МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Курс лекций

Направление подготовки:

09.04.02 «Информационные системы и технологии»

Квалификация выпускника: магистр

Ставрополь, 2025

Н 63 **Методы искусственного интеллекта:** курс лекций для студентов направления 09.04.02 «Информационные системы и технологии» / Николаев Е.И. – Ставрополь: Изд-во СКФУ, 2025. – 176 с.

Курс лекций по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning). Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки C, C++, C#, Python, R). Цель учебного пособия: сформировать у студентов целостный взгляд на современные тенденции в областях машинного обучения, искусственного интеллекта, анализа данныз; сформировать систему компетенций.

УДК 004.41 (075.8)
ББК 22.18я73

Автор:

канд. техн. наук, доцент **Е.И. Николаев**

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Курс лекций по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning).

Основная задача науки и реальной жизни – получение правильных предсказаний о будущем поведении сложных систем на основании их прошлого поведения. Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Как говорят, мы получаем данные из «черного ящика». В этих условиях ничего не остается, как только изучать доступную нам последовательность исходных данных и пытаться строить предсказания совершенствуя нашу схему в процессе предсказания. Подход, при котором про- шлые данные или примеры используются для первоначального формирования и совершенствования схемы предсказания, называется методом машинного обучения (Machine Learning). Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов. Данное пособие ни в какой мере не претендует на какое-либо исчерпывающее изложение содержания данной области. Основная цель – дать студентам теоретическое представление о современных математических проблемамах в области систем искусственного интеллекта, а также познакомить с путями их решения.

Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки C, C++, C#, Python, R). Цель

учебного пособия: сформировать у студентов целостный взгляд на современные тенденции в областях машинного обучения, искусственного интеллекта, анализа данныз; сформировать систему компетенций.

РАЗДЕЛ 1. ОБУЧЕНИЕ ПО ПРЕЦЕДЕНТАМ

В данном разделе даются базовые понятия и обозначения, которые будут использоваться на протяжении всего курса. Приводятся общие постановки задач обучения по прецедентам и некоторые примеры прикладных задач.

1.1. Основная терминология

Задано множество объектов X, множество допустимых ответов Y , и существует целевая функция (target function) $y^*: X \to Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \dots, x_\ell\} \subset X$. Пары «объект-ответ» (x_i, y_i) называются прецедентами. Совокупность пар $X^\ell = (x_i, y_i)_{i=1}^\ell$ называется обучающей выборкой (training sample).

Задача обучения по прецедентам заключается в том, чтобы по выборке X^ℓ восстановить зависимость y^*, то есть построить решающую функцию (decision function) $a: X \to Y$, которая приближала бы целевую функцию $y^*(x)$, причём не только на объектах обучающей выборки X^ℓ, но и на всём множестве X.

Решающая функция a должна допускать эффективную компьютерную реализацию; по этой причине будем называть её алгоритмом.

1.1.1. Объекты и признаки

Признак (feature) f объекта x – это результат измерения некоторой характеристики объекта. Формально признаком называется отображение $f : X \to D_f$, где D_f – множество допустимых значений признака. В частности, любой алгоритм $a: X \to Y$ также можно рассматривать как признак.

В зависимости от природы множества D_f признаки делятся на несколько типов.

Если $D_f = \{0,1\}$, то f – бинарный признак;

Если D_f – конечное множество, то f – номинальный признак;

Если D_f – конечное упорядоченное множество, то f – порядковый признак;

Если $D_f = \mathbb{R}$, то f – количественный признак.

Если все признаки имеют одинаковый тип, $D_{f_1} = \cdots = D_{f_n}$, то исходные данные называются однородными, в противном случае – разнородными.

Пусть имеется набор признаков $f_1, \ldots, f_n$. Вектор $(f_1(x), \ldots, f_n(x))$ называют признаковым описанием объекта $x \in X$. В дальнейшем мы не будем различать объекты из X и их признаковые описания, полагая $X = D_{f_1} \times \ldots \times D_{f_n}$. Совокупность признаковых описаний всех объектов выборки X^ℓ, записанную в виде таблицы размера $\ell \times n$, называют матрицей объектов-признаков:

$$F = \|f_j(x_i)\|_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_\ell) & \cdots & f_n(x_\ell) \end{pmatrix} \quad (1.1)$$

Матрица объектов-признаков является стандартным и наиболее распространённым способом представления исходных данных в прикладных задачах.

1.1.2. Ответы и типы задач

В зависимости от природы множества допустимых ответов Y задачи обучения по прецедентам делятся на следующие типы.

Если $Y = \{1, \ldots, M\}$, то это задача классификации (classification) на M непересекающихся классов. В этом случае всё множество объектов X разбивается на классы $K_y = \{x \in X: y^*(x) = y\}$, и алгоритм $a(x)$ должен давать ответ на вопрос «какому классу принадлежит x?». В некоторых приложениях классы называют образами и говорят о задаче распознавания образов (pattern recognition).

Если $Y = \{0,1\}^M$, то это задача классификации на M пересекающихся классов. В простейшем случае эта задача сводится к решению M независимых задач классификации с двумя непересекающимися классами.

Если $Y = R$, то это задача восстановления регрессии (regression estimation).

Задачи прогнозирования (forecasting) являются частными случаями классификации или восстановления регрессии, когда $x \in X$ – описание прошлого поведения объекта x, $y \in Y$ – описание некоторых характеристик его будущего поведения.

1.1.3. Модель алгоритмов и метод обучения

Моделью алгоритмов называется параметрическое семейство отображений $A = \{g(x,\theta)|\theta \in \Theta\}$, где $g: X \times \Theta \to Y$ – некоторая фиксированная функция, Θ – множество допустимых значений параметра θ, называемое пространством параметров или пространством поиска (search space).

Пример 1.1. В задачах с n числовыми признаками $f_j: X \to \mathbb{R}$, $j = 1,\ldots,n$ широко используются линейные модели с вектором параметров $\theta = (\theta_1,\ldots,\theta_n) \in \Theta = \mathbb{R}_n$:

$g(x,\theta) = \sum_{j=1}^{n} \theta_j f_j(x)$ – для задач восстановления регрессии, $Y = \mathbb{R}$;

$g(x,\theta) = \text{sign} \sum_{j=1}^{n} \theta_j f_j(x)$ – для задач классификации, $Y = \{-1,+1\}$.

Признаками могут быть не только исходные измерения, но и функции от них. В частности, многомерные линейные модели могут использоваться даже в задачах с единственным числовым признаком.

Пример 1.2. Один из классических подходов к аппроксимации функций одной переменной по заданным точкам $(x_i, y_i) \in \mathbb{R}^2$, $i = 1,\ldots,\ell$ заключается в построении полиномиальной модели. Если ввести n признаков $f_j(x) = x^{j-1}$, то функция $g(x,\theta)$ из примера 1.1 будет определять полином степени $n-1$ над исходным признаком x.

Процесс подбора оптимального параметра модели θ по обучающей выборке X^ℓ называют настройкой (fitting) или обучением (training, learning) алгоритма $a \in A$.

Согласно английской терминологии алгоритм является обучаемым, учеником (learning machine), а выборка данных – обучающей, учителем (training sample).

Метод обучения (learning algorithm) – это отображение μ: $(X \times Y)^\ell \to A$, которое произвольной конечной выборке $X^\ell = (x_i, y_i)_{i=1}^\ell$ ставит в соответствие некоторый алгоритм $a \in A$. Говорят также, что метод μ строит алгоритм a по выборке X^ℓ. Метод обучения должен допускать эффективную программную реализацию.

Итак, в задачах обучения по прецедентам чётко различаются два этапа.

На этапе обучения метод μ по выборке X^ℓ строит алгоритм $a = \mu(X^\ell)$.

На этапе применения алгоритм a для новых объектов x выдаёт ответы $y = a(x)$.

Этап обучения наиболее сложен. Как правило, он сводится к поиску параметров модели, доставляющих оптимальное значение заданному функционалу качества.

1.1.4. Функционал качества

Функция потерь (loss function) – это неотрицательная функция $\mathcal{L}(a, x)$, характеризующая величину ошибки алгоритма a на объекте x. Если $\mathcal{L}(a, x) = 0$, то ответ $a(x)$ называется корректным.

Функционал качества алгоритма a на выборке X^ℓ:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(a, x_i) \qquad (1.2)$$

Функционал Q называют также функционалом средних потерь или эмпирическим риском, так как он вычисляется по эмпирическим данным $(x_i, y_i)_{i=1}^\ell$.

Функция потерь, принимающая только значения 0 и 1, называется бинарной. В этом случае $\mathcal{L}(a,x)=1$ означает, что алгоритм a допускает ошибку на объекте x, а функционал Q называется частотой ошибок алгоритма а на выборке X^ℓ.

Наиболее часто используются следующие функции потерь, при $Y \subseteq \mathbb{R}$:

$\mathcal{L}(a,x)=[a(x)\neq y^*(x)]$ – индикатор ошибки, обычно применяется в задачах классификации; квадратные скобки переводят логическое значение в число по правилу $[\text{ложь}]=0$, $[\text{истина}]=1$.

$\mathcal{L}(a,x)=|a(x)-y^*(x)|$ – отклонение от правильного ответа; функционал Q называется средней ошибкой алгоритма а на выборке X^ℓ;

$\mathcal{L}(a,x)=(a(x)-y^*(x))^2$ – квадратичная функция потерь; функционал Q называется средней квадратичной ошибкой алгоритма а на выборке X^ℓ; обычно применяется в задачах регрессии.

Классический метод обучения, называемый минимизацией эмпирического риска (empirical risk minimization, ERM), заключается в том, чтобы найти в заданной модели A алгоритм a, доставляющий минимальное значение функционалу качества Q на заданной обучающей выборке X^ℓ:

$$\mu\left(X^\ell\right)=\arg\min_{a\in A} Q(a,X^\ell) \qquad (1.3)$$

Пример 1.3. В задаче восстановления регрессии ($Y=\mathbb{R}$) с n числовыми признаками $f_j: X\to\mathbb{R}, j=1,\ldots,n$, и квадратичной функцией потерь метод минимизации эмпирического риска есть ничто иное, как метод наименьших квадратов:

$\mu\left(X^\ell\right)=\arg\min_{\theta}\sum_{i=1}^{\ell}(g(x_i,\theta)-y_i)^2$.

1.1.5. Вероятностная постановка задачи обучения

В задачах обучения по прецедентам элементы множества X – это не реальные объекты, а лишь доступные данные о них. Данные могут быть неточными, поскольку измерения значений признаков fj(x) и целевой зависимости $y^*(x)$ обычно выполняются с погрешностями. Данные могут

быть неполными, поскольку измеряются не все мыслимые признаки, а лишь физически доступные для измерения.

В результате одному и тому же описанию x могут соответствовать различные объекты и различные ответы. В таком случае $y^*(x)$, строго говоря, не является функцией.

Устранить эту некорректность позволяет вероятностная постановка задачи. Вместо существования неизвестной целевой зависимости $y^*(x)$ предположим существование неизвестного вероятностного распределения на множестве $X \times Y$ с плотностью $p(x,y)$, из которого случайно и независимо выбираются ℓ наблюдений $X^\ell = (x_i, y_i)_{i=1}^{\ell}$. Такие выборки называются простыми или случайными одинаково распределёнными (independent identically distributed, i.i.d.).

Вероятностная постановка задачи считается более общей, так как функциональную зависимость $y^*(x)$ можно представить в виде вероятностного распределения $p(x,y) = p(x)p(y|x)$, положив $p(y|x) = \delta(y - y^*(x))$, где $\delta(z)$ – дельта-функция.

Принцип максимума правдоподобия. При вероятностной постановке задачи вместо модели алгоритмов $g(x,\theta)$, аппроксимирующей неизвестную зависимость $y^*(x)$, задаётся модель совместной плотности распределения объектов и ответов $\varphi(x,y,\theta)$, аппроксимирующая неизвестную плотность $p(x,y)$. Затем определяется значение параметра θ, при котором выборка данных X^ℓ максимально правдоподобна, то есть наилучшим образом согласуется с моделью плотности.

Если наблюдения в выборке X^ℓ независимы, то совместная плотность распределения всех наблюдений равна произведению плотностей $p(x,y)$ в каждом наблюдении: $p(X^\ell) = p\big((x_1,y_1),\dots,(x_\ell,y_\ell)\big) = p(x_1,y_1)\cdots p(x_\ell,y_\ell)$. Подставляя вместо $p(x,y)$ м о д е л ь плотности $\varphi(x,y,\theta)$, получаем функцию правдоподобия (likelihood):

$$L(\theta, X^{\ell}) = \prod_{i=1}^{\ell} \varphi(x_i, y_i, \theta).$$

Чем выше значение правдоподобия, тем лучше выборка согласуется с моделью. Значит, нужно искать значение параметра θ, при котором значение $L(\theta, X^{\ell})$ максимально. В математической статистике это называется принципом максимума правдоподобия.

После того, как значение параметра θ найдено, искомый алгоритм $a_\theta(x)$ строится по плотности $\varphi(x, y, \theta)$ несложно.

Связь максимизации правдоподобия с минимизацией эмпирического риска. Вместо максимизации L удобнее минимизировать функционал $-\ln L$, поскольку он аддитивен (имеет вид суммы) по объектам выборки:

$$-\ln L(\theta, X^{\ell}) = -\sum_{i=1}^{\ell} \ln \varphi(x_i, y_i, \theta) \to \min_{\theta} \qquad (1.4)$$

Этот функционал совпадает с функционалом эмпирического риска (1.2), если определить вероятностную функцию потерь $\mathcal{L}(a_\theta, x) = -\ell \ln \varphi(x, y, \theta)$. Такое определение потери вполне естественно – чем хуже пара (xi, yi) согласуется с моделью φ, тем меньше значение плотности $\varphi(x_i, y_i, \theta)$ и выше величина потери $\mathcal{L}(a_\theta, x)$.

Верно и обратное – для многих функций потерь возможно подобрать модель плотности $\varphi(x, y, \theta)$ таким образом, чтобы минимизация эмпирического риска была эквивалентна максимизации правдоподобия.

Пример 1.4. Пусть задана модель $g(x, \theta)$. Примем дополнительное вероятностное предположение, что ошибки $\varepsilon(x, \theta) = g(x, \theta) - y^*(x)$ имеют нормальное распределение $\mathcal{N}(\varepsilon, \theta, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$ с нулевым средним и дисперсией σ^2. Тогда модель плотности имеет вид:

$$\varphi(x, y, \theta) = p(x)\varphi(y|x, \theta) = p(x)\mathcal{N}(g(x, \theta) - y^*(x); \theta, \sigma^2).$$

Отсюда следует, что вероятностная функция потерь совпадает с квадратичной с точностью до констант C_0 и C_1, не зависящих от параметра θ:

$$-\ln \varphi(x, y, \theta) = -\ln p(x)\, \mathcal{N}(g(x, \theta) - y^*(x); \theta, \sigma^2)$$
$$= C_0 + C_1\big(g(x, \theta) - y^*(x)\big)^2$$

Таким образом, существуют два родственных подхода к формализации задачи обучения: первый основан на введении функции потерь, второй – на введении вероятностной модели порождения данных. Оба в итоге приводят к схожим (иногда даже в точности одинаковым) оптимизационным задачам. Таким образом, обучение – это оптимизация.

1.1.6. Проблема переобучения и понятие обобщающей способности

Минимизацию эмпирического риска следует применять с известной долей осторожности. Если минимум функционала $Q(a, X^\ell)$ достигается на алгоритме a, то это ещё не гарантирует, что a будет хорошо приближать целевую зависимость на произвольной контрольной выборке $X^k = (x'_i, y'_i)_{i=1}^k$.

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте переобучения (overtraining) или переподгонки (overfitting). При решении практических задач с этим явлением приходится сталкиваться очень часто.

Легко представить себе метод, который минимизирует эмпирический риск до нуля, но при этом абсолютно не способен обучаться. Получив обучающую выборку X^ℓ, он запоминает её и строит алгоритм, который сравнивает предъявляемый объект x с обучающими объектами x_i из X^ℓ. В случае совпадения $x = x_i$ алгоритм выдаёт правильный ответ y_i. Иначе выдаётся произвольный ответ. Эмпирический риск принимает наименьшее возможное значение, равное нулю. Однако этот алгоритм не способен восстановить зависимость вне материала обучения. Отсюда вывод: для успешного обучения необходимо не только запоминать, но и обобщать.

Обобщающая способность (generalization ability) метода μ характеризуется величиной $Q(\mu(X^\ell), X^k)$ при условии, что выборки X^ℓ и X^k являются представительными. Для формализации понятия «представительная выборка» обычно принимается стандартное предположение, что выборки X^ℓ и X^k – простые, полученные из одного и того же неизвестного вероятностного распределения на множестве X.

Метод обучения μ называется состоятельным, если при заданных достаточно малых значениях ε и η справедливо неравенство

$$\mathrm{P}_{X^\ell, X^k}\{Q(\mu(X^\ell), X^k) > \varepsilon\} < \eta \qquad (1.5)$$

Параметр ε называется точностью, параметр $(1-\eta)$ – надёжностью.

Допустима также эквивалентная формулировка: для любых простых выборок X^ℓ и X^k оценка $Q(\mu(X^\ell), X^k) \le \varepsilon$ справедлива с вероятностью не менее $1-\eta$.

Получение оценок вида (1.5) является фундаментальной проблемой статистической теории обучения. Первые оценки были получены в конце 60-х годов В.Н. Вапником и А.Я. Червоненкисом. В настоящее время статистическая теория развивается очень активно, однако для многих практически интересных случаев оценки обобщающей способности либо неизвестны, либо сильно завышены.

Эмпирические оценки обобщающей способности применяются в тех случаях, когда не удаётся воспользоваться теоретическими.

Пусть дана выборка $X^L = (x_i, y_i)_{i=1}^L$. Разобьём её N различными способами на две непересекающиеся подвыборки – обучающую X_n^ℓ длины ℓ и контрольную X_n^k длины $k = L - \ell$. Для каждого разбиения $n = 1, \ldots, N$ построим алгоритм $a_n = \mu(X_n^\ell)$ и вычислим значение $Q_n = Q(a_n, X_n^k)$. Среднее арифметическое значений Q_n по всем разбиениям называется оценкой скользящего контроля (cross-validation, CV):

$$CV(\mu, X^L) = \frac{1}{N}\sum_{n=1}^{N} Q(\mu(X_n^\ell), X_n^k) \quad (1.6)$$

Возможны различные варианты скользящего контроля, отличающиеся способами разбиения выборки X^L. В простейшем варианте разбиения генерируются случайным образом, число N берётся в диапазоне от 20 до 100. Стандартом «де факто» считается методика $t \times q$-кратного скользящего контроля ($t \times q$-fold cross-validation), когда выборка случайным образом разбивается на q блоков равной (или почти равной) длины, каждый блок по очереди становится контрольной выборкой, а объединение всех остальных блоков – обучающей. Выборка X^L по-разному t раз разбивается на q блоков. Итого получается $N = tq$ разбиений. Данная методика даёт более точные оценки за счёт того, что все объекты ровно по t раз встречаются в контроле.

Недостатками скользящего контроля являются: вычислительная неэффективность, высокая дисперсия, неполное использование имеющихся данных для обучения из-за сокращения длины обучающей выборки с L до ℓ.

1.2. Прикладные задачи

Прикладные задачи классификации, регрессии и прогнозирования встречаются в самых разных областях человеческой деятельности, и их число постоянно растёт.

1.2.1. Задачи классификации

Пример 1.5. В задачах медицинской диагностики в роли объектов выступают пациенты. Признаки характеризуют результаты обследований, симптомы заболевания и применявшиеся методы лечения. Примеры бинарных признаков – пол, наличие головной боли, слабости, тошноты, и т. д. Порядковый признак – тяжесть состояния (удовлетворительное, средней

тяжести, тяжёлое, крайне тяжёлое). Количественные признаки – возраст, пульс, артериальное давление, содержание гемоглобина в крови, доза препарата, и т. д. Признаковое описание пациента является, по сути дела, формализованной историей болезни. Накопив достаточное количество прецедентов, можно решать различные задачи: классифицировать вид заболевания (дифференциальная диагностика); определять наиболее целесообразный способ лечения; предсказывать длительность и исход заболевания; оценивать риск осложнений; находить синдромы – наиболее характерные для данного заболевания совокупности симптомов. Ценность такого рода систем в том, что они способны мгновенно анализировать и обобщать огромное количество прецедентов – возможность, недоступная человеку.

Пример 1.6. Задача оценивания заёмщиков решается банками при выдаче кредитов. Потребность в автоматизации процедуры выдачи кредитов впервые возникла в период бума кредитных карт 60-70-х годов в США и других развитых странах.

Объектами в данном случае являются заёмщики – физические или юридические лица, претендующие на получение кредита. В случае физических лиц признаковое описание состоит из анкеты, которую заполняет сам заёмщик, и, возможно, дополнительной информации, которую банк собирает о нём из собственных источников.

Примеры бинарных признаков: пол, наличие телефона. Номинальные признаки – место проживания, профессия, работодатель. Порядковые признаки – образование, занимаемая должность. Количественные признаки – возраст, стаж работы, доход семьи, размер задолженностей в других банках, сумма кредита. Обучающая выборка составляется из заёмщиков с известной кредитной историей. В простейшем случае принятие решений сводится к классификации заёмщиков на два класса: «хороших» и «плохих». Кредиты выдаются только заёмщикам первого класса. В более сложном случае оценивается суммарное число баллов (score) заёмщика, набранных по

совокупности информативных признаков. Чем выше оценка, тем более надёжным считается заёмщик. Отсюда и название – кредитный скоринг (credit scoring). На стадии обучения производится синтез и отбор информативных признаков и определяется, сколько баллов назначать за каждый признак, чтобы риск принимаемых решений был минимален. Следующая задача – решить, на каких условиях выдавать кредит: определить процентную ставку, срок погашения, и прочие параметры кредитного договора. Эта задача также сводится к обучению по прецедентам.

Пример 1.7. Задача предсказания ухода клиентов (churn prediction) возникает у крупных и средних компаний, работающих с большим количеством клиентов, как правило, с физическими лицами. Особенно актуальна эта задача для современных телекоммуникационных компаний. Когда рынок приходит в состояние, близкое к насыщению, основные усилия компаний направляются не на привлечение новых клиентов, а на удержание старых. Для этого необходимо как можно точнее выделить сегмент клиентов, склонных к уходу в ближайшее время. Классификация производится на основе информации, хранящейся у компании: клиентских анкет, данных о частоте пользования услугами компании, составе услуг, тарифных планах, регулярности платежей, и т. д. Наиболее информативны данные о том, что именно изменилось в поведении клиента за последнее время. Поэтому объектами, строго говоря, являются не сами клиенты, а пары «клиент x_i в момент времени t_i». Требуется предсказать, уйдёт ли клиент к моменту времени $t_i + \Delta t$. Обучающая выборка формируется из клиентов, о которых доподлинно известно, в какой момент они ушли.

1.2.2. Задачи восстановления регрессии

Пример 1.8. Термин «регрессия» был введён в 1886 году антропологом Фрэнсисом Гальтоном при изучении статистических закономерностей наследственности роста. Повседневный опыт подсказывает, что в среднем рост взрослых детей тем больше, чем выше их родители. Однако Гальтон

обнаружил, что сыновья очень высоких отцов часто имеют не столь высокий рост. Он собрал выборку данных по 928 парам отец-сын. Количественно зависимость неплохо описывалась линейной функцией $y = \frac{2}{3}x$, где x – отклонение роста отца от среднего, y – отклонение роста сына от среднего. Гальтон назвал это явление «регрессией к посредственности», то есть к среднему значению в популяции. Термин регрессия – движение назад – намекал также на нестандартный для того времени ход исследования: сначала были собраны данные, затем по ним угадана модель зависимости, тогда как традиционно поступали наоборот: данные использовались лишь для проверки теоретических моделей.

Это был один из первых случаев моделирования, основанного исключительно на данных. Позже термин, возникший в частной прикладной задаче, закрепился за широким классом методов восстановления зависимостей.

Огромное количество регрессионных задач возникает в физических экспериментах, в промышленном производстве, в экономике.

Пример 1.9. Задача прогнозирования потребительского спроса решается современными супермаркетами и торговыми розничными сетями. Для эффективного управления торговой сетью необходимо прогнозировать объёмы продаж для каждого товара на заданное число дней вперёд. На основе этих прогнозов осуществляется планирование закупок, управление ассортиментом, формирование ценовой политики, планирование промоакций (рекламных кампаний). Специфика задачи в том, что количество товаров может исчисляться десятками или даже сотнями тысяч. Прогнозирование и принятие решений по каждому товару «вручную» просто немыслимо.

Исходными данными для прогнозирования являются временны́е ряды цен и объёмов продаж по товарам и по отдельным магазинам. Современные технологии позволяют получать эти данные от кассовых аппаратов и накапливать в едином хранилище данных. Для увеличения точности прогнозов необходимо учитывать различные внешние факторы, влияющие на

спрос: рекламные кампании, социально-демографические условия, активность конкурентов, праздники, и даже погодные условия. В зависимости от целей анализа в роли объектов выступают либо товары, либо магазины, либо пары «магазин–товар». Ещё одна особенность задачи – несимметричность функции потерь. Если прогноз делается с целью планирования закупок, то потери от заниженного прогноза, как правило, существенно выше, чем от завышенного.

Пример 1.10. Задача предсказания рейтингов решается интернет-магазинами, особенно книжными, видео и аудио. Приобретая товар, клиент имеет возможность выставить ему рейтинг, например, целое число от 1 до 5. Система использует информацию о всех выставленных рейтингах для персонализации предложений. Когда клиент видит на сайте страницу с описанием товара, ему показывается также ранжированный список схожих товаров, пользующихся популярностью у схожих клиентов.

Основная задача – прогнозировать рейтинги товаров, которые данный клиент ещё не приобрёл. Роль матрицы объектов–признаков играет матрица клиентов–товаров, заполненная значениями рейтингов. Как правило, она сильно разрежена и может иметь более 99% пустых ячеек. Фиксированного целевого признака в этой задаче нет. Алгоритм должен предсказывать рейтинги для любых незаполненных ячеек матрицы. Данный тип задач выделяют особо и называют задачами коллаборативной фильтрации (collaborative filtering).

О трудности и актуальности этой задачи говорит следующий факт. В октябре 2006 года крупнейшая американская компания Netflix, занимающаяся видеопрокатом через Internet, объявила международный конкурс с призом в 1 миллион долларов тому, кто сможет на 10% улучшить точность прогнозирования рейтингов, по сравнению с системой Netflix Cinematch (см. http://www.netflixprize.com). Примечательно, что прогнозы самой Cinematch были лишь на те же 10% точнее элементарных прогнозов по средним рейтингам фильмов. Компания крайне заинтересована в увеличении точности

прогнозов, поскольку около 70% заказов поступают через рекомендующую систему. Конкурс успешно завершился только через два с половиной года.

1.2.3. Задачи ранжирования

Задачи ранжирования (ranking) возникают в области информационного поиска. Результатом поиска по запросу может оказаться настолько длинный список ответов, что пользователь физически не сможет его просмотреть. Поэтому ответы упорядочивают по убыванию релевантности – степени их соответствия запросу. Критерий упорядочения в явном виде неизвестен, хотя человек легко отличает более релевантные ответы от менее релевантных или совсем нерелевантных. Обычно для разметки выборки пар «запрос, ответ» привлекают команду экспертов (асессоров), чтобы учесть мнения различных людей, которые зачастую противоречат друг другу. Затем решают задачу обучения ранжированию (learning to rank).

Пример 1.11. Задача ранжирования текстовых документов, найденных в Интернете по запросу пользователя, решается всеми современными поисковыми машинами. Объектами являются пары «запрос, документ», ответами – оценки релевантности, сделанные асессорами. В зависимости от методологии формирования обучающей выборки оценки асессоров могут быть бинарными (релевантен, не релевантен) или порядковыми (релевантность в баллах). Признаками являются числовые характеристики, вычисляемые по паре «запрос, документ». Текстовые признаки основаны на подсчёте числа вхождений слов запроса в документы. Возможны многочисленные варианты: с учётом синонимов или без, с учётом числа вхождений или без, во всём документе или только в заголовках, и т. д. Ссылочные признаки основаны на подсчёте числа документов, ссылающихся на данный. Кликовые признаки основаны на подсчёте числа обращений к данному документу.

Пример 1.12. Задачу предсказания рейтингов из примера 1.10 на практике лучше ставить как задачу ранжирования. Пользователю выдаётся

список рекомендаций, поэтому важно обеспечить высокую релевантность относительно небольшого числа товаров, попадающих в вершину списка. Среднеквадратичная ошибка предсказания рейтингов, которую предлагалось минимизировать в условии конкурса Netflix, в данном случае не является адекватной мерой качества. Заметим, что в этой задаче в роли асессоров выступают все пользователи рекомендательного сервиса. Покупая товар или выставляя оценку, пользователь пополняет обучающую выборку и тем самым способствует улучшению качества сервиса.

1.2.4. Задачи кластеризации

Задачи кластеризации (clustering) отличаются от классификации (classification) тем, что в них не задаются ответы $y_i = y^*(x_i)$. Известны только сами объекты x_i, и требуется разбить выборку на подмножества (кластеры) так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Для этого необходимо задавать функцию расстояния на множестве объектов. Число кластеров также может задаваться, но чаще требуется определить и его.

Пример 1.13. Основным инструментом социологических и маркетинговых исследований является проведение опросов. Чтобы результаты опроса были объективны, необходимо обеспечить представительность выборки респондентов. С другой стороны, требуется минимизировать стоимость проведения опроса. Поэтому при планировании опросов возникает вспомогательная задача: отобрать как можно меньше респондентов, чтобы они образовывали репрезентативную выборку, то есть представляли весь спектр общественного мнения. Один из способов это сделать состоит в следующем. Сначала составляются признаковые описания достаточно большого числа точек опроса (это могут быть города, районы, магазины, и т. д.). Для этого используются недорогие способы сбора информации – пробные опросы или фиксация некоторых характеристик самих точек. Затем решается задача кластеризации, и из каждого кластера отбирается

по одной представительной точке. Только в отобранном множестве точек производится основной, наиболее ресурсоёмкий, опрос.

Задачи кластеризации, в которых часть объектов (как правило, незначительная) размечена по классам, называются задачами с частичным обучением (semisupervised learning). Считается, что они не сводятся непосредственно к классификации или кластеризации, и для их решения нужны особые методы.

Пример 1.14. Задача рубрикации текстов возникает при работе с большими коллекциями текстовых документов. Допустим, имеется некоторый иерархический рубрикатор, разработанный экспертами для данной предметной области (например, для спортивных новостей), или для всех областей (например, универсальный десятичный классификатор УДК). Имеется множество документов, классифицированных по рубрикам вручную. Требуется классифицировать по тем же рубрикам второе множество документов, которое может быть существенно больше первого. Для решения данной задачи используется функция расстояния, сравнивающая тексты по составу терминов. Терминами, как правило, являются специальные понятия предметной области, собственные имена, географические названия, и т. д. Документы считаются схожими, если множества их терминов существенно пересекаются.

1.2.5. Задачи поиска ассоциаций

Задача поиска ассоциативных правил (association rule induction) вынесена в отдельный класс и относится к задачам обучения без учителя, хотя имеет много общего с задачей классификации.

Пример 1.15. Задача анализа рыночных корзин (market basket analysis) состоит в том, чтобы по данным о покупках товаров в супермаркете (буквально, по чекам) определить, какие товары часто совместно покупаются. Эта информация может быть полезной для оптимизации размещения товаров на полках, планирования рекламных кампаний (промо-акций), управления

ассортиментом и ценами. В данной задаче объекты соответствуют чекам, признаки являются бинарными и соответствуют товарам.

Единичное значение признака $f_j(x_i) = 1$ означает, что в i-м чеке зафиксирована покупка j-го товара. Задача состоит в том, чтобы выявить все наборы товаров, которые часто покупают вместе. Например, «если куплен хлеб, то с вероятностью 60% будет куплено и молоко». Во многие учебники по бизнес-аналитике вошёл пример, когда система поиска ассоциативных правил обнаружила неочевидную закономерность: вечером перед выходными днями возрастают совместные продажи памперсов и пива.

Разместив дорогие сорта пива рядом с памперсами, менеджеры смогли увеличить продажи в масштабах всей розничной сети, что окупило внедрение системы анализа данных. Позже маркетологи и социологи предложили разумное объяснение данному явлению, однако обнаружено оно было именно путём анализа данных.

Пример 1.16. Задача выделения терминов (term extraction) из текстов, решаемая перед задачей рубрикации (см. пример 1.14), может быть сведена к поиску ассоциаций. Терминами считаются отдельные слова или устойчивые словосочетания, которые часто встречаются в небольшом подмножестве документов, и редко – во всех остальных. Множество часто совместно встречающихся терминов образует тему, скорее всего, соответствующую некоторой рубрике.

1.2.6. Методология тестирования обучаемых алгоритмов

Пока ещё не создан универсальный метод обучения по прецедентам, способный решать любые практические задачи одинаково хорошо. Каждый метод имеет свои преимущества, недостатки и границы применимости. На практике приходится проводить численные эксперименты, чтобы понять, какой метод из имеющегося арсенала лучше подходит для конкретной задачи. Обычно для этого методы сравниваются по скользящему контролю (1.6).

Существует два типа экспериментальных исследований, отличающихся целями и методикой проведения. Эксперименты на модельных данных. Их цель – выявление границ применимости метода обучения; построение примеров удачной и неудачной его работы; понимание, на что влияют параметры метода обучения. Модельные эксперименты часто используются на стадии отладки метода. Модельные выборки сначала генерируются в двумерном пространстве, чтобы работу метода можно было наглядно представить на плоских графиках. Затем исследуется работа метода на многомерных данных, при различном числе признаков. Генерация данных выполняется либо с помощью датчика случайных чисел по заданным вероятностным распределениям, либо детерминированным образом. Часто генерируется не одна модельная задача, а целая серия, параметризованная таким образом, чтобы среди задач оказались как заведомо «лёгкие», так и заведомо «трудные»; при такой организации эксперимента точнее выявляются границы применимости метода.

Эксперименты на реальных данных. Их цель – либо решение конкретной прикладной задачи, либо выявление «слабых мест» и границ применимости конкретного метода. В первом случае фиксируется задача, и к ней применяются многие методы, или, возможно, один и тот же метод при различных значениях параметров. Во втором случае фиксируется метод, и с его помощью решается большое число задач (обычно несколько десятков). Специально для проведения таких экспериментов создаются общедоступные репозитории реальных данных. Наиболее известный – репозиторий UCI (университета Ирвина, Калифорния), доступный по адресу http://archive.ics.uci.edu/ml. Он содержит около двух сотен задач, в основном классификации, из самых разных предметных областей.

Полигон алгоритмов классификации. В научных статьях по машинному обучению принято приводить результаты тестирования предложенного нового метода обучения в сравнении с другими методами на представительном наборе задач. Сравнение должно производиться в равных

условиях по одинаковой методике; если это скользящий контроль, то при одном и том же множестве разбиений. Несмотря на значительную стандартизацию таких экспериментов, результаты тестирования одних и тех же методов на одних и тех же задачах, полученные разными авторами, всё же могут существенно различаться. Проблема в том, что используются различные реализации методов обучения и методик тестирования, а проведённый кем-то ранее эксперимент практически невозможно воспроизвести во всех деталях. Для решения этой проблемы разработан Полигон алгоритмов классификации, доступный по адресу http://poligon.MachineLearning.ru. В этой системе реализована унифицированная расширенная методика тестирования и централизованное хранилище задач.

Реализация алгоритмов классификации, наоборот, децентрализована. Любой пользователь Интернет может объявить свой компьютер вычислительным сервером Полигона, реализующим один или несколько методов классификации. Все результаты тестирования сохраняются как готовые отчёты в базе данных системы и могут быть в любой момент выданы по запросу без проведения трудоёмких вычислений заново.

Конкурсы по решению задач анализа данных. В последние годы компании, заинтересованные в решении прикладных задач анализа данных, всё чаще стали обращаться к такой форме привлечения научного сообщества, как открытые конкурсы с денежными премиями для победителя. В каждом таком конкурсе публикуется обучающая выборка с известными ответами, тестовая выборка, ответы на которой известны только организатору конкурса, и критерий, по которому алгоритмы претендентов сравниваются на данных тестовой выборки. Информацию о текущих конкурсах можно найти на сайтах http://www.kaggle.com, http://tunedit.org. Существуют также сайты, на которых можно тестировать различные алгоритмы на различных наборах данных: http://poligon.MachineLearning.ru, http://mlcomp.org.

1.2.7. Приёмы генерации модельных данных

Генерация выборок данных может юыть полезна при исследовании алгоритмов. Поиск наборов данных с заданными характеристиками и требуемого объема не всегда возможны, поэтому разработаны различные методики получения «искусственных» данных.

Моделирование случайных данных. Следующие утверждения позволяют генерировать случайные выборки с заданными распределениями. Будем предполагать, что имеется стандартный способ получать равномерно распределённые на отрезке [0, 1] случайные величины.

Утв. 1. Если случайная величина r равномерно распределена на [0, 1], то случайная величина $\xi = [r < p]$ принимает значение 1 с вероятностью p и значение 0 с вероятностью $1 - p$.

Утв. 2. Если случайная величина r равномерно распределена на [0, 1], и задана возрастающая последовательность $F_0 = 0$, F_1, ..., F_{k-1}, $F_k = 1$, то дискретная случайная величина ξ, определяемая условием $F_{\xi-1} \le r \le F_\xi$, принимает значения $j = 1, \ldots, k$ с вероятностями $p_j = F_j - F_{j-1}$.

Утв. 3. Если случайная величина r равномерно распределена на [0,1], и задана возрастающая на $\mathbb{R}$ функция $F(x)$, $0 \le F(x) \le 1$, то случайная величина $\xi = F^{-1}(r)$ имеет непрерывную функцию распределения $F(x)$.

Утв. 4. Если r_1, r_2 – две независимые случайные величины, равномерно распределённые на [0,1], то преобразование Бокса-Мюллера

$\xi_1 = \sqrt{-2 \ln r_1} \sin 2\pi r_2$;

$\xi_2 = \sqrt{-2 \ln r_1} \cos 2\pi r_2$;

даёт две независимые нормальные случайные величины с нулевым матожиданием и единичной дисперсией: $\xi_1, \xi_2 \in \mathcal{N}(0,1)$.

РАЗДЕЛ 2. МЕТРИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, такие сложные объекты, как фотографии лиц, подписи, временные ряды или первичные структуры белков естественнее сравнивать непосредственно с друг с другом путём некоторого «наложения с выравниванием», чем изобретать какие-то признаки и сравнивать признаковые описания. Если мера сходства объектов введена достаточно удачно, то, как правило, оказывается, что схожим объектам очень часто соответствуют схожие ответы. В задачах классификации это означает, что классы образуют компактно локализованные подмножества. Это предположение принято называть гипотезой компактности.

Для формализации понятия «сходства» вводится функция расстояния в пространстве объектов X. Методы обучения, основанные на анализе сходства объектов, будем называть метрическими, даже если функция расстояния не удовлетворяет всем аксиомам метрики (в частности, аксиоме треугольника). В англоязычной литературе употребляются термины similarity-based learning или distance-based learning.

2.1. Метод ближайших соседей

Пусть на множестве объектов X задана функция расстояния $\rho: X \times X \to [0, \infty)$.

Существует целевая зависимость $y^*: X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^{\ell}$, $y_i = y^*(x_i)$. Множество классов Y конечно. Требуется построить алгоритм классификации $a: X \to Y$, аппроксимирующий целевую зависимость $y^*(x)$ на всём множестве X.

2.1.1. Обобщенный метрический классификатор

Для произвольного объекта $u \in X$ расположим элементы обучающей выборки $x_1, \ldots, x_\ell$ в порядке возрастания расстояний до u:

$$\rho\left(u, x_u^{(1)}\right) \leq \rho\left(u, x_u^{(2)}\right) \leq \cdots \leq \rho(u, x_u^{(\ell)}),$$

где через $x_u^{(i)}$ обозначается i-й сосед объекта u. Соответственно, ответ на i-м соседе объекта u есть $y_u^{(i)} = y^*(x_u^{(i)})$. Таким образом, любой объект $u \in X$ порождает свою перенумерацию выборки.

Метрический алгоритм классификации с обучающей выборкой X^ℓ относит объект u к тому классу $y \in Y$, для которого суммарный вес ближайших обучающих объектов $\Gamma_y(u, X^\ell)$ максимален:

$$\begin{gathered} a\left(u; X^\ell\right) = \arg\max_{y \in Y} \Gamma_y(u, X^\ell); \\ \Gamma_y\left(u, X^\ell\right) = \sum_{i=1}^{\ell} \left[y_u^{(i)} = y\right] \omega(i, u); \end{gathered} \tag{2.1}$$

где весовая функция $\omega(i, u)$ оценивает степень важности i-го соседа для классификации объекта u. Функция $\Gamma_y\left(u, X^\ell\right)$ называется оценкой близости объекта u к классу y.

Метрический классификатор определён с точностью до весовой функции $\omega(i, u)$. Обычно она выбирается неотрицательной, не возрастающей по i. Это соответствует гипотезе компактности, согласно которой чем ближе объекты u и $x_u^{(i)}$, тем выше шансы, что они принадлежат одному классу.

Обучающая выборка X^ℓ играет роль параметра алгоритма a. Настройка сводится к запоминанию выборки, и, возможно, оптимизации каких-то параметров весовой функции, однако сами объекты не подвергаются обработке и сохраняются «как есть». Алгоритм $a\left(u; X^\ell\right)$ строит локальную аппроксимацию выборки X^ℓ, причём вы-

числения откладываются до момента, пока не станет известен объект u. По этой причине метрические алгоритмы относятся к методам ленивого обучения (lazy learning), в отличие от усердного обучения (eager learning), когда на этапе обучения строится функция, аппроксимирующая выборку.

Метрические алгоритмы классификации относятся также к методам рассуждения по прецедентам (case-based reasoning, CBR). Здесь действительно можно говорить о «рассуждении», так как на вопрос «почему объект u был отнесён к классу y?» алгоритм может дать понятное экспертам объяснение: «потому, что имеются схожие с ним прецеденты класса y», и предъявить список этих прецедентов.

Выбирая весовую функцию $\omega(i,u)$, можно получать различные метрические классификаторы, которые подробно рассматриваются ниже.

2.1.2. Метод ближайших соседей

Алгоритм ближайшего соседа (nearest neighbor, NN) относит классифицируемый объект $u \in X^\ell$ к тому классу, которому принадлежит ближайший обучающий объект:

$\omega(i,u) = [i = 1]; a\left(u; X^\ell\right) = y_u^{(1)}$.

Этот алгоритм является, по всей видимости, самым простым классификатором. Обучение NN сводится к запоминанию выборки X^ℓ. Единственное достоинство этого алгоритма – простота реализации. Недостатков гораздо больше:

1. Неустойчивость к погрешностям. Если среди обучающих объектов есть выброс – объект, находящийся в окружении объектов чужого класса, то не только он сам будет классифицирован неверно, но и те окружающие его объекты, для которых он окажется ближайшим.

2. Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ.

3. В результате – низкое качество классификации.

Алгоритм k ближайших соседей (k nearest neighbors, kNN). Чтобы сгладить влияние выбросов, будем относить объект u к тому классу, элементов которого окажется больше среди k ближайших соседей $x_u^{(i)}$, $i = 1, \ldots, k$:

$$\omega(i,u) = [i \le k];\ a\big(u; X^\ell, k\big) = \arg\max_{y \in Y} \sum_{i=1}^{k} \left[y_u^{(i)} = y\right].$$

При $k = 1$ этот алгоритм совпадает с предыдущим, следовательно, неустойчив к шуму. При $k = \ell$, наоборот, он чрезмерно устойчив и вырождается в константу.

Таким образом, крайние значения k нежелательны. На практике оптимальное значение параметра k определяют по критерию скользящего контроля с исключением объектов по одному (leave-one-out, LOO). Для каждого объекта $x_i \in X^\ell$ проверяется, правильно ли он классифицируется по своим k ближайшим соседям.

$$\mathrm{LOO}\big(k, X^\ell\big) = \sum_{i=1}^{k} \left[\big(x_i; X^\ell \backslash \{x_i\}, k\big) \ne y_i\right] \to \min_k$$

Если классифицируемый объект x_i не исключать из обучающей выборки, то ближайшим соседом x_i всегда будет сам x_i, и минимальное (нулевое) значение функционала LOO(k) будет достигаться при $k = 1$.

Существует и альтернативный вариант метода kNN: в каждом классе выбирается k ближайших к u объектов, и объект u относится к тому классу, для которого среднее расстояние до k ближайших соседей минимально.

Алгоритм k взвешенных ближайших соседей. Недостаток kNN в том, что максимум может достигаться сразу на нескольких классах. В задачах с двумя классами этого можно избежать, если взять нечётное k. Более общая тактика, которая годится и для случая многих классов – ввести строго убывающую последовательность вещественных весов wi, задающих вклад i-го соседа в классификацию:

$$\omega(i,u) = [i \le k]\omega_i;\ a\big(u; X^\ell, k\big) = \arg\max_{y \in Y} \sum_{i=1}^{k} \left[y_u^{(i)} = y\right] \omega_i.$$

Выбор последовательности ω_i является эвристикой. Если взять линейно убывающие веса $\omega_i = \frac{k+1-i}{k}$, то неоднозначности также могут возникать, хотя и реже (пример: классов два; первый и четвёртый сосед голосуют за класс 1, второй и третий – за класс 2; суммы голосов совпадают). Неоднозначность устраняется окончательно, если взять нелинейно убывающую последовательность, скажем, геометрическую прогрессию: $\omega_i = q_i$, где знаменатель прогрессии $q \in (0,1)$ является параметром алгоритма. Его можно подбирать по критерию LOO, аналогично числу соседей k.

Недостатки простейших метрических алгоритмов типа kNN:

1. Приходится хранить обучающую выборку целиком. Это приводит к неэффективному расходу памяти и чрезмерному усложнению решающего правила. При наличии погрешностей (как в исходных данных, так и в модели сходства ρ) это может приводить к понижению точности классификации вблизи границы классов. Имеет смысл отбирать минимальное подмножество эталонных объектов, действительно необходимых для классификации.

2. Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Для задач с большими выборками или высокой частотой запросов это может оказаться накладно. Проблема решается с помощью эффективных алгоритмов поиска ближайших соседей, требующих в среднем $O(\ln \ell)$ операций.

3. В простейших случаях метрические алгоритмы имеют крайне бедный набор параметров, что исключает возможность настройки алгоритма по данным.

2.1.3. Метод парзеновского окна

Ещё один способ задать веса соседям – определить ω_i как функцию от расстояния $\rho(u, x_u^{(i)})$, а не от ранга соседа i. Введём функцию ядра $K(z)$,

невозрастающую на $[0,\infty)$. Положив $\omega(i,u) = K(\frac{1}{h}\rho(u,x_u^{(i)}))$ в общей формуле (2.1), получим алгоритм:

$$a(u;X^\ell,h) = \arg\max_{y\in Y}\sum_{i=1}^{k}\left[y_u^{(i)} = y\right]K(\frac{1}{h}\rho(u,x_u^{(i)})). \qquad (2.2)$$

Параметр h называется шириной окна и играет примерно ту же роль, что и число соседей k. «Окно» – это сферическая окрестность объекта u радиуса h, при попадании в которую обучающий объект x_i «голосует» за отнесение объекта u к классу y_i. Мы пришли к этому алгоритму чисто эвристическим путём, однако он имеет более строгое обоснование в байесовской теории классификации, и, фактически, совпадает с методом парзеновского окна.

Параметр h можно задавать априори или определять по скользящему контролю. Зависимость $\mathrm{LOO}(h)$, как правило, имеет характерный минимум, поскольку слишком узкие окна приводят к неустойчивой классификации; а слишком широкие – к вырождению алгоритма в константу.

Фиксация ширины окна h не подходит для тех задач, в которых обучающие объекты существенно неравномерно распределены по пространству X. В окрестности одних объектов может оказываться очень много соседей, а в окрестности других – ни одного. В этих случаях применяется окно переменной ширины. Возьмём финитное ядро – невозрастающую функцию $K(z)$, положительную на отрезке $[0,1]$, и равную нулю вне его. Определим h как наибольшее число, при котором ровно k ближайших соседей объекта u получают ненулевые веса: $h(u) = \rho\left(u,x_u^{(k+1)}\right)$. Тогда алгоритм принимает вид:

$$a(u;X^\ell,h) = \arg\max_{y\in Y}\sum_{i=1}^{k}\left[y_u^{(i)} = y\right]K(\frac{\rho(u,x_u^{(i)})}{\rho\left(u,x_u^{(k+1)}\right)}). \qquad (2.3)$$

При финитном ядре классификация объекта сводится к поиску его соседей, тогда как при не финитном ядре (например, гауссовском) требуется перебор всей обучающей выборки. Выбор ядра – отдельная тема.

2.1.4. Метод потенциальных функций

В методе парзеновского окна центр радиального ядра $K_h(u,x) = K(\frac{1}{h}\rho(u,x))$ помещается в классифицируемый объект u. В силу симметричности функции расстояния $\rho(u,x)$ возможен и другой, двойственный, взгляд на метрическую классификацию. Допустим, что ядро помещается в каждый обучающий объект x_i и «притягивает» объект u к классу y_i, если он попадает в его окрестность радиуса h_i:

Вход:
X^ℓ — обучающая выборка;
Выход:
Коэффициенты $\gamma_i,\ i = 1,\ldots,\ell$

Инициализация: $\gamma_i = 0;\ i = 1,\ldots,\ell;$
повторять
  выбрать объект $x_i \in X^\ell$;
  если $a(x_i) \neq y_i$ **то**
    $\gamma_i := \gamma_i + 1;$
пока число ошибок на выборке не окажется достаточно мало

Рисунок 2.1 – Алгоритм расчета коэффициентов в методе потенциальных функций

Данная идея лежит в основе метода потенциальных функций и имеет прямую физическую аналогию с электрическим потенциалом. При $Y = \{-1, +1\}$ обучающие объекты можно понимать как положительные и отрицательные электрические заряды; коэффициенты γ_i – как абсолютную величину этих зарядов; ядро K(z) – как зависимость потенциала от расстояния до заряда; а саму задачу классификации – как ответ на вопрос: какой знак имеет электростатический потенциал в заданной точке пространства u. Заметим, что в электростатике $K(z) = \frac{1}{z}$ либо $K(z) = \frac{1}{z+a}$, однако для наших целей совершенно не обязательно брать именно такое ядро.

Алгоритм (рисунок 2.1) имеет достаточно богатый набор из 2ℓ параметров γ_i, h_i. Простейший и исторически самый первый метод их настройки представлен в алгоритме на рисунке 2.1. Он настраивает только

веса γ_i, предполагая, что радиусы потенциалов h_i и ядро K выбраны заранее. Идея очень проста: если обучающий объект x_i классифицируется неверно, то потенциал класса y_i недостаточен в точке x_i, и вес γ_i увеличивается на единицу. Выбор объектов на шаге 3 лучше осуществлять не подряд, а в случайном порядке. Этот метод не так уж плох, как можно было бы подумать.

Достоинство алгоритма (рисунок 2.1) в том, что он очень эффективен, когда обучающие объекты поступают потоком, и хранить их в памяти нет возможности или необходимости. В те годы, когда метод потенциальных функций был придуман, хранение выборки действительно было большой проблемой. В настоящее время такой проблемы нет, и алгоритм представляет скорее исторический интерес.

Недостатков у данного алгоритма довольно много: он медленно сходится; результат обучения зависит от порядка предъявления объектов; слишком грубо (с шагом 1) настраиваются веса γ_i; центры потенциалов почему-то помещаются только в обучающие объекты; задача минимизации числа потенциалов (ненулевых γ_i) вообще не ставится; вообще не настраиваются параметры h_i. В результате данный алгоритм не может похвастаться высоким качеством классификации.

Существует более современный метод настройки линейных комбинаций потенциальных функций, основанный на EM-алгоритме. Он позволяет оптимизировать и ширину каждого потенциала, и положения центров, и даже количество потенциалов. Изменилось и название метода: теперь потенциальные функции предпочитают называть радиальными базисными функциями. Формально этот метод не подчиняется общей формуле (2.1), так как классифицируемый объект u сравнивается не с обучающими объектами, а с центрами потенциалов, которые в общем случае не совпадают ни с одним из обучающих объектов.

2.2. Отбор эталонных объектов

Обычно объекты обучения не являются равноценными. Среди них могут находиться типичные представители классов – эталоны. Если классифицируемый объект близок к эталону, то, скорее всего, он принадлежит тому же классу. Ещё одна категория объектов – неинформативные или периферийные. Они плотно окружены другими объектами того же класса. Если их удалить из выборки, это практически не отразится на качестве классификации. Наконец, в выборку может попасть некоторое количество шумовых выбросов – объектов, находящихся «в толще» чужого класса. Как правило, их удаление только улучшает качество классификации.

Эти соображения приводят к идее исключить из выборки шумовые и неинформативные объекты, оставив только минимальное достаточное количество эталонов.

Этим достигается несколько целей одновременно – повышается качество и устойчивость классификации, сокращается объём хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов. Кроме того, выделение небольшого числа эталонов в каждом классе позволяет понять структуру класса.

В первую очередь введём функцию отступа, которая позволит оценивать степень типичности объекта.

2.2.1. Понятие отступа объекта

Общая формула (2.1) позволяет ввести характеристику объектов, показывающую, насколько глубоко объект погружён в свой класс.

Отступом (margin) объекта $x_i \in X^\ell$ относительно алгоритма классификации, имеющего вид $a(u) = \arg\max\limits_{y \in Y \setminus y_i} \Gamma_y(u)$, называется величина:

$$M(x_i) = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus y_i} \Gamma_y(x_i).$$

Отступ показывает степень типичности объекта. Отступ отрицателен тогда и только тогда, когда алгоритм допускает ошибку на данном объекте.

В зависимости от значений отступа обучающие объекты условно делятся на пять типов, в порядке убывания отступа: эталонные, неинформативные, пограничные, ошибочные, шумовые (рисунок 2.2).

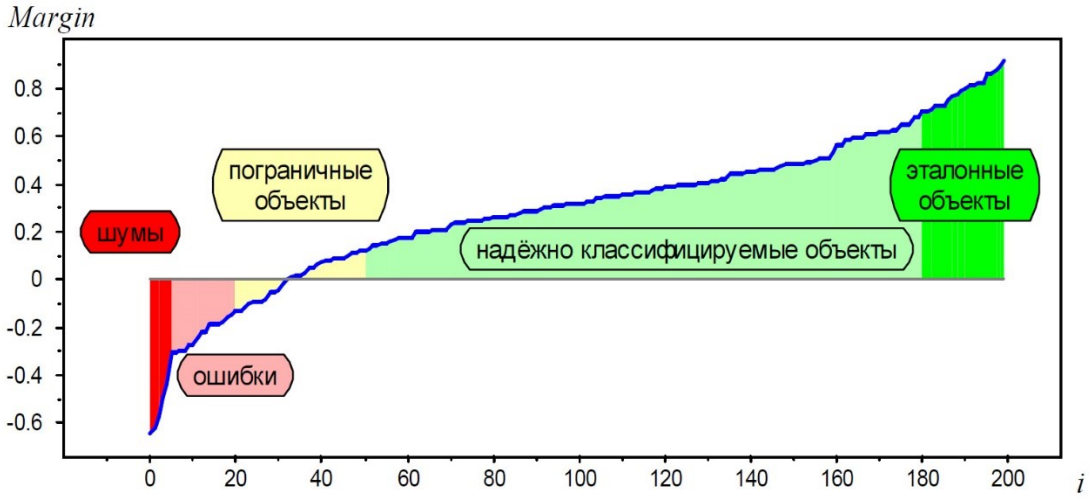


Рисунок 2.2 – Типы объектов обучающей выборки

1. Эталонные объекты имеют большой положительный отступ, плотно окружены объектами своего класса и являются наиболее типичными его представителями.

2. Неинформативные объекты также имеют положительный отступ. Изъятие этих объектов из выборки (при условии, что эталонные объекты остаются), не влияет на качество классификации. Фактически, они не добавляют к эталонам никакой новой информации. Наличие неинформативных объектов характерно для выборок избыточно большого объёма.

3. Пограничные объекты имеют отступ, близкий к нулю. Классификация таких объектов неустойчива в том смысле, что малые изменения метрики или состава обучающей выборки могут изменять их классификацию.

4. Ошибочные объекты имеют отрицательные отступы и классифицируются неверно. Возможной причиной может быть неадекватность алгоритмической модели (2.1), в частности, неудачная конструкция метрики ρ.

5. Шумовые объекты или выбросы – это небольшое число объектов с большими отрицательными отступами. Они плотно окружены объектами чужих классов и классифицируются неверно. Они могут возникать из-за грубых ошибок или пропусков в исходных данных, а также по причине отсутствия важной информации, которая позволила бы отнести эти объекты к правильному классу.

Приведённая типизация условна. Не существует чёткого различия между «соседними» типами объектов. В частности, легко строятся примеры выборок, содержащих такие пары близких объектов, что любой из них может быть объявлен эталонным, а второй – неинформативным.

Шумовые и неинформативные целесообразно удалять из выборки. Соответствующий эвристический алгоритм будет описан ниже.

Распределение значений отступов в выборке даёт полезную дополнительную информацию не только об отдельных объектах, но и о выборке в целом. Если основная масса объектов имеет положительные отступы, то разделение выборки можно считать успешным. Если в выборке слишком много отрицательных отступов, то гипотеза компактности не выполняется, и в данной задаче при выбранной метрике применять алгоритмы типа kNN нецелесообразно. Если значения отступов концентрируются вблизи нуля, то ждать надёжной классификации не приходится, так как слишком много объектов оказываются в пограничной «зоне неуверенности».

2.2.2 Алгоритм STOLP для отбора эталонных объектов

Идея отбора эталонов реализована в алгоритме STOLP. Рассмотрим его обобщённый вариант с произвольной весовой функцией $\omega(i,u)$. Будем строить метрический алгоритм $a(u;\Omega)$ вида (2.1), где $\Omega \subseteq X^{\ell}$ – множество эталонов.

Вход:
X^ℓ — обучающая выборка;
δ — порог фильтрации выбросов;
ℓ_0 — допустимая доля ошибок;
Выход:
Множество опорных объектов $\Omega \subseteq X^\ell$;

1: **для всех** $x_i \in X^\ell$ проверить, является ли x_i выбросом:
2: **если** $M(x_i, X^\ell) < \delta$ **то**
3: $X^{\ell-1} := X^\ell \setminus \{x_i\}$; $\ell := \ell - 1$;
4: Инициализация: взять по одному эталону от каждого класса:
$\Omega := \left\{ \arg\max_{x_i \in X_y^\ell} M(x_i, X^\ell) \;\middle|\; y \in Y \right\}$;
5: **пока** $\Omega \neq X^\ell$;
6: Выделить множество объектов, на которых алгоритм $a(u; \Omega)$ ошибается:
$E := \{x_i \in X^\ell \setminus \Omega : M(x_i, \Omega) < 0\}$;
7: **если** $|E| < \ell_0$ **то**
8: **выход**;
9: Присоединить к Ω объект с наименьшим отступом:
$x_i := \arg\min_{x \in E} M(x, \Omega)$; $\Omega := \Omega \cup \{x_i\}$;

Рисунок 2.3 – Отбор эталонных объектов STOLP

Обозначим через $M(x_i, \Omega)$ отступ объекта x_i относительно алгоритма $a(x_i, \Omega)$. Большой отрицательный отступ свидетельствует о том, что объект x_i окружён объектами чужих классов, следовательно, является выбросом. Большой положительный отступ означает, что объект окружён объектами своего класса, то есть является либо эталонным, либо периферийным.

Данный алгоритм начинает с отсева выбросов (шаги 1–3). Из выборки X^ℓ исключаются все объекты x_i с отступом $M(x_i, X^\ell)$, меньшим заданного порога δ. Если взять $\delta = 0$, то оставшиеся объекты будут классифицированы верно. Вместо δ можно задавать долю исключаемых объектов с наименьшими значениями отступа.

Затем формируется начальное приближение – в Ω заносится по одному наиболее типичному представителю от каждого класса (шаг 4).

После этого начинается процесс последовательного «жадного» наращивания множества. На каждом шаге к Ω присоединяется объект x_i, имеющий минимальное значение отступа. Так продолжается до тех пор, пока число ошибок не окажется меньше заданного порога ℓ_0. Если положить $\ell_0 =$

0, то будет построен алгоритм $a(u, \Omega)$, не допускающий ошибок на обучающих объектах, за исключением заранее исключённых выбросов.

В результате каждый класс будет представлен в Ω одним «центральным» эталонным объектом и массой «приграничных» эталонных объектов, на которых отступ принимал наименьшие значения в процессе итераций. Параметр δ позволяет регулировать ширину зазора между эталонами разных классов. Чем больше δ, тем дальше от границы классов будут располагаться «приграничные» эталоны, и тем более простой, менее «изрезанной» будет граница между классами.

В описанном варианте алгоритм STOLP имеет относительно низкую эффективность. Для присоединения очередного эталона необходимо перебрать множество объектов $X^{\ell} \backslash \Omega$, и для каждого вычислить отступ относительно множества эталонов Ω. Общее число операций составляет $O(|\Omega|^2 \ell)$. Для ускорения алгоритма можно добавлять сразу по несколько эталонов, не пересчитывая отступов. Если при этом выбирать добавляемые эталоны достаточно далеко друг от друга, то добавление одного из них практически не будет влиять на отступы остальных. Аналогично, на этапе отсева выбросов можно вычислить отступы только один раз, и потом отбросить все объекты с отступами ниже δ.

Результатом работы алгоритма STOLP является разбиение обучающих объектов на три категории: шумовые, эталонные и неинформативные. Если гипотеза компактности верна и выборка достаточно велика, то основная масса обучающих объектов окажется неинформативной и будет отброшена. Фактически, этот алгоритм выполняет сжатие исходных данных.

РАЗДЕЛ 3. ЛОГИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

Существует обширный класс алгоритмов классификации, в основе которых лежит принцип индуктивного вывода логических закономерностей или индукции правил (rule induction, rule learning).

Пусть $\varphi: X \to \{0,1\}$ – 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.

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 3.1. (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 3.2. (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:

1. Каков критерий информативности, позволяющий называть предикаты закономерностями?

2. Как строить закономерности?

3. Как строить алгоритмы классификации на основе закономерностей?

3.1. Понятие информативности

3.1.1. 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 $Y \backslash \{c\}$ 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) можно опускать. Предполагается, что $P \geq 1$, $N \geq 1$, $P + N = \ell$.

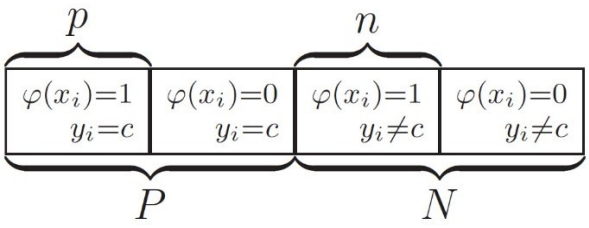


Рисунок 3.1 – Наглядное определение информативности

На рисунке 3.1 показаны четыре части выборки, на которые она разбивается условиями $y_i = c$ и $\varphi(x_i) = 1$.

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:

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

Предикат $\varphi(x)$ будем называть логической ε, δ-закономерностью для класса $c \in Y$, если $E_c\left(\varphi, X^\ell\right) \le \varepsilon$ и $D_c\left(\varphi, X^\ell\right) \ge \delta$ 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, 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.

В таблице 3.1 представлены различные расчетные значения критериев информативности предикатов.

Когда некоторая доля ошибок неизбежна, вполне допустимо пользоваться частичными закономерностями. Кроме того, существуют

способы построения корректных алгоритмов на основе частичных закономерностей, например, взвешенное голосование.

Табица 3.1 – Критерии информативности при $P = 200$ и $N = 100$

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

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. Следует учитывать, что вывести адекватную свертку критериев n и p достаточно сложно.

3.1.2. 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 $\{x: y^*(x) = c\}$ и $\{x: \varphi(x) = 1\}$. Тогда вероятность реализации пары (p, n) подчиняется гипергеометрическому распределению:

$$h_{P,N}(p,n) = \frac{C_P^p C_N^n}{C_{P+N}^{p+n}}, 0 \le p \le P, 0 \le n \le N, \qquad (3.1)$$

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

Если вероятность (3.1) мала, и тем не менее пара (p, n) реализовалась, то гипотеза о независимости должна быть отвергнута. Чем меньше значение вероятности 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.

Информативность предиката $\varphi(x)$ относительно класса $c \in Y$ по выборке X^ℓ есть:

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

Предикат $\varphi(x)$ будем называть статистической закономерностью для класса c, если $I_c(\varphi, X^\ell) \geq I_0$ 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. 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:

$$\ln(k!) \approx k \ln k - k + \frac{1}{2} \ln 2k\pi + \frac{1}{12k} - \frac{1}{360k^2}. \qquad (3.2)$$

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. Более точное приближение даёт формула Рамануджана:

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

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?

При разумных сочетаниях параметров ε и I0 эвристический критерий практически всегда оказывается строже статистического (рисунок 3.2). 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 будет показано, что в зависимости от ситуации оказывается более целесообразным применять то эвристический, то статистический критерий информативности.

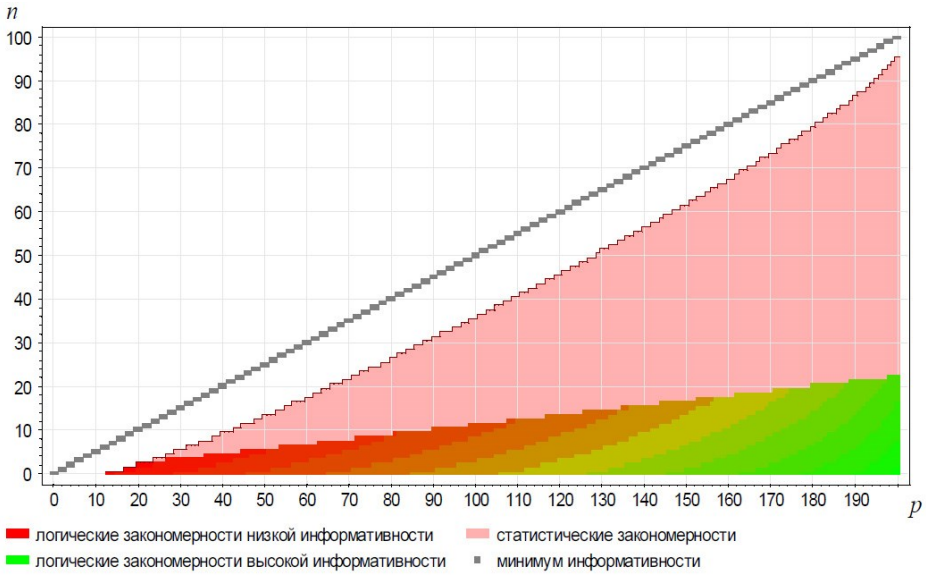


Рисунок 3.2 – Области логических ε, δ-закономерностей (при $\varepsilon = 0{,}1$) и статистических закономерностей (при $I_0 = 5$) в координатах (p, n) при $P = 200, N = 100$

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

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

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 $\{x \in X^\ell | \varphi(x) = 1\}$ есть $\widehat{H}(P, N)$. Вероятность

появления объекта из этой выборки оценивается как $\frac{p+n}{P+N}$. Аналогично, энтропия выборки $\{x \in X^\ell | \varphi(x) = 0\}$ есть $\widehat{H}(P-p, N-n)$, 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:

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

В итоге уменьшение энтропии составляет:

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

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.

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

Энтропийный критерий IGain_c асимптотически эквивалентен статистическому I_c:

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

Для доказательства достаточно применить к статистическому определению (3.1) 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 3.1, 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_c 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.

Однако на практике чаще используется энтропийный критерий, поскольку он проще вычисляется.

3.1.4. 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 3.2 на случай произвольного числа классов $Y = \{1, \dots, M\}$:

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

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 подразделе 3.1.3, 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 $h(z) \equiv -z \log_2 z$.

3.1.5. 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 б**о**льшим весом при поиске логических закономерностей.

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

$$P_c^\omega = \sum_{i=1}^{\ell} \omega_i [y_i = c]; \qquad p_c^\omega(\varphi) = \sum_{i=1}^{\ell} \omega_i [y_i = c][\varphi(x_i) = 1];$$
$$N_c^\omega = \sum_{i=1}^{\ell} \omega_i [y_i \neq c]; \qquad n_c^\omega(\varphi) = \sum_{i=1}^{\ell} \omega_i [y_i \neq c][\varphi(x_i) = 1]. \tag{3.3}$$

Определение логической ε, δ-закономерности, статистическая и энтропийная информативность легко обобщаются на этот случай. Для вычисления гипергеометрического распределения (3.1) можно воспользоваться обобщённым определением факториала через гамма-функцию, однако это сопряжено с трудоёмкими вычислениями. Другой подход – округлять значения P_c^ω, N_c^ω, p_c^ω, n_c^ω 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:

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

3.2. Методы поиска информативных закономерностей

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, $f_j: X \to \{0,1\}$, $j = 1, \ldots, n$. 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. Более того, качестве закономерностей можно брать только конъюнкции признаков и их отрицаний, а дизъюнкцию реализовать как корректирующую операцию, например, как голосование по большинству или старшинству.

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.

3.2.1. Бинаризация количественных признаков

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

1. Если f – номинальный признак:

$\beta(x) = [f(x) = d],\ d \in D_f;$

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

2. Если f – порядковый или количественный признак:

$\beta(x) = [f(x) \le d],\ d \in D_f$;

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

В случае количественных признаков $f: X \to \mathbb{R}$ 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:

$$d_i = \frac{f^{(i)} + f^{(i+1)}}{2}, f^{(i)} \ne f^{(i+1)}, i = 1, \dots, \ell - 1. \quad (3.4)$$

где $f^{(1)} \le \dots \le f^{(\ell)}$ – последовательность значений признака f на объектах выборки $f(x_1), \dots, f(x_\ell)$, упорядоченная по возрастанию (вариационный ряд, рисунок 3.3).

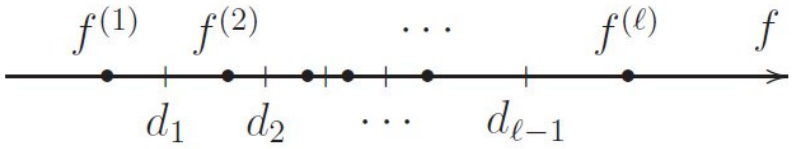


Рисунок 3.3 – Вариационный ряд значения признака $f(x)$ и пороги d_i

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 (рисунок 3.4).

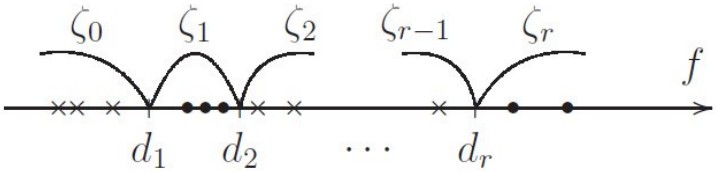


Рисунок 3.4 – Начальное разбиение на зоны позитивных (×) и неготивных (•) объектов

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

$\zeta_0(x) = [f(x) < d_1]$;

$\zeta_s(x) = [d_s \le f(x) < d_{s+1}], s = 1, \ldots, r-1$;

$\zeta_r(x) = [d_r \le f(x)]$.

Алгоритм (рисунок 3.5) начинает с разбиения на «мелкие зоны». Пороги определяются по формуле (3.4) 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 с или между точками не класса с приведёт только к уменьшению информативности зон. Итак, начальное разбиение состоит из чередующихся зон «только c – только не c», как показано на рисунке 3.4.

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;

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

Рисунок 3.5 – Жадный алгоритм слияния зон

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 $O(\ell/\sqrt{\tau})$.

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 с потеря не очень существенна, поскольку разбиение на зоны производилось по критерию информативности I_c. Дискретизацию можно применять как способ сжатия информации с наименьшими потерями.

Относительно другого класса $c^j \neq c$ разбиение диапазона значений признака на информативные зоны может оказаться совершенно иным. Алгоритм (рисунок 3.5) легко приспособить и для «универсального» разбиения на зоны, учитывающего сразу все классы. Для этого достаточно заменить критерий информативности I_c многоклассовым критерием.

3.2.2. Поиск закономерностей в форме конъюнкций

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

$$\mathcal{K}_K[\mathcal{B}] = \{\varphi(x) = \beta_1(x) \ldots \wedge \beta_k(x) | \beta_1, \ldots, \beta_k \in \mathcal{B},\ k \leq K\}.$$

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, опятьтаки 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 $O(|\mathcal{B}|^K)$, 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 заключается в том, чтобы не перебирать огромное количество заведомо неинформативных предикатов. Рассмотрим наиболее простой метод.

«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 последова тельность конъюнкций $\varphi_0, \varphi_1, \dots, \varphi_t, \dots$, 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. 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 (рисунок 3.6), 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.

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: $\varphi_t := \arg\max_{\varphi \in V_t} I_c(\varphi, X^\ell)$;
  наилучшая конъюнкция на t-й итерации: $\varphi_t^* := \arg\max_{\varphi \in V_t:\, E_c(\varphi) < \varepsilon} I_c(\varphi, X^\ell)$;
  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^*;

Рисунок 3.6 – «Градиентный» алгоритм синтеза конъюнкции

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.

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

3.2.3. Формы закономерностей

Конъюнкции над предикатами вида $\beta(x) = [d \le f(x) \le d']$ 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.

1. 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 $\omega \subseteq \{1, \dots, n\}$. 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».

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

Параметрическое семейство шаров:

$$\varphi_{\mathrm{c}}(x) = [\rho(x, x_0) \le r_0], \tag{3.5}$$

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 (3.5).

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

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

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.

Параметрическое семейство полуплоскостей:

$$\varphi_c(x) = [\langle x, \alpha \rangle \leq \alpha_0], \tag{3.6}$$

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 (3.6) 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 αj интерпретируются как степень важности j-го признака.

Параметрическое семейство областей, описываемых ядром:

$$\varphi(x) = [K(x, x_0) \leq K_0], \qquad (3.7)$$

где $K: X \times X \to \mathbb{R}$ – 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.

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 ¾потенциально полезные¿ функции, в том числе метрики и скалярные произведения. Тогда процедура поиска информативных закономерностей должна включать в себя перебор ядер. Ядра, зависящие только от небольшого числа признаков, проще поддаются содержательной интерпретации, но для их поиска приходится организовывать перебор подмножеств признаков.

3.3. Решающие списки

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

Решающий список – это алгоритм классификации $a: X \to Y$, 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 (рисунок 3.7).

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

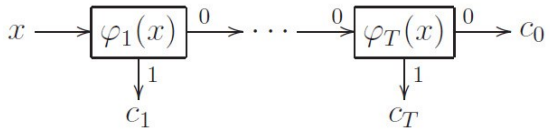


Рисунок 3.7 – Классификация объекта $x \in X$ решающим списком

«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».

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.

Решающий список закономерностей представляет собой частный случай алгоритмической композиции с голосованием по старшинству. Различия разве что терминологические: теперь базовые алгоритмы называются закономерностями или правилами. Рассмотрим альтернативный алгоритм, в котором, благодаря использованию критерия информативности, удаётся обойтись без априорного параметра λ, устанавливающего величину «штрафа за отказ».

3.3.1. Жадный алгоритм построения решающего списка

Алгоритм (рисунок 3.8) на каждой итерации строит ровно одно правило ϕt, 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 – может быть каким угодно, лишь бы для него существовала эффективная процедура поиска закономерностей. В частности, если Ф – конъюнкции, то на шаге 4 можно применить «градиентный» алгоритм (рисунок 3.6).

Вход:
X^ℓ — обучающая выборка;
Ф — 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:
$\varphi_t := \arg\max_{\varphi \in \Phi'} I_c(\varphi, U)$, где $\Phi' = \{\varphi \in \Phi : E_c(\varphi, U) \leqslant E_{\max}\}$;
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**;

Рисунок 3.8 – Жадный алгоритм построения решающего списка

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 $\{x: \varphi_t(x) = 1\}$. Поэтому решающий список называют также покрывающим набором закономерностей или машиной покрывающих множеств (set covering machine, SCM).

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. Это нежелательно, так как в решающем 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 (рисунок 3.8) 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 (рисунок 3.8) 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. 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.

3.3.2. 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 Ф, критерием информативности и методом поиска информативных предикатов.

1. 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 (рисунок 3.8), применяя на шаге 4 любой из методов поиска информативных конъюнкций, например, градиентный алгоритм (рисунок 3.6), либо алгоритм на рисунке 3.10 с параметром $T_0=1$.

2. Алгоритм (рисунок 3.8) с семейством предикатов Ф вида (3.5) строит покрытие обучающей выборки шарами (data dependent balls). Очень похожий алгоритм описан Маршандом и др. под названием BuildSCM. 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.

3. Алгоритм дробящихся эталонов ДРЭТ 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.

4. Непосредственное применение алгоритма (рисунок 3.8) 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 (рисунок 3.9).

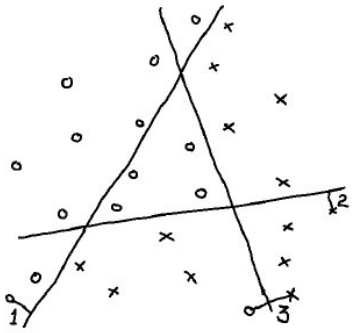


Рисунок 3.9 – Построение решающего списка из трёх полуплоскостей

Известно большое количество эвристик для последовательного построения разделяющих полуплоскостей.

5. Алгоритм Белецкого 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 (рисунок 3.8), после построения каждой полуплоскости делается попытка найти лучшие положения предыдущих полуплоскостей.

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

Достоинства решающих списков.

1. Интерпретируемость и простота классификации. Обученное по выборке правило классификации можно записать в виде инструкции и выполнять «вручную».

2. Гибкость: в зависимости от выбора множества можно строить весьма разнообразные алгоритмические конструкции.

3. Возможность обработки разнотипных данных и данных с пропусками.

Недостатки решающих списков.

1. Если множество правил выбрано неудачно, список может не построиться. При этом возможен высокий процент отказов от классификации.

2. 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.

3. 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.

3.4. 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.

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.

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, представленным на рисунке 3.10.

```
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.
```

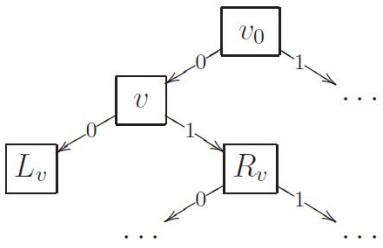


Рисунок 3.10 – Классификация объекта $x \in X$ бинарным решающим деревом

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 $v = \{x \in X: K_v(x) = 1\}$, 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:

$$a(x) = \arg\max_{y \in Y} \sum_{v \in T} [c_v = y] K_v(x), \tag{3.8}$$

причем 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 $\left|\Omega_v \cap X^\ell\right|$.

3.4.1. Синтез решающих деревьев

Задача построения дерева минимальной сложности, правильно классифицирующего заданную выборку, в общем случае является NP-полной задачей. 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».

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:

$v_0 := LearnID3(X^\ell)$.

На шаге 6 алгоритма (рисунок 3.11) выбирается предикат β из заданного семейства $\mathcal{B}$, задающий максимально информативное ветвление дерева – разбиение выборки на две части $U = U_0 \cup U_1$.

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

1: **ПРОЦЕДУРА** `LearnID3` (U);
2: **если** все объекты из U лежат в одном классе $c \in Y$ **то**
3: создать новый лист v;
4: $c_v := c$;
5: **вернуть** (v);
6: найти предикат с максимальной информативностью:
$\beta := \arg\max_{\beta \in \mathscr{B}} I(\beta, U)$;
7: разбить выборку на две части $U = U_0 \cup U_1$ по предикату β:
$U_0 := \{x \in U : \beta(x) = 0\}$;
$U_1 := \{x \in U : \beta(x) = 1\}$;
8: **если** $U_0 = \varnothing$ или $U_1 = \varnothing$ **то**
9: создать новый лист v;
10: $c_v :=$ класс, в котором находится большинство объектов из U;
11: **иначе**
12: создать новую внутреннюю вершину v;
13: $\beta_v := \beta$;
14: $L_v :=$ `LearnID3` (U_0); (построить левое поддерево)
15: $R_v :=$ `LearnID3` (U_1); (построить правое поддерево)
16: **вернуть** (v);

Рисунок 3.11 – Рекурсивный алгоритм синтеза бинарного решающего дерева ID3

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.

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

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

В алгоритме на рисунки 3.11 множество элементарных предикатов $\mathcal{B}$ может быть каким угодно, лишь бы существовал эффективный механизм выбора наиболее информативного предиката из $\mathcal{B}$ на шаге 6. Когда мощность $|\mathcal{B}|$ не велика, эта задача легко 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 $\beta(x) = [f_j(x) <> d_j]$. 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.

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 $\beta(x) = [f_j(x) <> d_j]$, 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 выборки при пропорциональном увеличении $p(\beta)$ и $n(\beta)$. Эвристический и энтропийный

критерии удовлетворяют этому требованию, а гипергеометрический – нет; 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.

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 $B = |\mathcal{B}|$, однако применение удачных эвристик для сокращения перебора на шаге 6 позволяет существенно уменьшить B.

Преимущества алгоритма ID3.

1. Простота и интерпретируемость классификации. Алгоритм (рисунок 3.10) 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.

2. Трудоёмкость алгоритма (рисунок 3.11) линейна по длине выборки.

3. Если множество предикатов $\mathcal{B}$ 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.

4. Не бывает отказов от классификации, в отличие от решающих списков.

5. 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.

1. 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.

2. 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.

3. Высокая чувствительность к составу выборки. Изменение данных в 1-2 объектах часто приводит к радикальному изменению структуры дерева.

4. 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.

3.4.2. 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. 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 $\beta \in B$ не дотягивает до заданного порогового

значения I_0. Для этого на шаге 8 условие ($U_0 = \emptyset$ или $U_1 = \emptyset$) заменяется условием $I(\beta, U) \leq I_0$. 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 Xk пропускается через построенное дерево. При этом в каждой внутренней вершине v запоминается подмножество $S_v \subseteq X^k$ попавших в неё контрольных объектов. Если $S_v = \emptyset$, 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 $c = \arg\min_{c \in Y} r_c(v)$.

3.4.3. Преобразование решающего дерева в решающий список

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

Согласно формуле (3.8) 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 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 3.2.2. Редукция приводит к расширению множеств v объектов, выделяемых конъюнкциями K_v, и они начинают 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.

3.4.4. 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 3.3. 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 рисунке 3.12. 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 рисунке 3.13.

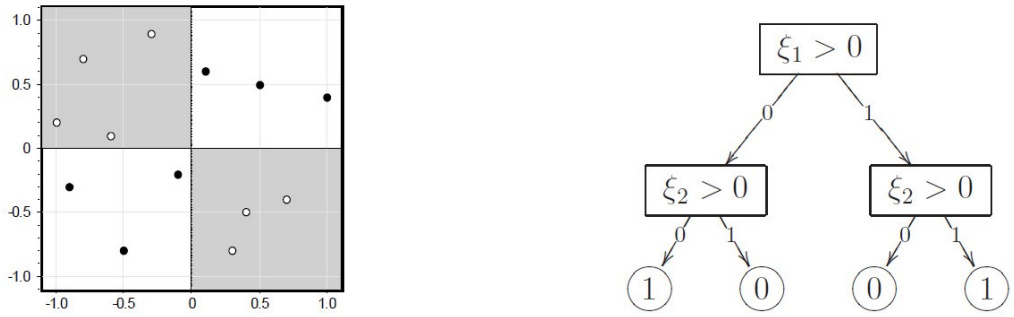


Рисунок 3.12 – Выборка типа XOR и идеальное дерево для неё

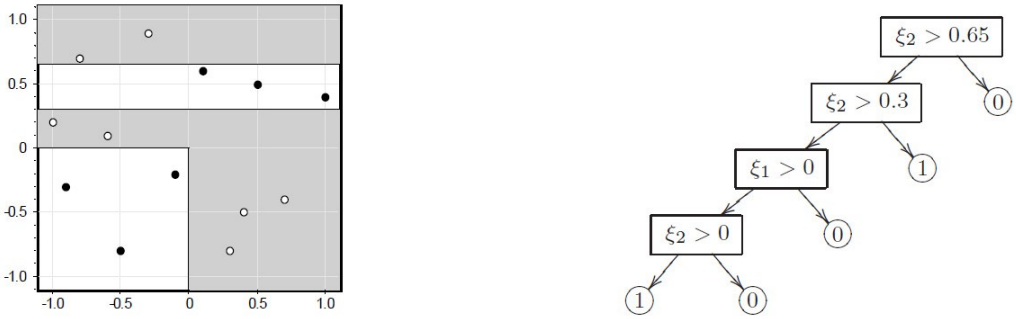


Рисунок 3.13 – Та же выборка и дерево, построенное жадным алгоритмом

Идея заглядывания вперёд (look ahead) заключается в том, чтобы на шаге 6, вместо вычисления информативности для каждого $\beta \in B$ 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. 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.

3.5. 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, принцип голосования применялся для построения композиций из произвольных алгоритмов классификации. Теперь рассмотрим композиции, состоящие из логических закономерностей.

3.5.1. Принцип голосования

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

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

Считается, что если $\varphi_c^t(x) = 1$, то правило φ_c^t с 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:

$$a(x) = \arg\max_{c \in Y} \Gamma_c(x). \tag{3.9}$$

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

Нормирующий множитель $\frac{1}{T_c}$ вводится для того, чтобы наборы с большим 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 $\alpha_c^t \geq 0$, и при голосовании берётся взвешенная сумма голосов:

$$\Gamma_c(x) = \frac{1}{T_c}\sum_{t=1}^{T_c} \varphi_c^t \alpha_c^t(x), \quad \alpha_c^t \geq 0. \tag{3.10}$$

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 $\varphi_c^1, \dots, \varphi_c^{T_c}$. Простое голосование является частным случаем взвешенного, когда веса одинаковы и равны $\frac{1}{T_c}$.

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.

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 А, если он регулярно подсматривает решения у эксперта В. Приведём простое теоретико-вероятностное обоснование принципа диверсификации, или повышения различности (diversity) правил. 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:

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

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:

$$\mathsf{P}\{M < 0\} \le \mathsf{P}\{\, |\, \mathsf{E}M - M\, | > \mathsf{E}M\, \} \le \mathsf{D}M/(\mathsf{E}M)^2.$$

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. Сначала для фиксированного класса $c \in Y$ строится покрывающий набор правил точно так, как это делалось для решающих списков. 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.

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:

1. Как построить много правил по одной и той же выборке?

2. Как избежать повторов и построения почти одинаковых правил?

3. Как избежать появления непокрытых объектов и обеспечить равномерное покрытие всей выборки правилами?

4. Как определять веса правил при взвешенном голосовании?

Рассмотрим, как эти проблемы решаются в известных алгоритмах.

3.5.2. Алгоритм КОРА

Алгоритм комбинаторного распознавания КОРА, предложенный М.М. Бонгардом в 1961-м году и реализованный М.Н. Вайнцвайгом, строит

набор конъюнктивных закономерностей. Этот алгоритм неоднократно доказал свою эффективность при решении прикладных задач.

В основе алгоритма лежат следующие эвристические предположения.

1. Множество элементарных предикатов B подобрано настолько удачно, что среди конъюнкций ранга 2 или 3 уже находится достаточное количество информативных закономерностей.

2. Поскольку ранг конъюнкций ограничен сверху числом 3, для поиска закономерностей можно применить полный перебор.

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

Алгоритм (рисунок 3.14) строит для каждого класса $c \in Y$ свой список конъюнкций R_c. Каждая конъюнкция содержит не более K термов, выбираемых из множества предикатов $\mathcal{B}$. 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 $D_c(\varphi) \geq D_{\min}$ и $E_c(\varphi) \leq E_{\max}$.

Перебор конъюнкций осуществляется методом поиска в глубину. На рисунке 3.15 показано дерево перебора всех конъюнкций при $|\mathcal{B}| = 4$. В процессе перебора необходимо избегать повторного просмотра конъюнкций, отличающихся только порядком записи термов. Для этого фиксируется некоторая исходная нумерация предикатов $\mathcal{B} = \{\beta_1, \ldots, \beta_{|\mathcal{B}|}\}$, и конъюнкции $\varphi = \beta_{j_1} \wedge \ldots \wedge \beta_{j_s}$ наращиваются так, чтобы номера предикатов строго возрастали: $1 \leq j_1 \leq \cdots \leq j_s \leq |\mathcal{B}|$.

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.

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');

Рисунок 3.14 – Построение списка информативных конъюнкций методом поиска в ширину (алгоритм КОРА)

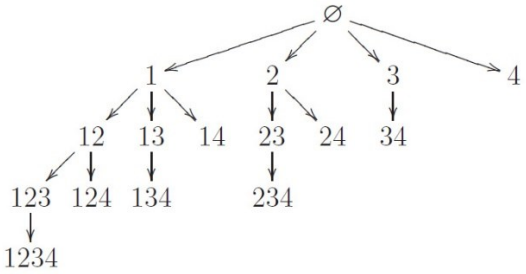


Рисунок 3.15 – Дерево полного перебора конъюнкций в алгоритме КОРА при $|\mathcal{B}| = 4$. Для краткости конъюнкции обозначены номерами составляющих их предикатов

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. Процедура показана на рисунке 3.16.

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;

Рисунок 3.16 – Включение конъюнкции φ в список R_c, содержащий не менее T самых информативных конъюнкций

Важные замечания по реализации алгоритмов:

1. 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.

2. 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.

3. Почему в алгоритме на рис. 3.16 нельзя сделать проще – после шага 2 удалять последнюю в списке наихудшую конъюнкцию? Проблема в том, что информативность $I_c(\varphi)$, 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.

1. Короткие конъюнкции легко интерпретируются в терминах предметной области. Алгоритм способен не только классифицировать объекты, но и объяснять свои решения на языке, понятном специалистам.

2. При малых $K, K \leq 3$, алгоритм очень эффективен.

3. Если короткие информативные конъюнкции существуют, они обязательно будут найдены, так как алгоритм осуществляет полный перебор.

Недостатки алгоритма КОРА.

1. При неудачном выборе множества предикатов В 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 $O(|\mathcal{B}|^K \ell)$.

2. Алгоритм не стремится диверсифицировать конъюнкции и обеспечивать равномерность покрытия объектов. Это отрицательно сказывается на обобщающей способности (вероятности ошибки) алгоритма.

3. Нет настройки коэффициентов α_c^t; предполагается простое голосование.

3.5.3. 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. 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 (рисунок 3.17) начинает процесс поиска закономерностей с построения конъюнкций ранга 1. Для этого отбираются не более T_1 самых информативных предикатов из базового множества $\mathcal{B}$. Затем к каждому из отобранных предикатов добавляется по одному терму из $\mathcal{B}$ всеми возможными способами. Получается не более $T_1|\mathcal{B}|$ 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 (рисунок 3.6). 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.

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;

Рисунок 3.17 – Построение списка информативных конъюнкций методом поиска в ширину (алгоритм ТЭМП)

Для улучшения конъюнкций к ним применяют эвристические методы «финальной шлифовки» – стабилизацию и редукцию.

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.

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

Достоинства алгоритма ТЭМП.

1. ТЭМП существенно более эффективен, чем КОРА, особенно при поиске конъюнкций ранга больше 3. Он решает поставленную задачу за $O(KT_1|B|\ell)$ операций, тогда как КОРА имеет трудоёмкость $O(|B|^K \ell)$.

2. Параметр T_1 позволяет управлять жадностью алгоритма и находить компромисс между качеством конъюнкций и скоростью работы алгоритма.

3. Благодаря простоте и эффективности алгоритм ТЭМП можно использовать в составе других алгоритмов как генератор конъюнкций, достаточно близких к оптимальным.

Недостатки алгоритма ТЭМП.

1. Нет гарантии, что будут найдены самые лучшие конъюнкции, особенно при малых значениях параметра T_1.

2. Алгоритм не стремится увеличивать различность конъюнкций, добиваясь равномерного покрытия объектов выборки. Стабилизация и редукция лишь отчасти компенсирует этот недостаток.

3. Нет настройки коэффициентов α_c^t; предполагается простое голосование.

3.5.4. 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.

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. 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.

3.6. Алгоритмы вычисления оценок

Алгоритмы вычисления оценок (АВО) были предложены Ю. И. Журавлёвым в начале 70-х годов. 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:

$$\rho_s(x, x') = |f_s(x) - f_s(x')|, s = 1, \ldots, n. \quad (3.11)$$

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 β(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 х 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 $\rho_s: X \times X \to \mathbb{R}_+, s = 1, \ldots, n$, в общем случае не обязательно вида (3.11) и даже не обязательно метрики.

2. Задаётся система опорных множеств

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

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

$$B_\omega(x, x') = \bigwedge_{s \in \omega} [\rho_s(x, x') \leq \varepsilon_s], \tag{3.12}$$

где ε_s – неотрицательные числа, называемые порогами, $s = 1, \ldots, n$. В случае (3.11) 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 с по всем опорным множествам:

$$\Gamma_c(x) = \sum_{i: y_i = c} \sum_{\omega \in \Omega} \alpha_{\omega i} B_\omega(x, x_i), \tag{3.13}$$

где веса $\alpha_{\omega i}$, как правило, предполагаются нормированными на единицу: $\sum_{i: y_i = c} \sum_{\omega \in \Omega} \alpha_{\omega i} = 1$, для всех $c \in Y$.

5. Алгоритм классификации $a(x)$ относит объект x к тому классу, который н а б р а л 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 (3.9)–(3.10), если в качестве правил Rc взять функции близости:

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

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 $\mathcal{B}$. В данном случае оно имеет вид

$$\mathcal{B} = \{\beta(x) = [\rho_s(x, x_i) \le \varepsilon_s] \mid s = 1,\ldots,n, y_i = c, i = 1,\ldots,\ell\}.$$

Для сокращения перебора при построении конъюнкций имеет смысл заранее отобрать лишь некоторые значения порогов $\{\varepsilon_s\}$, например, с помощью алгоритма (рисунок 3.5), выделяющего информативные зоны. Такой способ выбора порогов не даёт гарантии их оптимальности. Оптимизация порогов является сложной комбинаторной задачей, для практического решения которой разработаны достаточно удачные эвристические алгоритмы.

Ещё один способ сокращения перебора – заранее отобрать в качестве эталонов не все обучающие объекты, а только наиболее представительные. Это эквивалентно обнулению весов αωi 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

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

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 $\mathcal{B}$. Например, в Алгоритме ТЭМП (рисунок 3.17) 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.

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. 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:

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

При $\varepsilon = 0$ этот вариант совпадает с (3.12). 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:

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

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 (3.13). 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:

$\Gamma_c(x) > \Gamma_y(x) + \delta_1$, для всех $y \in Y\backslash\{c\}$;

$\Gamma_c(x) > \delta_2 \sum_{y\in Y\backslash\{c\}} \Gamma_y(x)$.

В остальных случаях алгоритм отказывается от классификации объекта x.

3.6.1. Тупиковые представительные наборы

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

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:

$$\Omega_i = \{\omega \mid \langle \omega, i \rangle - \text{ тупиковый представительный набор}\}, i = 1, \ldots, \ell,$$

и оценка близости объекта x к классу с 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 (3.13), если положить $\alpha_{\omega i} = 0$ для всех $\langle \omega, i \rangle$, не являющихся тупиковыми представительными наборами.

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

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

Число выделяемых позитивных объектов pc(ϕc) может оказаться недостаточным, чтобы представительный набор можно было называть закономерностью в соответствии с определениями.

Для построения тупиковых представительных наборов можно применять известные алгоритмы поиска информативных конъюнкций в семействе предикатов ϕ вида (3.12) по критериям $n_c(\varphi) = 0$ и $p_c(\varphi) \to \max$.

3.6.2. Тупиковые тесты

Множество $\omega \subseteq \{1, \ldots, n\}$ называется тестом, если для любых $x_i, x_j \in X$ из $y_i \neq y_j$ следует $B_\omega(x_i, x_j) = 0$, 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.

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

Алгоритм вычисления оценок, в котором системой опорных множеств 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 $\langle U, V, v^*, U^\ell \rangle$, в которой:

$U = X \times X$ – объекты из U есть пары объектов из X;

$V = \{0,1\}$ – множество допустимых ответов двухэлементно;

$v^*(x, x') = [y^*(x) = y^*(x')]$ – целевая зависимость;

$U^L = \{(x_i, x_j) \mid x_i, x_j \in X^\ell, i \le j\}$ – 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 $1 \in V$.

Всякой непротиворечивой закономерности вида $\varphi_1(x_i, x_j) = B_\omega(x_i, x_j)$ относительно класса $1 \in V$ на выборке U^L соответствует тест ω.

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

$$\begin{cases} n_1(\varphi_1) = 0, & - \text{по определению теста;} \\ p_1(\varphi_1) \ge \ell, & - \text{т. к. } B_\omega(x_i, x_j) = 1 \text{ при всех } i = 1, \dots, \ell. \end{cases}$$

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 определениями.

Утверждение о непротиворечивой закономерности вида $\varphi_1(x_i, x_j) = B_\omega(x_i, x_j)$ показывает, что для построения тупиковых тестов можно применять алгоритмы поиска непротиворечивых конъюнктивных закономерностей, переходя к вспомогательной задаче классификации и максимизируя функционал $p_1(\varphi_1)$.

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.

3.7. Поиск ассоциативных правил

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

Каждому набору признаков $\varphi \subseteq F$ 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 $\varphi \subseteq \mathcal{F}$ называется часто встречающимся набором (frequent itemset), если $\nu(\varphi) > \delta$. Параметр δ называется минимальной поддержкой, и в литературе обозначается MinSupp.

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

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

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

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. Задача поиска ассоциативных правил исходно возникла в связи с анализом рыночных корзин (market basket analysis). 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 $v(y|\varphi) = 60\%$; причём оба товара покупаются совместно с вероятностью $v(y \cup \varphi) = 2\%$». Заметим, что величина $v(y|\varphi)$ 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 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:

$D_1(\varphi) = v(\varphi \cup y) \geq \delta$;

$E_1(\varphi) = 1 - v(y|\varphi) \leq 1 - \varkappa \equiv \varepsilon$.

Таким образом, различия двух определений – чисто терминологические.

Алгоритм поиска ассоциативных правил. Исторически одним из первых стал алгоритм APriory (рисунок 3.18). 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 $\psi \in \mathcal{F}$ и любого его собственного подмножества $\varphi \subset \psi$ справедливо $\nu(\varphi) > \nu(\psi)$. Иными словами, добавление признака в набор может привести только к снижению частоты его встречаемости. Отсюда следствие: чтобы набор ψ был часто встречающимся, 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.

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:
$G_1 := \{f \in \mathcal{F} \mid \nu(f) \geqslant \delta\}$;
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: $\varphi' := \varphi \setminus \{f\}$; $y' := y \cup \{f\}$;
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');

Рисунок 3.18 – Алгоритм поиска ассоциативных правил Priory

Алнгоритм (рисунок 3.18) состоит из следующих 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 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 $\mathrm{AssocRules}(\varphi, y)$ 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.

Усовершенствования алгоритма.

1. Разработано большое количество эффективных алгоритмов поиска ассоциативных правил, основанных на специальных структурах хранения данных.

2. 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.

3. 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.

4. 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.

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.

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:

1. Во многих задачах существование логических закономерностей является объективным фактом. Если алгоритму обучения удаётся их отыскать, то, как правило, они служат надёжными правилами классификации.

2. Интерпретируемость. Построенный алгоритм может быть записан на естественном языке в виде набора правил – логических высказываний.

3. Простота реализации классификатора. После того, как алгоритм классификации обучен, собственно классификация может производиться даже вручную.

4. Возможность обработки разнотипных данных и данных с пропусками.

Недостатки логических алгоритмов классификации:

1. Существуют задачи, в которых применение пороговых решающих правил и дискретных методов обучения нежелательно. Например, когда требуется построение гладких границ между классами.

2. Процедура обучения, как правило, сводится к решению NP-полных задач. Для их приближённого решения приходится применять различные эвристические приёмы, не гарантирующие оптимальности решения.

3. 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.

РАЗДЕЛ 4. ЛИНЕЙНЫЕ МЕТОДЫ КЛАССИФИКАЦИИ

Линейные модели широко используются в машинном обучении благодаря их относительной простоте, в некоторых случаях хорошей интерпретируемости и наличию глубоко проработанных численных методов. Простейшим обоснованием линейного классификатора служит его аналогия с нервной клеткой – нейроном. Персептронные принципы обучения, первоначально заимствованные из нейрофизиологии, затем нашли математические обоснования и с точки зрения градиентных методов минимизации эмпирического риска, и с точки зрения байесовской теории классификации, и с точки зрения статистических оценок обобщающей способности.

4.1. Аппроксимация и регуляризация эмпирического риска

Рассмотрим задачу классификации с двумя классами, $Y = \{-1, +1\}$.

Пусть модель алгоритмов представляет собой параметрическое семейство отображений $a(x, w) = \operatorname{sign} f(x, w)$, где w – вектор параметров. Функция $f(x, w)$ называется дискриминантной функцией. Пока мы не будем предполагать, что она линейна по параметрам. Если $f(x, w) > 0$, то алгоритм a относит объект x к классу +1, иначе к классу −1. Уравнение $f(x, w) = 0$ описывает разделяющую поверхность.

Как обычно, задача обучения классификатора $a(x, w)$ заключается в том, чтобы настроить вектор параметров w, имея обучающую выборку пар $X^\ell = (x_i, y_i)_{i=1}^\ell$.

Величина $M_i(w) = y_i f(x_i, w)$ называется отступом (margin) объекта x_i относительно алгоритма классификации $a(x, w) = \operatorname{sign} f(x, w)$.

Если $M_i(w) < 0$, то алгоритм $a(x, w)$ допускает ошибку на объекте x_i. Чем больше отступ $M_i(w)$, тем правильнее и надёжнее классификация объекта x_i.

Минимизация аппроксимированного эмпирического риска. Определим функцию потерь вида $\mathcal{L}\big(M_i(w)\big)$, где $\mathcal{L}(M)$ – монотонно невозрастающая функция от с т у п а, мажорирующая пороговую функцию потерь: $[M < 0] \leq \mathcal{L}(M)$. Тогда минимизацию суммарных потерь можно рассматривать как приближённый метод минимизации эмпирического риска – числа ошибок на обучающей выборке:

$$Q\big(w, X^\ell\big) = \sum_{i=1}^{\ell} [M_i(w) < 0] \;\leq\; \tilde{Q}\big(w, X^\ell\big) = \sum_{i=1}^{\ell} \mathcal{L}\big(M_i(w)\big) \to \min_w. \qquad (4.1)$$

Некоторые применяемые на практике функции потерь показаны на рисунке 4.1. Квадратичная функция потерь не является монотонной, тем не менее, она соответствует линейному дискриминанту Фишера. Кусочно-линейная функция потерь соответствует методу опорных векторов (SVM), сигмоидная используется в нейронных сетях, логистическая – в логистической регрессии, экспоненциальная – в алгоритме бустинга AdaBoost. Каждый из перечисленных методов имеет свою разумную мотивацию, однако все они приводят к разным функциям потерь.

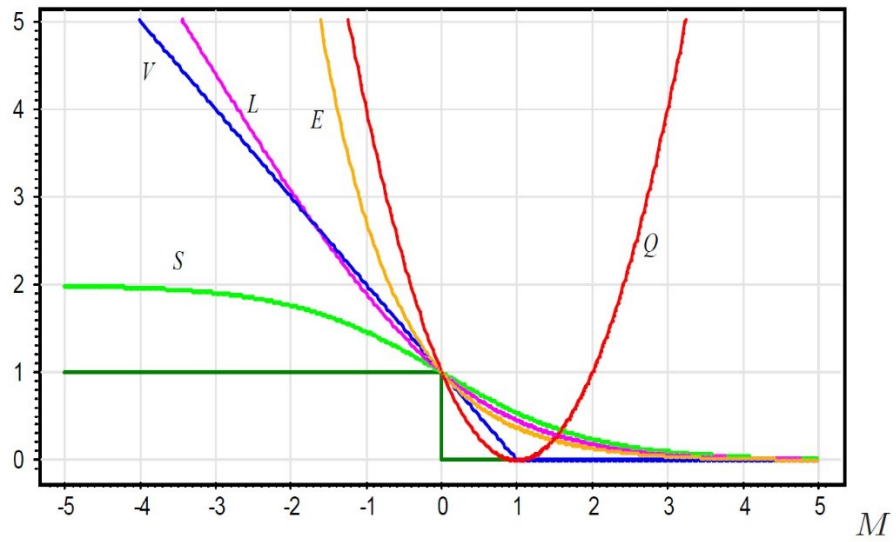


Рисунок 4.1 – Непрерывные аппроксимации пороговой функции потерь $[M < 0]$: $Q(M) = (1 - M)^2$ – квадратичная; $V(M) = (1 - M)_+$ – кусочно-линейная; $S(M) = 2(1 + e^M)^{-1}$ – сигмоидная; $L(M) = \log_2(1 + e^{-M})$ – логистическая; $E(M) = e^{-M}$ – экспоненциальная.

Исходя из эвристического принципа максимизации отступов, невозможно ответить на ряд вопросов: какие ещё функции $\mathcal{L}(M)$ допустимы, какие из них более предпочтительны и в каких ситуациях.

Вероятностная модель данных позволяет по-другому взглянуть на задачу. Допустим, множество $X \times Y$ является вероятностным пространством, и вместо модели разделяющей поверхности $f(x, w)$ задана параметрическая модель совместной плотности распределения объектов и классов $p(x, y|w)$.

Для настройки вектора параметров w по обучающей выборке X^ℓ применим принцип максимума правдоподобия. Найдём такое значение вектора параметров w, при котором наблюдаемая выборка X^ℓ максимально правдоподобна (совместная плотность распределения объектов выборки максимальна). Если выборка X^ℓ простая (состоит из независимых наблюдений, взятых из одного и того же распределения $p(x, y|w)$), то правдоподобие представляется в виде произведения:

$p(X^\ell|w) = \prod_{i=1}^{\ell} p(x_i, y_i|w) \to \max_w$.

Удобнее рассматривать логарифм правдоподобия:

$$L(w, X^\ell) = \ln p(X^\ell|w) = \sum_{i=1}^{\ell} \ln p(x_i, y_i|w) \to \max_w. \qquad (4.2)$$

Сопоставляя (4.2) с правой частью (4.1), легко заключить, что эти две задачи эквивалентны, если положить

$-\ln p(x_i, y_i|w) = \mathcal{L}(y_i f(x_i, w))$.

Зная модель плотности $p(x, y|w)$, можно выписать вид разделяющей поверхности f и функцию $\mathcal{L}$. С другой стороны, задавая вид разделяющей поверхности и функцию потерь $\mathcal{L}$, казалось бы, из чисто эвристических соображений, мы тем самым неявно принимаем некоторую вероятностную модель данных.

Принцип максимума совместного правдоподобия данных и модели. Допустим, что наряду с параметрической моделью плотности распределения $p(x, y|w)$ имеется ещё априорное распределение в пространстве параметров

модели $p(w)$. Никакая модель не может быть идеальной, поэтому вряд ли параметрическое семейство плотностей $p(x, y|w)$ содержит ту самую неизвестную плотность, которая породила выборку. Будем считать, что выборка может быть порождена каждой из плотностей $p(x, y|w)$ с вероятностью $p(w)$.

Чтобы ослабить априорные ограничения, вместо фиксированной функции $p(w)$ вводится параметрическое семейство априорных распределений $p(w; \gamma)$, где γ – неизвестная и не случайная величина, называемая гиперпараметром. Исследователю разрешается варьировать значение гиперпараметра. При этом свойства модели могут изменяться радикальным образом, и появляется возможность найти такое значение гиперпараметра, при котором модель наиболее адекватна.

Принцип максимума правдоподобия теперь будет записываться по-другому, поскольку не только появление выборки X^ℓ, но и появление модели w также является случайным. Их совместное появление описывается, согласно формуле условной вероятности, плотностью распределения $p(X^\ell, w; \gamma) = p(X^\ell|w)p(w; \gamma)$. Таким образом, приходим к принципу максимума совместного правдоподобия данных и модели:

$$L_\gamma\left(w, X^\ell\right) = \ln p(X^\ell, w; \gamma) = \sum_{i=1}^{\ell} \ln p(x_i, y_i|w) + \ln p(w; \gamma) \to \max_w. \quad (4.3)$$

Функционал L_γ распадается на два слагаемых: логарифм правдоподобия (4.2) и регуляризатор, не зависящий от данных. Второе слагаемое ограничивает вектор параметров модели, не позволяя ему быть каким угодно.

Нормальный регуляризатор. Пусть вектор $w \in \mathbb{R}^n$ имеет нормальное распределение, все его компоненты независимы и имеют равные дисперсии σ. Будем считать σ гиперпараметром. Логарифмируя, получаем квадратичный регуляризатор:

$$\ln p(w,\sigma) = \ln\left(\frac{1}{(2\pi\sigma)^{n/2}}\exp\left(-\frac{\|w\|^2}{2\sigma}\right)\right) = -\frac{1}{2\sigma}\|w\|^2 + \text{const}(w),$$

где $\text{const}(w)$ – слагаемое, не зависящее от w, которым можно пренебречь, поскольку оно не влияет на решение оптимизационной задачи (4.3). Минимизация регуляризованного эмпирического риска приводит в данном случае к выбору такого решения w, которое не слишком сильно отклоняется от нуля. В линейных классификаторах это позволяет избежать проблем мультиколлинеарности и переобучения.

Лапласовский регуляризатор. Пусть вектор $w \in \mathbb{R}^n$ имеет априорное распределение Лапласа, все его компоненты независимы и имеют равные дисперсии. Тогда

$$\ln p(w;C) = \ln\left(\frac{1}{(2C)^n}\exp\left(-\frac{\|w\|_1}{C}\right)\right) = -\frac{1}{C}\|w\|_1 + \text{const}(w),$$

$\|w\|_1 = \sum_{j=1}^{n}|w_j|$.

Распределение Лапласа имеет более острый пик и более тяжёлые «хвосты», по сравнению с нормальным распределением. Его дисперсия равна $2C^2$. Самое интересное и полезное для практики свойство этого регуляризатора заключается в том, что он приводит к отбору признаков. Это происходит из-за того, что априорная плотность не является гладкой в точке $w = 0$. Запишем оптимизационную задачу настройки вектора параметров w:

$$Q(w) = \sum_{i=1}^{\ell}\mathcal{L}_i(w) + \frac{1}{C}\sum_{j=1}^{n}|w_j| \to \min_w.$$

где $\mathcal{L}_i(w) = \mathcal{L}_i(y_i f(x_i,w))$ – некоторая ограниченная гладкая функция потерь. Сделаем замену переменных, чтобы функционал стал гладким. Каждой переменной w_j поставим в соответствие две новые неотрицательные переменные $u_j = \frac{1}{2}(|w_j| - w_j)$. Тогда $w_j = u_j - v_j$ и $|w_j| = v_j + u_j$. В новых переменных функционал становится гладким, но добавляются ограничения-неравенства:

$$Q(u,v) = \sum_{i=1}^{\ell}\mathcal{L}_i(u-v) + \frac{1}{C}\sum_{j=1}^{n}(v_j + u_j) \to \min_{u,v};$$

$$u_j \geq 0,\ v_j \geq 0,\ j = 1,\ldots,n.$$

Для любого j хотя бы одно из ограничений $u_j \geq 0$ и $v_j \geq 0$ является активным, то есть обращается в равенство, иначе второе слагаемое в $Q(u, v)$ можно было бы уменьшить, не изменив первое. Если гиперпараметр C устремить к нулю, то активными в какой-то момент станут все $2n$ ограничений. Постепенное уменьшение гиперпараметра C приводит к увеличению числа таких j, для которых оба ограничения активны, $u_j = v_j = 0$, откуда следует, что $w_j = 0$. В линейных моделях это означает, что значения j-го признака игнорируются, и его можно исключить из модели. Таким образом, для тех моделей, в которых обнуление коэффициента w_j означает исключение j-го признака, регуляризация Лапласа автоматически приводит к отбору признаков (features selection). Параметр C позволяет регулировать селективность метода. Чем меньше C, тем большее коэффициентов w_j окажутся нулевыми.

Независимые параметры с неравными дисперсиями. Возьмём гауссовскую модель априорного распределения $p(w), w \in \mathbb{R}_n$ с независимыми параметрами w_j, но теперь не будем предполагать, что дисперсии C_j параметров w_j одинаковы:

$$p(w) = \frac{1}{(2\pi)^{n/2}\sqrt{C_1 \dots C_n}} \exp\left(-\sum_{j=1}^{n} \frac{{w_j}^2}{2C_j}\right). \qquad (4.4)$$

Будем считать дисперсии C_j неизвестными и определять их наравне с самими параметрами w_j исходя из принципа максимума совместного правдоподобия:

$$Q(w) = \sum_{i=1}^{\ell} \mathcal{L}_i(w) + \frac{1}{C}\sum_{j=1}^{n}\left(\ln C_j + \frac{{w_j}^2}{C_j}\right) \to \min_{w,C}.$$

Результатом такой модификации также является отбор признаков. Если $C_j \to 0$, то параметр w_j стремится к нулю, и на практике часто достигает машинного нуля.

Если $C_j \to \infty$, то на параметр w_j фактически не накладывается никаких ограничений, и он может принимать любые значения.

4.2. Линейная модель классификации

Пусть X – пространство объектов; $Y = \{-1,1\}$ – множество допустимых ответов; объекты описываются n числовыми признаками $f_j: X \to \mathbb{R}, j = 1,\dots,n$. Вектор $x = (x_1,\dots,x_n) \in \mathbb{R}_n$, где $x_j = f_j(x)$, называется признаковым описанием объекта x.

Если дискриминантная функция определяется как скалярное произведение вектора x и вектора параметров $w \in \mathbb{R}_n$, то получается линейный классификатор:

$$a(x,w) = \text{sign}(\langle w,x\rangle - w_0) = \text{sign}\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right). \qquad (4.5)$$

Уравнение $\langle w,x\rangle = 0$ задаёт гиперплоскость, разделяющую классы в пространстве $\mathbb{R}_n$. Если вектор x находится по одну сторону гиперплоскости с её направляющим вектором w, то объект x относится к классу +1, иначе – к классу −1.

Параметр w_0 иногда опускают. Иногда полагают, что среди признаков есть константа, $f_j(x) \equiv -1$, и тогда роль свободного коэффициента w_0 играет параметр w_j.

Устройство нервной клетки и модель МакКаллока-Питтса. Линейный классификатор или персептрон является простейшей математической моделью нервной клетки – нейрона (рисунок 4.2). Нейрон имеет множество разветвлённых отростков – дендритов, и одно длинное тонкое волокно – аксон, на конце которого находятся синапсы, примыкающие к дендритам других нервных клеток. Нервная клетка может находиться в двух состояниях: обычном и возбуждённом. Клетка возбуждается, когда в ней накапливается достаточное количество положительных зарядов. В возбуждённом состоянии клетка генерирует электрический импульс величиной около 100 мВ и длительностью около 1 мс, который проходит по

аксону до синапсов. Синапс при приходе импульса выделяет вещество, способствующее проникновению положительных зарядов внутрь соседней клетки, примыкающей к данному синапсу. Синапсы имеют разную способность концентрировать это вещество, причём некоторые даже препятствуют его выделению – они называются тормозящими. После возбуждения клетки наступает период релаксации – некоторое время она не способна генерировать новые импульсы.

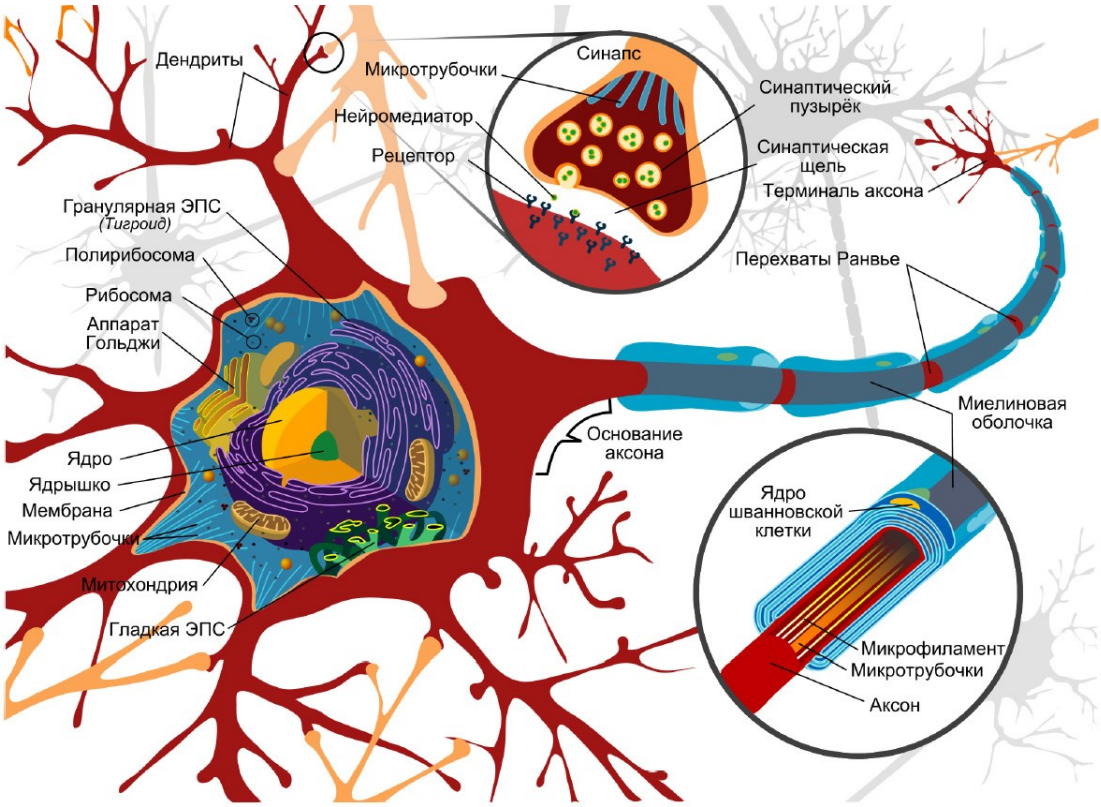


Рисунок 4.2 – Нервная клетка

Нервную клетку можно рассматривать как устройство, которое на каждом такте своей работы принимает заряды величиной $x_j = f_j(x)$ от n входов – синапсов, примыкающих к её дендритам. Поступающие заряды складываются с весами w_j.

Если вес w_j положительный, то j-й синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный заряд превышает порог активации w_0, то нейрон возбуждается и выдаёт на выходе +1, иначе выдаётся −1.

Функцию $\varphi(z) = \text{sign}(z)$, преобразующую значение суммарного импульса в выходное значение нейрона, называют функцией активации. В общем случае это не обязательно пороговая функция. Используют также ¾сглаженную¿ пороговую функцию – гиперболический тангенс $\varphi(z) = \text{th}(z)$ и другие (рисунок 4.3).

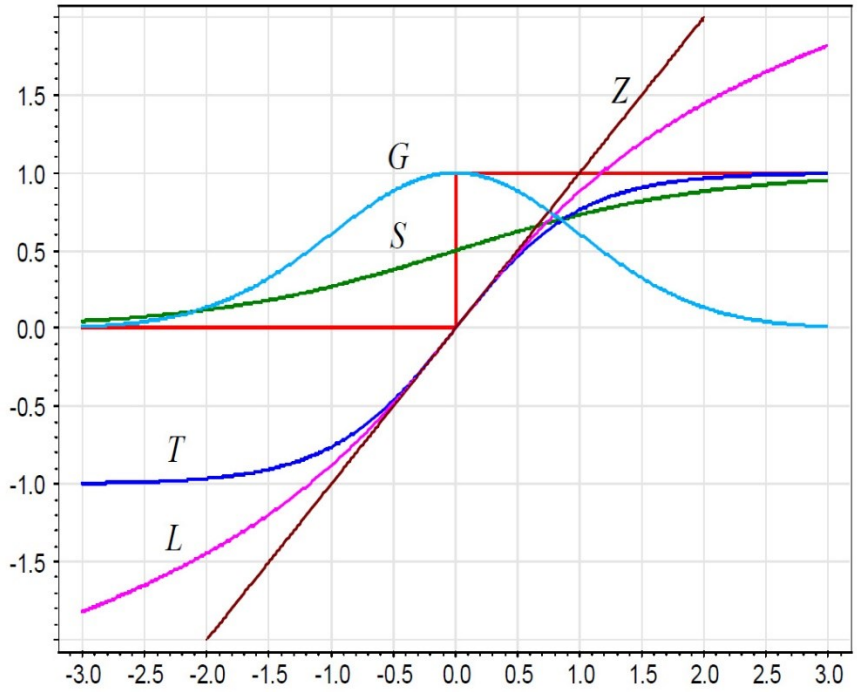


Рисунок 4.3 – Стандартные функции активации $\varphi(z)$:
$\theta(z) = [z \geq 0]$ – пороговая функция Хевисайда;
$S(z) = \sigma(z) = (1 + e^{-z})^{-1}$ – сигмоидная функция;
$T(z) = \text{th}(z) = 2\sigma(2z) - 1$ – гиперболический тангенс;
$L(z) = \ln(z + \sqrt{z^2 + 1})$ – логарифмическая функция;
$G(z) = \exp(-z^2/2)$ – гауссовская функция; $Z(z) = z$ – линейная функция.

Таким образом, линейный классификатор (4.5) является математической моделью нейрона. Эту модель предложили в 1943 году МакКаллок и Питтс.

Коннективизм и нейронные сети. Нервная система состоит из огромного числа связанных друг с другом нейронов, распространяющих направленные волны импульсов. Скорость распространения импульсов составляет приблизительно 100 м/с.

Человек способен решать сложные задачи распознавания и принятия решений за десятые доли секунды, откуда следует, что необходимые для этого нейровычисления выполняются не более чем за 102 последовательных тактов.

Кора головного мозга человека содержит порядка 1011 нейронов, и каждый нейрон имеет синаптические связи с 103–104 других нейронов. Нейровычисления выполняются с большой степенью параллелизма, и для принятия одного решения может быть задействовано огромное число нейронов.

Есть гипотеза, что, соединив большое число элементарных классификаторов (скажем, линейных, но обязательно через нелинейные функции активации), возможно создать универсальную машину, способную обучаться решению любых задач, подобно тому, как это делает человеческий мозг. На основе этой идеи, высказанной ещё в 50-е годы и названной принципом коннективизма, строятся искусственные нейронные сети.

На самом деле механизмы функционирования нервных клеток гораздо сложнее описанных выше. В нейрофизиологии известны десятки различных типов нейронов, и многие из них функционируют иначе. Однако в теории искусственных нейронных сетей не ставится задача максимально точного воспроизведения функций биологических нейронов. Цель в том, чтобы подсмотреть некоторые принципы в живой природе и использовать их для построения обучаемых устройств.

4.3. Метод стохастического градиента

Пусть задана обучающая выборка $X^\ell = \{(x_i, y_i)\}_{i=1}^{\ell}$, $x_i \in \mathbb{R}_n$, $yi \in \{-1,1\}$. Требуется найти вектор весов $w \in \mathbb{R}_n$, при котором достигается минимум аппроксимированного эмпирического риска:

$$Q\left(w, X^\ell\right) = \sum_{i=1}^{\ell} \mathcal{L}(\langle w, x_i\rangle, y_i) \to \min_w. \qquad (4.6)$$

Применим для минимизации $Q(w)$ метод градиентного спуска. В этом методе выбирается некоторое начальное приближение для вектора весов w, затем запускается итерационный процесс, на каждом шаге которого вектор w

изменяется в направлении наиболее быстрого убывания функционала Q. Это направление противоположно вектору градиента $Q'(w) = \left(\frac{\partial Q(w)}{\partial w_j}\right)_{j=1}^{n}$: $w := w - \eta Q'(x)$, где $\eta > 0$ – величина шага в направлении антиградиента, называемая также темпом обучения (learning rate). Предполагая, что функция потерь $\mathcal{L}$ дифференцируема, распишем градиент:

$$w := w - \eta \sum_{i=1}^{\ell} \mathcal{L}'(\langle w, x_i \rangle, y_i) x_i y_i. \tag{4.7}$$

Каждый прецедент (x_i, y_i) вносит аддитивный вклад в изменение вектора w, но вектор w изменяется только после перебора всех ℓ объектов. Сходимость итерационного процесса можно улучшить, если выбирать прецеденты (x_i, y_i) по одному, для каждого делать градиентный шаг и сразу обновлять вектор весов:

$$w := w - \eta \mathcal{L}'_a(\langle w, x_i \rangle, y_i) x_i y_i. \tag{4.8}$$

В методе стохастического градиента (stochastic gradient, SG) прецеденты перебираются в случайном порядке (рисунок 4.4). Если же объекты предъявлять в некотором фиксированном порядке, процесс может зациклиться или разойтись.

Вход:
X^ℓ — обучающая выборка; η — темп обучения; λ — параметр сглаживания.
Выход:
Синаптические веса $w_1, \ldots, w_n$;

1: инициализировать веса w_j, $j = 1, \ldots, n$;
2: инициализировать текущую оценку функционала:
$Q := \sum_{i=1}^{\ell} \mathscr{L}\left(\langle w, x_i \rangle y_i\right)$;
3: **повторять**
4: выбрать объект x_i из X^ℓ (например, случайным образом);
5: вычислить выходное значение алгоритма $a(x_i, w)$ и ошибку:
$\varepsilon_i := \mathscr{L}\left(\langle w, x_i \rangle y_i\right)$;
6: сделать шаг градиентного спуска:
$w := w - \eta \mathscr{L}'\left(\langle w, x_i \rangle y_i\right) x_i y_i$;
7: оценить значение функционала:
$Q := (1 - \lambda) Q + \lambda \varepsilon_i$;
8: **пока** значение Q не стабилизируется и/или веса w не перестанут изменяться;

Рисунок 4.4 – Алгоритм стохастического градиента

Инициализация весов может производиться различными способами. Стандартная рекомендация – взять небольшие случайные значения, $w_j := \text{random}(-\frac{1}{2n}, \frac{1}{2n})$. Иногда веса инициализируют нулём. Иногда берут оценки:

$$w_j := \frac{\langle y, f_j \rangle}{\langle f_j, f_j \rangle}, \qquad j = 1, \dots, n,$$

где $f_j = (f_j(x_i))_{i=1}^{\ell}$ – вектор значений j-го признака, $y = (y_i)_{i=1}^{\ell}$ – вектор ответов. Эти оценки являются точными в одном нереалистичном частном случае – когда функция потерь квадратична и признаки статистически независимы.

Критерий останова в алгоритме на рисунке 4.4 основан на приблизительной оценке функционала Q методом экспоненциальной скользящей средней. Вычисление точного значения по всем ℓ объектам слишком вычислительно трудоёмко. Когда градиентный метод подходит к окрестности минимума, оценка скользящего среднего стабилизируется и приближается к точному значению функционала. Параметр λ можно положить равным $1/\ell$. В случае избыточно длинной выборки его рекомендуется увеличивать.

Преимущества метода SG.

1. Метод легко реализуется и легко обобщается на нелинейные классификаторы и на нейронные сети – суперпозиции линейных классификаторов.

2. Метод подходит для динамического обучения, когда обучающие объекты поступают потоком, и вектор весов обновляется при появлении каждого объекта.

3. Метод позволяет настраивать веса на избыточно больших выборках, за счёт того, что случайной подвыборки может оказаться достаточно для обучения.

Недостатки метода SG.

1. Функционал Q, как правило, многоэкстремальный, и процесс может сходиться к локальному минимуму, сходиться очень медленно или не сходиться вовсе.

2. При большой размерности пространства n или малой длине выборки ℓ возможно переобучение. При этом резко возрастает норма вектора весов, появляются большие по абсолютной величине положительные и отрицательные веса, классификация становится неустойчивой – малые изменения обучающей выборки, начального приближения, порядка предъявления объектов или параметров алгоритма η, λ могут сильно изменить результирующий вектор весов, увеличивается вероятность ошибочной классификации новых объектов.

3. Если функция потерь имеет горизонтальные асимптоты, то процесс может попасть в состояние «паралича». Чем больше значение скалярного произведения $\langle w, x_i \rangle$, тем ближе значение производной $\mathcal{L}'$ к нулю, тем меньше приращение весов в (4.8). Если веса w_j попали в область больших значений, то у них практически не остаётся шансов выбраться из этой «мёртвой зоны».

Дополнительные материалы по теме «Метод стахостического градиента»: классические частные случаи, адаптивный линейный элемент, персептрон Розенблатта, эвристики для улучшения градиентных методов обучения, нормализация данных, порядок предъявления объектов, квадратичная регуляризация, выбор величины шага, выбивание из локальных минимумов, ранний останов.

4.4. Логистическая регрессия

Метод логистической регрессии основан на довольно сильных вероятностных предположениях, которые имеют сразу несколько интересных последствий. Во-первых, линейный алгоритм классификации оказывается оптимальным байесовским классификатором. Во-вторых, однозначно определяется функция потерь. В-третьих, возникает интересная

дополнительная возможность наряду с классификацией объекта получать численные оценки вероятности его принадлежности каждому из классов.

4.4.1. Обоснование логистической регрессии

В нормальном дискриминантном анализе доказывается, что если плотности классов нормальны и имеют равные матрицы ковариации, то оптимальный байесовский классификатор линеен. Возникает вопрос: а только ли в этом случае? Оказывается, нет – он остаётся линейным при менее жёстких предположениях.

Базовые предположения. Пусть классов два, $Y = \{-1, +1\}$, объекты описываются n числовыми признаками $f_j: X \to \mathbb{R}$, $j = 1, \ldots, n$. Будем полагать $X = \mathbb{R}_n$, отождествляя объекты с их признаковыми описаниями: $x \equiv (f_1(x), \ldots, f_n(x))$.

Гипотеза 4.1. Множество прецедентов $X \times Y$ является вероятностным пространством. Выборка прецедентов $X^\ell = \{(x_i, y_i)\}_{i=1}^\ell$ получена случайно и независимо согласно вероятностному распределению с плотностью $p(x, y) = P_y p_y(x) = \mathbb{P}(y|x)p(x)$, где P_y – априорные вероятности, $p_y(x)$ – функции правдоподобия, $\mathbb{P}(y|x)$ – апостериорные вероятности классов $y \in Y$.

Плотность распределения $p(x), x \in \mathbb{R}_n$ называется экспонентной, если $p(x) = \exp(c(\delta)\langle\theta, x\rangle + b(\delta, \theta) + d(x, \delta))$, где параметр $\theta \in \mathbb{R}_n$ называется сдвигом, параметр δ называется разбросом, b, c, d – произвольные числовые функции.

Класс экспонентных распределений очень широк. К нему относятся многие непрерывные и дискретные распределения: равномерное, нормальное, гипергеометрическое, пуассоновское, биномиальное, Г-распределение, и другие.

Гипотеза 4.2. Функции правдоподобия классов $p_y(x)$ принадлежат экспонентному семейству плотностей, имеют равные значения параметров d и δ, но отличаются значениями параметра сдвига θ_y.

Основная теорема. Оптимальный байесовский классификатор имеет вид $a(x) = \arg\max_{y \in Y} \lambda_y P(y|x)$, где λ_y – штраф за ошибку на объектах класса y.

В случае двух классов:

$$a(x) = \text{sign}\big(\lambda_+ \mathbb{P}(+1|x) - \lambda_- \mathbb{P}(-1|x)\big) = \text{sign}\left(\frac{\mathbb{P}(+1|x)}{\mathbb{P}(-1|x)} - \frac{\lambda_-}{\lambda_+}\right).$$

Теорема. Если справедливы гипотезы 4.1, 4.2, и среди признаков $f_1(x), \ldots, f_n(x)$ есть константа, то:

1) байесовский классификатор является линейным: $a(x) = \text{sign}(\langle w, x\rangle - w_0)$, где $w_0 = \ln(\lambda_-/\lambda_+)$, а вектор w не зависит от штрафов λ_-, λ_+;

2) апостериорная вероятность принадлежности произвольного объекта $x \in X$ к классу $y \in \{-1, +1\}$ может быть вычислена по значению дискриминантной функции: $P(y|x) = \sigma(\langle w, x\rangle y)$, где $\sigma(z) = \frac{1}{1+e^{-z}}$ – сигмоидная функция.

Доказательство. Рассмотрим отношение апостериорных вероятностей классов и воспользуемся тем, что $p_y(x)$ – экспонентные плотности с параметрами θ_y и δ:

$$\frac{\mathbb{P}(+1|x)}{\mathbb{P}(-1|x)} = \frac{\mathbb{P}_+ p_+(x)}{\mathbb{P}_- p_-(x)} = \exp\left(\langle \underbrace{(c_+(\delta)\theta_+ - c_-(\delta)\theta_-)}_{w=\text{const}(x)}, x\rangle + \underbrace{b_+(\delta,\theta_+) - b_-(\delta,\theta_-) + \ln\frac{\mathbb{P}_+}{\mathbb{P}_-}}_{\text{const}(x)}\right)$$

Здесь вектор w не зависит от x и является вектором свободных коэффициентов при признаках. Все слагаемые под экспонентой, не зависящие от x, можно считать аддитивной добавкой к коэффициенту при константном признаке. Поскольку свободные коэффициенты настраиваются по обучающей выборке, вычислять эту аддитивную добавку нет никакого смысла, и её можно включить в $\langle w, x\rangle$. Следовательно:

$\frac{\mathbb{P}(+1|x)}{\mathbb{P}(-1|x)} = e^{\langle w,x\rangle}$.

Используя формулу полной вероятности $\mathbb{P}(-1|x) + \mathbb{P}(+1|x) = 1$, нетрудно выразить апостериорные вероятности $\mathbb{P}(-1|x)$ и $\mathbb{P}(+1|x)$ через $\langle w, x\rangle$:

$\mathbb{P}(+1|x) = \sigma(+\langle w, x\rangle)$; $\mathbb{P}(-1|x) = \sigma(-\langle w, x\rangle)$.

О б ъ е д и н я я эти два равенства в одно, получаем требуемое: $\mathbb{P}(y|x) = \sigma(\langle w, x\rangle y)$.

Разделяющая поверхность в байесовском решающем правиле определяется уравнением $\lambda_-\mathbb{P}(-1|x) = \lambda_+\mathbb{P}(+1|x)$, которое равносильно $\langle w, x\rangle - \ln\frac{\lambda_-}{\lambda_+} = 0$, следовательно, разделяющая поверхность линейна.

4.4.2. Метод стохастического градиента для логистической регрессии

Принцип максимума правдоподобия. Для настройки вектора весов w по обучающей выборке X^ℓ будем максимизировать логарифм правдоподобия выборки:

$$L\left(w, X^\ell\right) = \log_2 \prod_{i=1}^{\ell} p(x_i, y_i) \to \max_w.$$

Согласно определению условной вероятности, $p(x, y) = \mathbb{P}(y|x)p(x)$, где плотности распределения объектов $p(x)$ не зависят от вектора параметров w. Апостериорные вероятности выражаются согласно рассмотренной в 4.4.1 теореме через линейную дискриминантную функцию: $\mathbb{P}(y|x) = \sigma(\langle w, x\rangle\, y)$. Таким образом,

$$L\left(w, X^\ell\right) = \sum_{i=1}^{\ell} \log_2 \sigma(\langle w, x_i\rangle\, y_i) + \text{const}(w) \to \max_w.$$

Максимизация правдоподобия $L\left(w, X^\ell\right)$ эквивалентна минимизации функционала $\tilde{Q}\left(w, X^\ell\right)$, гладко аппроксимирующего эмпирический риск (4.1):

$$\tilde{Q}(w, X^{\ell}) = \sum_{i=1}^{\ell} \log_2 1 + \exp(-\langle w, x_i \rangle\, y_i) \to \min_w. \qquad (4.9)$$

Таким образом, логистическая функция потерь $L(M) = \log_2(1 + e^{-M})$ является следствием экспонентности классов и принципа максимума правдоподобия.

Градиентный шаг. Запишем градиент функционала $\tilde{Q}(w)$, воспользовавшись выражением для производной сигмоидной функции $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$, и получим логистическое правило обновления весов для градиентного шага в методе стохастического градиента:

$$w := w + \eta y_i x_i \sigma(-\langle w, x_i \rangle\, y_i), \qquad (4.10)$$

где (x_i, y_i) – предъявляемый прецедент, η – темп обучения.

Аналогия с правилом Хэбба. Логистическая функция потерь является сглаженным вариантом кусочно-линейной функции потерь, соответствующей правилу Хэбба, рисунок. 4.5. Поэтому и логистическое правило (4.10) оказывается, в свою очередь, сглаженным вариантом правила Хэбба, рисунок. 4.6:

$$w := w + \eta y_i x_i [\langle w, x_i \rangle y_i < 0].$$

В правиле Хэбба смещение весов происходит только когда на объекте x_i допускается ошибка. В логистическом правиле смещение тем больше, чем меньше отступ $M_i(w) = \langle w, x_i \rangle y_i$, то есть чем серьёзнее ошибка. Даже если ошибки нет, но объект близок к границе классов, веса модифицируются так, чтобы граница прошла как можно дальше от объекта. Тем самым градиентная минимизация реализует стратегию увеличения зазора (margin) между обучающими объектами и разделяющей поверхностью, что способствует улучшению обобщающей способности.

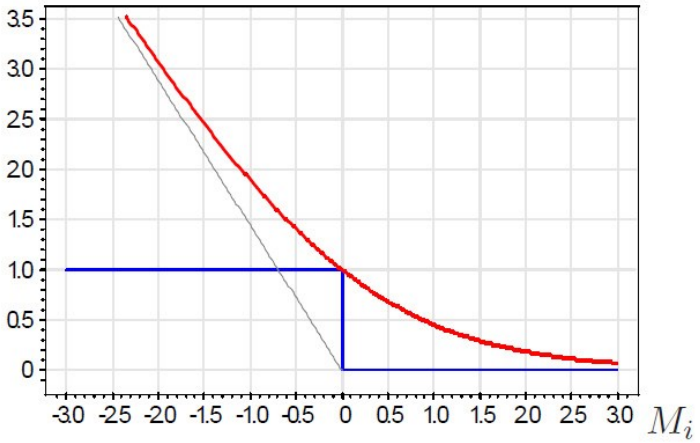


Рисунок 4.5 – Логарифмическая функция потерь $\log_2(1 + e^{-M_i})$ и её наклонная асимптота

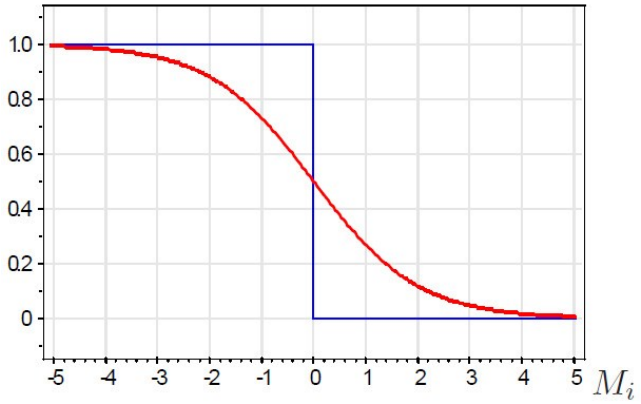


Рисунок 4.6 – Правило Хэбба: пороговое $[M_i < 0]$ и сглаженное $\sigma(-M_i)$

Методы второго порядка. Для оптимизации весов w чаще используется метод Ньютона-Рафсона. Он имеет более высокую скорость сходимости в окрестности локального оптимума, но требует решения линейной регрессионной задачи (следовательно, обращения ковариационной матрицы размера $n \times n$) на каждой итерации. Отсюда и название метода – логистическая регрессия. Это регрессия ещё и потому, что для классифицируемого объекта x оценивается вещественная величина – апостериорная вероятность $\mathbb{P}(+1|x)$.

Сравнение с линейным дискриминантом Фишера. Линейный дискриминант Фишера (ЛДФ) и логистическая регрессия исходят из байесовского решающего правила и принципа максимума правдоподобия, однако результат получается разный. В ЛДФ приходится оценивать $n|Y| +$

$n(n+1)/2$ параметров (векторы средних для каждого класса и общую ковариационную матрицу), в логистической регрессии – только n (вектор весов w). Это объясняется тем, что ЛДФ решает вспомогательную задачу восстановления плотностей распределения классов, предполагая, что плотности нормальны. Логистическая регрессия опирается на более слабые предположения о виде плотностей. С точки зрения философского принципа Оккама «не плодить сущности без необходимости» логистическая регрессия явно предпочтительнее, поскольку ЛДФ вводит избыточную сущность и сводит задачу классификации к более сложной задаче восстановления плотностей.

Достоинства логистической регрессии.

1. Как правило, логистическая регрессия даёт лучшие результаты по сравнению с линейным дискриминантом Фишера (поскольку она основана на менее жёстких гипотезах), а также по сравнению с дельта-правилом и правилом Хэбба (поскольку она использует «более правильную» функцию потерь).

2. Возможность оценивать апостериорные вероятности и риски.

Недостатки логистической регрессии.

1. Оценки вероятностей и рисков могут оказаться неадекватными, если не выполняются предположения основной теоремы из 4.4.1.

2. Градиентный метод обучения логистической регрессии наследует все недостатки метода стохастического градиента. Практичная реализация должна предусматривать стандартизацию данных, отсев выбросов, регуляризацию (сокращение весов), отбор признаков, и другие эвристики для улучшения сходимости.

Возможно применение метода второго порядка, но он требует обращения $n \times n$-матриц на каждом шаге и также не застрахован от плохой сходимости.

4.5. Метод опорных векторов

60–70-е годы коллективом советских математиков под руководством В.Н. Вапника был разработан метод обобщённого портрета, основанный на построении оптимальной разделяющей гиперплоскости. Требование оптимальности заключалось в том, что обучающие объекты должны быть удалены от разделяющей поверхности настолько далеко, насколько это возможно. На первый взгляд принцип оптимальности существенно отличается от методов минимизации эмпирического риска или максимизации правдоподобия, применяемых в других линейных классификаторах – персептроне, дискриминанте Фишера, логистической регрессии.

В 90-е годы метод получил широкую мировую известность и после некоторой переработки и серии обобщений стал называться машиной опорных векторов (support vector machine, SVM). В настоящее время он считается одним из лучших методов классификации.

Метод SVM обладает несколькими замечательными свойствами. Во-первых, обучение SVM сводится к задаче квадратичного программирования, имеющей единственное решение, которое вычисляется достаточно эффективно даже на выборках в сотни тысяч объектов. Во-вторых, решение обладает свойством разреженности: положение оптимальной разделяющей гиперплоскости зависит лишь от небольшой доли обучающих объектов. Они и называются опорными векторами; остальные объекты фактически не задействуются. Наконец, с помощью изящного математического приёма – введения функции ядра – метод обобщается на случай нелинейных разделяющих поверхностей. Вопрос о выборе ядра, оптимального для данной прикладной задачи, до сих пор остаётся открытой теоретической проблемой.

4.5.1. Линейно разделимая выборка

Рассмотрим задачу классификации на два непересекающихся класса, в которой объекты описываются n-мерными вещественными векторами: $X = \mathbb{R}^n$, $Y = \{-1, +1\}$.

Будем строить линейный пороговый классификатор:

$$a(x) = \text{sign}\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = \text{sign}(\langle w, x\rangle - w_0), \qquad (4.11)$$

где $x = (x^1, \dots, x^n)$ – признаковое описание объекта x; $w = (w_1, \dots, w_n) \in \mathbb{R}^n$ и скалярный порог $w_0 \in \mathbb{R}$ являются параметрами алгоритма. Уравнение $\langle w, x\rangle = w_0$ описывает гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^n$.

Предположим, что выборка $X^\ell = \{(x_i, y_i)\}_{i=1}^{\ell}$ линейно разделима и существуют значения параметров w, w_0, при которых функционал числа ошибок

$$Q(w, w_0) = \sum_{i=1}^{\ell} [y_i(\langle w, x_i\rangle - w_0) \le 0].$$

принимает нулевое значение. Но тогда разделяющая гиперплоскость не единственна. Можно выбрать другие её положения, реализующие такое же разбиение выборки на два класса. Идея метода заключается в том, чтобы разумным образом распорядиться этой свободой выбора.

Оптимальная разделяющая гиперплоскость. Потребуем, чтобы разделяющая гиперплоскость максимально далеко отстояла от ближайших к ней точек обоих классов. Первоначально данный принцип классификации возник из эвристических соображений: вполне естественно полагать, что максимизация зазора (margin) между классами должна способствовать более надёжной классификации. Позже этот принцип получил и теоретическое обоснование.

Нормировка. Заметим, что параметры линейного порогового классификатора определены с точностью до нормировки: алгоритм $a(x)$ не

изменится, если w и w_0 одновременно умножить на одну и ту же положительную константу. Удобно выбрать эту константу таким образом, чтобы выполнялось условие:

$$\min_{i=1,\dots,\ell} y_i(\langle w, x_i\rangle - w_0) = 1. \qquad (4.12)$$

Множество точек $\{x: -1 \le \langle w, x_i\rangle - w_0 \le 1\}$ описывает полосу, разделяющую классы, рисунок 4.7. Ни один из объектов обучающей выборки не попадает внутрь этой полосы.

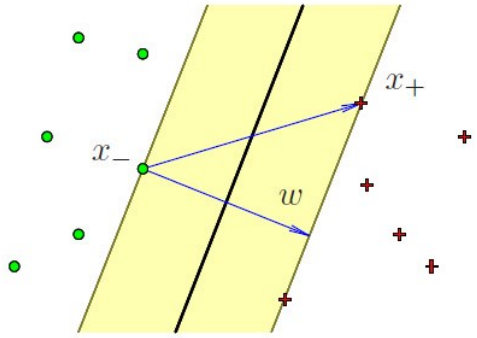


Рисунок 4.7 – Линейно разделимая выборка. Обучающие объекты x_- и x_+ находятся на границе разделяющей полосы. Вектор нормали w к разделяющей гиперплоскости определяет ширину полосы

Границами полосы служат две параллельные гиперплоскости с вектором нормали w. Разделяющая гиперплоскость проходит ровно по середине между ними. Объекты, ближайшие к разделяющей гиперплоскости, лежат на границах полосы, и именно на них достигается минимум (4.12). В каждом из классов имеется хотя бы один такой объект, в противном случае разделяющую полосу можно было бы ещё немного расширить и нарушался бы принцип максимального зазора.

Ширина разделяющей полосы. Чтобы разделяющая гиперплоскость как можно дальше отстояла от точек выборки, ширина полосы должна быть максимальной.

Пусть x_- и x_+ – два обучающих объекта классов −1 и +1 соответственно, лежащие на границе полосы. Тогда ширина полосы есть:

$$\langle (x_+ - x_-), \frac{w}{\|w\|}\rangle = \frac{\langle w, x_+\rangle - \langle w, x_-\rangle}{\|w\|} = \frac{(w_0 + 1) - (w_0 - 1)}{\|w\|} = \frac{2}{\|w\|}.$$

Ширина полосы максимальна, когда норма вектора w минимальна.

В случае линейно разделимой выборки получаем задачу квадратичного программирования: требуется найти значения параметров w и w_0, при которых выполняются ℓ ограничений-неравенств и норма вектора w минимальна:

$$\begin{cases} \langle w, w\rangle \to \min; \\ y_i(\langle w, x_i\rangle - w_0) \geq 1, \quad i = 1, \dots, \ell. \end{cases} \tag{4.13}$$

На практике линейно разделимые классы встречаются довольно редко. Поэтому постановку задачи (4.13) необходимо модифицировать так, чтобы система ограничений была совместна в любой ситуации.

4.5.2. Линейно неразделимая выборка

Чтобы обобщить постановку задачи на случай линейно неразделимой выборки, позволим алгоритму допускать ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше. Введём дополнительные переменные $\xi_i > 0$, характеризующие величину ошибки на объектах x_i, $i = 1, \dots, \ell$. Ослабим в (4.13) ограничения-неравенства и одновременно введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell} \xi_i \to \min_{w, w_0, \xi}; \\ y_i(\langle w, x_i\rangle - w_0) \geq 1 - \xi_i, \quad i = 1, \dots, \ell; \\ \xi_i \geq 0, \quad i = 1, \dots, \ell. \end{cases} \tag{4.14}$$

Положительная константа C является управляющим параметром метода и позволяет находить компромисс между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки.

Регуляризация эмпирического риска. В задачах с двумя классами $Y = \{-1, +1\}$ отступом (margin) объекта x_i от границы классов называется величина

$M_i(w, w_0) = y_i(\langle w, x_i\rangle - w_0)$.

Алгоритм (4.11) допускает ошибку на объекте x_i тогда и только тогда, когда отступ M_i отрицателен. Если $M_i \in (-1,+1)$, то объект x_i попадает внутрь разделяющей полосы. Если $M_i > 1$, то объект x_i классифицируется правильно, и находится на некотором удалении от разделяющей полосы.

Согласно (4.14) ошибка ξ_i выражается через отступ M_i. Действительно, из ограничений-неравенств следует, что $\xi_i > 0$ и $\xi_i > 1 - M_i$. В силу требования минимизации суммы $\sum_i \xi_i$ одно из этих неравенств обязательно должно обратиться в равенство. Следовательно, $\xi_i > (1 - M_i)_+$. Таким образом, задача (4.14) оказывается эквивалентной безусловной минимизации функционала Q, не зависящего от переменных ξ_i:

$$Q(w, w_0) = \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \frac{1}{2C}\|w\|^2 \to \min_{w,w_0}. \qquad (4.15)$$

В силу неравенства $[M_i < 0] \le (1 - M_i)_+$, рисунок 4.8, функционал (4.15) можно рассматривать как верхнюю оценку эмпирического риска (числа ошибочных классификаций объектов обучающей выборки), к которому добавлен регуляризатор $\|w\|^2$, умноженный на параметр регуляризации $\frac{1}{2C}$.

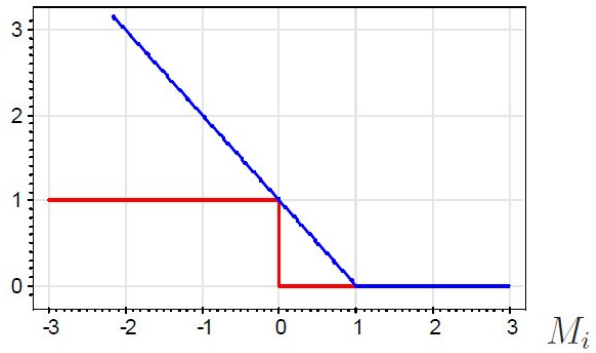


Рисунок 4.8 – Кусочно-линейная аппроксимация пороговой функции потерь: $[M_i < 0] \le (1 - M_i)_+$

Замена пороговой функции потерь $[M < 0]$ кусочно-линейной верхней оценкой $L(M) = (1 - M)_+$ делает функцию потерь чувствительной к величине ошибки. Функция потерь $L(M)$ штрафует объекты за приближение к границе классов.

Введение регуляризатора повышает устойчивость решения w. В случаях, когда минимум эмпирического риска достигается на множестве векторов w, регуляризация выбирает из них вектор с минимальной нормой. Тем самым устраняется проблема мультиколлинеарности, повышается устойчивость алгоритма, улучшается его обобщающая способность. Таким образом, принцип оптимальной разделяющей гиперплоскости или максимизации ширины разделяющей полосы тесно связан с регуляризацией некорректно поставленных задач по А.Н. Тихонову.

4.5.3. Ядра и спрямляющие пространства

Существует ещё один подход к решению проблемы линейной неразделимости. Это переход от исходного пространства признаковых описаний объектов X к новому пространству H с помощью некоторого преобразования $\psi: X \to H$. Если пространство H имеет достаточно высокую размерность, то можно надеяться, что в нём выборка окажется линейно разделимой (легко показать, что если выборка X^ℓ не противоречива, то всегда найдётся пространство размерности не более ℓ, в котором она будет линейно разделима). Пространство H называют спрямляющим.

Если предположить, что признаковыми описаниями объектов являются векторы $\psi(x_i)$, а не векторы x_i, то построение SVM проводится точно так же, как и ранее. Единственное отличие состоит в том, что скалярное произведение $\langle x, x' \rangle$ в пространстве X всюду заменяется скалярным произведением $\langle \psi(x), \psi(x') \rangle$ в пространстве H.

Отсюда вытекает естественное требование: пространство H должно быть наделено скалярным произведением, в частности, подойдёт любое евклидово, а в общем случае и гильбертово, пространство.

Преимущества SVM.

1. Задача квадратичного программирования имеет единственное решение, для нахождения которого разработаны достаточно эффективные методы.

2. Автоматически определяется сложность суперпозиции – число нейронов первого слоя, равное числу опорных векторов.

3. Максимизация зазора между классами улучшает обобщающую способность.

Недостатки SVM.

1. Неустойчивость к шуму в исходных данных. Объекты-выбросы являются опорными и существенно влияют на результат обучения.

2. До сих пор не разработаны общие методы подбора ядер под конкретную задачу. На практике «вполне разумные» ядра, построенные с учётом специфики задачи, могут и не обладать свойством положительной определённости.

3. Подбор параметра C требует многократного решения задачи.

4.6. ROC-кривая и оптимизация порога решающего правила

Рассмотрим задачу классификации на два класса, $Y = \{-1, +1\}$, и модель алгоритмов $a(x, w) = \text{sign}(f(x, w) - w_0)$, где $w_0 \in \mathbb{R}$ – аддитивный параметр дискриминантной функции. В теории нейронных сетей его называют порогом активации.

Согласно основной теореме в пп. 4.4.1 в случае линейной дискриминантной функции параметр w_0 определяется отношением потерь: $w_0 = \ln \frac{\lambda_-}{\lambda_+}$, где λ_+ и λ_- – величина потери при ошибке на объекте класса «+1» и «−1» соответственно.

На практике отношение потерь может многократно пересматриваться. Поэтому вводится специальная характеристика – ROC-кривая, которая

показывает, что происходит с числом ошибок обоих типов, если изменяется отношение потерь.

Термин операционная характеристика приёмника (receiver operating characteristic, ROC curve) пришёл из теории обработки сигналов. Эту характеристику впервые ввели во время Второй Мировой Войны, после поражения американского военного флота в Пёрл Харборе в 1941 году, когда была осознана проблема повышения точности распознавания самолётов противника по радиолокационному сигналу. Позже нашлись и другие применения: медицинская диагностика, приёмочный контроль качества, кредитный скоринг, предсказание лояльности клиентов, и т.д.

Каждая точка на ROC-кривой соответствует некоторому алгоритму. В общем случае это даже не обязательно кривая – дискретное множество алгоритмов может быть отображено в тех же координатах в виде точечного графика.

По оси X откладывается доля ошибочных положительных классификаций (false positive rate, FPR):

$$\mathrm{FPR}\left(a, X^{\ell}\right) = \frac{\sum_{i=1}^{\ell}[y_i = -1][a(x_i) = +1]}{\sum_{i=1}^{\ell}[y_i = -1]}.$$

Величина $1 - \mathrm{FPR}(a)$ равна доле правильных отрицательных классификаций (true negative rate, TNR) и называется специфичностью алгоритма a. Поэтому на горизонтальной оси иногда пишут «$1 -$ специфичность».

По оси Y откладывается доля правильных положительных классификаций (true positive rate, TPR), называемая также чувствительностью алгоритма a:

$$\mathrm{TPR}\left(a, X^{\ell}\right) = \frac{\sum_{i=1}^{\ell}[y_i = +1][a(x_i) = +1]}{\sum_{i=1}^{\ell}[y_i = +1]}.$$

Каждая точка ROC-кривой соответствует определённому значению параметра w_0. ROC-кривая монотонно не убывает и проходит из точки (0, 0) в точку (1, 1).

Для построения ROC-кривой нет необходимости вычислять FPR и TPR суммированием по всей выборке при каждом w_0. Более эффективный алгоритм построения ROC-кривой (рисунок 4.9) основан на простой идее, что в качестве значений порога w_0 достаточно перебрать только ℓ значений дискриминантной функции $f(x_i) = \langle w, x_i \rangle$, которые она принимает на объектах выборки.

Вход:
обучающая выборка X^ℓ; $f(x) = \langle w, x \rangle$ — дискриминантная функция;
Выход:
$\{(\mathrm{FPR}_i, \mathrm{TPR}_i)\}_{i=0}^{\ell}$ — последовательность точек ROC-кривой;
AUC — площадь под ROC-кривой.

$\ell_- := \sum_{i=1}^{\ell} [y_i = -1]$ — число объектов класса -1;
$\ell_+ := \sum_{i=1}^{\ell} [y_i = +1]$ — число объектов класса $+1$;
упорядочить выборку X^ℓ по убыванию значений $f(x_i)$;
поставить первую точку в начало координат:
$(\mathrm{FPR}_0, \mathrm{TPR}_0) := (0, 0)$; $\mathrm{AUC} := 0$;
для $i := 1, \ldots, \ell$
  если $y_i = -1$ **то**
    сместиться на один шаг вправо:
    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1} + \frac{1}{\ell_-}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1}$;
    $\mathrm{AUC} := \mathrm{AUC} + \frac{1}{\ell_-}\mathrm{TPR}_i$;
  иначе
    сместиться на один шаг вверх:
    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1} + \frac{1}{\ell_+}$;

Рисунок 4.9 – Эффективный алгоритм построения ROC-кривой

Чем выше проходит ROC-кривая, тем выше качество классификации. Идеальная ROC-кривая проходит через левый верхний угол – точку (0, 1). Наихудший алгоритм соответствует диагональной прямой, соединяющей точки (0, 0) и (1, 1); её также изображают на графике как ориентир.

В роли общей характеристики качества классификации, не зависящей от конъюнктурного параметра w_0, выступает площадь под ROC-кривой (area under curve, AUC). Её вычисление также показано в алгоритме (рисунок 4.9).

РАЗДЕЛ 5. МЕТОДЫ ВОССТАНОВЛЕНИЯ РЕГРЕССИИ

Задачу обучения по прецедентам при $Y = \mathbb{R}$ принято называть задачей восстановления регрессии. Основные обозначения остаются прежними. Задано пространство объектов X и множество возможных ответов Y. Существует неизвестная целевая зависимость $y^*: X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = \{(x_i, y_i)\}_{i=1}^{\ell}$. Требуется построить алгоритм, который в данной задаче принято называть «функцией регрессии» a: $X \to Y$, аппроксимирующий целевую зависимость y^*.

5.1. Метод наименьших квадратов

Пусть задана модель регрессии – параметрическое семейство функций $g(x, \alpha)$, где $\alpha \in \mathbb{R}^p$ – вектор параметров модели. Определим функционал качества аппроксимации целевой зависимости на выборке X^ℓ как сумму квадратов ошибок:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i)^2. \qquad (5.1)$$

Обучение по методу наименьших квадратов (МНК) состоит в том, чтобы найти вектор параметров α^*, при котором достигается минимум среднего квадрата ошибки на заданной обучающей выборке X^ℓ:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^p} Q(\alpha, X^\ell). \qquad (5.2)$$

Стандартный способ решения этой оптимизационной задачи – воспользоваться необходимым условием минимума. Если функция $g(x, \alpha)$ достаточное число раз дифференцируема по α, то в точке минимума выполняется система p уравнений относительно p неизвестных:

$$\frac{\partial Q}{\partial \alpha}\left(\alpha, X^{\ell}\right) = 2\sum_{i=1}^{\ell}(g(x_i,\alpha) - y_i)\frac{\partial g}{\partial \alpha}(x_i,\alpha) = 0. \qquad (5.3)$$

5.2. Непараметрическая регрессия: ядерное сглаживание

Непараметрическое восстановление регрессии основано на той же идее, что и непараметрическое восстановление плотности распределения, рассмотренное в 2.1.3. Значение $a(x)$ вычисляется для каждого объекта x по нескольким ближайшим к нему объектам обучающей выборки. Чтобы можно было говорить о «близости» объектов, на множестве X должна быть задана функция расстояния $\rho(x, x')$.

5.2.1. Формула Надарая-Ватсона

Возьмём самую простую модель регрессии, какая только возможна – константу $g(x, \alpha) = \alpha, \alpha \in \mathbb{R}$. Но при этом, чтобы не получить тривиального решения, введём веса объектов $w_i(x)$, зависящие от того объекта x, в котором мы собираемся вычислять значение $a(x) = g(x, \alpha)$. Можно сказать и так, что обучение регрессионной модели будет производиться отдельно в каждой точке x пространства объектов X.

Чтобы вычислить значение $a(x) = \alpha$ для произвольного $x \in X$, воспользуемся методом наименьших квадратов:

$$Q\left(\alpha; X^{\ell}\right) = \sum_{i=1}^{\ell} w_i(x)(\alpha - y_i)^2 \to \min_{\alpha\in\mathbb{R}}.$$

Зададим веса wi обучающих объектов так, чтобы они убывали по мере увеличения расстояния $\rho(x, x_i)$. Для этого введём невозрастающую, гладкую, ограниченную функцию K: $[0, \infty) \to [0, \infty)$, называемую ядром:

$$w_i(x) = K\left(\frac{\rho(x, x_i)}{h}\right).$$

Параметр h называется шириной ядра или шириной окна сглаживания. Чем меньше h, тем быстрее будут убывать веса $w_i(x)$ по мере удаления x_i от x. Приравняв нулю производную $\partial Q/\partial \alpha = 0$, получим формулу ядерного сглаживания Надарая-Ватсона:

$$a_h\left(x; X^\ell\right) = \frac{\sum_{i=1}^{\ell} y_i w_i(x)}{\sum_{i=1}^{\ell} w_i(x)} = \frac{\sum_{i=1}^{\ell} y_i K\left(\frac{\rho(x,x_i)}{h}\right)}{\sum_{i=1}^{\ell} K\left(\frac{\rho(x,x_i)}{h}\right)}. \qquad (5.4)$$

Эта формула интуитивно очевидна: значение $a(x)$ есть среднее y_i по объектам x_i, ближайшим к x.

В одномерном случае $X = \mathbb{R}^1$ метрика задаётся как $\rho(x, x_i) = |x - x_i|$.

5.2.2. Выбор ядра и ширины окна

Ядерное сглаживание – это довольно простой метод с точки зрения реализации. Обучение алгоритма $a_h\left(x; X^\ell\right)$ сводится к запоминанию выборки, подбору ядра K и ширины окна h.

Выбор ядра K мало влияет на точность аппроксимации, но определяющим образом влияет на степень гладкости функции $a_h(x)$. В одномерном случае функция $a_h(x)$ столько же раз дифференцируема, сколько и ядро $K(r)$. Для ядерного сглаживания чаще всего берут гауссовское ядро $K_G(r) = \exp\left(-\frac{1}{2}r^2\right)$ или квартическое $K_Q(r) = (1 - r^2)^2[|r| < 1]$.

Если ядро $K(r)$ финитно, то есть $K(r) = 0$ при $r > 1$, то ненулевые веса получат только те объекты xi, для которых $\rho(x, x_i) < h$. Тогда в формуле (5.4) достаточно суммировать только по ближайшим соседям объекта x. В одномерном случае $X = \mathbb{R}^1$ для эффективной реализации этой идеи выборка должна быть упорядочена по возрастанию x_i. В общем случае необходима специальная структура данных, позволяющая быстро находить множество ближайших соседей для любого объекта x.

Выбор ширины окна h решающим образом влияет на качество восстановления зависимости. При слишком узком окне ($h \to 0$) функция $a_h(x)$ стремится пройти через все точки выборки, реагируя на шум и претерпевая резкие скачки. При слишком широком окне функция чрезмерно сглаживается и в пределе $h \to \infty$ вырождается в константу. Таким образом, должно существовать оптимальное значение ширины окна h^* – компромисс между точностью описания выборки и гладкостью аппроксимирующей функции.

Оптимизация ширины окна. Чтобы оценить при данном h или k точность локальной аппроксимации в точке x_i, саму эту точку необходимо исключить из обучающей выборки. Если этого не делать, минимум ошибки будет достигаться при $h \to 0$.

Такой способ оценивания называется скользящим контролем с исключением объектов по одному (leave-one-out, LOO):

$$\mathrm{LOO}\left(h, X^\ell\right) = \sum_{i=1}^{\ell} \left(a_h(x_i; X^\ell \backslash \{x_i\}) - y_i\right)^2 \to \min_h,$$

где минимизация осуществляется по ширине окна h или по числу соседей k.

5.2.3. Проблема выбросов: робастная непараметрическая регрессия

Оценка Надарайя-Ватсона крайне чувствительна к большим одиночным выбросам. Идея обнаружения выбросов заключается в том, что чем больше величина ошибки $\varepsilon_i = \left|a_h(x_i; X^\ell \backslash \{x_i\}) - y_i\right|$, тем в большей степени прецедент (x_i, y_i) является выбросом, и тем меньше должен быть его вес. Эти соображения приводят к идее домножить веса $w_i(x)$ на коэффициенты $\gamma_i = \widetilde{K}(\varepsilon_i)$, где $\widetilde{K}$ – ещё одно ядро, вообще г о в о р я , отличное от $K(r)$.

Коэффициенты γ_i, как и ошибки ε_i, зависят от функции ah, которая, в свою очередь, зависит от γ_i. Разумеется, это не «порочный круг», а хороший повод для организации итерационного процесса (рисунок 5.1). На каждой итерации строится функция a_h, затем уточняются весовые множители γ_i. Как

правило, этот процесс сходится довольно быстро. Он называется локально взвешенным сглаживанием (locally weighted scatter plot smoothing, LOWESS).

Методы восстановления регрессии, устойчивые к шуму в исходных данных, называют робастными, что означает «разумный, здравый» (robust).

Вход:
X^ℓ — обучающая выборка;
Выход:
коэффициенты $\gamma_i,\ i = 1, \ldots, \ell$;

1: инициализация: $\gamma_i := 1,\ i = 1, \ldots, \ell$;
2: **повторять**
3: вычислить оценки скользящего контроля на каждом объекте:

$$a_i := a_h\big(x_i; X^\ell \backslash \{x_i\}\big) = \frac{\sum\limits_{j=1,\, j\neq i}^{\ell} y_j \gamma_j K\big(\frac{\rho(x_i,x_j)}{h(x_i)}\big)}{\sum\limits_{j=1,\, j\neq i}^{\ell} \gamma_j K\big(\frac{\rho(x_i,x_j)}{h(x_i)}\big)}, \quad i = 1, \ldots, \ell$$

4: вычислить коэффициенты γ_i:
$\gamma_i := \tilde{K}\big(|a_i - y_i|\big); \quad i = 1, \ldots, \ell;$
5: **пока** коэффициенты γ_i не стабилизируются;

Рисунок 5.1 – LOWESS – локально взвешенное сглаживание

Возможны различные варианты задания ядра $\widetilde{K}(\varepsilon)$. Жёсткая фильтрация: строится вариационный ряд ошибок $\varepsilon^{(1)} \leq \cdots \leq \varepsilon^{(\ell)}$, и отбрасывается некоторое количество t объектов с наибольшей ошибкой. Это соответствует ядру $\widetilde{K}(\varepsilon) = \left[\varepsilon \leq \varepsilon^{(\ell-t)}\right]$.

Мягкая фильтрация: используется квартическое ядро $\widetilde{K}(\varepsilon) = K_Q(\frac{\varepsilon}{6\ \mathrm{med}\{\varepsilon_i\}})$, где $\mathrm{med}\{\varepsilon_i\}$ – медиана вариационного ряда ошибок.

5.2.4. Проблема краевых эффектов

В одномерном случае $X = \mathbb{R}^1$ часто наблюдается значительное смещение аппроксимирующей функции $a_h(x)$ от истинной зависимости $y^*(x)$ вблизи минимальных и максимальных значений x_i. Смещение возникает, когда объекты выборки x_i располагаются только по одну сторону (а не вокруг) объекта x. Чем больше размерность пространства объектов, тем чаще возникает такая ситуация.

Для решения этой проблемы зависимость аппроксимируется в окрестности точки $x \in X$ не константой $a(u) = \alpha$, а линейной функцией $a(u) = \alpha(u - x) + \beta$.

Введём для краткости сокращённые обозначения $w_i = w_i(x)$, d_i=x_i−x и запишем задачу наименьших квадратов:

$$Q(\alpha, \beta; X^\ell) = \sum_{i=1}^{\ell} w_i(\alpha d_i + \beta - y_i)^2 \to \min_{\alpha,\beta \in \mathbb{R}}.$$

Приравнивая нулю производные $\frac{\partial Q}{\partial \alpha} = 0$ и $\frac{\partial Q}{\partial \beta} = 0$, получим систему линейных уравнений 2×2, решение которой даёт аналог формулы Надарая-Ватсона:

$$a_h(x; X^\ell) = \frac{\sum_{i=1}^{\ell} w_i d_i^2 \sum_{i=1}^{\ell} w_i y_i - \sum_{i=1}^{\ell} w_i d_i \sum_{i=1}^{\ell} w_i d_i y_i}{\sum_{i=1}^{\ell} w_i \sum_{i=1}^{\ell} w_i d_i^2 - \left(\sum_{i=1}^{\ell} w_i d_i\right)^2}.$$

В многомерном случае $X = \mathbb{R}^n$ для вычисления коэффициентов в линейной форме $a(u) = \alpha^{\mathrm{T}}(u - x) + \beta$ приходится решать задачу многомерной линейной регрессии. Причём она должна решаться заново для каждой точки $x \in X$, что сопряжено с большим объёмом вычислений.

5.3. Линейная регрессия

Пусть каждому объекту соответствует его признаковое описание $(f_1(x), \ldots, f_n(x))$, где $f_j: X \to \mathbb{R}$ – числовые признаки, $j = 1, \ldots, n$. Линейной моделью регрессии называется линейная комбинация признаков с коэффициентами $\alpha \in \mathbb{R}^n$:

$g(x, \alpha) = \sum_{j=1}^{n} \alpha_j f_j(x)$.

Введём матричные обозначения: $F = (f_j(x_i))_{\ell \times n}$ – матрица объекты-признаки; $y = (y_i)_{\ell \times 1}$ – целевой вектор; $\alpha = (\alpha_j)_{n \times 1}$ – вектор параметров.

В матричных обозначениях функционал Q принимает вид:

$Q(\alpha) = \|F\alpha - y\|^2$.

Запишем необходимое условие минимума (5.3) в матричном виде:

$\frac{\partial Q}{\partial \alpha}(\alpha) = 2F^{\mathrm{T}}(F\alpha - y) = 0.$

откуда следует $F^{\mathrm{T}}F\alpha = F^{\mathrm{T}}y$. Эта система линейных уравнений относительно α называется нормальной системой для задачи наименьших квадратов. Если матрица $F^{\mathrm{T}}F$ размера $n \times n$ невырождена, то решением нормальной системы является вектор:

$\alpha^* = (F^{\mathrm{T}}F)^{-1}F^{\mathrm{T}}y = F^{+}y.$

Матрица $F^{+} = (F^{\mathrm{T}}F)^{-1}F^{\mathrm{T}}$ называется псевдообратной для прямоугольной матрицы F. Подставляя найденное решение в исходный функционал, получаем:

$Q(\alpha^*) = \|P_F y - y\|^2,$

где $P_F = FF^{+} = (F^{\mathrm{T}}F)^{-1}F^{\mathrm{T}}$ – проекционная матрица.

Решение имеет простую геометрическую интерпретацию. Произведение $P_F y$ есть проекция целевого вектора y на линейную оболочку столбцов матрицы F. Разность $(P_F y - y)$ есть проекция целевого вектора y на ортогональное дополнение этой линейной оболочки. Значение функционала $Q(\alpha^*) = \|P_F y - y\|^2$ есть квадрат длины перпендикуляра, опущенного из y на линейную оболочку. Таким образом, МНК находит кратчайшее расстояние от y до линейной оболочки столбцов F.

Известно большое количество численных методов решения нормальной системы. Наибольшей популярностью пользуются методы, основанные на ортогональных разложениях матрицы F. Эти методы эффективны, обладают хорошей численной устойчивостью и позволяют строить различные модификации и обобщения.

5.3.1. Сингулярное разложение

Произвольную $\ell \times n$-матрицу ранга n можно представить в виде сингулярного разложения (singular value decomposition, SVD):

$F = VDU^{\mathrm{T}},$

обладающего рядом замечательных свойств:

1) $n \times n$-матрица D диагональна, $D = \mathrm{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$, где $\lambda_1, \ldots, \lambda_n$ – общие ненулевые собственные значения матриц $F^{\mathrm{T}}F$ и FF^{T}.

2) $\ell \times n$-матрица $V = (v_1, \ldots, v_n)$ ортогональна, $V^{\mathrm{T}}V = I_n$, столбцы v_j являются собственными векторами матрицы FF^{T}, соответствующими $\lambda_1, \ldots, \lambda_n$;

3) $n \times n$-матрица $U = (u_1, \ldots, u_n)$ ортогональна, $U^{\mathrm{T}}U = I_n$, столбцы u_j являются собственными векторами матрицы $F^{\mathrm{T}}F$, соответствующими $\lambda_1, \ldots, \lambda_n$.

Имея сингулярное разложение, легко записать псевдообратную матрицу:

$$F^{+} = (UDV^{\mathrm{T}}VDU^{\mathrm{T}})^{-1}UDV^{\mathrm{T}} = UD^{-1}V^{\mathrm{T}} = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j v_j^{\mathrm{T}};$$

вектор МНК-решения:

$$\alpha^{*} = F^{+}y = UD^{-1}V^{\mathrm{T}} = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j (v_j^{\mathrm{T}} y); \qquad (5.5)$$

вектор $F\alpha^{*}$ – МНК-аппроксимацию целевого вектора y:

$$F\alpha^{*} = P_F y = (VDU^{\mathrm{T}})UD^{-1}V^{\mathrm{T}}y = VV^{\mathrm{T}}y = \sum_{j=1}^{n} v_j (v_j^{\mathrm{T}} y); \qquad (5.6)$$

и норму вектора коэффициентов:

$$\|\alpha^{*}\|^2 = y^{\mathrm{T}}VD^{-1}U^{\mathrm{T}}UD^{-1}V^{\mathrm{T}}y = y^{\mathrm{T}}VD^{-2}V^{\mathrm{T}}y = \sum_{j=1}^{n} \frac{1}{\lambda_j} (v_j^{\mathrm{T}} y)^2; \qquad (5.7)$$

Итак, если есть сингулярное разложение, то обращать матрицы уже не нужно. Однако вычисление сингулярного разложения практически столь же трудоёмко, как и обращение. Эффективные численные алгоритмы, вычисляющие SVD, реализованы во многих стандартных математических пакетах.

5.3.2. Проблема мультиколлинеарности

Если ковариационная матрица $\Sigma = F^{\mathrm{T}}F$ имеет неполный ранг, то её обращение невозможно. Тогда приходится отбрасывать линейно зависимые признаки или применять описанные ниже методы – регуляризацию или метод главных компонент.

На практике чаще встречается проблема мультиколлинеарности – когда матрица Σ имеет полный ранг, но близка к некоторой матрице неполного ранга. Тогда говорят, что Σ – матрица неполного псевдоранга или что она плохо обусловлена. Геометрически это означает, что объекты выборки сосредоточены вблизи линейного подпространства меньшей размерности $m < n$. Признаком мультиколлинеарности является наличие у матрицы Σ собственных значений, близких к нулю.

Число обусловленности матрицы Σ есть:

$$\mu(\Sigma) = \|\Sigma\|\|\Sigma^{-1}\| = \frac{\max\limits_{u:\,\|u\|=1}\|\Sigma u\|}{\min\limits_{u:\,\|u\|=1}\|\Sigma u\|} = \frac{\lambda_{\max}}{\lambda_{\min}},$$

где $\lambda_{\max}$ и $\lambda_{\min}$ – максимальное и минимальное собственные значения матрицы Σ, все нормы евклидовы. Матрица считается плохо обусловленной, если $\mu(\Sigma) \gtrsim 10^2 \ldots 10^2$. Обращение такой матрицы численно неустойчиво. При умножении обратной матрицы на вектор, $z = \Sigma^{-1}u$, относительная погрешность усиливается в $\mu(\Sigma)$ раз:

$$\frac{\|\delta z\|}{\|z\|} \leq \mu(\Sigma)\frac{\|\delta u\|}{\|u\|}.$$

Именно это и происходит с МНК-решением в случае плохой обусловленности. В формуле (5.7) близкие к нулю собственные значения оказываются в знаменателе, в результате увеличивается разброс коэффициентов α^*, появляются большие по абсолютной величине положительные и отрицательные коэффициенты. МНК-решение становится неустойчивым – малые погрешности измерения признаков или ответов у обучающих объектов могут существенно повлиять на вектор решения α^*, а погрешности измерения признаков у тестового объекта x – на значения

функции регрессии $g(x, \alpha^*)$. Мультиколлинеарность влечёт не только неустойчивость и переобучение, но и неинтерпретируемость коэффициентов, так как по абсолютной величине коэффициента αj становится невозможно судить о степени важности признака f_j.

5.3.3. Гребневая регрессия

Для решения проблемы мультиколлинеарности добавим к функционалу Q регуляризатор, штрафующий большие значения нормы вектора весов $\|\alpha\|$:

$Q_\tau(\alpha) = \|F\alpha - y\|^2 + \tau\|\alpha\|^2$.

где τ – неотрицательный параметр. В случае мультиколлинеарности имеется бесконечно много векторов α, доставляющих функционалу Q значения, близкие к минимальному. Штрафное слагаемое выполняет роль регуляризатора, благодаря которому среди них выбирается решение с минимальной нормой. Приравнивая нулю производную $Q_\tau(\alpha)$ по параметру α, находим:

$\alpha_\tau^* = (F^{\mathrm{T}}F + \tau I_n)^{-1}F^{\mathrm{T}}y$.9

Таким образом, перед обращением матрицы к ней добавляется «гребень» – диагональная матрица τI_n. Отсюда и название метода – гребневая регрессия (ridge regression). При этом все её собственные значения увеличиваются на τ, а собственные векторы не изменяются. В результате матрица становится хорошо обусловленной, оставаясь в то же время «похожей» на исходную. Аналогичный приём мы уже упоминали в связи с обращением ковариационной матрицы – в линейном дискриминанте Фишера.

Выразим регуляризованное МНК-решение через сингулярное разложение:

$$\alpha_\tau^* = (UD^2U^{\mathrm{T}} + \tau I_n)^{-1}UDV^{\mathrm{T}}y = U(D^2 + \tau I_n)^{-1}DV^{\mathrm{T}}y = \sum_{j=1}^{n}\frac{\sqrt{\lambda_j}}{\lambda_j + \tau}u_j(v_j^{\mathrm{T}}y).$$

Теперь найдём регуляризованную МНК-аппроксимацию целевого вектора y:

$$F\alpha_\tau^* = VDU^{\mathrm{T}}\alpha_\tau^* = V\ \mathrm{diag}\left(\frac{\lambda_j}{\lambda_j+\tau}\right)V^{\mathrm{T}}y = \sum_{j=1}^{n}\frac{\lambda_j}{\lambda_j+\tau}v_j(v_j^{\mathrm{T}}y)^2. \qquad (5.8)$$

Как и прежде в (5.6), МНК-аппроксимация представляется в виде разложения целевого вектора y по базису собственных векторов матрицы $F^{\mathrm{T}}F$. Только теперь проекции на собственные векторы сокращаются, умножаясь на $\frac{\lambda_j}{\lambda_j+\tau} \in (0,1)$. В сравнении с (5.7) уменьшается и норма вектора коэффициентов:

$$\|\alpha_\tau^*\|^2 = \left\|(D^2+\tau I_n)^{-1}D^{-1}V^{\mathrm{T}}y\right\|^2 = \sum_{j=1}^{n}\frac{1}{\lambda_j+\tau}(v_j^{\mathrm{T}}y)^2 < \sum_{j=1}^{n}\frac{1}{\lambda_j}(v_j^{\mathrm{T}}y)^2 = \|\alpha^*\|^2$$

Отсюда ещё одно название метода – сжатие (shrinkage) или сокращение весов (weight decay).

Дополнительные темы по проблемам линейной регрессии

Понятие эффективной размерности. Проблема выбора константы регуляризации. Лассо Тибширани. Сравнение лассо и гребневой регрессии. Линейная монотонная регрессия.

5.4. Метод главных компонент

Ещё одно решение проблемы мультиколлинеарности заключается в том, чтобы подвергнуть исходные признаки некоторому функциональному преобразованию, гарантировав линейную независимость новых признаков, и, возможно, сократив их количество, то есть уменьшив размерность задачи.

В методе главных компонент (principal component analysis, PCA) строится минимальное число новых признаков, по которым исходные признаки восстанавливаются линейным преобразованием с минимальными погрешностями. PCA относится к методам обучения без учителя (unsupervised learning), поскольку матрица «объекты-признаки» F преобразуется без учёта целевого вектора y.

Важно отметить, что PCA подходит и для регрессии, и для классификации, и для многих других типов задач анализа данных, как вспомогательное преобразование, позволяющее определить эффективную размерность исходных данных.

Постановка задачи. Пусть имеется n числовых признаков $f_j(x), j = 1,\dots,n$. Как обычно, будем отождествлять объекты обучающей выборки и их признаковые описания: $x_i \equiv (f_1(x_i),\dots,f_n(x_i)),\ \ i = 1,\dots,\ell$. Рассмотрим матрицу F, строки которой соответствуют признаковым описаниям обучающих объектов:

$$F_{\ell\times n} = \begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_\ell) & \cdots & f_n(x_\ell) \end{pmatrix} = \begin{pmatrix} x_1 \\ \cdots \\ x_\ell \end{pmatrix}.$$

Обозначим через $z_i = (g_1(x_i),\dots,g_m(x_i))$ признаковые описания тех же объектов в новом пространстве $Z = \mathbb{R}^m$ меньшей размерности, $m < n$:

$$G_{\ell\times m} = \begin{pmatrix} g_1(x_1) & \cdots & g_m(x_1) \\ \vdots & \ddots & \vdots \\ g_1(x_\ell) & \cdots & g_m(x_\ell) \end{pmatrix} = \begin{pmatrix} z_1 \\ \cdots \\ z_\ell \end{pmatrix}.$$

Потребуем, чтобы исходные признаковые описания можно было восстановить по новым описаниям с помощью некоторого линейного преобразования, определяемого матрицей $U = (u_{js})_{n\times m}$:

$\hat{f}_j(x) = \sum_{s=1}^{m} g_s(x)u_{js}\,,\ \ j = 1,\dots,n,\ \ \ x \in X,$

или в векторной записи: $\hat{x} = zU^{\mathrm{T}}$. Восстановленное описание $\hat{x}$ не обязано в точности совпадать с исходным описанием x, но их отличие на объектах обучающей выборки должно быть как можно меньше при выбранной размерности m. Будем искать одновременно и матрицу новых признаковых описаний G, и матрицу линейного преобразования U, при которых суммарная невязка восстановленных описаний минимальна:

$$\Delta^2(G,U) = \sum_{i=1}^{\ell}\|\hat{x}_i - x_i\|^2 = \sum_{i=1}^{\ell}\left\|z_iU^{\mathrm{T}} - x_i\right\|^2 = \left\|GU^{\mathrm{T}} - F\right\|^2 \to \min_{G,U}, \tag{5.9}$$

где все нормы евклидовы. Напомним, что $\|A\|^2 = \mathrm{tr}\, AA^{\mathrm{T}}$, где tr – операция следа матрицы.

Будем предполагать, что матрицы G и U невырождены: $\mathrm{rk}\, G = \mathrm{rk}\, U = m$. Иначе существовало бы представление $\bar{G}\bar{U}^{\mathrm{T}} = GU^{\mathrm{T}}$ с числом столбцов в матрице $\bar{G}$, меньшим m. Поэтому интересны лишь случаи, когда $m \leq \mathrm{rk}\, F$.

Визуализация многомерных данных. Метод главных компонент часто используется для представления многомерной выборки данных на двумерном графике. Для этого полагают $m = 2$ и полученные пары значений $(g_1(x_i), g_2(x_i))$, $i = 1, \ldots, \ell$, наносят как точки на график. Проекция на главные компоненты является наименее искаженной из всех линейных проекций многомерной выборки на какую-либо пару осей. Как правило, в осях главных компонент удаётся увидеть наиболее существенные особенности исходных данных, даже несмотря на неизбежные искажения.

В частности, можно судить о наличии кластерных структур и выбросов. Две оси g_1 и g_2 отражают «две основные тенденции» в данных. Иногда их удаётся интерпретировать, если внимательно изучить, какие точки на графике являются «самыми левыми», «самыми правыми», «самыми верхними» и «самыми нижними». Этот вид анализа не позволяет делать точные количественные выводы и обычно используется с целью понимания данных. Аналогичную роль играют многомерное шкалирование и карты Кохонена.

Дополнительные темы по методу главных компонент

Эффективная размерность. Преобразование Карунена-Лоэва.

5.5. Нелинейные методы восстановления регрессии

Предположение о том, что модель регрессии линейна по параметрам, удобно для построения численных методов, но не всегда хорошо согласуется со знаниями о предметной области. В этом параграфе рассматриваются

случаи, когда модель регрессии нелинейна по параметрам, когда в линейную модель добавляются нелинейные преобразования исходных признаков или целевого признака, а также когда вводится неквадратичная функция потерь.

Общая идея во всех этих случаях одна: нелинейная задача сводится к решению последовательности более простых линейных задач.

5.5.1. Нелинейная модель регрессии

Пусть задана нелинейная модель регрессии $f(x, \alpha)$ и требуется минимизировать функционал качества по вектору параметров $\alpha \in \mathbb{R}^p$:

$$Q\left(\alpha; X^\ell\right) = \sum_{i=1}^{\ell} (f(x_i, \alpha) - y_i)^2.$$

Для выполнения численной минимизации функционала Q воспользуемся методом Ньютона-Рафсона. Выберем начальное приближение $\alpha^0 = (\alpha_1^0, \dots, \alpha_p^0)$ и организуем итерационный процесс:

$$\alpha^{t+1} := \alpha^t - h_t\left(Q''(\alpha^t)\right)^{-1} Q'(\alpha^t),$$

где $Q'(\alpha^t)$ – градиент функционала Q в точке α^t, $Q''(\alpha^t)$ – гессиан (матрица вторых производных) функционала Q в точке α^t, h_t – величина шага, который можно регулировать, а в простейшем варианте просто полагать равным единице.

Запишем компоненты градиента:

$$\frac{\partial}{\partial \alpha_j} Q(\alpha) = 2 \sum_{i=1}^{\ell} \left(f(x_i, \alpha) - y_i\right) \frac{\partial f}{\partial \alpha_j}(x_i, \alpha).$$

Запишем компоненты гессиана:

$$\frac{\partial^2}{\partial \alpha_j \partial \alpha_k} Q(\alpha) = 2 \sum_{i=1}^{\ell} \frac{\partial f}{\partial \alpha_j}(x_i, \alpha) \frac{\partial f}{\partial \alpha_k}(x_i, \alpha) - \underbrace{2 \sum_{i=1}^{\ell} \left(f(x_i, \alpha) - y_i\right) \frac{\partial^2 f}{\partial \alpha_j \partial \alpha_k}(x_i, \alpha)}_{\text{при линеаризации полагается равным } 0}.$$

Поскольку функция f задана, градиент и гессиан легко вычисляются численно. Основная сложность метода Ньютона-Рафсона заключается в обращении гессиана на каждой итерации.

Более эффективной с вычислительной точки зрения является следующая модификация этого метода. Если функция f достаточно гладкая (дважды непрерывно дифференцируема), то её можно линеаризовать в окрестности текущего значения вектора коэффициентов α^t:

$$f(x_i,\alpha) = f(x_i,\alpha^t) + \sum_{j=1}^{p} \frac{\partial f}{\partial \alpha_j}(x_i,\alpha_j)\left(\alpha_j - \alpha_j^t\right).$$

Заменим в гессиане функцию f на её линеаризацию. Это всё равно, что положить второе слагаемое в гессиане равным нулю. Тогда не нужно будет вычислять вторые производные $\frac{\partial^2 f}{\partial \alpha_j \partial \alpha_k}(x_i,\alpha)$. Этот метод называют методом Ньютона-Гаусса. В остальном он ничем не отличается от метода Ньютона-Рафсона. Введём матричные обозначения: $F_t = \left(\frac{\partial f}{\partial \alpha_j}(x_i,\alpha^t)\right)_{i=1,\ell}^{j=1,p}$ – матрица первых производных размера $\ell \times p$ на t-й итерации; $f_t = (f(x_i,\alpha^t))_{i=1,\ell}$ – вектор значений аппроксимирующей функции на t-й итерации. Тогда формула t-й итерации метода Ньютона-Гаусса в матричной записи примет вид:

$$\alpha^{t+1} := \alpha^t - h_t \underbrace{(F_t^\intercal F_t)^{-1} F_t^\intercal (f^t - y)}_{\delta}.$$

В правой части записано решение стандартной задачи многомерной линейной регрессии $\|F_t\delta - (f^t - y)\|^2 \to \min_\delta$. Таким образом, в методе Ньютона-Гаусса нелинейная регрессия сводится к последовательности линейных регрессионных задач. Скорость сходимости у него практически такая же, как и у метода Ньютона-Рафсона (оба являются методами второго порядка), но вычисления несколько проще и выполняются стандартными методами линейной регрессии.

5.5.2 Нелинейные одномерные преобразования признаков

На практике встречаются ситуации, когда линейная модель регрессии представляется необоснованной, но предложить адекватную нелинейную

модель $f(x,\alpha)$ также не удаётся. Тогда в качестве компромисса строится модель вида:

$$f(x,\alpha)=\sum_{j=1}^{n}\varphi_j(f_j(x)),$$

где $\varphi_j\colon \mathbb{R}\to\mathbb{R}$ – некоторые преобразования исходных признаков, в общем случае нелинейные. Задача состоит в том, чтобы подобрать неизвестные одномерные преобразования φ_j, при которых достигается минимум квадратичного функционала (5.1).

Метод настройки с возвращениями предложен Хасти и Тибширани в 1986 году. Схема реализации показана в Алгоритме 5.2.

Метод основан на итерационном уточнении функций φ_j. На первом шаге они полагаются линейными, $\varphi_j(x)=\alpha_j f_j(x)$, и неизвестные коэффициенты α_j настраиваются методами многомерной линейной регрессии. На каждом последующем шаге выбирается одна из функций φ_j, все остальные фиксируются, и выбранная функция строится заново. Для этого решается стандартная задача наименьших квадратов:

$$Q(\varphi_j,X^\ell)=\sum_{i=1}^{\ell}\Big(\varphi_j(f_j(x_i))-\underbrace{\big(y_i-\textstyle\sum_{k=1,k\neq j}^{n}\varphi_k(f_k(x_i))\big)}_{z_i=\mathrm{const}(\varphi_j)}\Big)^2\to\min_{\varphi_j}$$

с обучающей выборкой $Z_j^\ell=(f_j(x_i),z_i)_{i=1}^{\ell}$. Для решения данной задачи годятся любые одномерные методы: ядерное сглаживание, сплайны, полиномиальная или Фурье-аппроксимация.

Дополнительные темы по нелинейным методам восстановления регрессии

Обобщенные линейные модели. Неквадратичные функции потерь. Ненормальный шум. Проблемно-зависимые функции потерь. Робастная регрессия. Логистическая регрессия и итерационный взвешенный МНК. Метод опорных векторов в задачах регрессии.

РАЗДЕЛ 6. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Человеку и высшим животным буквально на каждом шагу приходится распознавать, принимать решения и обучаться. Нейросетевой подход возник из стремления понять, каким образом мозг решает столь сложные задачи, и реализовать эти принципы в автоматических устройствах. Пока искусственные нейронные сети (artificial neural networks, ANN) являются лишь предельно упрощёнными аналогами естественных нейронных сетей. Нервные системы животных и человека гораздо сложнее тех устройств, которые можно создать с помощью современных технологий. Однако для успешного решения многих практических задач оказалось вполне достаточно «подсмотреть» лишь общие принципы функционирования нервной системы. Некоторые разновидности ANN представляют собой математические модели, имеющие лишь отдалённое сходство с нейрофизиологией, что отнюдь не препятствует их практическому применению.

6.1. Проблема полноты

Итак, отдельно взятый нейрон позволяет реализовать линейный классификатор или линейную регрессию. При решении практических задач линейность оказывается чрезмерно сильным ограничением. На ограниченность персептрона указывали Минский и Пайперт в своей знаменитой книге «Персептроны». Следующий классический контрпример иллюстрирует невозможность нейронной реализации даже очень простых функций.

6.1.1. Задача «исключающего ИЛИ»

Легко построить нейроны, реализующие логические функции И, ИЛИ, НЕ от бинарных переменных x_1 и x_2, рисунок 6.1:

$$x^1 \vee x^2 = \left[x^1 + x^2 - \frac{1}{2} > 0\right];$$

$$x^1 \wedge x^2 = \left[x^1 + x^2 - \frac{3}{2} > 0\right];$$

$$\neg x^1 = \left[-x^1 + \frac{1}{2} > 0\right].$$

Однако функцию $x^1 \oplus x^2 = [x^1 \neq x^2]$ – исключающее ИЛИ (exclusive or, XOR) принципиально невозможно реализовать одним нейроном с двумя входами x^1 и x^2, поскольку множества нулей и единиц этой функции линейно неразделимы.

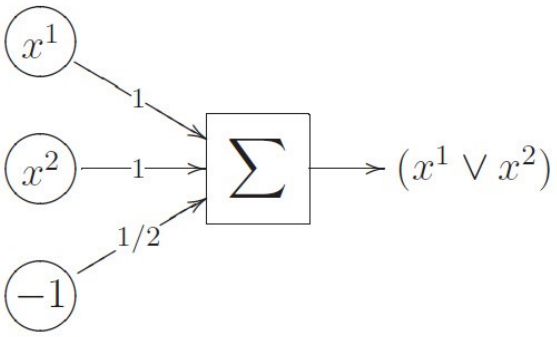


Рисунок 6.1 – Однослойный персептрон, реализующий операцию ИЛИ

Возможны два пути решения этой проблемы, рисунок 6.2.

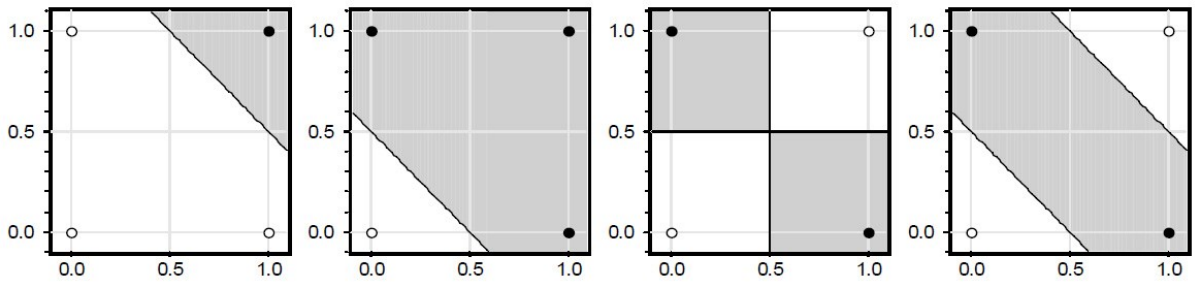


Рисунок 6.2 – Персептронная реализация элементарных логических функций. Слева направо: И, ИЛИ, XOR с помощью добавления признака x^1x^2, XOR с помощью двухслойной сети.

Первый путь – пополнить состав признаков, подавая на вход нейрона нелинейные преобразования исходных признаков. В частности, если разрешить образовывать всевозможные произведения исходных признаков, то нейрон будет строить уже не линейную, а полиномиальную разделяющую поверхность. В случае исключающего ИЛИ достаточно добавить только один

вход x^1x^2, чтобы в расширенном пространстве множества нулей и единиц оказались линейно разделимыми:

$$x^1 \oplus x^2 = [x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0].$$

Расширенные пространства признаков, в которых линейный классификатор безошибочно разделяет обучающую выборку, называют спрямляющими. Проблема заключается в том, что подбор нужных нелинейных преобразований является нетривиальной задачей, которая для общего случая до сих пор остаётся нерешённой.

Второй путь – построить композицию из нескольких нейронов. Например, исключающее ИЛИ можно реализовать, подав выходы И-нейрона и ИЛИ-нейрона на вход ещё одному ИЛИ-нейрону, рисунок 6.3:

$$x^1 \oplus x^2 = [(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0].$$

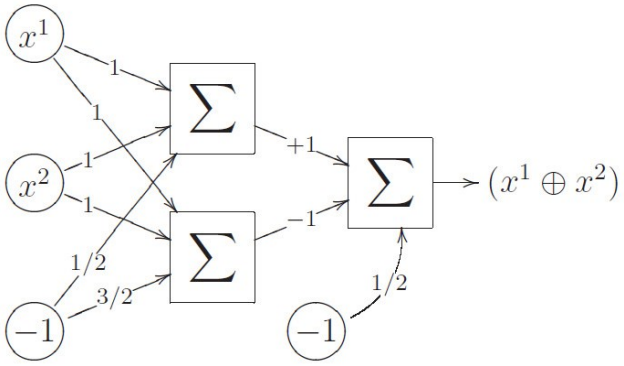


Рисунок 6.3 – Двухслойная сеть, реализующая операцию исключающего ИЛИ

Дальнейшее обобщение этой идеи приводит к построению многослойных нейронных сетей, состоящих из огромного количества связанных нейронов и напоминающих естественные нейронные сети. Пример такой композиции показан на рисунке 6.4.

Значения всех n признаков одновременно подаются на вход всех H нейронов первого слоя. Затем их выходные значения подаются на вход всех M нейронов следующего слоя. В данном случае этот слой является выходным – такая сеть называется двухслойной. В общем случае сеть может содержать

произвольное число слоёв. Все слои, за исключением последнего, называются скрытыми (hidden layers).

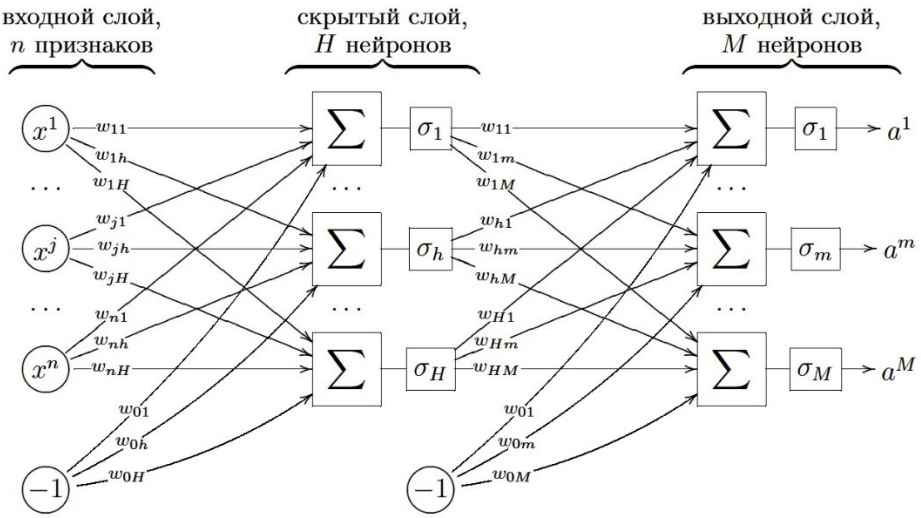


Рисунок 6.4 – Многослойная сеть с одним скрытым слоем

Вычисление выходных значений сети может осуществляться с высокой степенью параллелизма, за число тактов, равное числу слоёв. Существуют эффективные аппаратные реализации нейронных сетей, в которых вычисления действительно происходят параллельно. Но пока на практике чаще используются программные реализации, выполненные на обычных последовательных компьютерах.

6.1.2. Вычислительные возможности нейронных сетей

Возникает вопрос: любую ли функцию можно представить (хотя бы приближённо) с помощью нейронной сети? Следующие факты позволяют ответить на этот вопрос утвердительно.

1. Любая булева функция представима в виде двухслойной сети. Это тривиальное следствие нейронной представимости функций И, ИЛИ, НЕ и представимости произвольной булевой функции в виде дизъюнктивной нормальной формы.

2. Из простых геометрических соображений вытекает, что двухслойная сеть с пороговыми функциями активации позволяет выделить произвольный выпуклый многогранник в n-мерном пространстве признаков. Трёхслойная

сеть позволяет вычислить любую конечную линейную комбинацию характеристических функций выпуклых многогранников, следовательно, аппроксимировать любые области с непрерывной границей, включая невыпуклые и даже неодносвязные, а также аппроксимировать любые непрерывные функции.

3. В 1900 году Гильберт предложил список из 23 нерешённых задач, которые, по его мнению, должны были стать вызовом для математиков XX века. Тринадцатая проблема заключалась в следующем: возможно ли произвольную непрерывную функцию n аргументов представить в виде суперпозиции функций меньшего числа аргументов. Ответ был дан А.Н. Колмогоровым.

Теорема (Колмогоров, 1957). Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции сложения:

$$f(x^1, x^2, \dots, x^n) = \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^{n} \varphi_{ik}(x^i) \right),$$

где h_k, φ_{ik} – непрерывные функции, причём φ_{ik} не зависят от выбора f.

Нетрудно видеть, что записанное здесь выражение имеет структуру нейронной сети с одним скрытым слоем из $2n + 1$ нейронов. Таким образом, двух слоёв уже достаточно, чтобы вычислять произвольные непрерывные функции, и не приближённо, а точно. К сожалению, представление Колмогорова не является персептроном: функции φ_{ik} не линейны, а функции h_k зависят от f, и в общем случае не являются дифференцируемыми.

4. Известна классическая теорема Вейерштрасса о том, что любую непрерывную функцию n переменных можно равномерно приблизить полиномом с любой степенью точности. Более общая теорема Стоуна утверждает, что любую непрерывную функцию на произвольном компакте X можно приблизить не только многочленом от исходных переменных, но и многочленом от любого конечного набора функций F, разделяющих точки.

Определение. Набор функций F называется разделяющим точки множества X, если для любых различных $x, x' \in X$ существует функция $f \in F$ такая, что $f(x) \neq f(x')$.

Теорема (Стоун, 1948). Пусть X – компактное пространство, $C(X)$ – алгебра непрерывных на X вещественных функций, F – кольцо в $C(X)$, содержащее константу $(1 \in F)$ и разделяющее точки множества X. Тогда F плотно в $C(X)$.

На самом деле справедливо ещё более общее утверждение. Оказывается, вместо многочленов (суперпозиции операций сложения и умножения) можно пользоваться суперпозициями сложения и какой-нибудь (практически произвольной) непрерывной нелинейной функции одного аргумента. Этот результат имеет прямое отношение к нейронным сетям, поскольку они строятся из операций сложения, умножения и нелинейных функций активации.

Определение. Набор функций $F \subseteq C(X)$ называется замкнутым относительно функции $\varphi: \mathbb{R} \to \mathbb{R}$, если для любого $f \in F$ выполнено $\varphi(f) \in F$.

Теорема (Горбань, 1998). Пусть X – компактное пространство, $C(X)$ – алгебра непрерывных на X вещественных функций, F – линейное подпространство в $C(X)$, замкнутое относительно нелинейной непрерывной функции φ, содержащее константу $(1 \in F)$ и разделяющее точки множества X. Тогда F плотно в $C(X)$.

Это интерпретируется как утверждение об универсальных аппроксимационных возможностях произвольной нелинейности: с помощью линейных операций и единственного нелинейного элемента ϕ можно получить устройство, вычисляющее любую непрерывную функцию с любой желаемой точностью. Однако данная теорема ничего не говорит о количестве слоёв нейронной сети (уровней вложенности суперпозиции) и о количестве нейронов, необходимых для аппроксимации произвольной функции.

Таким образом, нейронные сети являются универсальными аппроксиматорами функций. Возможности сети возрастают с увеличением числа слоёв и числа нейронов в них. Двух-трёх слоёв, как правило, достаточно для решения подавляющего большинства практических задач классификации, регрессии и прогнозирования.

6.2. Многослойные нейронные сети

Многослойные сети, так же, как и однослойный персептрон (линейный классификатор), можно настраивать градиентными методами, несмотря на огромное количество весовых коэффициентов. В середине 80-х одновременно несколькими исследователями был предложен эффективный способ вычисления градиента, при котором каждый градиентный шаг выполняется за число операций, лишь немногим большее, чем при обычном вычислении сети на одном объекте. Это кажется удивительным – ведь количество операций, необходимых для вычисления градиента, обычно возрастает пропорционально размерности, то есть числу весовых коэффициентов. Здесь этого удаётся избежать благодаря аналитическому дифференцированию суперпозиции с сохранением необходимых промежуточных величин. Метод получил название обратного распространения ошибок (error back-propagation).

6.2.1. Метод обратного распространения ошибок

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, рисунок 6.4. Такая сеть называется полносвязной. Для большей общности положим $X = \mathbb{R}^n$, $Y = \mathbb{R}^M$.

Введём следующие обозначения. Пусть выходной слой состоит из M нейронов с функциями активации σ_m и выходами a^m, $m = 1, \dots, M$. Перед ним находится скрытый слой из H нейронов с функциями активации σ_h и выходами u^h, $h = 1, \dots, H$.

Веса синаптических связей между h-м нейроном скрытого слоя и m-м нейроном выходного слоя будем обозначать через w_{hm}. Перед этим слоем может находиться либо входной слой признаков (называемый также распределительным слоем), либо ещё один скрытый слой с выходами v^j, $j = 1, \dots, J$ и синаптическими весами w_{jh}. В общем случае число слоёв может быть произвольным. Если сеть двухслойная, то v^j есть просто j-й признак: $v^j(x) \equiv f_j(x) \equiv x^j$, и $J = n$. Обозначим через w вектор всех синаптических весов сети.

Выходные значения сети на объекте x_i вычисляются как суперпозиция:

$$\begin{aligned} a^m(x_i) &= \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right), \\ u^h(x_i) &= \sigma_h\left(\sum_{j=0}^{J} w_{jh} v^j(x_i)\right). \end{aligned} \tag{6.1}$$

Зафиксируем объект x_i и запишем функционал среднеквадратичной ошибки (для других функций потерь выкладки могут быть проделаны аналогично):

$$Q(w) = \frac{1}{2}\sum_{m=1}^{M} (a^m(x_i) - y_i^m)^2. \tag{6.2}$$

В дальнейшем нам понадобятся частные производные Q по выходам нейронов. Выпишем их сначала для выходного слоя:

$$\frac{\partial Q(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m.$$

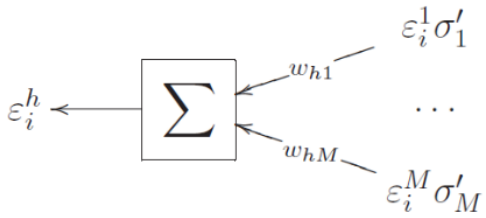
Оказывается, частная производная Q по am равна величине ошибки ε_i^m на объекте x_i. Теперь выпишем частные производные по выходам скрытого слоя:

$$\frac{\partial Q(w)}{\partial u^h} = \sum_{m=1}^{M} (a^m(x_i) - y_i^m)\sigma_m' w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma_m' w_{hm} = \varepsilon_i^h.$$

Эту величину, по аналогии с ε_i^m, будем называть ошибкой сети на скрытом слое и обозначать через ε_i^h. Через σ'_m обозначена производная функции активации, вычисленная при том же значении аргумента, что и в (6.1). Если используется сигмоидная функция активации, то для эффективного вычисления производной можно воспользоваться формулой $\sigma'_m = \sigma_m(1 - \sigma_m) = a^m(x_i)(1 - a^m(x_i))$.

Заметим, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд», подав на выходы нейронов скрытого слоя значения $\varepsilon_i^m \sigma'_m$, а результат ε^h получив на входе.

При этом входной вектор скалярно умножается на вектор весов w_{hm}, находящихся справа от нейрона, а не слева, как при прямом вычислении (отсюда и название алгоритма – обратное распространение ошибок):



Имея частные производные по a^m и u^h, легко выписать градиент Q по весам:

$$\frac{\partial Q(w)}{\partial w_{hm}} = \frac{\partial Q(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i), m = 1, \dots, M, h = 0, \dots, H. \qquad (6.3)$$

$$\frac{\partial Q(w)}{\partial w_{jh}} = \frac{\partial Q(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h v^j(x_i), h = 1, \dots, H, j = 0, \dots, J. \qquad (6.4)$$

и так далее для каждого слоя. Если слоёв больше двух, то остальные частные производные вычисляются аналогично – обратным ходом по слоям сети справа налево. Теперь мы обладаем всем необходимым, чтобы полностью выписать алгоритм обратного распространения, рисунок 6.5.

Достоинства метода обратного распространения.

1. Достаточно высокая эффективность. В случае двухслойной сети прямой ход, обратный ход и вычисления градиента требуют порядка $O(Hn + HM)$ операций.

2. Через каждый нейрон проходит информация только о связных с ним нейронах. Поэтому back-propagation легко реализуется на вычислительных устройствах с параллельной архитектурой.

3. Высокая степень общности. Алгоритм легко записать для произвольного числа слоёв, произвольной размерности выходов и выходов, произвольной функции потерь и произвольных функций активации, возможно, различных у разных нейронов. Кроме того, back-propagation можно применять совместно с различными градиентными методами оптимизации: методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

Вход:
$X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}^M$;
H — число нейронов в скрытом слое;
η — темп обучения;
Выход:
синаптические веса w_{jh}, w_{hm};

1: инициализировать веса небольшими случайными значениями:
$w_{jh} := \text{random}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$;
$w_{hm} := \text{random}\left(-\frac{1}{2H}, \frac{1}{2H}\right)$;
2: **повторять**
3: выбрать прецедент $(x_i, y_i) \in X^\ell$ случайным образом;
4: прямой ход:
$u_i^h := \sigma_h\left(\sum_{j=0}^J w_{jh} x_i^j\right)$, для всех $h = 1, \ldots, H$;
$a_i^m := \sigma_m\left(\sum_{h=0}^H w_{hm} u_i^h\right)$, для всех $m = 1, \ldots, M$;
$\varepsilon_i^m := a_i^m - y_i^m$, для всех $m = 1, \ldots, M$;
$Q_i := \sum_{m=1}^M (\varepsilon_i^m)^2$;
5: обратный ход:
$\varepsilon_i^h := \sum_{m=1}^M \varepsilon_i^m \sigma'_m w_{hm}$, для всех $h = 1, \ldots, H$;
6: градиентный шаг:
$w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma'_m u_i^h$, для всех $h = 0, \ldots, H$, $m = 1, \ldots, M$;
$w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma'_h x_i^j$, для всех $j = 0, \ldots, n$, $h = 1, \ldots, H$;
7: $Q := \frac{\ell-1}{\ell} Q + \frac{1}{\ell} Q_i$;
8: **пока** Q не стабилизируется;

Рисунок 6.5 – Обучение двухслойной сети методом back-propagation – обратного распространения ошибки

Недостатки метода обратного распространения.

1. Метод наследует известные недостатки градиентной настройки весов в однослойном персептроне. Здесь также возникают проблемы медленной сходимости или расходимости, ¾застревания¿ в локальных минимумах функционала Q, переобучения и паралича. Причём парализоваться могут отдельные связи, нейроны, или вся сеть в целом.

2. Приходится заранее фиксировать число нейронов скрытого слоя H. В то же время, это критичный параметр сложности сети, от которого может существенно зависеть качество обучения и скорость сходимости.

6.2.2. Эвристики для улучшения сходимости

Для улучшения сходимости и качества градиентного обучения применяются все те же приёмы, которые рекомендовались при рассмотрении градиентных методов для обучения однослойного персептрона методом стохастического градиента. К ним добавляется ряд новых рекомендаций, связанных с многослойностью.

Выбор начального приближения. Для предотвращения паралича синаптические веса должны инициализироваться небольшими по модулю значениями. В алгоритме на рисунке 6.5 на шаге 1 веса инициализируются случайными – значениями из отрезка $\left[-\frac{1}{2k}, \frac{1}{2k}\right]$, где k – число нейронов в том слое, из которого выходит данный синапс.

В этом случае (и при условии, что все признаки нормализованы) значения скалярных произведений гарантированно попадают в «рабочую зону» функций активации, представленных на рисунке 4.3.

Существует и более тонкий способ формирования начального приближения. Идея заключается в том, чтобы сначала настроить нейроны первого слоя поотдельности, как H однослойных персептронов. Затем по-отдельности настраиваются нейроны второго слоя, которым на вход подаётся вектор выходных значений первого слоя. Чтобы сеть не получилась

вырожденной, нейроны первого слоя должны быть существенно различными. Ещё лучше, если они будут хоть как-то приближать целевую зависимость, тогда второму слою останется только усреднить результаты первого слоя, сгладив ошибки некоторых нейронов. Добиться этого совсем несложно, если обучать нейроны первого слоя на различных случайных подвыборках, либо подавать им на вход различные случайные подмножества признаков. Отметим, что при формировании начального приближения не требуется особая точность настройки, поэтому отдельные нейроны можно обучать простейшими градиентными методами.

Выбор градиентного метода оптимизации. К сожалению, градиентные методы первого порядка сходятся довольно медленно, и потому редко применяются на практике. Ньютоновские методы второго порядка также непрактичны, но по другой причине – они требуют вычисления матрицы вторых производных функционала $Q(w)$, имеющей слишком большой размер. Метод сопряжённых градиентов этого не требует, однако применять его непосредственно нельзя, так как он существенно опирается на предположение неизменности функционал $Q(w)$, а в методе стохастического градиента функционал меняется при предъявлении каждого нового объекта. Необходимы специальные ухищрения, чтобы приспособить стандартные методы оптимизации для настройки нейронных сетей. Существуют следующие рекомендации.

1. Если обучающая выборка имеет большой объём (порядка нескольких сотен объектов и более), или если решается задача классификации, то можно использовать метод стохастического градиента с адаптивным шагом.

2. Диагональный метод Левенберга-Марквардта сходится в несколько раз быстрее. В этом методе величина шага вычисляется индивидуально для каждого весового коэффициента, при этом используется только один диагональный элемент матрицы вторых производных:

$$\eta_{jh} = \frac{\eta}{\frac{\partial^2 Q}{\partial w_{jh}^2} + \mu},$$

где η остаётся глобальным параметром темпа обучения, μ – новый параметр, предотвращающий обнуление знаменателя, и, соответственно, неограниченное увеличение шага. Отношение η/μ есть темп обучения на ровных участках функционала $Q(w)$, где вторая производная обращается в нуль. Диагональный элемент матрицы вторых производных вычисляется с помощью специального варианта back-propagation.

3. Если обучающая выборка имеет небольшой объём, или если решается задача регрессии, то лучше использовать адаптированные варианты метода сопряжённых градиентов. Адаптация заключается в том, что объекты предъявляются не по одному, а пакетами (batch learning). Состав пакета может формироваться случайным образом. Для каждого пакета минимизируемый функционал остаётся фиксированным, что позволяет применить метод сопряжённых градиентов.

6.2.3. Оптимизация структуры сети

Выбор структуры сети, то есть числа слоёв, числа нейронов и числа связей для каждого нейрона, является, пожалуй, наиболее сложной проблемой. Существуют различные стратегии поиска оптимальной структуры сети: постепенное наращивание, построение заведомо слишком сложной сети с последующим упрощением, поочерёдное наращивание и упрощение.

Проблема выбора структуры тесно связана с проблемами недообучения и переобучения. Слишком простые сети не способны адекватно моделировать целевые зависимости в реальных задачах. Слишком сложные сети имеют избыточное число свободных параметров, которые в процессе обучения настраиваются не только на восстановление целевой зависимости, но и на воспроизведение шума.

Выбор числа слоёв. Если в конкретной задаче гипотеза о линейной разделимости классов выглядит правдоподобно, то можно ограничиться однослойным персептроном. Двухслойные сети позволяют представлять извилистые нелинейные границы, и в большинстве случаев этого хватает.

Трёхслойными сетями имеет смысл пользоваться для представления сложных многосвязных областей. Чем больше слоёв, тем более богатый класс функций реализует сеть, но тем хуже сходятся градиентные методы, и тем труднее её обучить.

Выбор числа нейронов в скрытом слое H производят различными способами, но ни один из них не является лучшим.

1. Визуальный способ. Если граница классов (или кривая регрессии) слишком сглажена, значит, сеть переупрощена, и необходимо увеличивать число нейронов в скрытом слое. Если граница классов (или кривая регрессии) испытывает слишком резкие колебания, на тестовых данных наблюдаются большие выбросы, веса сети принимают большие по модулю значения, то сеть переусложнена, и скрытый слой следует сократить. Недостаток этого способа в том, что он подходит только для задач с низкой размерностью пространства (небольшим числом признаков).

2. Оптимизация H по внешнему критерию, например, по критерию скользящего контроля или средней ошибки на независимой контрольной выборке $Q(X^k)$. Зависимость внешних критериев от параметра сложности, каким является H, обычно имеет характерный оптимум. Недостаток этого способа в том, что приходится много раз заново строить сеть при различных значениях параметра H, а в случае скользящего контроля – ещё и при различных разбиениях выборки на обучающую и контрольную части.

Динамическое добавление нейронов. Сначала сеть обучается при заведомо недостаточной мощности среднего слоя $H \ll \ell$. Обучение происходит до тех пор, пока ошибка не перестанет убывать. Тогда добавляется один или несколько новых нейронов. Веса новых связей инициализируются небольшими случайными числами, либо добавленные нейроны обучаются по-отдельности как однослойные персептроны.

Во втором случае можно рекомендовать обучать новый персептрон на случайной подвыборке, возможно, добавив в неё те объекты, на которых текущая сеть допустила наибольшие ошибки. Веса старых связей не меняются.

Затем проводится настройка сети методом обратного распространения. После добавления новых нейронов ошибка, как правило, сначала резко возрастает, затем быстро сходится к меньшему значению. Интересно, что общее время обучения обычно оказывается лишь в 1.5–2 раза больше, чем если бы в сети сразу было нужное количество нейронов. Это означает, что информация, накопленная в сети, является полезной и не теряется при добавлении новых нейронов.

При постепенном наращивании сети целесообразно наблюдать за динамикой какого-нибудь внешнего критерия. Прохождение значения $Q(X^k)$ через минимум является надёжным критерием останова, так как свидетельствует о переобученности, вызванной чрезмерным усложнением сети.

Удаление избыточных связей. Метод оптимального прореживания сети (optimal brain damage, OBD) удаляет те связи, к изменению которых функционал Q наименее чувствителен. Уменьшение числа весов снижает склонность сети к переобучению.

Метод OBD основан на предположении, что после стабилизации функционала ошибки Q вектор весов w находится в локальном минимуме, где функционал может быть аппроксимирован квадратичной формой:

$$Q(w+\delta) = Q(w) + \frac{1}{2}\delta^{\mathrm{T}} H(w)\delta + o(\|\delta\|^2).$$

где $H(w) = \left(\frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}\right)$ – гессиан, матрица вторых производных. Как и в диагональном методе Левенберга-Марквардта, предполагается, что диагональные элементы доминируют в гессиане, а остальными частными производными можно пренебречь, положив их равными нулю. Это предположение носит эвристический характер и вводится для того, чтобы избежать трудоёмкого вычисления всего гессиана.

Если гессиан $H(w)$ диагонален, то

$$\delta^{\mathrm{T}} H(w)\delta = \sum_{j=0}^{J}\sum_{h=1}^{H} \delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^w}.$$

Обнуление веса w_{jh} эквивалентно выполнению условия $w_{jh} + \delta_{jh} = 0$. Введём величину значимости (salience) синаптической связи, равную изменению функционала $Q(w)$ при обнулении веса: $S_{jh} = w_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^w}$.

Эвристика OBD заключается в том, чтобы удалить из сети d синапсов, соответствующих наименьшим значениям S_{jh}. Здесь d – это ещё один параметр метода настройки. После удаления производится цикл итераций до следующей стабилизации функционала Q. При относительно небольших значениях d градиентный алгоритм довольно быстро находит новый локальный минимум Q. Процесс упрощения сети останавливается, когда внутренний критерий стабилизируется, либо когда заданный внешний критерий начинает возрастать.

Обнуление веса w_{jh} между входным и скрытым слоями означает, что h-й нейрон скрытого слоя не будет учитывать j-й признак. Тем самым происходит отбор информативных признаков для h-го нейрона скрытого слоя.

Метод OBD легко приспособить и для настоящего отбора признаков. Вводится суммарная значимость признака $S_j = \sum_{h=1}^{H} S_{jh}$, и из сети удаляется один или несколько признаков с наименьшим значением S_j.

Обнуление веса w_{jh} между скрытым и выходным слоями означает, что m-е выходное значение не зависит от h-го нейрона скрытого слоя. Если выход одномерный ($M = 1$), то h-й нейрон можно удалить. В случае многомерного выхода для удаления нейронов скрытого слоя вычисляется суммарная значимость $S_h = \sum_{m=1}^{M} S_{hm}$.

РАЗДЕЛ 7. БАЙЕСОВСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

Байесовский подход является классическим в теории распознавания образов и лежит в основе многих методов. Он опирается на теорему о том, что если плотности распределения классов известны, то алгоритм классификации, имеющий минимальную вероятность ошибок, можно выписать в явном виде. Для оценивания плотностей классов по выборке применяются различные подходы. В этом курсе лекций рассматривается три: параметрический, непараметрический и оценивание смесей распределений.

7.1. Вероятностная постановка задачи классификации

Пусть X – множество объектов, Y – конечное множество имён классов, множество $X \times Y$ является вероятностным пространством с плотностью распределения $p(x,y) = \mathbb{P}(y)p(x|y)$. Вероятности появления объектов каждого из классов $P_y = \mathbb{P}(y)$ называются априорными вероятностями классов. Плотности распределения $p_y(x) = p(x|y)$ называются функциями правдоподобия классов. Вероятностная постановка задачи классификации разделяется на две независимые подзадачи.

Задача 1. Имеется простая выборка $X^\ell = (x_i, y_i)_{i=1}^\ell$ из неизвестного распределения $p(x,y) = P_y p_y(x)$. Требуется построить эмпирические оценки априорных вероятностей $\hat{P}_y$ и функций правдоподобия $\hat{p}_y(x)$ для каждого из классов $y \in Y$.

Задача 2. По известным плотностям распределения $p_y(x)$ и априорным вероятностям P_y всех классов $y \in Y$ построить алгоритм $a(x)$, минимизирующий вероятность ошибочной классификации.

Вторая задача решается относительно легко, и мы сразу это сделаем. Первая задача имеет множество решений, поскольку многие распределения $p(x,y)$ могли бы породить одну и ту же выборку X^ℓ. Приходится привлекать

различные предположения о плотностях, что и приводит к большому разнообразию байесовских методов.

7.1.1. Функционал среднего риска

Знание функций правдоподобия позволяет находить вероятности событий вида «$x \in \Omega$ при условии, что x принадлежит классу y»:

$$\mathbb{P}(\Omega|\mathrm{y}) = \int_{\Omega} p_y(x)dx\,, \Omega \subset X,$$

Рассмотрим произвольный алгоритм $a: X \to Y$. Он разбивает множество X на непересекающиеся области $A_y = \{x \in X | a(x) = y\}$, $y \in Y$. Вероятность того, что появится объект класса y и алгоритм а отнесёт его к классу s, равна $P_y\mathbb{P}(A_s|y)$. Каждой паре $(y, s) \in Y \times Y$ поставим в соответствие величину потери λ_{ys} при отнесении объекта класса y к классу s. Обычно полагают $\lambda_{yy} = 0$, и $\lambda_{ys} > 0$ при $y \neq s$.

Соотношения потерь на разных классах, как правило, известны заранее.

Определение. Функционалом среднего риска называется ожидаемая величина потери при классификации объектов алгоритмом a:

$$R(a) = \sum_{y \in Y} \sum_{s \in Y} \lambda_{ys} P_y \mathbb{P}(A_s|y).$$

Если величина потерь одинакова для ошибок любого рода, $\lambda_{ys} = [y \neq s]$, то средний риск $R(a)$ совпадает с вероятностью ошибки алгоритма a.

7.1.2. Оптимальное байесовское решающее правило

Теорема 1. Если известны априорные вероятности P_y и функции правдоподобия $p_y(x)$, то минимум среднего риска $R(a)$ достигается алгоритмом

$$a(x) = \arg\min_{s \in Y} \sum_{y \in Y} \lambda_{ys} P_y p_y(x).$$

Доказательство теоремы в данномкурсе лекций не приводится.

Часто можно полагать, что величина потери зависит только от истинной классификации объекта, но не от того, к какому классу он был ошибочно отнесён. В этом случае формула оптимального алгоритма упрощается.

Теорема 2. Если P_y и $p_y(x)$ известны, $\lambda_{yy} = 0$ и $\lambda_{ys} \equiv \lambda_y$ для всех $y, s \in Y$, то минимум среднего риска достигается алгоритмом

$$a(x) = \arg\max_{y \in Y} \lambda_y P_y p_y(x). \qquad (7.1)$$

Разделяющая поверхность между классами s и t – это геометрическое место точек $x \in X$ таких, что максимум в (7.1) достигается одновременно при $y = s$ и $y = t$: $\lambda_t P_t p_t(x) = \lambda_s P_s p_s(x)$. Объекты x, удовлетворяющие этому уравнению, можно относить к любому из двух классов s, t, что не повлияет на средний риск $R(a)$.

Апостериорная вероятность класса y для объекта x – это условная вероятность $\mathbb{P}(y|x)$. Она может быть вычислена по формуле Байеса, если известны $p_y(x)$ и P_y:

$$\mathbb{P}(y|x) = \frac{p(x,y)}{p(x)} = \frac{p_y(x)P_y}{\sum_{s \in Y} p_s(x)P_s}.$$

Во многих приложениях важно не только классифицировать объект x, но и сказать, с какой вероятностью $\mathbb{P}(y|x)$ он принадлежит каждому из классов. Через апостериорные вероятности выражается величина ожидаемых потерь на объекте x:

$$R(x) = \sum_{y \in Y} \lambda_y \mathbb{P}(y|x)$$

Принцип максимума апостериорной вероятности. Оптимальный алгоритм классификации (7.1) можно переписать через апостериорные вероятности:

$$a(x) = \arg\max_{y \in Y} \lambda_y \mathbb{P}(y|x).$$

Поэтому выражение (7.1) называют байесовским решающим правилом.

Минимальное значение среднего риска $R(a)$, достигаемое байесовским решающим правилом, называется байесовским риском или байесовским уровнем ошибки.

Если классы равнозначны ($\lambda_y \equiv 1$), то байесовское правило называется также принципом максимума апостериорной вероятности. Если классы ещё и равновероятны ($P_y \equiv \frac{1}{|Y|}$), то объект x просто относится к классу y с наибольшим значением плотности распределения $p_y(x)$ в точке x.

2.1.3. Задача восстановления плотности распределения

Перейдём к **Задаче 1**. Требуется оценить, какой могла бы быть плотность вероятностного распределения $p(x,y) = P_y p_y(x)$, сгенерировавшего выборку X^ℓ.

Обозначим подвыборку прецедентов класса y через $X_y^\ell = \{(x_i, y_i)_{i=1}^\ell | y_i = y\}$.

Проще всего оценить априорные вероятности классов P_y. Согласно закону больших чисел, частота появления объектов каждого из классов

$$\hat{P}_y = \frac{\ell_y}{\ell}, \ell_y = \left|X_y^\ell\right|, y \in Y, \tag{7.2}$$

сходится по вероятности к P_y при $\ell_y \to \infty$. Чем больше длина выборки, тем точнее выборочная оценка $\hat{P}_y$. Оценка (7.2) является несмещённой лишь в том случае, если все без исключения наблюдавшиеся объекты заносились в обучающую выборку.

На практике применяются и другие принципы формирования данных. Например, в задачах с несбалансированными классами (unbalanced classes) один из классов может встречаться в тысячи раз реже остальных; это может затруднять построение алгоритмов, поэтому выборку формируют неслучайным образом, чтобы объекты всех классов были представлены поровну. Возможна также ситуация, когда обучающая выборка формируется

в ходе планируемого эксперимента, а применять построенный алгоритм классификации предполагается в реальной среде с другими априорными

вероятностями классов. Во всех подобных ситуациях оценка $\hat{P}_y$ должна делаться не по доле обучающих объектов (7.2), а из других содержательных соображений.

Задача восстановления плотности имеет самостоятельное значение, поэтому мы сформулируем её в более общем виде, обозначая выборку через X^m вместо X_y^ℓ, что позволит несколько упростить обозначения.

Задача 3. Задано множество объектов $X^m = \{x_1, \dots, x_m\}$, выбранных случайно и независимо согласно неизвестному распределению $p(x)$. Требуется построить эмпирическую оценку плотности – функцию $\hat{p}(x)$, приближающую $p(x)$ на всём X.

«Наивный» байесовский классификатор. Допустим, что объекты $x \in X$ описываются n числовыми признаками $f_j: X \to \mathbb{R}$, $j = 1, \dots, n$. Обозначим через $x = (\xi_1, \dots, \xi_n)$ произвольный элемент пространства объектов $X = \mathbb{R}^n$, где $\xi_j = f_j(x)$.

Гипотеза. Признаки $f_1(x), \dots, f_n(x)$ являются независимыми случайными величинами. Следовательно, функции правдоподобия классов представимы в виде

$$p_y(x) = p_{y1}(\xi_1) \dots p_{yn}(\xi_n), y \in Y, \qquad (7.3)$$

где $p_{yj}(\xi_j)$ – плотность распределения значений j-го признака для класса y.

Предположение о независимости существенно упрощает задачу, так как оценить n одномерных плотностей гораздо проще, чем одну n-мерную плотность. Однако оно крайне редко выполняется на практике, поэтому алгоритмы классификации, использующие (7.3), называются наивными байесовскими.

Подставим эмпирические оценки одномерных плотностей $\hat{p}_{yj}(\xi)$ в (7.3) и затем в (7.1). Получим алгоритм:

$$a(x) = \arg\max_{y \in Y} \left(\ln \lambda_y \hat{P}_y + \sum_{j=1}^{n} \ln \hat{p}_{yj}(\xi_j) \right). \qquad (7.4)$$

Основные его преимущества – простота реализации и низкие вычислительные затраты при обучении и классификации. В тех редких случаях, когда признаки (почти) независимы, наивный байесовский классификатор (почти) оптимален.

Основной его недостатокнизкое качество классификации. Он используется либо как эталон при экспериментальном сравнении алгоритмов, либо как элементарный «строительный блок» в алгоритмических композициях.

7.2. Непараметрическая классификация

Непараметрические методы классификации основаны на локальном оценивании плотностей распределения классов $p_{\text{н}}(x)$ в окрестности классифицируемого объекта $x \in X$. Для классификации объекта x применяется основная формула (7.1).

7.2.1. Непараметрические оценки плотности

Локальное оценивание опирается на само определение плотности. Начнём с простейших одномерных оценок. Они уже могут оказаться полезными на практике, в частности, при построении «наивных» байесовских классификаторов (7.4). Кроме того, они подскажут нам идеи обобщения на многомерный случай.

Дискретный случай. Пусть X – конечное множество, причём $|X| \ll m$. Оценкой плотности служит гистограмма значений x_i, встретившихся в выборке $X^m = (x_i)_{i=1}^m$:

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1}^{m} [x_i = x].$$

Эта оценка не применима, если $|X| \gg m$, и, тем более, в непрерывном случае, так как её значение почти всегда будет равно нулю.

Одномерный непрерывный случай. Пусть $X = \mathbb{R}$. Согласно определению плотности, $p(x) = \lim\limits_{h \to 0} \frac{1}{2h} P[x-h, x+h]$, где $P[a,b]$ – вероятностная мера отрезка $[a,b]$. Соответственно, эмпирическая оценка плотности определяется как доля точек выборки, лежащих внутри отрезка $[x-h, x+h]$, где h – неотрицательный параметр, называемый шириной окна:

$$\hat{p}_h(x) = \frac{1}{2mh} \sum_{i=1}^{m} [|x - x_i| < h]. \qquad (7.5)$$

Функция $\hat{p}_h(x)$ является кусочно-постоянной, что приводит к появлению широких зон неуверенности, в которых максимум (7.1) достигается одновременно для нескольких классов $y \in Y$. Проблема решается с помощью локальной непараметрической оценки Парзена-Розенблатта:

$$\hat{p}_h(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right), \qquad (7.6)$$

где $K(z)$ – функция, называемая ядром, чётная и нормированная $\int K(z)dz = 1$.

Функция $\hat{p}_h(x)$ обладает той же степенью гладкости, что и ядро $K(z)$, и, благодаря нормировке, действительно может интерпретироваться как плотность вероятности: $\int \hat{p}_h(x) = 1$ при любом h.

На практике часто используются ядра, показанные на рисунке 7.1.

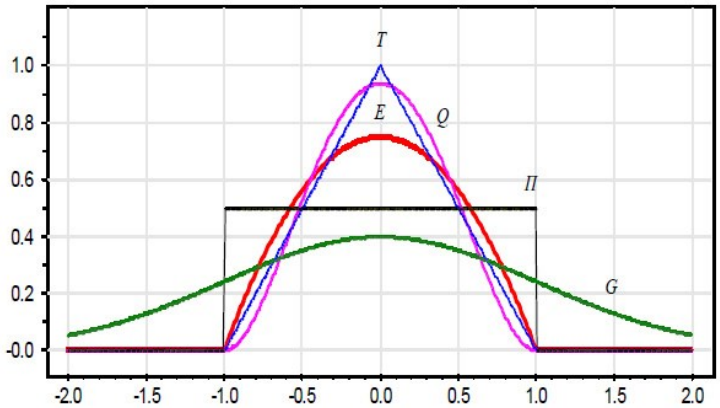


Рисунок 7.1 – Часто используемые ядра: E – Епанечникова; Q – квартическое; T – треугольное; G – гауссовское; П – прямоугольное

Прямоугольное ядро $K(z) = \frac{1}{2}[|z| < 1]$ соответствует простейшей оценке (7.5).

Точечное ядро $K(z) = [z = 0]$ при $h = 1$ соответствует дискретному случаю (7.4).

Обоснованием оценки (7.5) служит следующая теорема, утверждается, что $\hat{p}_h(x)$ поточечно сходится к истинной плотности $p(x)$ для широкого класса ядер при увеличении длины выборки m и одновременном уменьшении ширины окна h.

Теорема 3. Пусть выполнены следующие условия:

1) выборка X^m простая, получена из плотности распределения $p(x)$;

2) ядро $K(z)$ непрерывно, его квадрат ограничен: $\int K^2(z)dz < \infty$;

3) последовательность h_m такова, что $\lim\limits_{m\to\infty} h_m = 0$ и $\lim\limits_{m\to\infty} mh_m = \infty$.

Тогда $\hat{p}_{h_m}(x)$ сходится к $p(x)$ при $m \to \infty$ для почти всех $x \in X$, причём скорость сходимости имеет порядок $O(m^{-2/5})$.

Многомерный непрерывный случай. Пусть объекты описываются n числовыми признаками $f_j: X \to \mathbb{R}$, $j = 1, \dots, n$. Тогда непараметрическая оценка плотности в точке $x \in X$ записывается в следующем виде:

$$\hat{p}_h(x) = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{n}\frac{1}{h_j}K\left(\frac{f_j(x) - f_j(x_i)}{h_j}\right). \qquad (7.7)$$

Таким образом, в каждой точке x_i многомерная плотность представляется в виде произведения одномерных плотностей. Заметим, что это никак не связано с «наивным» байесовским предположением о независимости признаков. При «наивном» подходе плотность представлялась бы как произведение одномерных парзеновских оценок (7.6), то есть как произведение сумм, а не как сумма произведений.

Произвольное метрическое пространство. Пусть на X задана функция расстояния $\rho(x, x')$, вообще говоря, не обязательно метрика. Одномерная оценка Парзена-Розенблатта (7.6) легко обобщается и на этот случай:

$$\hat{p}_h(x) = \frac{1}{mV(h)} \sum_{i=1}^{m} K\left(\frac{\rho(x, x_i)}{h}\right). \qquad (7.8)$$

где $V(h)$ – нормирующий множитель, гарантирующий, что $\hat{p}_h(x)$ действительно является плотностью. Сходимость оценки (7.8) доказана при некоторых дополнительных ограничениях на ядро K и метрику ρ, причём скорость сходимости лишь немного хуже, чем в одномерном случае.

Дополнительные темы по бейесовским методам классификации

Метод парзеновского окна. Нормальный дискриминантный анализ. Разделение смеси распределений.

ЗАКЛЮЧЕНИЕ

Курс лекций по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning).

В пособии были рассмотрены теоретические аспекты построения информационных систем для решения различных задач с использованием следующих подходов: линейных методов классификации, аппарата нейронных сетей, байесовских методов классификации, алгоритмов восстановления регрессии, алгоритмов логической классификации.

Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Как говорят, мы получаем данные из «черного ящика». В этих условиях ничего не остается, как только изучать доступную нам последовательность исходных данных и пытаться строить предсказания совершенствуя нашу схему в процессе предсказания. Подход, при котором про- шлые данные или примеры используются для первоначального формирования и совершенствования схемы предсказания, называется методом машинного обучения (Machine Learning). Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов.

СПИСОК ЛИТЕРАТУРЫ

Список основной литературы

1. Уэс, Маккинли. Python и анализ данных Электронный ресурс / Маккинли Уэс ; пер. А. А. Слинкин. - Python и анализ данных,2025-04-19. - Саратов : Профобразование, 2017. - 482 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0046-7, экземпляров неограниченно.

2. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python,2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0058-2, экземпляров неограниченно

Список дополнительной литературы

3. Стенли, Липпман. Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++,2025-04-19. - Саратов : Профобразование, 2017. - 1104 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0136-5, экземпляров неограниченно

4. Седжвик, Р. Алгоритмы на C++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно

Методы искусственного интеллекта

КУРС ЛЕКЦИЙ

Автор

Николаев Евгений Иванович

Редактор, техническая правка,

компьютерная верстка:

Подписано в печать _____г.

Формат 60х84 1/16 Усл.п.л. – 4,25. Уч.-изд.л. – 3,4.

Бумага газетная. Печать офсетная. Заказ № ___ Тираж ___ экземпляров

ФГАОУ ВПО «Северо-Кавказский федеральный университет»

355000, г. Ставрополь, ул. Пушкина, 1

Издательство СКФУ