками: номинальными, порядковыми, количественными, и т. д, или когда объекты представляют собой более сложные структуры: тексты, изображения или сигналы. Подобного рода ситуации чаще возникают на практике, чем «рафинированный» бинарный случай. Тогда пространством поиска становятся всевозможные бинарные функции от исходных признаков. Процесс построения таких функций называют *бинаризацией* исходной информации. Как правило, допустимых способов бинаризации оказывается настолько много, что возникает новый вопрос: как сократить пространство поиска, избежав оценивания информативности для огромного числа заведомо неинформативных или почти одинаковых предикатов.

В этом параграфе рассматриваются наиболее употребительные методы построения и отбора бинарных признаков.

1.2.1 Бинаризация количественных признаков

Произвольный признак $f\colon X \to D_f$ порождает семейство предикатов, проверяющих попадание значения $f(x)$ в определённые подмножества множества D_f. Ниже перечисляются наиболее типичные конструкции такого вида.

- Если f — номинальный признак:

$$\beta(x) = [f(x) = d\,], \quad d \in D_f;$$
$$\beta(x) = [f(x) \in D'], \quad D' \subset D_f.$$

- Если f — порядковый или количественный признак:

$$\beta(x) = [f(x) \leqslant d\,], \quad d \in D_f;$$
$$\beta(x) = [\,d \leqslant f(x) \leqslant d'\,], \quad d, d' \in D_f, \quad d < d'.$$

В случае количественных признаков $f\colon X \to \mathbb{R}$ имеет смысл брать только такие значения порогов d, которые по-разному разделяют выборку X^ℓ. Если исключить тривиальные разбиения, обращающие $\beta(x)$ в 0 или 1 на всей выборке, то таких значений окажется не более $\ell - 1$. Например, можно взять пороги вида

$$d_i = \frac{f^{(i)} + f^{(i+1)}}{2}, \quad f^{(i)} \neq f^{(i+1)}, \quad i = 1, \ldots, \ell - 1, \tag{1.4}$$

где $f^{(1)} \leqslant \ldots \leqslant f^{(\ell)}$ — последовательность значений признака f на объектах выборки $f(x_1), \ldots, f(x_\ell)$, упорядоченная по возрастанию (вариационный ряд), см. Рис. 3.

Описанные способы позволяют получить огромное количество предикатов. Если в дальнейшем они будут использоваться для синтеза конъюнкций, то для сокращения перебора имеет смысл сразу отобрать из них наиболее информативные. В случае порядковых и количественных признаков данная задача решается путём оптимального разбиения диапазона значений признака на зоны.

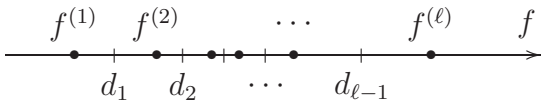


Рис. 3. Вариационный ряд значений признака $f(x)$ и пороги d_i.

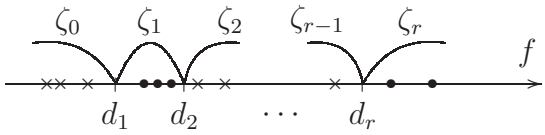


Рис. 4. Начальное разбиение на зоны позитивных ($\times$) и негативных ($\bullet$) объектов.

Разбиение диапазона значений признака на информативные зоны. Пусть $f\colon X \to \mathbb{R}$ — числовой признак, $d_1, \ldots, d_r$ — возрастающая последовательность порогов. *Зонами* значений признака f будем называть предикаты вида

$$\begin{aligned}
&\zeta_0(x) = [f(x) < d_1];\\
&\zeta_s(x) = [d_s \leqslant f(x) < d_{s+1}], \qquad s = 1, \ldots, r-1;\\
&\zeta_r(x) = [d_r \leqslant f(x)].
\end{aligned}$$

Алгоритм 1.1 начинает с разбиения на «мелкие зоны». Пороги определяются по формуле (1.4) и проходят между всеми парами точек x_{i-1}, x_i, ровно одна из которых принадлежит классу c (шаги 2–4). Нетрудно показать, что расстановка порогов между точками класса c или между точками не класса c приведёт только к уменьшению информативности зон. Итак, начальное разбиение состоит из чередующихся зон «только c — только не c», как показано на Рис. 4.

Далее зоны укрупняются путём слияния троек соседних зон. Именно троек — слияние пар приводит к нарушению чередования «c — не c», в результате некоторые «мелкие зоны» могут так и остаться неслитыми. Зоны сливаются до тех пор, пока информативность некоторой слитой зоны $\zeta_{i-1} \vee \zeta_i \vee \zeta_{i+1}$ превышает информативность исходных зон ζ_{i-1}, ζ_i и ζ_{i+1}, либо пока не будет получено заданное количество зон r. Каждый раз выбирается та тройка, при слиянии которой достигается максимальный выигрыш информативности.

Этот алгоритм имеет трудоёмкость $O(\ell^2)$. Его можно заметно ускорить, если на каждой итерации сливать не одну тройку зон, а $\tau\ell$ троек с достаточно большим выигрышем δI_i, при условии, что они не перекрываются. Число τ — ещё один параметр алгоритма, $0 < \tau < 0.5$. В этом случае трудоёмкость составляет $O\bigl(\ell/\sqrt{\tau}\bigr)$.

Замечание 1.1. Значения признака можно перекодировать в номера зон. Такое преобразование признаков называют *дискретизацией*. При этом описания объектов упрощаются и часть информации теряется. Однако с точки зрения классификации объектов класса c потеря не очень существенна, поскольку разбиение на зоны производилось по критерию информативности I_c. Дискретизацию можно применять как способ сжатия информации с наименьшими потерями.

Замечание 1.2. Относительно другого класса $c' \neq c$ разбиение диапазона значений признака на информативные зоны может оказаться совершенно иным. Алгоритм 1.1 легко приспособить и для «универсального» разбиения на зоны, учитывающего сразу все классы. Для этого достаточно заменить критерий информативности I_c многоклассовым критерием, стр. 7.

Алгоритм 1.1. Жадный алгоритм слияния зон

Вход:
$f(x)$ — признак;
$c \in Y$ — выделенный класс;
$X^\ell = \{x_i, y_i\}_{i=1}^\ell$ — выборка, упорядоченная по возрастанию $f(x_i)$;
r — желаемое количество зон;
δ_0 — порог слияния зон (по умолчанию $\delta_0 = 0$).
Выход:
$D = \{d_1, \ldots, d_r\}$ — строго возрастающая последовательность порогов;

$D := \varnothing$;
для всех $i = 2, \ldots, \ell$
  если $f(x_{i-1}) \neq f(x_i)$ и $[y_{i-1} = c] \neq [y_i = c]$ **то**
    добавить новый порог $\frac{1}{2}(f(x_{i-1}) + f(x_i))$ в конец последовательности D;
повторять
  для всех $d_i \in D$, $i = 1, \ldots, |D| - 1$
    вычислить выигрыш от слияния тройки соседних зон ζ_{i-1}, ζ_i, ζ_{i+1}:
    $\delta I_i := I_c(\zeta_{i-1} \vee \zeta_i \vee \zeta_{i+1}) - \max\{I_c(\zeta_{i-1}), I_c(\zeta_i), I_c(\zeta_{i+1})\}$;
  найти тройку зон, для которой слияние наиболее выгодно:
  $i := \arg\max_s \delta I_s$;
  если $\delta I_i > \delta_0$ **то**
    слить зоны ζ_{i-1}, ζ_i, ζ_{i+1} удалив пороги d_i и d_{i+1} из последовательности D;
пока $|D| > r + 1$.

1.2.2 Поиск закономерностей в форме конъюнкций

Пусть $\mathscr{B}$ — конечное множество предикатов, которые мы будем называть *элементарными*. Введём множество конъюнкций с ограниченным числом термов из $\mathscr{B}$:

$$\mathscr{K}_K[\mathscr{B}] = \left\{\varphi(x) = \beta_1(x) \wedge \ldots \wedge \beta_k(x) \;\middle|\; \beta_1, \ldots, \beta_k \in \mathscr{B},\ k \leqslant K\right\}.$$

Число термов k в конъюнкции называется её *рангом*. Конъюнкции небольшого ранга обладают важным преимуществом — они имеют вид привычных для человека логических высказываний и легко поддаются содержательной интерпретации.

Максимальный ранг конъюнкций K обычно устанавливают от 3 до 7, опять-таки из соображений интерпретируемости — почти невозможно уследить за смыслом высказываний, содержащих слишком большое число условий.

Поиск наиболее информативных конъюнкций в общем случае требует полного перебора. Число допустимых конъюнкций есть $O(|\mathscr{B}|^K)$, и может оказаться настолько большим, что полный перебор станет практически неосуществим. Представим вполне реалистичную ситуацию: имеется 100 числовых признаков; диапазон значений каждого признака разбит на 10 зон, т. е. порождает 10 элементарных предикатов. Тогда число конъюнкций ранга K равно $C_{100}^K 10^K$. Уже при $K = 5$ эта величина имеет порядок 10^{13}, что исключает возможность полного перебора на современных компьютерах.

На практике используют различные эвристики для сокращённого целенаправленного поиска конъюнкций, близких к оптимальным. Идея всех этих методов заклю-

чается в том, чтобы не перебирать огромное количество заведомо неинформативных предикатов. Начнём с наиболее простого метода.

«Градиентный» алгоритм синтеза конъюнкций. Поставим каждой конъюнкции φ в соответствие её *окрестность* — множество конъюнкций $V(\varphi)$, получаемых из φ путём элементарных модификаций: добавлением, изъятием или модификацией одного из термов конъюнкции.

Начиная с заданной конъюнкции φ_0 (например, пустой), строится последовательность конъюнкций $\varphi_0, \varphi_1, \ldots, \varphi_t, \ldots$, в которой каждая следующая конъюнкция φ_t выбирается из окрестности предыдущей $V_t = V(\varphi_{t-1})$ по критерию максимума информативности (шаг 3).

Наиболее перспективными с точки зрения дальнейших модификаций считаются конъюнкции с максимальной информативностью. Однако «перспективная» — не означает лучшая, так как конъюнкции с высокой информативностью могут допускать много ошибок (см. сравнение логической и статистической информативности на стр. 6). Поэтому на каждом шаге t выделяется «наилучшая» конъюнкция, удовлетворяющая дополнительному условию $E_c(\varphi_t^*) < \varepsilon$, и в общем случае не совпадающая с наиболее перспективной φ_t.

Поскольку множество конъюнкций, по-разному классифицирующих выборку, конечно, итерационный процесс сходится за конечное число шагов к некоторой «локально неулучшаемой» конъюнкции.

Разумеется, ни о каком градиенте в прямом смысле слова речь не идёт. Имеется в виду, что данный алгоритм, по аналогии с градиентным спуском, выбирает на каждой итерации наилучшую из ближайших точек пространства поиска.

Критерий информативности I и функция окрестности V, вообще говоря, являются параметрами алгоритма. При формировании окрестности V можно применять различные эвристики:

— ограничивать максимальный ранг конъюнкций K;
— отбирать из окрестности только ε, δ-закономерности;
— разрешать только добавления термов;
— чередовать серии добавлений термов с сериями удалений;
— разрешать модификацию нескольких термов одновременно;
— разрешать изменение порога α, но не признака f, в термах $[f(x) < \alpha]$;
— варьировать пороги α только в пределах соседних зон.

Варьируя параметры Алгоритма 1.2, можно получать различные процедуры поиска или улучшения информативных конъюнкций. Рассмотрим более подробно четыре варианта этого алгоритма.

Жадный алгоритм синтеза конъюнкции использует только операцию добавления термов. Начальным приближением является пустая конъюнкция (не содержащая термов). Недостаток жадной стратегии в том, что она может уводить в сторону от глобального максимума информативности. Терм, найденный на k-м шаге, перестаёт быть оптимальным после добавления последующих термов. Тем не менее, в ряде практических задач эта простая эвристика демонстрирует способность находить неплохие закономерности.

Алгоритм 1.2. «Градиентный» алгоритм синтеза конъюнкции

Вход:
X^ℓ — обучающая выборка;
φ_0 — начальное приближение;
$c \in Y$ — класс, для которого строится конъюнкция;
$t_{\max}$ — максимальное число итераций;
d — параметр критерия останова;
ε — порог доли ошибок для отбора конъюнкций;

Выход:
конъюнкция φ;

$I^* := I_c(\varphi_0, X^\ell)$;
для всех $t = 1, \ldots, t_{\max}$
  текущее множество перебираемых конъюнкций: $V_t := V(\varphi_{t-1})$;
  наиболее перспективная конъюнкция: $\varphi_t := \arg\max\limits_{\varphi \in V_t} I_c(\varphi, X^\ell)$;
  наилучшая конъюнкция на t-й итерации: $\varphi_t^* := \arg\max\limits_{\varphi \in V_t:\, E_c(\varphi) < \varepsilon} I_c(\varphi, X^\ell)$;
  если $I_c(\varphi_t^*) > I^*$ **то**
    запомнить, на какой итерации была получена наилучшая конъюнкция:
    $t^* := t$; $\varphi^* := \varphi_t^*$; $I^* := I_c(\varphi^*)$
  если $t - t^* > d$ (конъюнкция не улучшилась за последние d шагов) **то**
    выход;
вернуть φ^*;

Стохастический локальный поиск (stochastic local search, SLS) также начинает с пустой конъюнкции, но использует полный набор возможных модификаций. Это преимущество по сравнению с жадным алгоритмом, так как появляется возможность удалять и заменять неоптимальные термы. С другой стороны, мощность окрестности $|V(\varphi)|$ может оказаться настолько большой, что перебор всех допустимых модификаций станет нерентабельным. Поэтому в SLS строится не вся окрестность, а только некоторое её случайное подмножество. Максимальная допустимая мощность этого подмножества задаётся как дополнительный параметр алгоритма $V_{\max}$.

Для улучшения конъюнкции φ, построенной жадным наращиванием или SLS, к ней применяют методы «финальной шлифовки» — стабилизацию и редукцию.

Процедура стабилизации пытается улучшить конъюнкцию φ, удаляя или заменяя по одному терму. В отличие от SLS, перебираются все возможные удаления и замены. Модификации производятся до тех пор, пока возрастает информативность конъюнкции $I_c(\varphi, X^\ell)$. Стабилизация повышает устойчивость алгоритма относительно малых вариаций обучающей выборки или других условий обучения (например, генератора псевдослучайной последовательности в SLS). В результате стабилизации найденные конъюнкции часто сходятся к одним и тем же локальным максимумам информативности. Обычно это положительно сказывается на интерпретируемости правил. Эксперт больше доверяет правилу, когда видит, что попытки скорректировать его «вручную» приводят только к его ухудшению.

Процедура редукции отличается тем, что термы только удаляются, а информативность вычисляется по независимой *контрольной выборке* X^k, составленной из объектов, не участвовавших в построении конъюнкции φ. Контрольную выборку формируют до начала обучения, выделяя из массива исходных данных около 30% объектов, как правило, случайным образом. При этом объекты разных классов распределяются в той же пропорции, что и во всей выборке (этот принцип отбора называется *стратификацией* выборки). Смысл редукции в том, чтобы проверить, не является ли найденная конъюнкция избыточно сложной, и либо упростить её, либо вовсе признать неудачной. Упрощение повышает общность логического правила, поскольку множество выделяемых им объектов расширяется. Недостаток редукции в том, что она требует оставить значительную долю данных для контроля, уменьшив представительность обучающей выборки. Однако при разумном выборе соотношения $\ell : k$ поиск правил по X^ℓ с последующей редукцией по X^k может давать лучшие результаты, чем поиск по $X^\ell \cup X^k$ без редукции.

Генетический алгоритм синтеза конъюнкций (Genetic Algorithm, GA) можно рассматривать как дальнейшее усовершенствование SLS на основе идей дарвиновской эволюции. Главное отличие GA от SLS в том, что на каждом шаге отбирается не одна наилучшая конъюнкция, а целое множество лучших конъюнкций, называемое *популяцией*. Из них порождается большое количество конъюнкций-*потомков* с помощью двух *генетических операций* — скрещивания и мутации. *Скрещивание* (crossingover) — это образование новой конъюнкции путём обмена термами между двумя членами популяции. В роли *мутаций* выступают уже знакомые операции добавления, замещения и удаления термов. Таким способом можно получить огромное количество потомков, но на практике строят лишь ограниченное число, не более T_1 потомков путём случайных скрещиваний и мутаций. Затем производится *естественный отбор*, в результате которого в следующее поколение переходят не более T_0 наиболее информативных потомков. Обычно берут $T_0 \ll T_1$.

Генетические алгоритмы отличаются большим разнообразием всевозможных эвристик, заимствованных непосредственно из живой природы. Например, в GA легко встроить процедуру селекции или *искусственного отбора*, порождая потомков только от наилучших конъюнкций, или задавая распределение вероятностей на популяции так, чтобы вероятность стать родителем увеличивалась с ростом информативности. Можно организовывать несколько параллельно развивающихся популяций (островная модель эволюции), чтобы увеличить разнообразие конъюнкций. Эти и другие эвристики описаны в обширной литературе по генетическим алгоритмам, см. например [37, 15, 7, 3].

Поиск информативных конъюнкций как задача отбора признаков. Для поиска конъюнктивных закономерностей можно также приспособить многие методы отбора признаков (features selection), описанные в **??**. Для этого достаточно заменить в них функционал качества — вместо минимизации средней ошибки искать максимум информативности. В частности, метод шаговой регрессии Add-Del (см. **??**) приводит к построению конъюнкций путём попеременного наращивания и редукции. Многорядный итерационный алгоритм МГУА (см. **??**) аналогичен «полужадному» алгоритму ТЭМП, который будет рассмотрен подробнее в разделе 1.5.3. Случайный

поиск с адаптацией (см. **??**) и генетические алгоритмы представляют собой, по сути дела, продвинутые обобщения стохастического локального поиска.

Информативные конъюнкции являются универсальными «строительными блоками» для многих логических алгоритмов классификации. Далее мы увидим, как из них конструируются решающие списки и алгоритмы взвешенного голосования.

1.2.3 Формы закономерностей

Конъюнкции над предикатами вида $\beta(x) = [d \leqslant f(x) \leqslant d']$ описывают области пространства X, имеющие форму гиперпараллелепипедов. На практике применяются и другие форму закономерностей, например, шары или гиперплоскости. Выбор формы определяется особенностями конкретной задачи. В общем случае к форме предъявляются два требования.

- *Интерпретируемость*. Условие $\varphi(x) = 1$ должно выражаться на естественном языке в форме, понятной эксперту. Для этого, в частности, закономерность $\varphi(x)$ должна зависеть от небольшого числа признаков $\omega \subseteq \{1, \ldots, n\}$. Обычно $|\omega|$ ограничивают сверху числом от 2 до 7. Найденные сочетания признаков ω могут либо подтверждать существующие представления о взаимосвязях между признаками, либо указывать на существование ранее неизвестных взаимосвязей, и тогда можно говорить о получении новых знаний из данных (knowledge discovery). Некоторые из найденных сочетаний признаков ω могут оказаться не интерпретируемыми с содержательной точки зрения; такие закономерности отвергаются экспертами как «ложные».

- *Эффективность поиска*. Должны существовать эффективные методы поиска закономерностей по конечной выборке U в рамках выбранного семейства предикатов Φ: $I(\varphi, U) \to \max_{\varphi \in \Phi}$. Как правило, наиболее трудоёмким является поиск информативных наборов признаков $\omega \subseteq \{1, \ldots, n\}$.

Параметрическое семейство шаров:

$$\varphi_c(x) = \big[\rho(x, x_0) \leqslant r_0\big], \tag{1.5}$$

где $\rho(x, x_0)$ — метрика в пространстве объектов X. Параметрами являются центр шара x_0, его радиус r_0, и сама функция расстояния ρ. В качестве центров берут либо обучающие объекты, либо центры локальных сгущений, найденные каким-либо алгоритмом кластеризации. Радиусы шаров можно подбирать аналогично выделению зон — формируется множество $\ell - 1$ пороговых значений, по-разному разделяющих выборку, и из них выбирается такое значение радиуса, при котором достигается максимум информативности предиката (1.5).

Функция расстояния ρ, как правило, задаётся в виде линейной комбинации элементарных метрик по некоторому набору признаков $\omega \subseteq \{1, \ldots, n\}$:

$$\rho_\omega(x, x') = \sum_{j \in \omega} \alpha_j |f_j(x) - f_j(x')|,$$

где набор ω и коэффициенты α_j предполагается оптимизировать по выборке.

Выделение объекта x шарообразной закономерностью $\varphi(x)$ легко интерпретируется в терминах сходства: «объект x относится к классу c потому, что он близок к эталонному объекту x_0, лежащему в классе c, по совокупности признаков ω». Такого рода объяснения хорошо воспринимаются в тех предметных областях, где распространён прецедентный стиль мышления — в медицине, геологии, социологии, юриспруденции, и др. На закономерностях данного типа основаны *алгоритмы вычисления оценок*, подробно рассматриваемые в §1.6.

Параметрическое семейство полуплоскостей:

$$\varphi_c(x) = \big[\langle x, \alpha\rangle \leqslant \alpha_0\big], \tag{1.6}$$

где $\langle x, \alpha\rangle$ — скалярное произведение в пространстве объектов X. Параметрами являются вектор нормали α и смещение α_0 разделяющей гиперплоскости. Максимизация информативности сводится к подбору таких α и α_0, при которых по одну сторону гиперплоскости лежат объекты преимущественно одного класса.

Закономерности вида (1.6) хорошо интерпретируются, когда среди компонент вектора α мало ненулевых, и разделяющая гиперплоскость проводится в подпространстве небольшой размерности:

$$\langle x, \alpha\rangle = \sum_{j\in\omega} \alpha_j f_j(x).$$

При этом желательно, чтобы признаки обладали следующим свойством монотонности: «чем выше значение признака $f_j(x)$, тем скорее объект x относится к классу c». Тогда коэффициенты α_j интерпретируются как степень важности j-го признака.

Параметрическое семейство областей, описываемых ядром:

$$\varphi(x) = \big[K(x, x_0) \leqslant K_0\big], \tag{1.7}$$

где $K\colon X \times X \to \mathbb{R}$ — *функция ядра* (kernel function), $x_0 \in X$ и $K_0 \in \mathbb{R}$ — параметры предиката. Шары, полуплоскости и гиперпараллелепипеды являются частными случаями ядер. Аналогичный приём *введения ядра* (kernel trick) использовался при обобщении линейной *машины опорных векторов* (SVM) на нелинейную в разделе **??**. Однако здесь, в отличие от SVM, ядро $K(x, x_0)$ не обязано быть скалярным произведением. Функция K может свободно выбираться, исходя из любых априорных соображений. Если таковых в конкретной задаче не имеется, единственное, что остаётся — организовать банк ядер, содержащий «потенциально полезные» функции, в том числе метрики и скалярные произведения. Тогда процедура поиска информативных закономерностей должна включать в себя перебор ядер.

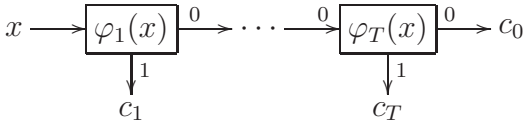
Опять-таки, ядра, зависящие только от небольшого числа признаков, проще поддаются содержательной интерпретации, но для их поиска приходится организовывать перебор подмножеств признаков.

§1.3 Решающие списки

Решающий список (decision list, DL) — это наиболее простой логический алгоритм, как по своей структуре, так и по способу построения.

Алгоритм 1.3. Классификация объекта $x \in X$ решающим списком

1: **для всех** $t = 1, \ldots, T$
2: **если** $\varphi_t(x) = 1$ **то**
3: **вернуть** c_t;
4: **вернуть** c_0.



Опр. 1.4. *Решающий список — это алгоритм классификации* $a\colon X \to Y$, *который задаётся набором закономерностей* $\varphi_1(x), \ldots, \varphi_T(x)$, *приписанных к классам* $c_1, \ldots, c_T \in Y$ *соответственно, и вычисляется согласно Алгоритму 1.3.*

«Особый ответ» c_0 означает отказ алгоритма от классификации объекта x. Обычно такие объекты приписывают классу, имеющему минимальную цену ошибки. Например, в задаче выдачи кредитов отказ алгоритма приведёт к более осторожному решению «не выдавать». В задаче распознавания спама более осторожным будет решение «не спам».

Замечание 1.3. Соседние правила в списке φ_{t-1}, φ_t можно переставлять местами, если только они приписаны к одному классу, $c_{t-1} = c_t$. В общем случае перестановка правил в списке изменяет алгоритм.

Замечание 1.4. Решающий список закономерностей представляет собой частный случай алгоритмической композиции с голосованием по старшинству, рассмотренный в разделах **??** и **??**. Различия разве что терминологические: теперь *базовые алгоритмы* называются закономерностями или правилами. Для построения решающего списка можно было бы применять Алгоритм **??** в неизменном виде. Тем не менее, мы рассмотрим альтернативный алгоритм, в котором, благодаря использованию критерия информативности, удаётся обойтись без априорного параметра λ, устанавливающего величину «штрафа за отказ».

1.3.1 Жадный алгоритм построения решающего списка

Алгоритм 1.4 на каждой итерации строит ровно одно правило φ_t, выделяющее максимальное число объектов некоторого класса c_t и минимальное число объектов всех остальных классов. Для этого на шаге 4 производится поиск наиболее информативного правила $\varphi_t \in \Phi$, допускающего относительно мало ошибок. Семейство правил Φ может быть каким угодно, лишь бы для него существовала эффективная процедура поиска закономерностей. В частности, если Φ — конъюнкции, то на шаге 4 можно применить «градиентный» Алгоритм 1.2. После построения правила φ_t выделенные им объекты изымаются из выборки и алгоритм переходит к поиску следующего правила φ_{t+1} по оставшимся объектам. В итоге выборка оказывается покрытой множествами вида $\{x\colon \varphi_t(x) = 1\}$. Поэтому решающий список называют также *покрывающим набором закономерностей* или *машиной покрывающих множеств* (set covering machine, SCM) [31].

Критерии отбора правил. Почему приходится использовать сразу два критерия отбора правил I_c и E_c? Правило с высокой информативностью I_c вполне может допускать значительную долю ошибок E_c, см. Рис. 2. Это нежелательно, так как в ре-

Алгоритм 1.4. Жадный алгоритм построения решающего списка

Вход:
X^ℓ — обучающая выборка;
Φ — семейство предикатов, из которого выбираются закономерности;
$T_{\max}$ — максимальное допустимое число правил в списке;
$I_{\min}$ — минимальная допустимая информативность правил в списке;
$E_{\max}$ — максимальная допустимая доля ошибок на обучающей выборке;
ℓ_0 — максимальное допустимое число отказов;

Выход:
решающий список $\{\varphi_t, c_t\}_{t=1}^{T}$;

1: $U := X^\ell$;
2: **для всех** $t := 1, \ldots, T_{\max}$
3: $c := c_t$ — выбрать класс из Y, для которого будет строиться правило;
4: найти наиболее информативное правило при ограничении на долю ошибок:
$\varphi_t := \arg\max\limits_{\varphi \in \Phi'} I_c(\varphi, U)$, где $\Phi' = \{\varphi \in \Phi \colon E_c(\varphi, U) \leqslant E_{\max}\}$;
5: **если** $I_c(\varphi_t, U) < I_{\min}$ **то выход**;
6: исключить из выборки объекты, выделенные правилом φ_t:
$U := \{x \in U : \varphi_t(x) = 0\}$;
7: **если** $|U| \leqslant \ell_0$ **то выход**;

шающем списке каждое правило принимает окончательное решение. С другой стороны, правило с небольшой долей ошибок может выделять слишком мало объектов, и по этой причине не являться закономерностью. Совместное использование обоих критериев позволяет отобрать предикаты, удовлетворяющие условиям как статистической, так и логической закономерности.

Критерии останова. В Алгоритме 1.4 одновременно работают три критерия останова: (1) построение заданного числа правил $T_{\max}$; (2) покрытие всей выборки, за исключением не более ℓ_0 объектов; (3) невозможность найти правило с информативностью выше $I_{\min}$ по остатку выборки.

Оптимизация сложности решающего списка. Параметр $E_{\max}$ позволяет найти компромисс между точностью классификации обучающего материала и сложностью списка. Уменьшение $E_{\max}$ приводит к снижению числа ошибок на обучении. С другой стороны, оно ужесточает отбор правил, способствует уменьшению числа объектов, выделяемых отдельными правилами, и увеличению длины списка T. Правила, выделяющие слишком мало объектов, статистически не надёжны и могут допускать много ошибок на независимых контрольных данных. Иными словами, увеличение длины списка при одновременном «измельчении» правил может приводить к эффекту переобучения. Из общих соображений $E_{\max}$ должно быть приблизительно равно доле ошибок, которую мы ожидаем получить как на обучающей выборке, так и вне её. На практике параметр $E_{\max}$ подбирается экспериментально.

Стратегия выбора класса. В описании шага 3 Алгоритма 1.4 ничего не говорится о том, как выбирается класс c_t. Рассмотрим два варианта.

Первый вариант — сначала строятся все правила для первого класса, затем для второго, и так далее. Классы берутся в порядке убывания важности или цены ошибки. Преимущество данного варианта в том, что правила оказываются независимыми — в пределах своего класса их можно переставлять местами. Это улучшает интерпретируемость правил.

Второй вариант — совместить шаги 3 и 4 и выбирать пару $(\varphi_t, c_t) \in \Phi \times Y$, для которой информативность $I_{c_t}(\varphi_t, U)$ максимальна. Тогда правила различных классов могут следовать вперемежку. Доказано, что списки такого типа реализуют более широкое множество функций [34]. При этом улучшается разделяющая способность списка, но ухудшается его интерпретируемость.

На практике первый вариант часто оказывается более удобным. В некоторых случаях правила строятся только для $(M-1)$ классов, а в последний, наименее важный, класс c_0 объекты заносятся «по остаточному принципу».

Обработка пропусков в данных. Решающие списки позволяют легко обойти проблему пропущенных данных. Если для вычисления предиката $\varphi_t(x)$ не хватает данных, то считается, что $\varphi_t(x) = 0$, и обработку объекта x берут на себя следующие правила в списке. Это относится и к стадии обучения, и к стадии классификации.

1.3.2 Разновидности решающих списков

Логику решающего списка или, что то же самое, комитета старшинства, имеют многие алгоритмы, предлагавшиеся в разное время разными авторами под разными названиями. Многочисленные варианты отличаются выбором семейства предикатов Φ, критерием информативности и методом поиска информативных предикатов.

Пример 1.3. Наиболее распространены решающие списки конъюнкций. Они почти идеально соответствуют человеческой логике принятия решений, основанной на последовательной проверке достаточно простых правил. Поэтому решающие списки часто используются для представления знаний, извлекаемых непосредственно из эмпирических данных. Для построения отдельных правил можно использовать Алгоритм 1.4, применяя на шаге 4 любой из методов поиска информативных конъюнкций, например, градиентный Алгоритм 1.2, либо Алгоритм 1.9 с параметром $T_0 = 1$.

Пример 1.4. Алгоритм 1.4 с семейством предикатов Φ вида (1.5) строит покрытие обучающей выборки шарами (data dependent balls). Очень похожий алгоритм описан Маршандом и др. под названием BuildSCM [31]. Решающие списки шаров хорошо работают, когда метрика $\rho(x, x')$ удовлетворяет гипотезе компактности, т. е. близкие объекты часто оказываются в одном классе. Если это не так, то будет построено слишком большое количество шаров небольшого радиуса. Такие алгоритмы обладают невысокой обобщающей способностью.

Пример 1.5. Алгоритм дробящихся эталонов ДРЭТ [12] также основан на покрытии выборки шарами, и отличается тем, что список строится от конца к началу. На первом шаге для каждого из классов определяется шар минимального радиуса, включающий все обучающие объекты данного класса. Если шары разных классов

пересекаются, то для объектов, попавших в пересечение, снова строятся покрывающие шары, но уже меньшего радиуса. Процесс построения шаров продолжается, пока в пересечениях шаров остаются представители разных классов. Построенные шары образуют решающий список в порядке возрастания их радиусов.

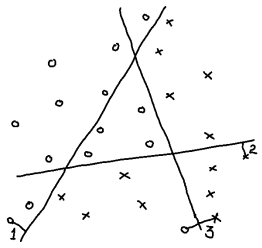


Рис. 5. Построение решающего списка из трёх полуплоскостей.

Непосредственное применение Алгоритма 1.4 к семейству предикатов Ф вида (1.6) позволяет построить покрытие обучающей выборки полуплоскостями. В этом случае решающий список описывает кусочно-линейную разделяющую поверхность между классами, Рис. 5.

Известно большое количество эвристик для последовательного построения разделяющих полуплоскостей.

Пример 1.6. Алгоритм Белецкого [1] строит полуплоскости, отделяющие как можно больше объектов одного класса, что равносильно максимизации информативности предиката (1.6). Для этого применяются методы линейного программирования. В отличие от жадного Алгоритма 1.4, после построения каждой полуплоскости делается попытка найти лучшие положения предыдущих полуплоскостей.

Пример 1.7. В алгоритме Маршанда и др. [30] перебираются всевозможные гиперплоскости, разделяющие какие-нибудь три точки (data dependent half-spaces), и из них выбирается полуплоскость с максимальной информативностью.

Достоинства решающих списков.

- Интерпретируемость и простота классификации. Обученное по выборке правило классификации можно записать в виде инструкции и выполнять «вручную».
- Гибкость: в зависимости от выбора множества Ф можно строить весьма разнообразные алгоритмические конструкции.
- Возможность обработки разнотипных данных и данных с пропусками.

Недостатки решающих списков.

- Если множество правил Ф выбрано неудачно, список может не построиться. При этом возможен высокий процент отказов от классификации.
- Возможна утрата интерпретируемости, если список длинный и правила различных классов следуют вперемежку. В этом случае правила не могут быть интерпретированы по-отдельности, без учёта предшествующих правил, и логика их взаимодействия становится довольно запутанной.
- Каждый объект классифицируется только одним правилом, что не позволяет правилам компенсировать неточности друг друга. Данный недостаток устраняется путём голосования правил, но это уже совсем другой алгоритм.

§1.4 Решающие деревья

Решающее дерево (decision tree, DT) — это ещё один логический алгоритм классификации, основанный на поиске конъюнктивных закономерностей. Но, в отличие от решающего списка, при синтезе дерева все конъюнкции строятся одновременно.

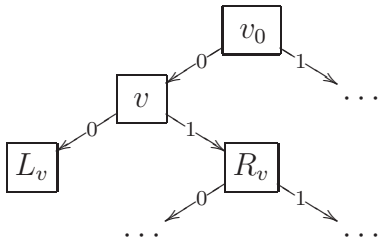
Напомним некоторые понятия теории графов.

Деревом называется конечный связный граф с множеством вершин V, не содержащий циклов и имеющий выделенную вершину $v_0 \in V$, в которую не входит ни одно ребро. Эта вершина называется *корнем* дерева. Вершина, не имеющая выходящих рёбер, называется *терминальной* или *листом*. Остальные вершины называются *внутренними*. Дерево называется *бинарным*, если из любой его внутренней вершины выходит ровно два ребра. Выходящие рёбра связывают каждую внутреннюю вершину v с *левой дочерней* вершиной L_v и с *правой дочерней* вершиной R_v.

Опр. 1.5. *Бинарное решающее дерево — это алгоритм классификации, задающийся бинарным деревом, в котором каждой внутренней вершине $v \in V$ приписан предикат $\beta_v : X \to \{0,1\}$, каждой терминальной вершине $v \in V$ приписано имя класса $c_v \in Y$. При классификации объекта $x \in X$ он проходит по дереву путь от корня до некоторого листа, в соответствии с Алгоритмом 1.5.*

Алгоритм 1.5. Классификация объекта $x \in X$ бинарным решающим деревом

```
1: v := v_0;
2: пока вершина v внутренняя
3:     если β_v(x) = 1 то
4:         v := R_v;   (переход вправо)
5:     иначе
6:         v := L_v;   (переход влево)
7: вернуть c_v.
```



Объект x доходит до вершины v тогда и только тогда, когда выполняется конъюнкция $K_v(x)$, составленная из всех предикатов, приписанных внутренним вершинам дерева на пути от корня v_0 до вершины v. Пусть T — множество всех терминальных вершин дерева. Множества объектов $\Omega_v = \{x \in X : K_v(x) = 1\}$, выделяемых терминальными конъюнкциями $v \in T$, попарно не пересекаются, а их объединение совпадает со всем пространством X (это легко доказывается индукцией по числу вершин дерева). Отсюда следует, что решающее дерево никогда не отказывается от классификации, в отличие от решающего списка. Отсюда также следует, что алгоритм классификации $a : X \to Y$, реализуемый бинарным решающим деревом, можно представить в виде простого голосования конъюнкций:

$$a(x) = \arg\max_{y \in Y} \sum_{v \in T} [c_v = y] K_v(x), \tag{1.8}$$

причём для любого $x \in X$ одно и только одно слагаемое во всех этих суммах равно единице. Вместо суммирования можно было бы использовать и дизъюнкцию.

Естественное требование максимизации информативности конъюнкций $K_v(x)$ означает, что каждая из них должна выделять как можно больше обучающих объектов, допуская при этом как можно меньше ошибок. Для повышения обобщающей способности решающего дерева число листьев должно быть как можно меньше, и они должны покрывать подвыборки примерно одинаковой мощности $|\Omega_v \cap X^\ell|$. Строгое доказательство этого утверждения можно найти в [5].

1.4.1 Синтез решающих деревьев

Задача построения дерева минимальной сложности, правильно классифицирующего заданную выборку, в общем случае является NP-полной задачей [5]. На практике применяют различные, более или менее удачные, эвристики, нацеленные на построение как можно более простого дерева, обладающего как можно лучшим качеством классификации. Выбор эвристик неоднозначен, как обычно бывает в таких случаях. Придумано огромное количество различных методов синтеза бинарных решающих деревьев по обучающей выборке.

Рассмотрим сначала простой жадный алгоритм, основанный на принципе «разделяй и властвуй» [33].

Алгоритм построения решающего дерева ID3 (Induction of Decision Tree). Идея алгоритма заключается в последовательном дроблении выборки на две части до тех пор, пока в каждой части не окажутся объекты только одного класса. Проще всего записать этот алгоритм в виде рекурсивной процедуры `LearnID3`, которая строит дерево по заданной подвыборке U.

Для построения полного дерева она применяется ко всей выборке и возвращает указатель на корень построенного дерева:

$$v_0 := \texttt{LearnID3}\,(X^\ell).$$

На шаге 6 Алгоритма 1.6 выбирается предикат β из заданного семейства $\mathscr{B}$, задающий максимально информативное ветвление дерева — разбиение выборки на две части $U = U_0 \cup U_1$.

На практике применяются различные *критерии ветвления*.

1. Критерий, ориентированный на отделение заданного класса $c \in Y$.

$$I(\beta, U) = \max_{c \in Y} I_c(\beta, U).$$

2. Более эффективны (особенно на верхних уровнях дерева) критерии, ориентированные на отделение не одного, а сразу нескольких классов. Они были введены в разделе 1.1.4.

3. D-критерий — число пар объектов из разных классов, на которых предикат β принимает разные значения. В случае двух классов он имеет вид

$$I(\beta, U) = p(\beta)\big(N - n(\beta)\big) + n(\beta)\big(P - p(\beta)\big).$$

В Алгоритме 1.6 множество элементарных предикатов $\mathscr{B}$ может быть каким угодно, лишь бы существовал эффективный механизм выбора наиболее информативного предиката из $\mathscr{B}$ на шаге 6. Когда мощность $|\mathscr{B}|$ не велика, эта задача легко

Алгоритм 1.6. Рекурсивный алгоритм синтеза бинарного решающего дерева ID3

Вход:
U — обучающая выборка;
$\mathscr{B}$ — множество элементарных предикатов;
Выход:
возвращает корневую вершину дерева, построенного по выборке U;

ПРОЦЕДУРА `LearnID3` (U);
если все объекты из U лежат в одном классе $c \in Y$ **то**
  создать новый лист v;
  $c_v := c$;
  вернуть (v);
найти предикат с максимальной информативностью:
$\beta := \arg\max\limits_{\beta \in \mathscr{B}} I(\beta, U)$;
разбить выборку на две части $U = U_0 \cup U_1$ по предикату β:
$U_0 := \big\{x \in U : \beta(x) = 0\big\}$;
$U_1 := \big\{x \in U : \beta(x) = 1\big\}$;
если $U_0 = \varnothing$ или $U_1 = \varnothing$ **то**
  создать новый лист v;
  $c_v :=$ класс, в котором находится большинство объектов из U;
иначе
  создать новую внутреннюю вершину v;
  $\beta_v := \beta$;
  $L_v :=$ `LearnID3` (U_0); (построить левое поддерево)
  $R_v :=$ `LearnID3` (U_1); (построить правое поддерево)
вернуть (v);

решается полным перебором. В противном случае приходится применять эвристические процедуры направленного поиска.

На практике в качестве элементарных предикатов чаще всего берут простые пороговые условия вида $\beta(x) = [f_j(x) \lessgtr d_j]$. Конъюнкции, составленные из таких термов, хорошо интерпретируются и допускают запись на естественном языке. Однако никто не запрещает использовать в вершинах дерева любые разделяющие правила: шары, гиперплоскости, и, вообще говоря, произвольные бинарные классификаторы. Большое число таких примеров можно найти в обзоре [13].

Обработка пропусков. В практических задачах, особенно в области медицины или социологии, часто встречаются данные с пропусками. Что делать, если предикат $\beta(x)$ не может быть вычислен для данного объекта $x \in X$? Самая типичная ситуация — когда $\beta(x) = [f_j(x) \lessgtr d_j]$, и значение признака $f_j(x)$ для данного x не измерено.

1. Стадия обучения. Если значение $\beta(x_i)$ не определено для обучающего объекта $x_i \in X^\ell$, то при вычислении информативности $I(\beta, U)$ этот объект не учитывается. Соответственно, длина выборки при вычислении информативности уменьшается. Чтобы сравнение информативности по выборкам разной длины было «законно», критерий I должен быть инвариантен относительно увеличения длины выбор-

ки при пропорциональном увеличении $p(\beta)$ и $n(\beta)$. Эвристический и энтропийный критерии удовлетворяют этому требованию, а гипергеометрический — нет; согласно Теореме 1.1 величину I_c надо нормировать на длину выборки, то есть на шаге 6 надо сравнивать удельные информативности $\frac{1}{|U|}I_c(\beta, U)$.

2. Стадия классификации. Допустим, что дерево уже построено, и при классификации объекта x значение $\beta_v(x)$ во внутренней вершине v не определено. Возможны несколько стратегий обработки этой ситуации. Самая распространённая — *пропорциональное распределение* (proportional distribution). На стадии обучения для каждой внутренней вершины оценивается вероятность левой ветви $\hat{p}_L = |U_0|/|U|$ и вероятность правой ветви $\hat{p}_R = |U_1|/|U|$. На стадии классификации объект x пропускается через обе ветви, и результаты классификации взвешиваются с весами $\hat{p}_L$ и $\hat{p}_R$. Такие ветвления могут происходить в поддеревьях многократно. Поэтому для каждой вершины v оценивается *апостериорное распределение* вероятностей классов, согласно рекуррентной формуле

$$\hat{P}_v(y|x) = \begin{cases} [y = c_v], & v \in T; \\ \hat{p}_L \hat{P}_{L_v}(y|x) + \hat{p}_R \hat{P}_{R_v}(y|x), & v \notin T. \end{cases}$$

Оценивание вероятностей. Во многих приложениях наряду с классификацией объекта x необходимо получать оценки апостериорных вероятностей классов $\hat{P}(y|x)$.

Проще всего оценить их как долю обучающих объектов каждого из классов $y \in Y$, попавших в терминальную вершину v, классифицировавшую объект x. Однако такая оценка может оказаться смещённой по причине переобучения, поскольку предикаты $\beta(x)$ во всех внутренних вершинах дерева выбирались по той же самой обучающей выборке.

Влияние переобучения можно снизить различными способами: строить несколько различных деревьев и использовать усреднённые оценки; использовать для оценивания апостериорных вероятностей контрольную выборку; использовать дерево, редуцированное по контрольной выборке (см. ниже).

Трудоёмкость алгоритма ID3 имеет порядок $O(Bh\ell)$, где h — глубина дерева, B — среднее число предикатов, для которых оценивается информативность на шаге 6. Действительно, больше всего времени занимает вычисление информативности подвыборки U, и это время прямо пропорционально мощности $|U|$. Суммарная мощность всех подвыборок, оцениваемых в вершинах одного уровня, в точности равна ℓ. Значит, число операций, производимых для построения одного полного уровня дерева, имеет порядок $B\ell$. В наихудшем случае $B = |\mathscr{B}|$, однако применение удачных эвристик для сокращения перебора на шаге 6 позволяет существенно уменьшить B.

Преимущества алгоритма ID3.

- Простота и интерпретируемость классификации. Алгоритм 1.5 способен не только классифицировать объект, но и выдать объяснение классификации в терминах предметной области. Объяснение строится путём выписывания последовательности условий, проверенных для данного объекта на пути от корня дерева до листа v. Эти условия образуют конъюнкцию K_v, то есть легко интерпретируемое логическое правило.

- Трудоёмкость Алгоритма 1.6 линейна по длине выборки.
- Если множество предикатов $\mathscr{B}$ настолько богато, что на шаге 6 всегда находится предикат, разбивающий выборку U на непустые подмножества U_0 и U_1, то алгоритм строит бинарное решающее дерево, безошибочно классифицирующее выборку X^ℓ.
- Не бывает отказов от классификации, в отличие от решающих списков.
- Алгоритм очень прост для реализации и легко поддаётся различным усовершенствованиям. Можно использовать различные критерии ветвления и критерии останова, вводить редукцию, и т. д.

Недостатки алгоритма ID3.

- Жадность. Локально оптимальный выбор предиката β_v не является глобально оптимальным. В случае неудачного выбора алгоритм не способен вернуться на уровень вверх и заменить неудачный предикат.
- Чем дальше вершина v расположена от корня дерева, тем меньше длина подвыборки U, по которой принимается решение о ветвлении в вершине v. Тем менее статистически надёжным является выбор предиката β_v. В худшем случае всё дерево может оказаться составленным из ненадёжных закономерностей.
- Высокая чувствительность к составу выборки. Изменение данных в 1–2 объектах часто приводит к радикальному изменению структуры дерева.
- Алгоритм ID3 переусложняет структуру дерева, и, как следствие, склонен к переобучению. Его обобщающая способность (качество классификации новых объектов) относительно невысока.

Основная причина недостатков — неоптимальность жадной стратегии наращивания дерева. Для их устранения применяют различные эвристические приемы: редукцию, элементы глобальной оптимизации, «заглядывание вперёд» (look ahead), построение совокупности деревьев — решающего леса.

1.4.2 Редукция решающих деревьев

Суть редукции состоит в удалении поддеревьев, имеющих недостаточную статистическую надёжность. При этом дерево перестаёт безошибочно классифицировать обучающую выборку, зато качество классификации новых объектов (способность к обобщению), как правило, улучшается.

Придумано огромное количество эвристик для проведения редукции, однако ни одна из них, вообще говоря, не гарантирует улучшения качества классификации, см. обзоры [21, 6]. Мы рассмотрим лишь наиболее простые варианты редукции.

Предредукция (pre-pruning) или критерий *раннего останова* досрочно прекращает дальнейшее ветвление в вершине дерева, если информативность $I(\beta, U)$ для всех предикатов $\beta \in \mathscr{B}$ не дотягивает до заданного порогового значения I_0. Для этого на шаге 8 условие ($U_0 = \varnothing$ или $U_1 = \varnothing$) заменяется условием $I(\beta, U) \leqslant I_0$. Порог I_0 является управляющим параметром метода.

Предредукция считается не самым эффективным способом избежать переобучения, так как жадное ветвление по-прежнему остаётся глобально неоптимальным. Более эффективной считается стратегия постредукции.

Постредукция (post-pruning) просматривает все внутренние вершины дерева и заменяет отдельные вершины либо одной из дочерних вершин (при этом вторая дочерняя удаляется), либо терминальной вершиной. Процесс замен продолжается до тех пор, пока в дереве остаются вершины, удовлетворяющие критерию замены.

Критерием замены является сокращение числа ошибок на контрольной выборке, отобранной заранее, и не участвовавшей в обучении дерева. Стандартная рекомендация — оставлять в контроле около 30% объектов.

Для реализации постредукции контрольная выборка X^k пропускается через построенное дерево. При этом в каждой внутренней вершине v запоминается подмножество $S_v \subseteq X^k$ попавших в неё контрольных объектов. Если $S_v = \varnothing$, то вершина v считается ненадёжной и заменяется терминальной по *мажоритарному правилу*: в качестве c_v берётся тот класс, объектов которого больше всего в обучающей подвыборке U, пришедшей в вершину v.

Затем для каждой внутренней вершины v вычисляется число ошибок, полученных при классификации выборки S_v следующими способами:

1) $r(v)$ — классификация поддеревом, растущим из вершины v;
2) $r_L(v)$ — классификация поддеревом левой дочерней вершины L_v;
3) $r_R(v)$ — классификация поддеревом правой дочерней вершины R_v;
4) $r_c(v)$ — отнесение всех объектов выборки S_v к классу $c \in Y$.

Эти величины сравниваются, и, в зависимости от того, какая из них оказалась минимальной, принимается, соответственно, одно из четырёх решений:

1) сохранить поддерево вершины v;
2) заменить поддерево вершины v поддеревом левой дочерней вершины L_v;
3) заменить поддерево вершины v поддеревом правой дочерней вершины R_v;
4) заменить поддерево v терминальной вершиной класса $c = \arg\min\limits_{c \in Y} r_c(v)$.

1.4.3 Преобразование решающего дерева в решающий список

Существует ещё одна стратегия редукции решающих деревьев, при которой меняется сама структура классификатора.

Согласно формуле (1.8) всякое решающее дерево эквивалентно решающему списку, составленному из терминальных конъюнкций $K_v(x)$, $v \in T$. Порядок правил в списке не имеет значения, так как области Ω_v, $v \in T$ не пересекаются. Кроме того, такой список никогда не отказывается от классификации, так как объединение этих областей совпадает со всем множеством X.

Полученный список можно упростить, применив процедуру редукции ко всем конъюнкциям K_v по очереди, как это было описано в разделе 1.2.2. Редукция приводит к расширению множеств Ω_v объектов, выделяемых конъюнкциями K_v, и они на-

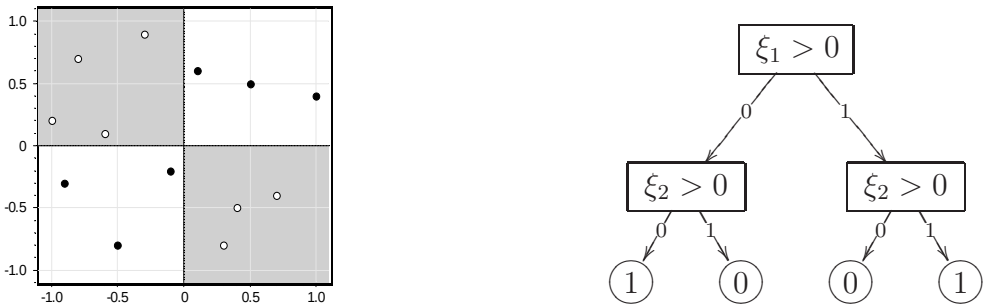


Рис. 6. Выборка типа XOR и идеальное дерево для неё.

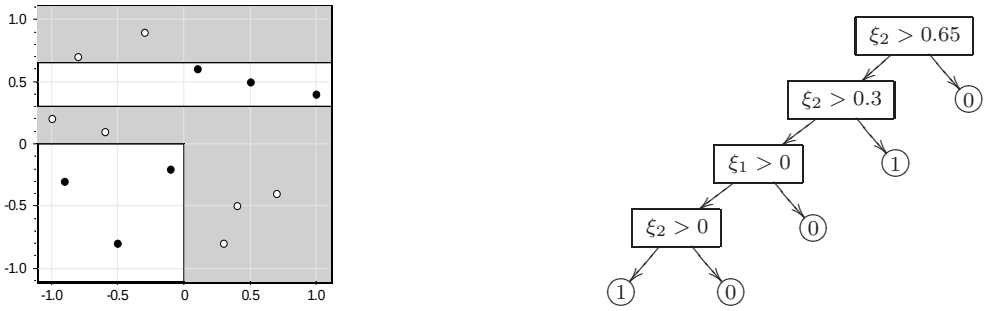


Рис. 7. Та же выборка и дерево, построенное жадным алгоритмом.

чинают перекрываться. Возникает вопрос: в каком порядке расположить правила K_v в списке. Представляется разумным добавлять правила к списку в порядке убывания информативности, и каждое добавленное правило сразу редуцировать. Очевидно, редуцированный список также никогда не отказывается от классификации. В отличие от редукции, стабилизация может привести к образованию непокрытых областей пространства X, следовательно, к отказам алгоритма на некоторых объектах.

1.4.4 Заглядывание вперёд

Во многих задачах жадное ветвление приводит к построению деревьев, существенно отличающихся от оптимальных. В качестве примера приведём знаменитую задачу «исключающего или» (XOR).

Пример 1.8. Пусть классов два, выборка двумерная, целевая зависимость имеет вид $y^*(\xi_1, \xi_2) = [\xi_1\xi_2 > 0]$. «Идеальное» дерево для этого случая показано на Рис. 6. Жадный алгоритм не сможет построить такое дерево, так как правильное ветвление в корневой вершине не увеличивает информативность, следовательно, алгоритм ID3 никогда не выберет его, и весь дальнейший процесс построения дерева пойдёт неоптимальным образом. Результат показан на Рис. 7.

Идея *заглядывания вперёд* (look ahead) заключается в том, чтобы на шаге 6, вместо вычисления информативности для каждого $\beta \in \mathscr{B}$ построить поддерево небольшой глубины h. Во внутреннюю вершину v помещается тот предикат β, при котором поддерево допускает наименьшее число ошибок. Этот алгоритм работает заметно дольше, но строит более надёжные и простые деревья.

В статье [24] указывается, что при небольших фиксированных значениях h данная стратегия практически не даёт выигрыша, и предлагается *неограниченный*

по времени алгоритм (anytime algorithm), который в фоновом режиме постоянно улучшает дерево, выполняя всё более и более глубокое заглядывание вперёд. Эта работа может быть в любой момент приостановлена для получения готового дерева, и затем возобновлена вновь. Такая технология удобна, и даже предпочтительна, в тех практических ситуациях, когда выборки большие и имеется возможность задействовать простаивающий вычислительный ресурс.

§1.5 Взвешенное голосование правил

Допустим, имеется консилиум экспертов, каждый член которого может допустить ошибку. Процедура голосования — это способ повышения качества принимаемых решений, при котором ошибки отдельных экспертов компенсируют друг друга.

Ранее, в разделе **??**, принцип голосования применялся для построения композиций из произвольных алгоритмов классификации. Теперь рассмотрим композиции, состоящие из логических закономерностей.

1.5.1 Принцип голосования

Пусть для каждого класса $c \in Y$ построено множество логических закономерностей (правил), специализирующихся на различении объектов данного класса:

$$R_c = \{\varphi_c^t : X \to \{0,1\} \mid t = 1, \ldots, T_c\}.$$

Считается, что если $\varphi_c^t(x) = 1$, то правило φ_c^t относит объект $x \in X$ к классу c. Если же $\varphi_c^t(x) = 0$, то правило φ_c^t воздерживается от классификации объекта x.

Алгоритм простого голосования (simple voting) подсчитывает долю правил в наборах R_c, относящих объект x к каждому из классов:

$$\Gamma_c(x) = \frac{1}{T_c} \sum_{t=1}^{T_c} \varphi_c^t(x), \quad c \in Y,$$

и относит объект x к тому классу, за который подана наибольшая доля голосов:

$$a(x) = \arg\max_{c \in Y} \Gamma_c(x). \tag{1.9}$$

Если максимум достигается одновременно на нескольких классах, выбирается тот, для которого цена ошибки меньше.

Нормирующий множитель $1/T_c$ вводится для того, чтобы наборы с бóльшим числом правил не перетягивали объекты в свой класс.

Алгоритм взвешенного голосования (weighted voting, WV) действует более тонко, учитывая, что правила могут иметь различную ценность. Каждому правилу φ_c^t приписывается вес $\alpha_c^t \geqslant 0$, и при голосовании берётся взвешенная сумма голосов:

$$\Gamma_c(x) = \sum_{t=1}^{T_c} \alpha_c^t \varphi_c^t(x), \quad \alpha_c^t \geqslant 0. \tag{1.10}$$

Веса принято нормировать на единицу: $\sum_{t=1}^{T_c} \alpha_c^t = 1$, для всех $c \in Y$. Поэтому функцию $\Gamma_c(x)$ называют также *выпуклой комбинацией* правил $\varphi_c^1, \ldots, \varphi_c^{T_c}$. Очевидно, простое голосование является частным случаем взвешенного, когда веса одинаковы и равны $1/T_c$.

На первый взгляд, вес правила должен определяться его информативностью. Однако, важно ещё, насколько данное правило уникально. Если имеется 10 хороших, но одинаковых (или почти одинаковых) правил, их суммарный вес должен быть сравним с весом столь же хорошего правила, не похожего на все остальные. Таким образом, веса должны учитывать не только ценность правил, но и их различность.

Простой общий подход к настройке весов заключается в том, чтобы сначала найти набор правил $\{\varphi_c^t(x)\}$, затем принять их за новые (бинарные) признаки и построить в этом новом признаковом пространстве линейную разделяющую поверхность (кусочно-линейную, если $|Y| > 2$). Для этого можно использовать логистическую регрессию, однослойный персептрон или метод опорных векторов. Существуют и другие подходы. Например, в 1.5.4 будет рассмотрен метод бустинга, в котором правила настраиваются последовательно, и для каждого правила сразу вычисляется его вес.

Проблема диверсификации правил. Голосующие правила должны быть существенно различны, иначе они будут бесполезны для классификации. Продолжая аналогию с консилиумом, заметим, что нет никакого смысла держать в консилиуме эксперта A, если он регулярно подсматривает решения у эксперта B.

Приведём простое теоретико-вероятностное обоснование *принципа диверсификации*, или повышения *различности* (diversity) правил [14]. Пусть X — вероятностное пространство, множество ответов Y конечно. Введём случайную величину $M(x)$, равную перевесу голосов в пользу правильного класса; её называют также *отступом* (margin) объекта x от границы классов:

$$M(x) = \Gamma_c(x) - \Gamma_{\bar{c}}(x), \quad \Gamma_{\bar{c}}(x) = \max_{y \in Y \setminus \{c\}} \Gamma_y(x), \quad c = y^*(x).$$

Если отступ положителен, $M(x) > 0$, то алгоритм голосования правильно классифицирует объект x. Предположим, что в среднем наш алгоритм классифицирует хотя бы немного лучше, чем наугад: $\mathsf{E}M > 0$. Тогда можно оценить вероятность ошибки по неравенству Чебышева:

$$\mathsf{P}\{M < 0\} \leqslant \mathsf{P}\big\{|\mathsf{E}M - M| > \mathsf{E}M\big\} \leqslant \frac{\mathsf{D}M}{(\mathsf{E}M)^2}.$$

Отсюда вывод: для уменьшения вероятности ошибки необходимо максимизировать ожидание перевеса голосов $\mathsf{E}M$ и минимизировать его дисперсию $\mathsf{D}M$. Для выполнения этих условий каждый объект должен выделяться примерно одинаковым числом правил. Обычно ни одно из правил не выделяет класс целиком, поэтому правила должны быть существенно различны, то есть выделять существенно различные подмножества объектов.

Неплохая эвристика, усиливающая различия между правилами и позволяющая равномернее выделять объекты обучения, используется в алгоритме CORAL [12]. Сначала для фиксированного класса $c \in Y$ строится покрывающий набор правил

точно так, как это делалось для решающих списков (стр. 18). Затем строится второй покрывающий набор, но при этом запрещается использовать признаки, часто входившие в закономерности первого набора. Поэтому второй набор неминуемо окажется отличным от первого. Затем запрещаются признаки, часто входившие в оба набора, и строится третий набор. И так далее, для каждого класса $c \in Y$.

Другая стратегия используется в алгоритме бустинга. После построения каждой закономерности веса выделенных ею объектов уменьшаются, поощряя выделение других объектов следующими закономерностями. Этот алгоритм будет подробно рассмотрен в разделе 1.5.4.

Отказы от классификации. Возможны ситуации, когда ни одно из правил не выделяет классифицируемый объект x. Тогда алгоритм должен либо отказываться от классификации, либо относить объект к классу, имеющему наименьшую цену ошибки. Отказ алгоритма означает, что данный объект является нетипичным, не подпадающим ни под одну из ранее обнаруженных закономерностей. Вообще, *обнаружение нетипичности* (novelty detection) принято считать отдельным видом задач обучения по прецедентам, наряду с классификацией и кластеризацией. Способность алгоритмов отказываться от классификации нетипичных объектов во многих приложениях является скорее преимуществом, чем недостатком. В то же время, число отказов не должно быть слишком большим.

Итак, при построении алгоритмов взвешенного голосования правил возникает четыре основных вопроса:

- Как построить много правил по одной и той же выборке?
- Как избежать повторов и построения почти одинаковых правил?
- Как избежать появления непокрытых объектов и обеспечить равномерное покрытие всей выборки правилами?
- Как определять веса правил при взвешенном голосовании?

Рассмотрим, как эти проблемы решаются в известных алгоритмах.

1.5.2 Алгоритм КОРА

Алгоритм комбинаторного распознавания КОРА, предложенный М. М. Бонгардом в 1961-м году и реализованный М. Н. Вайнцвайгом [2], строит набор конъюнктивных закономерностей. Этот алгоритм неоднократно доказал свою эффективность при решении прикладных задач.

В основе алгоритма лежат следующие эвристические предположения.

- Множество элементарных предикатов $\mathscr{B}$ подобрано настолько удачно, что среди конъюнкций ранга 2 или 3 уже находится достаточное количество информативных закономерностей.
- Поскольку ранг конъюнкций ограничен сверху числом 3, для поиска закономерностей можно применить полный перебор.

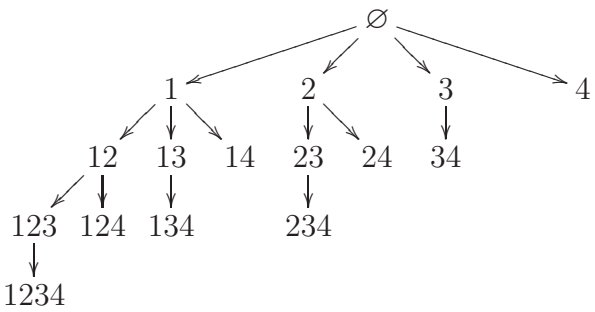


Рис. 8. Дерево полного перебора конъюнкций в алгоритме КОРА при $|\mathscr{B}| = 4$. Для краткости конъюнкции обозначены номерами составляющих их предикатов.

- Наибольший интерес представляют непротиворечивые закономерности (в исходном варианте алгоритма только они и строились).

Алгоритм 1.7 строит для каждого класса $c \in Y$ свой список конъюнкций R_c. Каждая конъюнкция содержит не более K термов, выбираемых из множества предикатов $\mathscr{B}$. Ядром алгоритма является рекурсивная процедура `Нарастить`(φ), которая добавляет термы в конъюнкцию $\varphi(x)$ всевозможными способами и заносит в список закономерностей R_c только наилучшие конъюнкции — удовлетворяющие критериям отбора $D_c(\varphi) \geqslant D_{\min}$ и $E_c(\varphi) \leqslant E_{\max}$.

Перебор конъюнкций осуществляется методом поиска в глубину. На Рис. 8 показано дерево перебора всех конъюнкций при $|\mathscr{B}| = 4$. В процессе перебора необходимо избегать повторного просмотра конъюнкций, отличающихся только порядком записи термов. Для этого фиксируется некоторая исходная нумерация предикатов $\mathscr{B} = \{\beta_1, \ldots, \beta_{|\mathscr{B}|}\}$, и конъюнкции $\varphi = \beta_{j_1} \wedge \ldots \wedge \beta_{j_s}$ наращиваются так, чтобы номера предикатов строго возрастали: $1 \leqslant j_1 < \cdots < j_s \leqslant |\mathscr{B}|$.

Процедура наращивания использует два приёма для сокращения перебора.

Во-первых, конъюнкция φ перестаёт наращиваться, если она выделяет слишком мало объектов своего класса, $D_c(\varphi) < D_{\min}$, поскольку с увеличением числа термов количество выделяемых объектов может только уменьшиться.

Во-вторых, конъюнкция, удовлетворяющая критериям отбора и добавленная в список R_c, больше не наращивается. Тем самым предпочтение отдаётся более коротким конъюнкциям.

Параметры $D_{\min}$ и $E_{\max}$ решающим образом влияют на количество получаемых конъюнкций. Если критерии отбора заданы слишком жёстко, алгоритм может вообще не найти ни одной конъюнкции. Если же критерии слишком слабы, алгоритм будет тратить время на перебор и оценивание большого количества малоинформативных конъюнкций. Если бы не эта проблема, алгоритм сводился бы к однократному применению процедуры `Нарастить` к пустой конъюнкции (шаг 3). Более сложный внешний цикл алгоритма (шаги 2–9) необходим для того, чтобы скорректировать параметры $D_{\min}$ и $E_{\max}$ в зависимости от количества получаемых конъюнкций T и времени, затраченного на поиск. На практике эти параметры зачастую подбирают экспериментальным путём, фактически, выполняя внешний цикл вручную.

Процедура `Добавить_в_список` заносит конъюнкцию φ в список R_c так, чтобы в итоге в нём осталось около $T_{\max}$ наилучших конъюнкций. Эта процедура ещё понадобится нам в дальнейшем, поэтому она вынесена в отдельный Алгоритм 1.8.

Алгоритм 1.7. Построение списка информативных конъюнкций методом поиска в ширину (алгоритм КОРА)

Вход:
X^ℓ — обучающая выборка;
$\mathscr{B}$ — семейство элементарных предикатов;
K — максимальный ранг конъюнкций;
$E_{\max}$ — максимальная доля ошибок $E_c(\varphi)$ для конъюнкций $\varphi \in R_c$;
$D_{\min}$ — минимальная доля позитивных объектов $D_c(\varphi)$ для конъюнкций $\varphi \in R_c$;
$T_{\min}, T_{\max}$ — ограничение на число конъюнкций T_c;
Выход:
списки конъюнкций $R_c = \{\varphi_c^t(x) : t = 1, \ldots, T_c\}$, для всех $c \in Y$;

1: инициализировать списки: $R_c = \varnothing$ для всех $c \in Y$;
2: **повторять**
3: `Нарастить` $(\varnothing)$;
4: $T := \min\limits_{c \in Y} |R_c|$;
5: **если** $T < T_{\min}$ **то**
6: уменьшить $D_{\min}$ и/или увеличить $E_{\max}$;
7: **если** $T \geqslant T_{\max}$ или время поиска слишком велико **то**
8: увеличить $D_{\min}$ и/или уменьшить $E_{\max}$;
9: **пока** $T \notin [T_{\min}, T_{\max})$

Процедура рекурсивного наращивания конъюнкции $\varphi = \beta_{j_1} \wedge \ldots \wedge \beta_{j_s}$ путём добавления термов всевозможными способами.

10: **ПРОЦЕДУРА** `Нарастить` (φ);
11: **если** $\varphi = \varnothing$ **то** $j_s := 0$;
12: **для всех** $j \in \{j_s + 1, \ldots, |\mathscr{B}|\}$
13: добавить терм β_j к исходной конъюнкции:
$\varphi' := \varphi \wedge \beta_j$;
14: **если** $|\varphi'| \leqslant K$ и $\exists c \in Y$: $\big(D_c(\varphi') \geqslant D_{\min}$ и $E_c(\varphi') \leqslant E_{\max}\big)$ **то**
15: `Добавить_в_список` $(R_c, \varphi', T_{\max})$;
16: **иначе если** $|\varphi'| < K$ и $\exists c \in Y$: $\big(D_c(\varphi') \geqslant D_{\min}\big)$ **то**
17: `Нарастить` (φ');

Замечания о деталях реализации.

- Вместе с каждой конъюнкцией имеет смысл хранить бинарный вектор её значений на всех объектах из X^ℓ. Тогда после наращивания конъюнкции (шаг 13) не придётся заново вычислять конъюнкцию всех предыдущих термов.

- Список R_c поддерживается отсортированным по убыванию информативности I_c. Это позволяет эффективно реализовать вставку конъюнкции на шаге 2 за $O(\log_2 |R_c|)$ операций, а выделение множества наихудших конъюнкций на шагах 3–4 за $O(1)$ операций.

- Почему в Алгоритме 1.8 нельзя сделать проще — после шага 2 удалять последнюю в списке наихудшую конъюнкцию? Проблема в том, что информатив-

Алгоритм 1.8. Включение конъюнкции φ в список R_c, содержащий не менее T самых информативных конъюнкций

1: **ПРОЦЕДУРА** `Добавить_в_список` (R_c, φ, T);
2: вставить конъюнкцию φ в список R_c в порядке убывания информативности $I_c(\varphi)$;
3: наихудшая информативность конъюнкций в списке R_c:
$J := \min\limits_{\psi \in R_c} I_c(\psi)$;
4: число конъюнкций с наихудшей информативностью:
$\Delta := \#\{\psi \in R_c : I_c(\psi) = J\}$;
5: **если** $|R_c| - \Delta \geqslant T$ **то**
6: удалить из списка R_c все конъюнкции с наихудшей информативностью J;

ность $I_c(\varphi)$, как функция дискретных величин $p_c(\varphi)$ и $n_c(\varphi)$, принимает конечное число значений. Число конъюнкций φ', для которых вызывается процедура `Добавить_в_список`, как правило, значительно больше. Конъюнкции, составленные из предикатов с меньшими порядковыми номерами, будут попадать в список раньше, но не будут вытесняться другими конъюнкциями с такой же информативностью. Это приведёт к тому, что предикаты с меньшими номерами будут чаще входить в отобранные конъюнкции, хотя никаких объективных оснований для такого предпочтения нет.

Достоинства алгоритма КОРА.

- Короткие конъюнкции легко интерпретируются в терминах предметной области. Алгоритм способен не только классифицировать объекты, но и объяснять свои решения на языке, понятном специалистам.
- При малых K, $K \leqslant 3$, алгоритм очень эффективен.
- Если короткие информативные конъюнкции существуют, они обязательно будут найдены, так как алгоритм осуществляет полный перебор.

Недостатки алгоритма КОРА.

- При неудачном выборе множества предикатов $\mathscr{B}$ коротких информативных конъюнкций может просто не существовать. В то же время, увеличение числа K приводит к экспоненциальному падению эффективности, так как число операций, выполняемых алгоритмом, составляет $O(|\mathscr{B}|^K \ell)$.
- Алгоритм не стремится диверсифицировать конъюнкции и обеспечивать равномерность покрытия объектов. Это отрицательно сказывается на обобщающей способности (вероятности ошибки) алгоритма.
- Нет настройки коэффициентов α_c^t; предполагается простое голосование.

1.5.3 Алгоритм ТЭМП

Полный перебор всех конъюнкций ранга не более K требует экспоненциального по K числа операций. В реальных задачах объём вычислений становится огромным уже при $K > 3$, и от идеи полного перебора приходится отказаться.

Существует две стандартные стратегии перебора конъюнкций: *поиск в глубину* (depth-first search) и *поиск в ширину* (breadth-first search). Первая применяется в алгоритме КОРА, вторая — в алгоритме ТЭМП, предложенным Г. С. Лбовым в 1976 году [12]. Поиск в ширину работает немного быстрее, и в него легче встраивать различные эвристики, сокращающие перебор.

В исходном варианте алгоритм ТЭМП выполнял полный перебор всех конъюнкций ранга не более K. Ниже описан слегка модифицированный вариант, позволяющий ограничить перебор и увеличить максимальный ранг конъюнкций K.

Алгоритм 1.9 начинает процесс поиска закономерностей с построения конъюнкций ранга 1. Для этого отбираются не более T_1 самых информативных предикатов из базового множества $\mathscr{B}$. Затем к каждому из отобранных предикатов добавляется по одному терму из $\mathscr{B}$ всеми возможными способами. Получается не более $T_1|\mathscr{B}|$ конъюнкций ранга 2, из которых снова отбираются T_1 самых информативных. И так далее. На каждом шаге процесса делается попытка добавить один терм к каждой из имеющихся конъюнкций. Наращивание конъюнкций прекращается либо при достижении максимального ранга K, либо когда ни одну из конъюнкций не удаётся улучшить путём добавления терма.

Лучшие конъюнкции, собранные со всех шагов, заносятся в списки R_c. Таким образом, списки R_c могут содержать конъюнкции различного ранга.

Параметр T_1 позволяет найти компромисс между качеством и скоростью работы алгоритма. При $T_1 = 1$ алгоритм ТЭМП работает исключительно быстро и строит единственную конъюнкцию, добавляя термы по очереди. Фактически, он совпадает с жадным Алгоритмом 1.2. При увеличении T_1 пространство поиска расширяется, алгоритм начинает работать медленнее, но находит больше информативных конъюнкций. На практике выбирают максимальное значение параметра T_1, при котором поиск занимает приемлемое время. Однако стратегия поиска всё равно остаётся жадной — термы оптимизируются по-отдельности, и при подборе каждого терма учитываются только предыдущие, но не последующие термы.

Для улучшения конъюнкций к ним применяют эвристические методы «финальной шлифовки» — стабилизацию и редукцию (стр. 13).

В результате стабилизации конъюнкции становятся локально неулучшаемыми. Алгоритм в целом становится более устойчивым — при незначительных изменениях в составе обучающей выборки он чаще находит одни и те же закономерности, а значит, улучшается его способность обобщать эмпирические факты.

В результате стабилизации некоторые конъюнкции могут совпасть, и в списке появятся дубликаты. Их удаление предусмотрено на шаге 13. Если список R_c поддерживается отсортированным по информативности, то удаление дубликатов является недорогой операцией, так как достаточно проверять на совпадение только соседние конъюнкции с одинаковой информативностью.

Если задать $T_1 = \infty$, то алгоритм выполнит полный перебор, как в исходном варианте ТЭМП. «Финальная шлифовка» в этом случае не нужна.

Алгоритм 1.9. Построение списка информативных конъюнкций методом поиска в ширину (алгоритм ТЭМП)

Вход:
X^ℓ — обучающая выборка;
$\mathscr{B}$ — семейство элементарных предикатов;
$c \in Y$ — класс, для которого строится список конъюнкций;
K — максимальный ранг конъюнкций;
T_1 — число лучших конъюнкций, отбираемых на каждом шаге;
T_0 — число лучших конъюнкций, отбираемых на последнем шаге, $T_0 \leqslant T_1$;
$I_{\min}$ — порог информативности;
$E_{\max}$ — порог допустимой доли ошибок;
X^k — контрольная выборка для проведения редукции;

Выход:
список конъюнкций $R_c = \{\varphi_c^t(x) : t = 1, \ldots, T_c\}$;

$R_c := \varnothing$;
для всех $\beta \in \mathscr{B}$
  `Добавить_в_список` (R_c, β, T_1);
для всех $k = 2, \ldots, K$
  для всех конъюнкций $\varphi \in R_c$ ранга $(k-1)$
    для всех предикатов $\beta \in \mathscr{B}$, которых ещё нет в конъюнкции φ
      добавить терм β к конъюнкции φ:
      $\varphi' := \varphi \wedge \beta$;
      если $I_c(\varphi') \geqslant I_{\min}$ и $E_c(\varphi') \leqslant E_{\max}$ и конъюнкции φ' нет в R_c, **то**
        `Добавить_в_список` (R_c, φ', T_1);
для всех конъюнкций $\varphi \in R_c$
  `Стабилизация` (φ);
  `Редукция` (φ, X^k);
удалить из списка R_c дублирующие конъюнкции;
оставить в списке R_c не более T_0 лучших конъюнкций;

И ещё одно отличие описанного алгоритма от исходного ТЭМПа: здесь конъюнкции отбираются по информативности, тогда как в исходном варианте использовался эвристический критерий $p_c(\varphi) \geqslant p_{\min}$, $n_c(\varphi) \leqslant n_{\max}$.

Достоинства алгоритма ТЭМП.

- ТЭМП существенно более эффективен, чем КОРА, особенно при поиске конъюнкций ранга больше 3. Он решает поставленную задачу за $O(KT_1|\mathscr{B}|\ell)$ операций, тогда как КОРА имеет трудоёмкость $O(|\mathscr{B}|^K\ell)$.
- Параметр T_1 позволяет управлять жадностью алгоритма и находить компромисс между качеством конъюнкций и скоростью работы алгоритма.
- Благодаря простоте и эффективности алгоритм ТЭМП можно использовать в составе других алгоритмов как генератор конъюнкций, достаточно близких к оптимальным.

Недостатки алгоритма ТЭМП.

- Нет гарантии, что будут найдены самые лучшие конъюнкции, особенно при малых значениях параметра T_1.
- Алгоритм не стремится увеличивать различность конъюнкций, добиваясь равномерного покрытия объектов выборки. Стабилизация и редукция лишь отчасти компенсирует этот недостаток.
- Нет настройки коэффициентов α_c^t; предполагается простое голосование.

1.5.4 Алгоритм бустинга

Алгоритмы КОРА и ТЭМП имеют общий недостаток — они не стремятся увеличивать различность конъюнкций. Эта проблема решается в алгоритме бустинга.

Бустинг (boosting) предложили американские учёные Фройнд и Шапир как универсальный метод построения выпуклой комбинации классификаторов [25]. Мы уже разбирали алгоритм AdaBoost в **??**, а теперь рассмотрим его специальный вариант, приспособленный для взвешенного голосования закономерностей.

В бустинге закономерности строятся последовательно, и после построения очередной закономерности веса выделенных ею объектов изменяются — уменьшаются у позитивных и увеличиваются у негативных объектов. Обновлённый вектор весов w используется при поиске следующей закономерности φ по критерию максимума *взвешенной информативности* (см. 1.1.5). В результате каждая следующая закономерность стремится выделить «наименее покрытые» объекты, оказавшиеся «наиболее трудными» для предыдущих закономерностей. Это способствует повышению различности закономерностей, более равномерному покрытию объектов и повышению обобщающей способности выпуклой комбинации закономерностей.

Описанная стратегия напоминает алгоритм построения решающего списка. Разница в том, что там было достаточно покрыть объект один раз, после чего он исключался из рассмотрения. Здесь же каждое покрытие только изменяет вес объекта.

Для реализации этой идеи остаётся понять, как именно должны вычисляться веса объектов и веса закономерностей на каждом шаге алгоритма.

Рассмотрим задачу классификации с двумя классами, $Y = \{-1, +1\}$ и алгоритм взвешенного голосования (1.9)–(1.10), состоящий из $T = T_{-1} + T_{+1}$ закономерностей:

$$a_T(x) = \operatorname{sign}\Bigg(\underbrace{\sum\nolimits_{t=1}^{T_{+1}} \alpha_{+1}^t \varphi_{+1}^t(x)}_{\Gamma_{+1}(x)} - \underbrace{\sum\nolimits_{t=1}^{T_{-1}} \alpha_{-1}^t \varphi_{-1}^t(x)}_{\Gamma_{-1}(x)}\Bigg), \qquad \alpha_c^t > 0,\ c \in Y.$$

Экспоненциальная аппроксимация пороговой функции потерь. Пусть уже построено T закономерностей, вместе составляющих алгоритм классификации $a_T(x)$. При добавлении ещё одной закономерности $\varphi_c(x)$ в список R_c взвешенная сумма голосов за класс $c \in \{-1, +1\}$ примет вид

$$\Gamma'_c(x) = \Gamma_c(x) + \alpha\varphi_c(x).$$

Задача состоит в том, чтобы найти закономерность φ_c и её вес α, при которых алгоритм $a_{T+1}(x)$ допускает минимальное число ошибок на обучающей выборке X^ℓ.

Число ошибок алгоритма $a_T(x)$ перед добавлением закономерности φ_c:

$$Q_T = \sum_{i=1}^{\ell} \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) < 0\right].$$

Число ошибок алгоритма $a_{T+1}(x)$ после добавления закономерности φ_c:

$$Q_{T+1}(\varphi_c, \alpha) = \sum_{i=1}^{\ell} [y_i = c] \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) + \alpha\varphi_c(x_i) < 0\right] + \\ + \sum_{i=1}^{\ell} [y_i \neq c] \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) - \alpha\varphi_c(x_i) < 0\right].$$

Выписанный функционал содержит параметр α внутри пороговой функции вида $[z(\alpha) < 0]$, следовательно, является разрывной функцией от α. Минимизация такого функционала является нетривиальной задачей комбинаторной оптимизации. Попробуем решить её приближенно. Заменим пороговую функцию непрерывно дифференцируемой оценкой сверху. Тогда минимизацию по α можно будет выполнить аналитически.

Выбор конкретной аппроксимирующей функции является эвристикой. В частности, подойдёт любая из представленных на Рис. **??**, стр. **??**. Наиболее простые выкладки получаются, если воспользоваться оценкой $[z < 0] \leqslant e^{-z}$.

Запишем верхнюю оценку $\tilde{Q}_T$ функционала Q_T:

$$Q_T \leqslant \tilde{Q}_T \equiv \sum_{i=1}^{\ell} \underbrace{\exp\bigl(\Gamma_{-y_i}(x_i) - \Gamma_{y_i}(x_i)\bigr)}_{w_i} = \sum_{i=1}^{\ell} w_i.$$

Если алгоритм $a_T(x)$ правильно классифицирует объект x_i, то $w_i < 1$. Если ошибается, то $w_i > 1$. Чем больше перевес голосов в пользу ошибочного класса, тем больше вес w_i. Таким образом, больший вес получают наиболее «трудные» объекты.

Введём вектор $w = (\tilde{w}_i)_{i=1}^{\ell}$ с компонентами $\tilde{w}_i = w_i\ell/\tilde{Q}_T$. Тогда будет выполнено условие нормировки $\sum_{i=1}^{\ell} \tilde{w}_i = \ell$. Следующая теорема показывает, что выбор $\tilde{w}_i$ в качестве весов объектов оптимален для построения $(T+1)$-й закономерности.

Теорема 1.2. *Минимум функционала $\tilde{Q}_{T+1}(\varphi_c, \alpha)$ достигается при*

$$\varphi_c^* = \arg\max_{\varphi} J_c^w(\varphi, X^\ell), \qquad J_c^w(\varphi, X^\ell) = \sqrt{p_c^w(\varphi)} - \sqrt{n_c^w(\varphi)};$$
$$\alpha^* = \frac{1}{2} \ln \frac{p_c^w(\varphi_c^*)}{n_c^w(\varphi_c^*)}, \text{ при } n_c^w(\varphi_c) \neq 0, \tag{1.11}$$

где функции $p_c^w(\varphi)$ и $n_c^w(\varphi)$ определяются по формулам (1.3).

Доказательство.

Распишем функционал $\tilde{Q}_{T+1}$, воспользовавшись тождеством $e^{A\varphi} = (1-\varphi)+\varphi e^A$, которое справедливо при любых $A \in \mathbb{R}$ и $\varphi \in \{0, 1\}$:

$$\tilde{Q}_{T+1} = \sum_{i=1}^{\ell} w_i e^{-\alpha\varphi_c(x_i)}\,[y_i = c] + \sum_{i=1}^{\ell} w_i e^{\alpha\varphi_c(x_i)}\,[y_i \neq c] =$$
$$= \sum_{i=1}^{\ell} w_i(1-\varphi_c(x_i)) + e^{-\alpha}\sum_{i=1}^{\ell} w_i\varphi_c(x_i)\,[y_i = c] + e^{\alpha}\sum_{i=1}^{\ell} w_i\varphi_c(x_i)\,[y_i \neq c]\,.$$

Подставим в эту формулу $w_i = \tilde{w}_i\tilde{Q}_T/\ell$ и обозначим для краткости $p = p_c^w(\varphi_c)$, $n = n_c^w(\varphi_c)$. Тогда, согласно определению (1.3),

$$\tilde{Q}_{T+1} = \frac{\tilde{Q}_T}{\ell}\Bigg(\underbrace{\sum_{i=1}^{\ell}\tilde{w}_i}_{\ell} - \underbrace{\sum_{i=1}^{\ell}\tilde{w}_i\varphi_c(x_i)}_{p+n} + e^{-\alpha}\underbrace{\sum_{i:\,y_i=c}\tilde{w}_i\varphi_c(x_i)}_{p} + e^{\alpha}\underbrace{\sum_{i:\,y_i\neq c}\tilde{w}_i\varphi_c(x_i)}_{n}\Bigg) =$$
$$= \frac{\tilde{Q}_T}{\ell}(\ell - p - n + e^{-\alpha}p + e^{\alpha}n). \tag{1.12}$$

Минимум этого выражения по α достигается при $e^{-\alpha}p = e^{\alpha}n$, откуда вытекает $\alpha^* = \frac{1}{2}\ln\frac{p}{n}$, если только $n \neq 0$. Подставляя α^* в функционал $\tilde{Q}_{T+1}$, получаем:

$$\tilde{Q}_{T+1} = \frac{\tilde{Q}_T}{\ell}\left(\ell - p - n + p\sqrt{\tfrac{n}{p}} + n\sqrt{\tfrac{p}{n}}\right) = \tilde{Q}_T\left(1 - \frac{1}{\ell}\left(\sqrt{p} - \sqrt{n}\right)^2\right).$$

Минимизация этого функционала по предикату φ_c эквивалентна максимизации функционала $J_c^w(\varphi_c) = \sqrt{p} - \sqrt{n}$ по φ_c, что и требовалось доказать. ■

Замечание 1.5. Теорема не позволяет определить вес непротиворечивой закономерности φ_c^*, так как знаменатель (1.11) обращается в нуль. Это происходит потому, что, согласно (1.12), при $n = 0$ функционал $\tilde{Q}_{T+1}$ неограниченно убывает по α, и непротиворечивые закономерности бесконечно выгодны с точки зрения минимизации $\tilde{Q}_{T+1}$. Чтобы предотвратить этот нежелательный эффект, вводят дополнительный параметр $\lambda \in (0,1)$, слегка модифицируя формулу расчёта веса:

$$\alpha^* = \frac{1}{2}\ln\frac{p_c^w(\varphi_c^*)}{\max\{n_c^w(\varphi_c^*),\lambda\}}.$$

Это всё равно, что ввести в функционал $\tilde{Q}_{T+1}$ дополнительное штрафное слагаемое с коэффициентом λ:

$$\tilde{Q}_{T+1} \leqslant \tilde{\tilde{Q}}_{T+1} = \frac{\tilde{Q}_T}{\ell}\big(\ell - p - n + e^{-\alpha}p + e^{\alpha}n + \lambda e^{\alpha}[n=0]\big).$$

Чем меньше λ, тем больший вес получают непротиворечивые закономерности. Таким образом, управляя параметром λ, можно заставить алгоритм искать преимущественно непротиворечивые закономерности.

Алгоритм 1.10. Построение выпуклой комбинации закономерностей при классификации на два класса, $Y = \{-1, +1\}$ (алгоритм бустинга)

Вход:
X^ℓ — обучающая выборка;
Φ — семейство базовых предикатов;
T — общее число закономерностей всех классов;
λ — коэффициент поощрения непротиворечивых закономерностей;

Выход:
списки закономерностей и их весов $\left\{\varphi_c^t(x), \alpha_c^t \;\middle|\; t = 1, \ldots, T_c\right\}$ для всех $c \in Y$;

1: инициализировать веса: $w_i := 1$ для всех $i = 1, \ldots, \ell$;
2: **для всех** $t = 1, \ldots, T$
3: $c := c_t$ — выбрать класс, для которого будет строиться закономерность;
4: $\varphi_c^t := \arg\max\limits_{\varphi \in \Phi} \sqrt{p_c^w(\varphi)} - \sqrt{n_c^w(\varphi)}$;
5: $\alpha_c^t := \dfrac{1}{2} \ln \dfrac{p_c^w(\varphi)}{\max\{n_c^w(\varphi), \lambda\}}$;
6: **для всех** $i = 1, \ldots, \ell$ пересчитать вес w_i:
7: $$w_i := \begin{cases} w_i, & \varphi_c(x_i) = 0; \\ w_i \exp(-\alpha_c^t), & \varphi_c(x_i) = 1 \text{ и } y_i = c; \\ w_i \exp(\alpha_c^t), & \varphi_c(x_i) = 1 \text{ и } y_i \neq c; \end{cases}$$
8: нормировать веса:
$Z := \frac{1}{\ell} \sum_{i=1}^{\ell} w_i; \quad w_i := w_i / Z$ для всех $i = 1, \ldots, \ell$;

Замечание 1.6. После того, как закономерность $\varphi_c(x)$ найдена и вычислен соответствующий ей коэффициент α, легко пересчитать веса объектов w_i' для следующего шага алгоритма:

$$w_i' = w_i \Big([y_i = c]\, e^{-\alpha \varphi_c(x_i)} + [y_i \neq c]\, e^{\alpha \varphi_c(x_i)} \Big) = \begin{cases} w_i, & \varphi_c(x_i) = 0; \\ w_i e^{-\alpha}, & \varphi_c(x_i) = 1 \text{ и } y_i = c; \\ w_i e^{\alpha}, & \varphi_c(x_i) = 1 \text{ и } y_i \neq c. \end{cases}$$

Таким образом, при выделении позитивного объекта его вес уменьшается в e^α раз, а при выделении негативного — увеличивается во столько же раз. В результате многократного применения этого правила бустинг стремится выделять позитивные объекты «равномерно часто», а негативные — «равномерно редко».

Замечание 1.7. Фактически, теорема 1.2 вводит ещё один функционал информативности предикатов $J_c^w(\varphi) = \sqrt{p_c^w(\varphi)} - \sqrt{n_c^w(\varphi)}$. Как видно из таблицы 1, он достаточно адекватно оценивает качество закономерностей, а вычисляется существенно проще, чем I_c или IGain_c. Его можно применять не только в алгоритме бустинга, но и отдельно, поскольку теорема 1.2 остаётся верна для функционала $\tilde{Q}_1$, если положить все веса w_i равными единице.

Замечание 1.8. В процессе работы алгоритма имеет смысл проанализировать распределение весов объектов. Объекты с наибольшими весами w_i являются наиболее

«трудными» для всех построенных закономерностей. Возможно, это «шумовые выбросы» — объекты, в описании которых допущены грубые ошибки. Исключение таких объектов, называемое *цензурированием выборки*, как правило, повышает качество классификации. После исключения выбросов построение закономерностей лучше начать заново, поскольку выбросы мешали адекватно оценивать информативность закономерностей.

Замечание 1.9. В Алгоритме 1.10 основная работа выполняется на шаге 4, когда ищется закономерность, доставляющая максимальное значение функционалу информативности $J_c^w(\varphi)$. Для решения данной задачи можно воспользоваться любым доступным алгоритмом синтеза закономерностей — градиентным, генетическим, ТЕМПом с параметром $T_0 = 1$, и т. д. Единственная модификация, которая для этого потребуется — заменить критерий информативности I_c на J_c^w. Отметим, что симбиоз «бустинг + ТЭМП с редукцией правил» практически совпадает с алгоритмом SLIPPER (simple learner with iterative pruning to produce error reduction), который считается одним из лучших логических алгоритмов классификации [22].

Теорема сходимости. Бустинг гарантирует построение корректного алгоритма, не допускающего ошибок на обучающей выборке. Оказывается, для этого достаточно потребовать, чтобы на шаге 4 всякий раз удавалось найти закономерность φ_c^t со строго положительной информативностью $J_c^w(\varphi_c^t)$.

Теорема 1.3. *Для числа ошибок на обучении справедлива оценка*

$$Q_T \leqslant \ell \prod_{t=1}^{T} \left(1 - \frac{J_t^2}{\ell}\right), \quad \text{где} \quad J_t = \sqrt{p_c^w(\varphi_c^t)} - \sqrt{n_c^w(\varphi_c^t)}. \tag{1.13}$$

Если на каждом шаге алгоритма бустинга находится закономерность φ_c^t с информативностью $J_t > J > 0$, то для обращения функционала Q_T в нуль потребуется не более $T_0 = (\ell \ln \ell)/J^2$ шагов.

Доказательство.

При доказательстве теоремы 1.2 была получена рекуррентная формула

$$\tilde{Q}_{t+1} = \tilde{Q}_t \left(1 - J_t^2/\ell\right).$$

Аналогично доказывается, что на первом шаге алгоритма, когда веса w_i единичны,

$$\tilde{Q}_1 = \ell \left(1 - J_1^2/\ell\right).$$

Поскольку $Q_T \leqslant \tilde{Q}_T$, отсюда немедленно вытекает (1.13).

Для доказательства сходимости за конечное число шагов оценим Q_T сверху:

$$Q_T \leqslant \tilde{Q}_T \leqslant \ell \left(1 - J^2/\ell\right)^T \leqslant \ell \exp\left(-J^2 T/\ell\right).$$

При $T > T_0$ эта оценка становится меньше 1 и функционал Q_T обращается в нуль, поскольку он может принимать только целые неотрицательные значения. ■

Достоинства бустинга.

- *Высокая обобщающая способность* достигается за счёт более равномерного покрытия объектов закономерностями.
- *Корректность на обучающей выборке* гарантируется при достаточно слабых дополнительных ограничениях.
- *Универсальность*. Можно использовать любое семейство базовых предикатов Ф, не обязательно конъюнкции. Предполагается только, что на шаге 4 применяется достаточно эффективный алгоритм перебора по множеству Ф.
- *Эффективность* бустинга целиком определяется этим внешним алгоритмом. Собственные накладные расходы бустинга невелики.
- *Интерпретируемость*. Линейная комбинация правил легко интерпретируется, если правил немного и они имеют вид конъюнкций. Веса правил всегда положительны и показывают степень их важности для классификации.
- *Фильтрация шума*. Возможность выделения шумовых объектов.

Недостатки бустинга.

- Эффект переобучения может всё же наблюдаться, если на шаге 4 не удаётся находить достаточно хорошие закономерности.
- В этом случае резко возрастает число правил T, необходимых для обеспечения корректности и линейная комбинация теряет свойство интерпретируемости. Алгоритм голосования становится «чёрным ящиком» с неочевидной внутренней логикой, когда число правил превышает несколько десятков.

§1.6 Алгоритмы вычисления оценок

Алгоритмы вычисления оценок (АВО) были предложены Ю. И. Журавлёвым в начале 70-х годов [8, 11]. Они совмещают метрические и логические принципы классификации, отбор признаков и взвешенное голосование в рамках единой конструкции. Поэтому АВО можно рассматривать как универсальный язык для описания очень широкого класса алгоритмов классификации.

От метрических алгоритмов АВО наследует принцип оценивания сходства объектов. Понятие сходства часто допускает несколько альтернативных формализаций. Например, если объекты описываются числовыми признаками $f_1, \ldots, f_n$, то можно ввести n метрик, оценивающих сходство объектов по каждому из признаков:

$$\rho_s(x, x') = |f_s(x) - f_s(x')|, \quad s = 1, \ldots, n. \tag{1.14}$$

В некоторых практических задачах проще измерять расстояния между объектами, чем формировать их признаковые описания. Например, к таким задачам относится идентификация подписей или классификация белковых молекул по их первичной структуре. Причём, как правило, имеется несколько альтернативных способов определить функцию расстояния.

В общем случае предполагается, что задан набор метрик $\rho_s(x, x')$, $s = 1, \ldots, n$, причём невозможно априори сказать, какая из них «самая правильная».

От логических алгоритмов АВО наследует принцип поиска конъюнктивных закономерностей. В отличие от рассмотренных выше алгоритмов КОРА и ТЭМП, конъюнкции строятся не над бинарными признаками $\beta(x)$, а над бинарными функциями близости вида $\beta(x, x') = [\rho_s(x, x') < \varepsilon_s]$. В этом случае каждой закономерности соответствует не подмножество признаков, а подмножество метрик, называемое *опорным множеством*. Как правило, одного опорного множества не достаточно для надёжной классификации, поэтому в АВО применяется взвешенное голосование по системе опорных множеств.

Опорное множество обладает высокой информативностью в том случае, когда объекты, близкие по всем метрикам опорного множества, существенно чаще оказываются лежащими в одном классе, чем в разных. Таким образом, в АВО гипотеза о существовании информативных закономерностей, свойственная логическим алгоритмам, оказывается эквивалентной гипотезе компактности, свойственной метрическим алгоритмам.

Принцип классификации, применяемый в АВО, называется также *принципом частичной прецедентности — объект x относится к тому классу, в котором имеется больше обучающих объектов, близких к x по информативным частям признаковых описаний (опорным множествам).*

Строение АВО. Перейдём теперь к формальному описанию модели АВО.

1. Задаются функции расстояния $\rho_s \colon X \times X \to \mathbb{R}_+$, $s = 1, \ldots, n$, в общем случае не обязательно вида (1.14) и даже не обязательно метрики.

2. Задаётся *система опорных множеств*

$$\Omega = \big\{\omega \;\big|\; \omega \subseteq \{1, \ldots, n\}\big\}.$$

3. Вводится бинарная *пороговая функция близости*, оценивающая сходство пары объектов $x, x' \in X$ по опорному множеству $\omega \in \Omega$,

$$B_\omega(x, x') = \bigwedge_{s \in \omega} \left[\rho_s(x, x') \leqslant \varepsilon_s\right], \tag{1.15}$$

где ε_s — неотрицательные числа, называемые *порогами*, $s = 1, \ldots, n$. В случае (1.14) порог ε_s называют *точностью измерения* признака f_s.

4. Вводится *оценка близости объекта $x \in X$ к классу $c \in Y$* как результат взвешенного голосования близостей объекта x ко всем обучающим объектам класса c по всем опорным множествам:

$$\Gamma_c(x) = \sum_{i \colon y_i = c} \sum_{\omega \in \Omega} \alpha_{\omega i} B_\omega(x, x_i), \tag{1.16}$$

где веса $\alpha_{\omega i}$, как правило, предполагаются нормированными на единицу:

$$\sum_{i \colon y_i = c} \sum_{\omega \in \Omega} \alpha_{\omega i} = 1, \quad \text{для всех } c \in Y.$$

5. Алгоритм классификации $a(x)$ относит объект x к тому классу, который набрал наибольшую сумму голосов:

$$a(x) = \arg\max_{c \in Y} \Gamma_c(x).$$

Итак, алгоритм вычисления оценок задаётся системой опорных множеств Ω, порогами ε_s, $s = 1, \ldots, n$ и весами $\alpha_{\omega i}$, $\omega \in \Omega$, $i = 1, \ldots, \ell$. Эти параметры оптимизируются по критерию минимума числа ошибок на обучающей выборке.

Обучение АВО. Описанный алгоритм совпадает с общим алгоритмом взвешенного голосования (1.9)–(1.10), если в качестве правил R_c взять функции близости:

$$R_c = \big\{\varphi_c(x) = B_\omega(x, x_i) \;\big|\; \omega \in \Omega,\ y_i = c,\ i = 1, \ldots, \ell\big\}, \quad c \in Y.$$

Поскольку функции близости являются конъюнкциями, для оптимизации описанного варианта АВО можно приспособить стандартные алгоритмы, описанные в предыдущих параграфах. Для поиска информативных конъюнкций подходят градиентные методы, КОРА или ТЭМП. Для настройки весов $\alpha_{\omega i}$ можно применить бустинг или любой линейный классификатор, например, SVM.

Для применения алгоритмов поиска информативных конъюнкций требуется задать семейство элементарных предикатов $\mathscr{B}$. В данном случае оно имеет вид

$$\mathscr{B} = \big\{\beta(x) = [\rho_s(x, x_i) \leqslant \varepsilon_s] \;\big|\; s = 1, \ldots, n,\ y_i = c,\ i = 1, \ldots, \ell\big\}.$$

Для сокращения перебора при построении конъюнкций имеет смысл заранее отобрать лишь некоторые значения порогов $\{\varepsilon_s\}$, например, с помощью Алгоритма 1.1, выделяющего информативные зоны. Заметим, что такой способ выбора порогов не даёт гарантии их оптимальности. Оптимизация порогов является сложной комбинаторной задачей, для практического решения которой разработаны достаточно удачные эвристические алгоритмы [16].

Ещё один способ сокращения перебора — заранее отобрать в качестве эталонов не все обучающие объекты, а только наиболее представительные. Это эквивалентно обнулению весов $\alpha_{\omega i}$ для некоторых объектов x_i, $i = 1, \ldots, \ell$. Разумеется, эти объекты по-прежнему остаются в выборке и используются при настройке параметров ε_s, $\alpha_{\omega i}$ и оценивании информативности закономерностей.

Наконец, необходимо учесть, что строение АВО накладывает специфическое ограничение на процедуру наращивания конъюнкций — конъюнкция не должна содержать элементарных предикатов, относящихся к различным эталонным объектам. То есть должны быть запрещены конструкции вида

$$\varphi_c(x) = [\rho_s(x, x_i) \leqslant \varepsilon_s] \wedge [\rho_t(x, x_j) \leqslant \varepsilon_t],$$

в которых $i \neq j$, поскольку они не являются функциями пары объектов (x, x_i). Это обстоятельство несложно учесть при реализации цикла перебора по множеству элементарных предикатов $\mathscr{B}$. Например, в Алгоритме 1.9 (ТЭМП) это отразится только на шаге 6.

Таким образом, рассмотренные ранее логические алгоритмы легко приспособить для обучения АВО, что является несложным техническим упражнением.

Другие варианты АВО. Модель АВО допускает различные варианты задания системы опорных множеств, функций близости и решающего правила. Различным вариантам АВО, методам их оптимизации и приложениям посвящена обширная библиография, см., например, [9, 10].

1. В качестве системы опорных множеств можно взять все подмножества $\omega \subseteq \Omega$ заданной мощности k. В этом случае возможно аналитическое преобразование формул вычисления оценок, приводящее к эффективным вариантам АВО [10]. Другой подход к выбору системы Ω — построение тупиковых тестов или тупиковых представительных наборов, которые рассматриваются в следующем разделе.

2. Конъюнктивную функцию близости можно обобщить, введя ещё один параметр алгоритма ε:

$$B_\omega(x, x') = \left[\sum_{s\in\omega} [\rho_s(x, x') > \varepsilon_s] \leqslant \varepsilon\right],$$

При $\varepsilon = 0$ этот вариант совпадает с (1.15). При $\varepsilon > 0$ функция близости уже не является конъюнкцией.

Другой способ обобщения функции близости заключается в том, чтобы вместо конъюнкции пороговых функций использовать взвешенное голосование метрик:

$$B_\omega(x, x') = \left[\sum_{s\in\omega} \alpha_s \rho_s(x, x') \leqslant \varepsilon\right].$$

В этом случае вместо параметров точности измерения признаков ε_s задаются параметры α_s — веса метрик, образующих опорное множество ω.

3. Веса можно вводить для объектов и признаков, и уже через них определять веса функций близости, входящие в (1.16). Чаще всего используется следующее аддитивно-мультипликативное представление весов:

$$\alpha_{\omega i} = \frac{1}{Z}\gamma_i \sum_{s\in\omega} p_s,$$

где Z — нормирующий множитель, γ_i — вес i-го обучающего объекта, p_s — вес s-го признака. В этом случае алгоритм бустинга для настройки весов уже неприменим, так как классификатор становится билинейной функцией параметров.

4. Решающее правило иногда усложняют, вводя дополнительные параметры δ_1 и δ_2. Объект x относится к классу c, если выполняются два условия:

$$\Gamma_c(x) > \Gamma_y(x) + \delta_1, \quad \text{для всех } y \in Y \setminus \{c\};$$
$$\Gamma_c(x) > \delta_2 \sum_{y\in Y\setminus\{c\}} \Gamma_y(x);$$

В остальных случаях алгоритм отказывается от классификации объекта x.

1.6.1 Тупиковые представительные наборы

Опр. 1.6. *Совокупность $\langle\omega, i\rangle$ называется представительным набором, если объект x_i отличим от любого объекта другого класса $x_j \in X$, $y_j \neq y_i$ по опорному множеству ω: $B_\omega(x_j, x_i) = 0$.*

Опр. 1.7. *Представительный набор $\langle\omega, i\rangle$ называется тупиковым, если для любого собственного подмножества $\omega' \subset \omega$ совокупность $\langle\omega', i\rangle$ не является представительным набором.*

При голосовании по тупиковым представительным наборам для каждого объекта $x_i \in X$ фактически строится своя подсистема опорных множеств:

$$\Omega_i = \big\{\omega \;\big|\; \langle \omega, i\rangle \text{ — тупиковый представительный набор}\big\}, \quad i = 1, \ldots, \ell,$$

и оценка близости объекта x к классу c вычисляется по формуле

$$\Gamma_c(x) = \sum_{i:\; y_i = c} \sum_{\omega \in \Omega_i} \alpha_{\omega i} B_\omega(x, x_i),$$

которая является частным случаем (1.16), если положить $\alpha_{\omega i} = 0$ для всех $\langle \omega, i\rangle$, не являющихся тупиковыми представительными наборами.

Утв. 1.4. *Всякой непротиворечивой закономерности вида $\varphi_c(x) = B_\omega(x, x_i)$ относительно класса $c = y_i$ соответствует представительный набор $\langle \omega, i\rangle$.*

Обратное, вообще говоря, неверно. Представительному набору $\langle \omega, i\rangle$ соответствует предикат $\varphi_c(x) = B_\omega(x, x_i)$, где $c = y_i$, про который известно только, что

$$\begin{cases} n_c(\varphi_c) = 0 & \text{— по определению представительного набора;} \\ p_c(\varphi_c) \geqslant 1 & \text{— поскольку } B_\omega(x_i, x_i) = 1 \text{ для любого } i = 1, \ldots, \ell. \end{cases}$$

Число выделяемых позитивных объектов $p_c(\varphi_c)$ может оказаться недостаточным, чтобы представительный набор можно было называть закономерностью в соответствии с определением 1.1 или 1.2.

Утверждение 1.4 показывает, что для построения тупиковых представительных наборов можно применять известные алгоритмы поиска информативных конъюнкций в семействе предикатов φ вида (1.15) по критериям $n_c(\varphi) = 0$ и $p_c(\varphi) \to \max\limits_{\varphi}$.

1.6.2 Тупиковые тесты

Опр. 1.8. *Множество $\omega \subseteq \{1, \ldots, n\}$ называется тестом, если для любых $x_i, x_j \in X$ из $y_j \neq y_i$ следует $B_\omega(x_j, x_i) = 0$, то есть любые два объекта из разных классов различимы по опорному множеству ω.*

В отличие от представительного набора, тест не привязан к какому-либо конкретному объекту x_i.

Опр. 1.9. *Тест ω называется тупиковым, если любое его собственное подмножество $\omega' \subset \omega$ не является тестом.*

Алгоритм вычисления оценок, в котором системой опорных множеств Ω является множество всех тупиковых тестов, называется *тестовым алгоритмом*. Исторически это был первый вариант АВО, предложенный Ю. И. Журавлёвым.

Построим по исходной задаче классификации $\langle X, Y, y^*, X^\ell\rangle$ вспомогательную задачу классификации $\langle U, V, v^*, U^L\rangle$, в которой:

$U = X \times X$ — объекты из U есть пары объектов из X;
$V = \{0, 1\}$ — множество допустимых ответов двухэлементно;
$v^*(x, x') = \big[y^*(x) = y^*(x')\big]$ — целевая зависимость;

$U^L = \{(x_i, x_j) \mid x_i, x_j \in X^\ell, i \leqslant j\}$ — обучающая выборка, $L = \ell(\ell+1)/2$.

Таким образом, в этой задаче распознаётся попадание пары объектов $u = (x, x')$ в один класс. Нас интересуют закономерности относительно класса $1 \in V$.

Утв. 1.5. *Всякой непротиворечивой закономерности вида* $\varphi_1(x, x') = B_\omega(x, x')$ *относительно класса* $1 \in V$ *на выборке* U^L *соответствует тест* ω.

Обратное, вообще говоря, неверно. Тесту ω соответствует предикат $\varphi_1(x, x') = B_\omega(x, x')$, про который известно только, что

$$\begin{cases} n_1(\varphi_1) = 0 & \text{— по определению теста;} \\ p_1(\varphi_1) \geqslant \ell & \text{— поскольку } B_\omega(x_i, x_i) = 1 \text{ для всех } i = 1, \ldots, \ell. \end{cases}$$

Число $p_1(\varphi_1)$ выделяемых пар объектов, лежащих в одном классе, может оказаться недостаточным, чтобы предикат φ_1 можно было называть закономерностью в соответствии с определением 1.1 или 1.2.

Утверждение 1.5 показывает, что для построения тупиковых тестов можно применять алгоритмы поиска непротиворечивых конъюнктивных закономерностей, переходя к вспомогательной задаче классификации и максимизируя функционал $p_1(\varphi_1)$.

Замечание 1.10. При переходе к вспомогательной задаче размер выборки фактически возрастает квадратично — с ℓ до $\ell(\ell+1)/2$. Поэтому тестовый алгоритм особенно эффективен при решении задач классификации с малым числом объектов и большим числом признаков.

§1.7 Поиск ассоциативных правил

Пусть на множестве объектов X задано n бинарных признаков $\mathscr{F} = \{f_1, \ldots, f_n\}$, $f_j \colon X \to \{0, 1\}$, и имеется выборка $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$.

Каждому набору признаков $\varphi \subseteq \mathscr{F}$ поставим в соотвествие предикат $\varphi(x)$, равный конъюнкции всех признаков из φ:

$$\varphi(x) = \bigwedge_{f \in \varphi} f(x), \quad x \in X.$$

Если $\varphi(x) = 1$, то говорят, что признаки набора φ *совместно встречаются* у объекта x. Частотой встречаемости или *поддержкой* (support) набора φ в выборке X^ℓ называется функция

$$\nu(\varphi) = \frac{1}{\ell} \sum_{i=1}^{\ell} \varphi(x_i).$$

Набор $\varphi \subseteq \mathscr{F}$ называется *часто встречающимся набором* (frequent itemset), если $\nu(\varphi) \geqslant \delta$. Параметр δ называется минимальной поддержкой, и в литературе обозначается `MinSupp`.

Опр. 1.10. *Пара непересекающихся наборов $\varphi, y \subseteq \mathscr{F}$ называется ассоциативным правилом (association rule) «$\varphi \to y$», если выполнены два условия:*

$$\nu(\varphi \cup y) \geqslant \delta; \qquad \nu(y|\varphi) \equiv \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \geqslant \varkappa.$$

Величина $\nu(y|\varphi)$ называется *значимостью* (confidence) правила. Параметр $\varkappa$ называется минимальной значимостью и обозначается `MinConf`.

Итак, в ассоциативном правиле «$\varphi \to y$» наборы φ и y совместно часто встречаются, и в значительной доле случаев (не менее $\varkappa$) если встречается набор φ, то встречается также и набор y.

Пример 1.9. Задача поиска ассоциативных правил исходно возникла в связи с *анализом рыночных корзин* (market basket analysis) [19, 18]. В наиболее распространённом частном случае признаки соответствуют товарам в супермаркете, в общем случае — некоторым *предметам* (items). Объекты соответствуют покупкам, точнее, чекам или *транзакциям*. Если $f_j(x_i) = 1$, то в i-м чеке зафиксирована покупка j-го товара. Задача заключается в том, чтобы выявить наборы товаров, которые часто покупают вместе. Например, «если куплен хлеб φ, то будет куплено и молоко y с вероятностью $\nu(y|\varphi) = 60\%$; причём оба товара покупаются совместно с вероятностью $\nu(\varphi \cup y) = 2\%$». Заметим, что величина $\nu(y|\varphi)$ является оценкой условной вероятности того, что покупатель приобретёт набор товаров y, если он уже приобрёл набор φ. Выявление совместно продаваемых товаров позволяет оптимизировать размещение товаров на полках, планировать рекламные кампании (промо-акции), более эффективно управлять ценами и ассортиментом.

Поиск ассоциативных правил принято относить к задачам *обучения без учителя* (unsupervised learning).

Покажем, что понятие ассоциативного правила является прямым обобщением понятия логической ε, δ-закономерности (стр. 3) на тот случай, когда закономерность ищется в форме конъюнкции, при этом целевой признак не фиксирован и тоже ищется в форме конъюнкции. Для ассоциативного правила «$\varphi \to y$» возьмём в качестве целевого признака $y^*(x) = \bigwedge_{f \in y} f(x)$. Тогда, согласно определению 1.1,

$$D_1(\varphi) = \nu(\varphi \cup y) \geqslant \delta;$$
$$E_1(\varphi) = 1 - \nu(y|\varphi) \leqslant 1 - \varkappa \equiv \varepsilon.$$

Таким образом, различия двух определений — чисто терминологические.

Алгоритм поиска ассоциативных правил. Исторически одним из первых стал алгоритм APriory [20]. Позже было придумано множество более эффективных алгоритмов. Все они так или иначе направлены на усовершенствование APriory, который считается базовым. Пик исследований в этой области пришёлся на конец 90-х, когда эти алгоритмы стали чрезвычайно востребованными в бизнес-приложениях.

Алгоритм основан на свойстве *антимонотонности*: для любого набора $\psi \in \mathscr{F}$ и любого его собственного подмножества $\varphi \subset \psi$ справедливо $\nu(\varphi) \geqslant \nu(\psi)$. Иными словами, добавление признака в набор может привести только к снижению частоты его встречаемости. Отсюда следствие: чтобы набор ψ был часто встречающимся,

Алгоритм 1.11. APriory — поиск ассоциативных правил

Вход:
X^ℓ — обучающая выборка;
$\delta, \varkappa$ — минимальная поддержка и минимальная значимость;
Выход:
$R = \{(\varphi, y)\}$ — список всех найденных ассоциативных правил;

1: множество всех часто встречающихся исходных признаков:
$G_1 := \{f \in \mathscr{F} \mid \nu(f) \geqslant \delta\}$;
2: **для всех** $j = 2, \ldots, n$
3: множество всех часто встречающихся наборов мощности j:
$G_j := \{\varphi \cup \{f\} \mid \varphi \in G_{j-1},\ f \in G_1,\ \nu(\varphi \cup \{f\}) \geqslant \delta\}$;
4: **если** $G_j = \varnothing$ **то**
5: **выход** из цикла по j;
6: $R := \varnothing$;
7: **для всех** $\psi \in G_j,\ j = 2, \ldots, n$
8: `AssocRules` $(\psi, \varnothing)$;

9: **ПРОЦЕДУРА** `AssocRules` (φ, y);
10: **для всех** $f \in \varphi$
11: $\varphi' := \varphi \setminus \{f\}; \quad y' := y \cup \{f\}$;
12: **если** $\nu(y'|\varphi') \geqslant \varkappa$ **то**
13: добавить ассоциативное правило (φ', y') в список результатов R;
14: **если** $|\varphi'| > 1$ **то**
15: `AssocRules` (ψ', y');

необходимо (но не достаточно), чтобы все его подмножества $\varphi \subset \psi$ были часто встречающимися. Это свойство позволяет реализовать перебор более эффективно. Понятно, что если некоторый набор φ не является часто встречающимся, то все наборы, содержащие его, также не являются часто встречающимся, и нет никакого смысла их проверять.

Алгоритм 1.11 состоит из двух этапов.

На первом этапе (шаги 1–5) ищутся все часто встречающиеся наборы мощности $j = 1, 2, 3, \ldots$ Это наиболее трудоёмкий в вычислительном плане этап. В реальных приложениях исходные данные о транзакциях имеют огромные объёмы и хранятся в базе данных. В Алгоритме 1.11 приводится только основная идея — выполнение поиска в ширину, а её конкретная реализация может сильно зависеть от способа хранения данных.

На втором этапе (шаги 6–8) рекурсивно вызывается процедура `AssocRules`(φ, y) для синтеза всех ассоциативных правил, которые можно получить, переводя некоторые признаки из набора φ в набор y. Второй этап может быть проведён целиком в оперативной памяти, поскольку частоты всех часто встречающихся наборов (и, следовательно, всех их собственных подмножеств) уже вычислены на первом этапе.

Некоторые усовершенствования

- Разработано большое количество эффективных алгоритмов поиска ассоциативных правил, основанных на специальных структурах хранения данных, см. обзоры [20, 29, 38, 28].

- *Онлайновые алгоритмы* учитывают то обстоятельство, что база транзакций X^ℓ накапливается в реальном масштабе времени. Некоторые правила со временем перестают удовлетворять определению, с другой стороны, появляются новые. Существуют способы быстрой перепроверки существующих правил и «кандидатов в правила» при появлении новой порции транзакций [27].

- Радикальное повышение эффективности может быть достигнуто путём *поиска по случайной подвыборке* (sampling). На первом шаге правила строятся по относительно небольшой подвыборке при несколько заниженных порогах δ, $\varkappa$. Известны статистические оценки, показывающие, на сколько необходимо занизить пороги, чтобы доля пропущенных правил не превысила заданной величины, с заданной вероятностью. На втором шаге построенные правила один раз проверяются по полной базе, что может быть сделано довольно быстро [36].

- *Обобщённые ассоциативные правила* учитывают то обстоятельство, что в реальных приложениях над признаками существует многоуровневая иерархия разделов. Например, в базах данных супермаркетов все товары распределяются по товарным группам, группы в свою очередь объединяются в более крупные группы. Нет особого смысла учитывать, что именно данный сорт хлеба часто покупают именно с данным сортом молока, и, скорее всего, такое правило окажется незначимым. Задача заключается в том, чтобы во всей товарной иерархии найти часто встречающиеся сочетания не самих товаров, а их товарных групп. Учёт дополнительной информации о группировании товаров усложняет алгоритм, но повышает эффективность поиска правил [35].

§1.8 Выводы

Все без исключения методы синтеза логических алгоритмов стремятся найти закономерности получше (с наибольшей информативностью), построить из них алгоритм попроще (чтобы он лучше интерпретировался), но при этом высокого качества, и сделать это как можно быстрее. Удовлетворить столь противоречивым требованиям не так просто. Поэтому методов синтеза логических классификаторов придумано очень много, и постоянно появляются новые. Каждый имеет свои достоинства и недостатки, и свой набор эвристик для сокращения времени перебора.

Снова о статистических и логических закономерностях. При построении логических алгоритмов классификации в одних ситуациях лучше применять эвристический критерий информативности, в других ситуациях — статистический (гипергеометрический, энтропийный, и др.). Попробуем резюмировать эти различия.

1. Решающие списки и деревья. Решения, принимаемые отдельными правилами, являются окончательными и непосредственно выдаются алгоритмом. Поэтому требования к ним жёстче — они должны допускать мало ошибок, то есть быть ε, δ-закономерностями.

2. Алгоритмы голосования. Решения, принимаемые отдельными правилами, усредняются, при этом их ошибки компенсируют друг друга. Здесь важнее сместить пропорцию $n : p \ll N : P$, чем минимизировать число ошибок n. Для этой цели больше подходят статистические критерии.

3. Промежуточные стадии построения конъюнкций. Во многих алгоритмах (включая списки и деревья) правила имеют вид конъюнкций, которые наращиваются постепенно, терм за термом. В процессе наращивания качество «полуфабрикатов» также удобнее оценивать по статистическому критерию. И только для окончательного отбора закономерностей используется более строгий ε, δ-критерий.

Преимущества логических алгоритмов классификации.

- Во многих задачах существование логических закономерностей является объективным фактом. Если алгоритму обучения удаётся их отыскать, то, как правило, они служат надёжными правилами классификации.
- Интерпретируемость. Построенный алгоритм может быть записан на естественном языке в виде набора правил — логических высказываний.
- Простота реализации классификатора. После того, как алгоритм классификации обучен, собственно классификация может производиться даже вручную.
- Возможность обработки разнотипных данных и данных с пропусками.

Недостатки логических алгоритмов классификации.

- Существуют задачи, в которых применение пороговых решающих правил и дискретных методов обучения нежелательно. Например, когда требуется построение гладких границ между классами.
- Процедура обучения, как правило, сводится к решению NP-полных задач. Для их приближённого решения приходится применять различные эвристические приёмы, не гарантирующие оптимальности решения.
- Неоднозначность в выборе хороших эвристик для сокращения перебора. Как правило, не удаётся доказать, что одна эвристика работает лучше другой. Это можно только продемонстрировать эмпирически, путём решения большого числа реальных задач.

Резюме

1.

Упражнения

Упр. 1.1. Доказать: если случайные величины $y^*(x)$ и $\varphi(x)$ независимы, выборка X^ℓ простая, то вероятность реализации пары (p_φ, n_φ) подчиняется гипергеометрическому распределению (1.1).

Упр. 1.2. Доказать теорему 1.1.

Упр. 1.3. Доказать, что в Алгоритме 1.1 начальное слияние зон «по границам классов» на шагах 2–4 может только увеличивать информативность зон.

Упр. 1.4. Доказать: решающий список длины k реализует более широкое множество функций, чем k-DNF, k-KNF, k-DT. Если же запретить классам перемежаться, то решающий список длины k реализует то же самое множество функций, что k-DNF.

Упр. 1.5. Доказать: множества объектов $\{x \in X\colon K_v(x) = 1\}$, выделяемых конъюнкциями (1.8), попарно не пересекаются, а их объединение совпадает со всем пространством объектов X.

Список литературы

[1] *Белецкий Н. Г.* Применение комитетов для многоклассовой классификации // Численный анализ решения задач линейного и выпуклого программирования. — Свердловск, 1983. — С. 156–162.

[2] *Вайнцвайг М. Н.* Алгоритм обучения распознаванию образов «кора» // Алгоритмы обучения распознаванию образов / Под ред. В. Н. Вапник. — М.: Советское радио, 1973. — С. 110–116.

[3] *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы. — М.: Физматлит, 2006. — 320 с.

[4] *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. — М.: Мир, 1998.

[5] *Донской В. И., Башта А. И.* Дискретные модели принятия решений при неполной информации. — Симферополь: Таврия, 1992. — 166 с.

[6] *Дюличева Ю. Ю.* Стратегии редукции решающих деревьев (обзор) // *Таврический вестник информатики и математики.* — 2002. — № 1. — С. 10–17.

[7] *Емельянов В. В., Курейчик В. В., Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003. — 432 с.

[8] *Журавлёв Ю. И.* Экстремальные задачи, возникающие при обосновании эвристических процедур // Приблемы прикладной математики и механики. — М.: Наука, 1971. — С. 67–74.
http://www.ccas.ru/frc/papers/zhuravlev71extremal.pdf.

[9] *Журавлёв Ю. И.* Непараметрические задачи распознавания образов // *Кибернетика.* — 1976. — № 6.

[10] *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // *Проблемы кибернетики.* — 1978. — Т. 33. — С. 5–68.
http://www.ccas.ru/frc/papers/zhuravlev78prob33.pdf.

[11] *Журавлёв Ю. И., Никифоров В. В.* Алгоритмы распознавания, основанные на вычислении оценок // *Кибернетика.* — 1971. — № 3.

[12] *Лбов Г. С.* Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.

[13] *Норушис А.* Построение логических (древообразных) классификаторов методами нисходящего поиска (обзор) // Статистические проблемы управления. Вып. 93 / Под ред. Ш. Раудис. — Вильнюс, 1990. — С. 131–158.

[14] *Полякова М. П., Вайнцвайг М. Н.* Об использовании метода «голосования» признаков в алгоритмах распознавания // Моделирование обучения и поведения. — М., 1975. — С. 25—28.

[15] *Редько В. Г.* Эволюционная кибернетика. — М.: Наука, 2003. — 155 с.

[16] *Рязанов В. В., Сенько О. В.* О некоторых моделях голосования и методах их оптимизации // *Распознавание, классификация, прогноз.* — 1990. — Т. 3. — С. 106–145.

[17] *Яблонский С. В.* Введение в дискретную математику. — М.: Наука, 1986.

[18] *Agrawal R., Imielinski T., Swami A.* Database mining: A performance perspective // Special Issue on Learning and Discovery in Knowledge-Based Databases / Ed. by N. Cercone, M. Tsuchiya. — Washington, U.S.A.: Institute of Electrical and Electronics Engineers, 1993. — 6 no. 5. — Pp. 914–925.
http://citeseer.ist.psu.edu/agrawal93database.html.

[19] *Agrawal R., Imielinski T., Swami A. N.* Mining association rules between sets of items in large databases // Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data / Ed. by P. Buneman, S. Jajodia. — Washington, D.C.: 1993. — Pp. 207–216.
http://citeseer.ist.psu.edu/agrawal93mining.html.

[20] *Agrawal R., Srikant R.* Fast algorithms for mining association rules // Proc. 20th Int. Conf. Very Large Data Bases, VLDB / Ed. by J. B. Bocca, M. Jarke, C. Zaniolo. — Morgan Kaufmann, 1994. — Pp. 487–499.
http://citeseer.ist.psu.edu/agrawal94fast.html.

[21] *Breslow L. A., Aha D. W.* Simplifying decision trees: a survey // *Knowledge Engineering Review.* — 1997. — Vol. 12, no. 1. — Pp. 1–40.
http://citeseer.ist.psu.edu/breslow96simplifying.html.

[22] *Cohen W. W., Singer Y.* A simple, fast and effective rule learner // Proc. of the 16 National Conference on Artificial Intelligence. — 1999. — Pp. 335–342.
http://citeseer.ist.psu.edu/198445.html.

[23] *Dubner P. N.* Statistical tests for feature selection in KORA recognition algorithms // *Pattern Recognition and Image Analysis.* — 1994. — Vol. 4, no. 4. — P. 396.

[24] *Esmeir S., Markovitch S.* Lookahead-based algorithms for anytime induction of decision trees // Proceedings of the 21st International Conference on Machine Learning (ICML-2004). — 2004.
http://citeseer.ist.psu.edu/esmeir04lookaheadbased.html.

[25] *Freund Y., Schapire R. E.* A decision-theoretic generalization of on-line learning and an application to boosting // European Conference on Computational Learning Theory. — 1995. — Pp. 23–37.
http://citeseer.ist.psu.edu/article/freund95decisiontheoretic.html.

[26] *Fürnkranz J., Flach P. A.* Roc 'n' rule learning-towards a better understanding of covering algorithms // *Machine Learning*. — 2005. — Vol. 58, no. 1. — Pp. 39–77.
http://dblp.uni-trier.de/db/journals/ml/ml58.html#FurnkranzF05.

[27] *Hidber C.* Online association rule mining // SIGMOD Conf. — 1999. — Pp. 145–156.
http://citeseer.ist.psu.edu/hidber98online.html.

[28] *Hipp J., Güntzer U., Nakhaeizadeh G.* Algorithms for association rule mining — a general survey and comparison // *SIGKDD Explorations*. — 2000. — Vol. 2, no. 1. — Pp. 58–64.
http://citeseer.ist.psu.edu/hipp00algorithms.html.

[29] *Mannila H., Toivonen H., Verkamo A. I.* Efficient algorithms for discovering association rules // AAAI Workshop on Knowledge Discovery in Databases (KDD-94) / Ed. by U. M. Fayyad, R. Uthurusamy. — Seattle, Washington: AAAI Press, 1994. — Pp. 181–192.
http://citeseer.ist.psu.edu/mannila94efficient.html.

[30] *Marchand M., Shah M., Shawe-Taylor J., Sokolova M.* The set covering machine with data-dependent half-spaces // Proc. 20th International Conf. on Machine Learning. — Morgan Kaufmann, 2003. — Pp. 520–527.
http://www.hpl.hp.com/conferences/icml03/titlesAndAuthors.html.

[31] *Marchand M., Shawe-Taylor J.* Learning with the set covering machine // Proc. 18th International Conf. on Machine Learning. — Morgan Kaufmann, San Francisco, CA, 2001. — Pp. 345–352.
http://citeseer.ist.psu.edu/452556.html.

[32] *Martin J. K.* An exact probability metric for decision tree splitting and stopping // *Machine Learning*. — 1997. — Vol. 28, no. 2-3. — Pp. 257–291.
http://citeseer.ist.psu.edu/martin97exact.html.

[33] *Quinlan J.* Induction of decision trees // *Machine Learning*. — 1986. — Vol. 1, no. 1. — Pp. 81–106.

[34] *Rivest R. L.* Learning decision lists // *Machine Learning*. — 1987. — Vol. 2, no. 3. — Pp. 229–246.
http://citeseer.ist.psu.edu/rivest87learning.html.

[35] *Srikant R., Agrawal R.* Mining generalized association rules // *Future Generation Computer Systems*. — 1997. — Vol. 13, no. 2–3. — Pp. 161–180.
http://citeseer.ist.psu.edu/srikant95mining.html.

[36] *Toivonen H.* Sampling large databases for association rules // In Proc. 1996 Int. Conf. Very Large Data Bases / Ed. by T. M. Vijayaraman, A. P. Buchmann, C. Mohan,

N. L. Sarda. — Morgan Kaufman, 1996. — Pp. 134–145.
http://citeseer.ist.psu.edu/toivonen96sampling.html.

[37] *Whitley D.* An overview of evolutionary algorithms: practical issues and common pitfalls // *Information and Software Technology.* — 2001. — Vol. 43, no. 14. — Pp. 817–831.
http://citeseer.ist.psu.edu/whitley01overview.html.

[38] *Zheng Z., Kohavi R., Mason L.* Real world performance of association rule algorithms // Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining / Ed. by F. Provost, R. Srikant. — 2001. — Pp. 401–406.
http://citeseer.ist.psu.edu/zheng01real.html.