теория обучения машин

1 Введение: задачи обучения по прецедентам

В этой вводной лекции 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 Основные понятия и определения

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 $\{x_1, \ldots, x_\ell\} \subset X$. Пары «объект–ответk (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, но и на всём множестве X.

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

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 $D_f = \mathrm{R}$, 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*:

$$F = \|f_j(x_i)\|_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix}. \tag{1.1}$$

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

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 $Y = \mathbb{R}$, 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.

Модель алгоритмов и метод обучения

Опр. 1.1. *Моделью алгоритмов называется параметрическое семейство отображений $A = \{g(x, \vartheta) \mid \vartheta \in \Theta\}$, 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 $\vartheta = (\vartheta_1, \ldots, \vartheta_n) \in \Theta = \mathbb{R}^n$:

$$g(x, \vartheta) = \sum_{j=1}^{n} \vartheta_j f_j(x) \quad \text{— для задач восстановления регрессии, } Y = \mathbb{R};$$

$$g(x, \vartheta) = \operatorname{sign} \sum_{j=1}^{n} \vartheta_j f_j(x) \quad \text{— для задач классификации, } Y = \{-1, +1\}.$$

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 $g(x, \vartheta)$ 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)[1] алгоритма $a \in A$.

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

Опр. 1.2. *Метод обучения (learning algorithm) — это отображение* $\mu\colon (X \times Y)^\ell \to A$, *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.

Функционал качества

Опр. 1.3. *Функция потерь (loss function) — это неотрицательная функция* $\mathsf{L}(a, x)$, *характеризующая величину ошибки алгоритма* a *на объекте* x. *Если* $\mathsf{L}(a, x) = 0$, *то ответ* $a(x)$ *называется корректным.*

Опр. 1.4. *Функционал качества алгоритма* a *на выборке* X^ℓ:

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

Функционал Q называют также функционалом *средних потерь* или *эмпирическим риском* [4], 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 $\mathsf{L}(a, x) = 1$ означает, что алгоритм a допускает ошибку на объекте x, а функционал Q называется *частотой ошибок* алгоритма a на выборке X^ℓ.

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

$\mathsf{L}(a, x) = [a(x) = y^*(x)]$ — индикатор ошибки, обычно применяется в задачах классификации[2];

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

$\mathsf{L}(a, x) = (a(x) - y^*(x))^2$ — квадратичная функция потерь; функционал Q называется *средней квадратичной ошибкой* алгоритма a 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:

$$\mu(X^\ell) = \arg\min_{a \in A} Q(a, X^\ell). \tag{1.3}$$

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

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

[2] Квадратные скобки переводят логическое значение в число по правилу [ложь] = 0, [истина] = 1.

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 $f_j(x)$ 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 $g(x, \vartheta)$, аппроксимирующей неизвестную зависимость $y^*(x)$, задаётся модель совместной плотности распределения объектов и ответов $\phi(x, y, \vartheta)$, 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: $p(X^\ell) = p(x_1, y_1), \ldots, (x_\ell, y_\ell) = p(x_1, y_1) \cdots p(x_\ell, y_\ell)$. Подставляя вместо $p(x, y)$ модель плотности $\phi(x, y, \vartheta)$, получаем *функцию правдоподобия* (likelihood):

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

Чем выше значение правдоподобия, тем лучше выборка согласуется с моделью. Значит, нужно искать значение параметра ϑ, при котором значение $L(\vartheta, X^\ell)$ максимально. В математической статистике это называется *принципом максимума правдоподобия*. Его формальные обоснования можно найти, например, в [13].

После того, как значение параметра ϑ найдено, искомый алгоритм $a_\vartheta(x)$ строится по плотности $\phi(x, y, \vartheta)$ 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:

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

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

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

Пример 1.4. Пусть задана модель $g(x, \vartheta)$. Примем дополнительное вероятностное предположение, что ошибки $\varepsilon(x, \vartheta) = g(x, \vartheta) - y^*(x)$ имеют нормальное распределение $\mathsf{N}(\varepsilon; 0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\varepsilon}{2\sigma^2}\right)$ с нулевым средним и дисперсией σ^2. Тогда модель плотности имеет вид

$$\phi(x, y, \vartheta) = p(x)\phi(y \mid x, \vartheta) = p(x)\mathsf{N}\left(g(x, \vartheta) - y^*(x); 0, \sigma^2\right).$$

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

$$-\ln \phi(x, y, \vartheta) = -\ln p(x)\mathsf{N}\left(g(x, \vartheta) - y^*(x); 0, \sigma^2\right) = C_0 + C_1\left(g(x, \vartheta) - y^*(x)\right)^2.$$

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. Обучение — это оптимизация.

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 выборкаk обычно принимается стандартное предположение, что выборки X^ℓ и X^k — простые, полученные из одного и того же неизвестного вероятностного распределения на множестве X.

Опр. 1.5. *Метод обучения μ называется состоятельным, если при заданных достаточно малых значениях ε и η справедливо неравенство*

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

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

Допустима также эквивалентная формулировка: для любых простых выборок X^ℓ и X^k оценка $Q(\mu(X^\ell), X^k) \leq \varepsilon$ 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 [5, 6, 7]. В настоящее время статистическая теория развивается очень активно [34], 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):

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

Возможны различные варианты скользящего контроля, отличающиеся способами разбиения выборки X^L [48]. В простейшем варианте разбиения генерируются случайным образом, число N берётся в диапазоне от 20 до 100. Стандартом «де фактоk считается методика *t×q-кратного скользящего контроля* (t×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 Примеры прикладных задач

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

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: «хорошихk и «плохихk. 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, явля-

ются не сами клиенты, а пары «клиент x_i в момент времени t_ik. 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.

Задачи восстановления регрессии

Пример 1.8. Термин «регрессияk 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 посредственностиk, 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 «вручнуюk 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–товарk. 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.

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, ответk 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, документk, 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, документk. 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.

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* (semi-supervised 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.

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 молокоk. 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.

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 «лёгкиеk, так и заведомо «трудныеk; при такой организации эксперимента точнее выявляются границы применимости метода.

Эксперименты на реальных данных. Их цель — либо решение конкретной прикладной задачи, либо выявление «слабых местk 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 [30].

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.

Приёмы генерации модельных данных

Данный раздел носит справочный характер. В нём перечислены некоторые сведения, полезные при генерации модельных выборок данных.

Моделирование случайных данных. Следующие утверждения позволяют генерировать случайные выборки с заданными распределениями [10]. 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 $F_0 = 0, F_1, \ldots, F_{k-1}, F_k = 1$, то дискретная случайная величина ξ, определяемая условием $F_{\xi-1} \le r < F_\xi$, 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 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: $\xi_1, \xi_2 \in \mathrm{N}(0, 1)$.

Утв. 5. Если ξ — нормальная случайная величина из $\mathrm{N}(0, 1)$, то случайная величина $\eta = \mu + \sigma\xi$ имеет нормальное распределение $\mathrm{N}(\mu, \sigma^2)$ с матожиданием μ и дисперсией σ^2.

Утв. 6. Пусть n-мерный вектор $x = (\xi_1, \ldots, \xi_n)$ составлен из независимых нормальных случайных величин $\xi_i \sim \mathrm{N}(0, 1)$. Пусть V — невырожденная $n \times n$-матрица, $\mu \in \mathrm{R}^n$. Тогда вектор $x' = \mu + V^{\mathsf{T}}x$ имеет многомерное нормальное распределение $\mathrm{N}(\mu, \Sigma)$ с вектором матожидания μ и ковариационной матрицей $\Sigma = V^{\mathsf{T}}V$.

Утв. 7. Пусть на вероятностном пространстве X заданы k плотностей распределения $p_1(x), \ldots, p_k(x)$. Пусть дискретная случайная величина ξ принимает значения $1, \ldots, k$ с вероятностями $w_1, \ldots, w_k$. Тогда случайный элемент $x \in X$, полученный согласно распределению $p_\xi(x)$, подчиняется *смеси распределений* $p(x) = \sum_{j=1}^{k} w_j p_j(x)$. На практике часто используют смеси многомерных нормальных распределений.

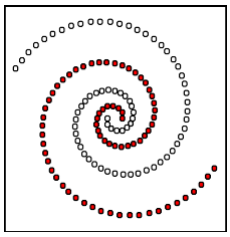


Рис. 1. Модельная выборка «спиралис.

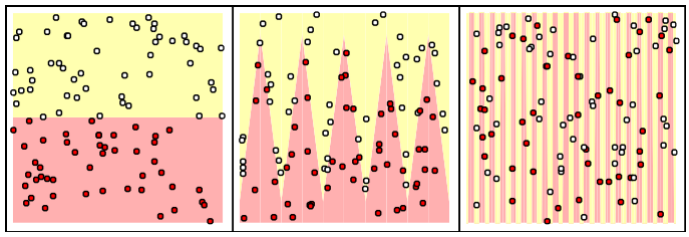


Рис. 2. Серия модельных выборок «пилас.

Утв. 8. Предыдущий случай обобщается на континуальные смеси распределений. Пусть на вероятностном пространстве X задано параметрическое семейство плотностей распределения $p(x, t)$, где $t \in \mathrm{R}$ — параметр. Пусть значение $\tau \in \mathrm{R}$ взято из распределения с плотностью $w(t)$. Тогда случайный элемент $x \in X$, полученный согласно распределению $p(x, \tau)$, подчиняется распределению $p(x) = \int_{-\infty}^{+\infty} w(t)p(x, t)\,dt$. Этот метод, называемый *методом суперпозиций*, позволяет моделировать широкий класс вероятностных распределений, представимых интегралом указанного вида.

Утв. 9. Пусть в R^n задана прямоугольная область $\Pi = [a_1, b_1] \times \ldots \times [a_n, b_n]$ и произвольное подмножество $G \subset \Pi$. Пусть $r = (r_1, \ldots, r_n)$ — вектор из n независимых случайных величин r_i, равномерно распределённых на $[a_i, b_i]$. *Метод исключения* состоит в том, чтобы генерировать случайный вектор r до тех пор, пока не выполнится условие $r \in G$. Тогда результирующий вектор r равномерно распределён на G. Этот метод вычислительно неэффективен, если объём G много меньше объёма Π.

Неслучайные модельные данные позволяют наглядно продемонстрировать, в каких случаях одни методы работают лучше других.

Один из классических примеров — две спирали на Рис. 1. Эта выборка хорошо классифицируется методом ближайших соседей, но непреодолимо трудна для линейных разделяющих правил. Если витки спиралей расположить ближе друг к другу, задача станет трудна и для метода ближайших соседей. Некоторые кусочно-линейные разделители справляются с задачей и в этом случае.

Обычно при создании модельных данных, как случайных, так и неслучайных, вводится параметр, плавно изменяющий задачу от предельно простой до предельно трудной. Это позволяет исследовать границы применимости метода. На Рис. 2 показана серия модельных задач классификации с двумя классами, обладающая таким свойством относительно метода ближайших соседей и некоторых других алгоритмов.

2 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.

§2.1 Вероятностная постановка задачи классификации

Пусть X — множество объектов, Y — конечное множество имён классов, множество $X \times Y$ является вероятностным пространством с плотностью распределения $p(x, y) = \mathsf{P}(y)p(x\,|\,y)$. Вероятности появления объектов каждого из классов $P_y = \mathsf{P}(y)$ называются *априорными вероятностями* классов. Плотности распределения $p_y(x) = p(x\,|\,y)$ называются *функциями правдоподобия* классов[3]. Вероятностная постановка задачи классификации разделяется на две независимые подзадачи.

Задача 2.1. *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[4] априорных вероятностей $\hat{P}_y$ и функций правдоподобия $\hat{p}_y(x)$ для каждого из классов $y \in Y$.*

Задача 2.2. *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.

Функционал среднего риска

Знание функций правдоподобия позволяет находить вероятности событий вида «$x \in \Omega$ при условии, что x принадлежит классу yk:

$$\mathsf{P}(\Omega|y) = \int_\Omega p_y(x)\,dx, \quad \Omega \subset X.$$

Рассмотрим произвольный алгоритм $a\colon X \to Y$. Он разбивает множество X на непересекающиеся области $A_y = \{\, x \in X \mid a(x) = y \,\}$ $y \in Y$. Вероятность того, что появится объект класса y и алгоритм a отнесёт его к классу s, равна $P_y\,\mathsf{P}(A_s|\,y)$. 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 $y = s$. Соотношения потерь на разных классах, как правило, известны заранее.

[3] Большой буквой P будем обозначать вероятности, а строчной p — плотности распределения.

[4] Символами с «крышечкойс принято обозначать *выборочные* (*эмпирические*) оценки вероятностей, функций распределения или случайных величин, вычисляемые по выборке.

Опр. 2.1. *Функционалом среднего риска называется ожидаемая величина потери при классификации объектов алгоритмом a:*

$$R(a) = \sum_{y\in Y}\sum_{s\in Y} \lambda_{ys} P_y \, \mathsf{P}(A_s|y).$$

Если величина потерь одинакова для ошибок любого рода, $\lambda_{ys} = [y = s]$, то средний риск $R(a)$ совпадает с вероятностью ошибки алгоритма a.

Оптимальное байесовское решающее правило

Теорема 2.1. *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. Для произвольного $t \in Y$ запишем функционал среднего риска:

$$R(a) = \sum_{y\in Y}\sum_{s\in Y} \lambda_{ys} P_y \, \mathsf{P}(A_s|y) =$$
$$= \sum_{y\in Y} \lambda_{yt} P_y \, \mathsf{P}(A_t|y) + \sum_{s\in Y\setminus\{t\}} \sum_{y\in Y} \lambda_{ys} P_y \, \mathsf{P}(A_s|y).$$

Применив формулу полной вероятности, $\mathsf{P}(A_t|y) = 1 - \sum_{s\in Y\setminus\{t\}} \mathsf{P}(A_s|y)$, получим:

$$R(a) = \sum_{y\in Y} \lambda_{yt} P_y + \sum_{s\in Y\setminus\{t\}} \sum_{y\in Y} (\lambda_{ys} - \lambda_{yt}) P_y \, \mathsf{P}(A_s|y) =$$
$$= \text{const}(a) + \sum_{s\in Y\setminus\{t\}} \int_{A_s} \sum_{y\in Y} (\lambda_{ys} - \lambda_{yt}) P_y p_y(x) \; dx. \qquad (2.1)$$

Введём для сокращения записи обозначение $g_s(x) = \sum_{y\in Y} \lambda_{ys} P_y p_y(x)$, тогда

$$R(a) = \text{const}(a) + \sum_{s\in Y\setminus\{t\}} \int_{A_s} g_s(x) - g_t(x) \quad dx.$$

В выражении (2.1) неизвестны только области A_s. Функционал $R(a)$ есть сумма $|Y|-1$ слагаемых $I(A_s) = \int_{A_s} g_s(x) - g_t(x) \; dx$, каждое из которых зависит только от одной области A_s. Минимум $I(A_s)$ достигается, когда A_s совпадает с областью неположительности подынтегрального выражения. В силу произвольности t

$$A_s = \{ x \in X \cdot g_s(x) \le g_t(x), \; \forall t \in Y, \; t \neq s \}.$$

С другой стороны, $A_s = \{ x \in X \cdot a(x) = s \}$. Значит, $a(x) = s$ тогда и только тогда, когда $s = \arg\min_{t\in Y} g_t(x)$. Если минимум $g_t(x)$ достигается при нескольких значениях t, то можно взять любое из них, что не повлияет на риск $R(a)$, так как подынтегральное выражение в этом случае равно нулю.

Теорема доказана. □

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 2.2. *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*

$$a(x) = \arg\max_{y \in Y} \lambda_y P_y p_y(x). \tag{2.2}$$

Доказательство. Рассмотрим выражение (2.1) из доказательства Теоремы 2.1. Поскольку λ_{ys} не зависит от второго индекса, то для любых $s, t \in Y$

$$\lambda_{ys} - \lambda_{yt} = \lambda_t[y = t] - \lambda_s[y = s].$$

Следовательно,

$$\sum_{y \in Y} (\lambda_{ys} - \lambda_{yt}) P_y p_y(x) = \lambda_t P_t p_t(x) - \lambda_s P_s p_s(x) = \tilde{g}_t(x) - \tilde{g}_s(x),$$

где $\tilde{g}_y(x) = \lambda_y P_y p_y(x)$ для всех $y \in Y$. Аналогично доказательству Теоремы 2.1 отсюда вытекает, что $a(x) = s$ при тех x, для которых $\tilde{g}_s(x)$ максимально по $s \in Y$. □

Разделяющая поверхность между классами s и t — это геометрическое место точек x таких, что максимум в (2.2) 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 $\mathsf{P}(y|x)$. Она может быть вычислена по формуле Байеса, если известны $p_y(x)$ и P_y:

$$\mathsf{P}(y|x) = \frac{p(x, y)}{p(x)} = \frac{p_y(x) P_y}{\sum_{s \in Y} p_s(x) P_s}.$$

Во многих приложениях важно не только классифицировать объект x, но и сказать, с какой вероятностью $\mathsf{P}(y|x)$ он принадлежит каждому из классов. Через апостериорные вероятности выражается величина ожидаемых потерь на объекте x:

$$R(x) = \sum_{y \in Y} \lambda_y \mathsf{P}(y|x).$$

Принцип максимума апостериорной вероятности. Оптимальный алгоритм классификации (2.2) можно переписать через апостериорные вероятности:

$$a(x) = \arg\max_{y \in Y} \lambda_y \mathsf{P}(y|x).$$

Поэтому выражение (2.2) 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.

Задача восстановления плотности распределения

Перейдём к Задаче 2.1. Требуется оценить, какой могла бы быть плотность вероятностного распределения $p(x, y) = P_y p_y(x)$, сгенерировавш его выборк у X^ℓ. }

Обозначим подвыборку прецедентов класса y через $X_y^\ell = (x_i, y_i)_{i=1}^\ell \cdot y_i = y$.

Проще всего оценить априорные вероятности классов P_y. Согласно закону больших чисел, частота появления объектов каждого из классов

$$\hat{P}_y = \frac{\ell_y}{\ell}, \quad \ell_y = |X_y^\ell|, \quad y \in Y, \tag{2.3}$$

сходится по вероятности к P_y при $\ell_y \to \infty$. Чем больше длина выборки, тем точнее выборочная оценка $\hat{P}_y$. Оценка (2.3) 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 (2.3), 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 2.3. *Задано множество объектов $X^m = \{x_1, \ldots, x_m\}$, 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 $f_j : X \to \mathrm{R}$, $j = 1, \ldots, n$. Обозначим через $x = (\xi_1, \ldots, \xi_n)$ произвольный элемент пространства объектов $X = \mathrm{R}^n$, где $\xi_j = f_j(x)$.

Гипотеза 2.1. *Признаки $f_1(x), \ldots, f_n(x)$ являются независимыми случайными величинами. Следовательно, функции правдоподобия классов представимы в виде*

$$p_y(x) = p_{y1}(\xi_1) \cdots p_{yn}(\xi_n), \qquad y \in Y, \tag{2.4}$$

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 (2.4), называются *наивными байесовскими* (na¨ıve Bayes).

Подставим эмпирические оценки одномерных плотностей $\hat{p}_{yj}(\xi)$ в (2.4) и затем в (2.2). Получим алгоритм

$$a(x) = \arg\max_{y \in Y} \left(\ln \lambda_y \hat{P}_y + \sum_{j=1}^{} \ln \hat{p}_{yj}(\xi_j) \right) . \tag{2.5}$$

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 недостаток — низкое качество классификации. Он используется либо как эталон при экспериментальном сравнении алгоритмов, либо как элементарный «строительный блокk в алгоритмических композициях, **??**.

§2.2 Непараметрическая классификация

Непараметрические методы классификации основаны на локальном оценивании плотностей распределения классов $p_y(x)$ в окрестности классифицируемого объекта $x \in X$. Для классификации объекта x применяется основная формула (2.2).

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 «наивныхk байесовских классификаторов (2.5). 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 $X^m = (x_i)_{i=1}^m$:

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1} [x_i = x]. \tag{2.6}$$

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 $X = \mathrm{R}$. Согласно определению плотности, $p(x) = \lim_{h \to} \frac{1}{2h} P[x - h, x + h]$, 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*:

$$\hat{p}_h(x) = \frac{1}{2mh} \sum_{i=1} |x - x_i| < h \ . \tag{2.7}$$

Функция $\hat{p}_h(x)$ является кусочно-постоянной, что приводит к появлению широких *зон неуверенности*, в которых максимум (2.2) достигается одновременно для нескольких классов $y \in Y$. Проблема решается с помощью *локальной непараметрической оценки* Парзена-Розенблатта [57, 56]:

$$\hat{p}_h(x) = \frac{1}{mh} \sum_{i=1} K \left(\frac{x - x_i}{h} \right), \tag{2.8}$$

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: $\int \hat{p}_h(x)\,dx = 1$ при любом h.

На практике часто используются ядра, показанные на рис. 3, стр. 25.

Прямоугольное ядро $K(z) = \frac{1}{2}|z| < 1$ соответствует простейшей оценке (2.7).

Точечное ядро $K(z) = [z = 0]$ при $h = 1$ соответствует дискретному случаю (2.6).

Обоснованием оценки (2.7) 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 2.3 ([56, 57, 25]). *Пусть выполнены следующие условия:*

1) *выборка X^m простая, получена из плотности распределения $p(x)$;*
2) *ядро $K(z)$ непрерывно, его квадрат ограничен: $\int_X K^2(z)\,dz < \infty$;*
3) *последовательность h_m такова, что $\lim_{m\to\infty} h_m = 0$ и $\lim_{m\to\infty} mh_m = \infty$.*

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 $f_j : X \to \mathrm{R}$, $j = 1, \ldots, n$. Тогда непараметрическая оценка плотности в точке $x \in X$ записывается в следующем виде [9, 17]:

$$\hat{p}_h(x) = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{n}\frac{1}{h_j}K\left(\frac{f_j(x) - f_j(x_i)}{h_j}\right). \tag{2.9}$$

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 (2.8), 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 (2.8) легко обобщается и на этот случай:

$$\hat{p}_h(x) = \frac{1}{mV(h)}\sum_{i=1}^{m} K\left(\frac{\rho(x, x_i)}{h}\right), \tag{2.10}$$

где $V(h)$ — нормирующий множитель, гарантирующий, что $\hat{p}_h(x)$ действительно является плотностью. Сходимость оценки (2.10) доказана при некоторых дополнительных ограничениях на ядро K и метрику ρ, причём скорость сходимости лишь немного хуже, чем в одномерном случае [22].

Метод парзеновского окна

Запишем парзеновскую оценку плотности (2.10) для каждого класса $y \in Y$:

$$\hat{p}_{y,h}(x) = \frac{1}{\ell_y V(h)}\sum_{i=1}^{m} [y_i = y]\, K\left(\frac{\rho(x, x_i)}{h}\right), \tag{2.11}$$

где K — ядро, h — ширина окна. Если нормирующий множитель $V(h)$ не зависит от y, то в байесовском классификаторе (2.2) его можно убрать из-под знака arg max и вообще не вычислять. Подставим оценку плотности (2.11) и оценку априорной вероятности классов $\hat{P}_y = \ell_y/\ell$ в формулу (2.2):

$$a(x; X^\ell, h) = \arg\max_{y \in Y} \lambda_y \sum_{i=1}^{\ell} [y_i = y] K\left(\frac{\rho(x, x_i)}{h}\right). \qquad (2.12)$$

Выборка X^ℓ сохраняется «как есть» и играет роль параметра алгоритма.

Если метрика ρ фиксирована, то обучение парзеновского классификатора (2.12) сводится к подбору ширины окна h и вида ядра K.

Ширина окна h существенно влияет на качество восстановления плотности. При $h \to 0$ плотность концентрируется вблизи обучающих объектов, и функция $\hat{p}_h(x)$ претерпевает резкие скачки. При $h \to \infty$ плотность чрезмерно сглаживается и вырождается в константу. Следовательно, должно существовать компромиссное значение ширины окна h^*. На практике его находят по скользящему контролю:

$$\mathrm{LOO}(h, X^\ell) = \sum_{i=1}^{\ell} a\left(x_i; X^\ell \setminus x_i, h\right) \neq y_i \to \min_h,$$

где $a(x; X^\ell \setminus x_i, h)$ — алгоритм классификации, построенный по обучающей выборке X^ℓ без объекта x_i. Обычно зависимость $\mathrm{LOO}(h)$ имеет характерный минимум, соответствующий оптимальной ширине окна h^*.

Переменная ширина окна $h(x)$. Если распределение объектов в пространстве X сильно неравномерно, то возникает *проблема локальных сгущений*. Одно и то же значение ширины окна h приводит к чрезмерному сглаживанию плотности в одних областях пространства X, и недостаточному сглаживанию в других. Проблему решает *переменная ширина окна*, определяемая в каждой точке $x \in X$ как расстояние до $(k+1)$-го соседа $h(x) = \rho\left(x, x^{(k+1)}\right)$, если считать, что обучающие объекты ранжированы по возрастанию расстояний до x.

Нормирующий множитель $V(h)$ не должен зависеть от y, поэтому в числе соседей должны учитываться объекты всех классов, хотя плотности $\hat{p}_{y,h(x)}(x)$ оцениваются по подвыборкам X_y^ℓ для каждого класса $y \in Y$ в отдельности. Оптимальное значение k^* определяется по критерию скользящего контроля, аналогично h^*.

Функция ядра K практически не влияет на качество восстановления плотности и на качество классификации. В то же время, она определяет степень гладкости функции $\hat{p}_h(x)$. Вид ядра может также влиять на эффективность вычислений. Гауссовское ядро G требует просмотра всей выборки для вычисления значения $\hat{p}_h(x)$ в произвольной точке x. Ядра E, Q, T, Π являются финитными (имеют ограниченный носитель, рис. 3), и для них достаточно взять только те точки выборки, которые попадают в окрестность точки x радиуса h.

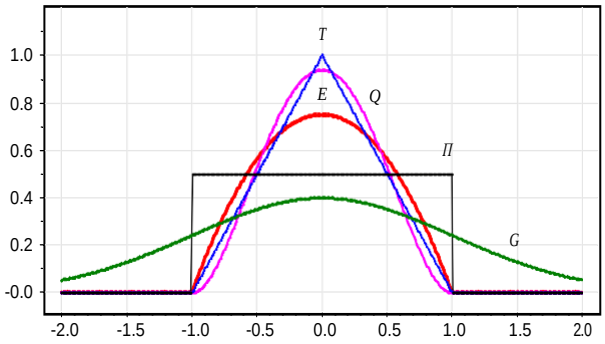


Рис. 3. Часто используемые ядра:
E — Епанечникова;
Q — квартическое;
T — треугольное;
G — гауссовское;
П — прямоугольное.

Проблема «проклятия размерности». Если используемая метрика $\rho(x, x')$ основана на суммировании различий по всем признакам, а число признаков очень велико, то все точки выборки могут оказаться практически одинаково далеки друг от друга. Тогда парзеновские оценки плотности становятся неадекватны. Это явление называют *проклятием размерности* (curse of dimensionality). Выход заключается в понижении размерности с помощью преобразования пространства признаков (см. раздел **??**), либо путём отбора информативных признаков (см. раздел **??**). Можно строить несколько альтернативных метрик в подпространствах меньшей размерности, и полученные по ним алгоритмы классификации объединять в композицию. На этой идее основаны алгоритмы вычисления оценок, подробно описанные в **??**.

§2.3 Нормальный дискриминантный анализ

В *параметрическом подходе* предполагается, что плотность распределения выборки $X^m = (x_1, \ldots, x_m)$ известна с точностью до параметра, $p(x) = \phi(x; \vartheta)$, где ϕ — фиксированная функция. Вектор параметров ϑ оценивается по выборке X^m с помощью *принципа максимума правдоподобия* (maximum likelihood).

Нормальный дискриминантный анализ — это специальный случай байесовской классификации, когда предполагается, что плотности всех классов $p_y(x)$, $y \in Y$ являются многомерными нормальными. Этот случай интересен и удобен тем, что задача оценивания параметров распределения по выборке решается аналитически.

Многомерное нормальное распределение

Пусть $X = \mathrm{R}^n$, то есть объекты описываются n числовыми признаками.

Опр. 2.2. *Вероятностное распределение с плотностью*

$$\mathrm{N}(x; \mu, \Sigma) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu)\right), \quad x \in \mathrm{R}^n,$$

называется n-мерным нормальным (гауссовским) распределением с математическим ожиданием (центром) $\mu \in \mathrm{R}^n$ и ковариационной матрицей $\Sigma \in \mathrm{R}^{n\times n}$. Предполагается, что матрица Σ симметричная, невырожденная, положительно определённая.

Интегрируя по R^n, можно убедиться в том, что это действительно распределение, а параметры μ и Σ оправдывают своё название:

$$\int \mathrm{N}(x;\mu,\Sigma)\,dx = 1;$$

$$\mathrm{E}x = \int x\mathrm{N}(x;\mu,\Sigma)\,dx = \mu;$$

$$\mathrm{E}(x-\mu)(x-\mu)^{\mathsf{T}} = \int (x-\mu)(x-\mu)^{\mathsf{T}}\mathrm{N}(x;\mu,\Sigma)\,dx = \Sigma.$$

Геометрическая интерпретация нормальной плотности. Если признаки некоррелированы, $\Sigma = \mathrm{diag}(\sigma_1^2,\ldots,\sigma_n^2)$, то линии уровня плотности распределения имеют форму эллипсоидов с центром μ и осями, параллельными линиям координат. Если признаки имеют одинаковые дисперсии, $\Sigma = \sigma^2 I_n$, то эллипсоиды являются сферами.

Если признаки коррелированы, то матрица Σ не диагональна и линии уровня имеют форму эллипсоидов, оси которых повёрнуты относительно исходной системы координат. Действительно, как всякая симметричная матрица, Σ имеет спектральное разложение $\Sigma = V S V^{\mathsf{T}}$, где $V = (v_1,\ldots,v_n)$ — ортогональные собственные векторы матрицы Σ, соответствующие собственным значениям $\lambda_1,\ldots,\lambda_n$, матрица S диагональна, $S = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$. Тогда $\Sigma^{-1} = V S^{-1} V^{\mathsf{T}}$, следовательно,

$$(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu) = (x-\mu)^{\mathsf{T}} V S^{-1} V^{\mathsf{T}}(x-\mu) = (x'-\mu')^{\mathsf{T}} S^{-1}(x'-\mu').$$

Это означает, что в результате ортогонального преобразования координат $x' = V^{\mathsf{T}}x$ оси эллипсоидов становятся параллельны линиям координат. В новых координатах ковариационная матрица S является диагональной. Поэтому линейное преобразование V называется *декоррелирующим*. В исходных координатах оси эллипсоидов направлены вдоль собственных векторов матрицы Σ.

Квадратичный дискриминант

Рассмотрим задачу классификации с произвольным числом классов.

Теорема 2.4. *Если классы имеют n-мерные нормальные плотности распределения*

$$p_y(x) = \mathrm{N}(x;\mu_y,\Sigma_y), \qquad y \in Y.$$

то байесовский классификатор задаёт квадратичную разделяющую поверхность. Она вырождается в линейную, если ковариационные матрицы классов равны.

Доказательство. Поверхность, разделяющая классы s и t, описывается уравнением $\lambda_s P_s p_s(x) = \lambda_t P_t p_t(x)$, которое после логарифмирования принимает вид

$$\ln p_s(x) - \ln p_t(x) = C_{st},$$

где $C_{st} = \ln(\lambda_t P_t / \lambda_s P_s)$ — константа, не зависящая от x. Разделяющая поверхность в общем случае квадратична, поскольку $\ln p_y(x)$ является квадратичной формой по x:

$$\ln p_y(x) = -\frac{n}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_y| - \frac{1}{2}(x-\mu_y)^{\mathsf{T}}\Sigma_y^{-1}(x-\mu_y).$$

Если $\Sigma_s = \Sigma_t \equiv \Sigma$, то квадратичные члены сокращаются и уравнение поверхности вырождается в линейную форму:

$$x^\intercal \Sigma^{-1}(\mu_s - \mu_t) - \tfrac{1}{2}\mu_s^\intercal \Sigma^{-1}\mu_s + \tfrac{1}{2}\mu_t^\intercal \Sigma^{-1}\mu_t = C_{st};$$
$$(x - \mu_{st})^\intercal \Sigma^{-1}(\mu_s - \mu_t) = C_{st};$$

где $\mu_{st} = \frac{1}{2}(\mu_s + \mu_t)$ — точка посередине между центрами классов. □

Геометрия разделяющих поверхностей. Рассмотрим простейший случай, когда классы s, t равновероятны и равнозначны ($\lambda_s P_s = \lambda_t P_t$), ковариационные матрицы равны, признаки некоррелированы и имеют одинаковые дисперсии ($\Sigma_s = \Sigma_t = \sigma I_n$). Это означает, что классы имеют одинаковую сферическую форму. В этом случае разделяющая гиперплоскость проходит посередине между классами, ортогонально линии, соединяющей центры классов. Нормаль гиперплоскости обладает свойством оптимальности — это та прямая, в одномерной проекции на которую классы разделяются наилучшим образом, то есть с наименьшим значением байесовского риска $R(a)$.

Если признаки коррелированы ($\Sigma_s = \Sigma_t = \sigma I_n$), то ортогональность исчезает, однако разделяющая гиперплоскость по-прежнему проходит посередине между классами, касательно к линиям уровня обоих распределений.

Если классы не равновероятны или не равнозначны ($\lambda_s P_s = \lambda_t P_t$), то разделяющая гиперплоскость отодвигается дальше от более значимого класса.

Если ковариационные матрицы не диагональны и не равны, то разделяющая поверхность становится квадратичной и «прогибаетсяk так, чтобы менее плотный класс «охватывалk более плотный.

В некоторых случаях более плотный класс «разрезаетk менее плотный на две несвязные области. Это может приводить к парадоксальным ситуациям. Например, может возникнуть область, в которой объекты будут относиться к менее плотному классу, хотя объекты более плотного класса находятся ближе.

Если число классов превышает 2, то разделяющая поверхность является кусочно-квадратичной, а при равных ковариационных матрицах — кусочно-линейной.

Расстояние Махаланобиса. Если классы равновероятны и равнозначны, ковариационные матрицы равны, то уравнение разделяющей поверхности принимает вид

$$(x - \mu_s)^\intercal \Sigma^{-1}(x - \mu_s) = (x - \mu_t)^\intercal \Sigma^{-1}(x - \mu_t);$$
$$\|x - \mu_s\|_\Sigma = \|x - \mu_t\|_\Sigma;$$

где $\|u - v\|_\Sigma \equiv \sqrt{(u - v)^\intercal \Sigma^{-1}(u - v)}$ — метрика в $\mathbb{R}^n$, называемая *расстоянием Махаланобиса*. Разделяющая поверхность является геометрическим местом точек, равноудалённых от центров классов в смысле расстояния Махаланобиса.

Если признаки независимы и имеют одинаковые дисперсии, то расстояние Махаланобиса совпадает с евклидовой метрикой. В этом случае оптимальным (байесовским) решающим правилом является *классификатор ближайшего среднего* (nearest mean classifier), относящий объект к классу с ближайшим центром.

Принцип максимума правдоподобия составляет основу параметрического подхода. Пусть задано множество объектов $X^m = \{x_1, \ldots, x_m\}$, выбранных независимо друг от друга из вероятностного распределения с плотностью $\phi(x; \vartheta)$. *Функцией правдоподобия* называется совместная плотность распределения всех объектов выборки:

$$p(X^m; \vartheta) = p(x_1, \ldots, x_m; \vartheta) = \prod_{i=1} \phi(x_i; \vartheta).$$

Значение параметра ϑ, при котором выборка максимально правдоподобна, то есть функция $p(X^m; \vartheta)$ принимает максимальное значение, называется *оценкой максимума правдоподобия*. Как известно из математической статистики, эта оценка обладает рядом оптимальных свойств [13, 16].

Вместо максимизации правдоподобия удобнее максимизировать его логарифм:

$$L(X^m; \vartheta) = \sum_{i=1} \ln \phi(x_i; \vartheta) \to \max_{\vartheta}. \tag{2.13}$$

Для решения этой задачи можно использовать стандартные методы оптимизации. В некоторых случаях решение выписывается в явном виде, исходя из необходимого условия оптимума (если функция $\phi(x; \vartheta)$ достаточно гладкая по параметру ϑ):

$$\frac{\partial}{\partial \vartheta} L(X^m; \vartheta) = \sum_{i=1} \frac{\partial}{\partial \vartheta} \ln \phi(x_i; \vartheta) = 0. \tag{2.14}$$

Выборочные оценки параметров нормального распределения. В случае гауссовской плотности с параметрами $\vartheta = (\mu, \Sigma)$ задача максимизации правдоподобия имеет аналитическое решение, которое получается из уравнений (2.14).

Теорема 2.5. *Пусть задана независимая выборка объектов $X^m = (x_1, \ldots, x_m)$. Тогда оценки параметров гауссовской плотности $\phi(x; \vartheta) \equiv \mathrm{N}(x; \mu, \Sigma)$, доставляющие максимум функционалу правдоподобия (2.13), имеют вид*

$$\hat{\mu} = \frac{1}{m} \sum_{i=1} x_i; \qquad \hat{\Sigma} = \frac{1}{m} \sum_{i=1} (x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathsf{T}}.$$

Поправка на смещение. Естественным требованием к оценке параметра распределения является её несмещённость.

Опр. 2.3. *Пусть X^m есть выборка случайных независимых наблюдений, полученная согласно распределению $\phi(x; \vartheta)$ при фиксированном $\vartheta = \vartheta_0$. Оценка $\hat{\vartheta}(X^m)$ параметра ϑ, вычисленная по выборке X^m, называется несмещённой, если $\mathsf{E}_{X^m} \hat{\vartheta}(X^m) = \vartheta_0$.*

Оценка $\hat{\mu}$ является несмещённой, $\mathsf{E}\hat{\mu} = \mu$. Однако оценка $\hat{\Sigma}$ уже смещена: $\mathsf{E}\hat{\Sigma} = \frac{m}{m-1}\Sigma$. Это связано с тем, что при вычислении $\hat{\Sigma}$ вместо неизвестного точного значения матожидания μ приходится подставлять его выборочную оценку $\hat{\mu}$. Для учёта этого небольшого смещения вводится *поправка на смещение*:

$$\hat{\Sigma} = \frac{1}{m-1} \sum_{x=1} (x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathsf{T}}. \tag{2.15}$$

Подстановочный алгоритм. Оценим параметры функций правдоподобия $\hat{\mu}_y$ и $\hat{\Sigma}_y$ по частям обучающей выборки $X_y^\ell = \{ x_i \in X^\ell \mid y_i = y \}$ для каждого класса $y \in Y$. Затем эти выборочные оценки подставим в формулу (2.2). Получим *байесовский нормальный классификатор*, который называется также *подстановочным* (plug-in).

В асимптотике $\ell_y \to \infty$ оценки $\hat{\mu}_y$ и $\hat{\Sigma}_y$ обладают рядом оптимальных свойств: они не смещены, состоятельны и эффективны. Однако оценки, сделанные по коротким выборкам, могут быть не достаточно точными.

Недостатки подстановочного алгоритма вытекают из нескольких чрезмерно сильных базовых предположений, которые на практике часто не выполняются.

- Функции правдоподобия классов могут существенно отличаться от гауссовских. В частности, когда имеются признаки, принимающие дискретные значения, или когда классы распадаются на изолированные сгустки.
- Если длина выборки меньше размерности пространства, $\ell_y < n$, или среди признаков есть линейно зависимые, то матрица $\hat{\Sigma}_y$ становится вырожденной. В этом случае обратная матрица не существует и метод вообще неприменим.
- На практике встречаются задачи, в которых признаки «почти линейно зависимы». Тогда матрица $\hat{\Sigma}_y$ является *плохо обусловленной*, то есть близкой к некоторой вырожденной матрице. Это так называемая *проблема мультиколлинеарности*, которая влечёт неустойчивость как самой обратной матрицы $\hat{\Sigma}_y^{-1}$, так и вектора $\hat{\Sigma}_y^{-1}(x - \mu_{st})$. Они могут непредсказуемо и сильно изменяться при незначительных вариациях исходных данных, например, связанных с погрешностями измерений. Неустойчивость снижает качество классификации.
- Выборочные оценки чувствительны к нарушениям нормальности распределений, в частности, к редким большим выбросам.

Наивный нормальный байесовский классификатор. Предположим, что все признаки $f_j(x)$ независимы и нормально распределены с матожиданием μ_{yj} и дисперсией σ_{yj}, вообще говоря, отличающимися для разных классов:

$$p_{yj}(\xi) = \frac{1}{\sqrt{2\pi\sigma_{yj}}} \exp\left(-\frac{(\xi - \mu_{yj})^2}{2\sigma_{yj}^2}\right), \quad y \in Y, \quad j = 1, \ldots, n.$$

Тогда, как нетрудно убедиться, ковариационные матрицы Σ_y и их выборочные оценки $\hat{\Sigma}_y$ будут диагональны. В этом случае проблемы вырожденности и мультиколлинеарности не возникают. Метод обучения до крайности прост и сводится к вычислению параметров $\hat{\mu}_{yj}$ и $\hat{\sigma}_{yj}$ для всех $y \in Y$ и всех признаков $j = 1, \ldots, n$.

Линейный дискриминант Фишера

В 1936 г. Р. Фишер предложил простую эвристику, позволяющую увеличить число объектов, по которым оценивается ковариационная матрица, повысить

её устойчивость и заодно упростить алгоритм обучения [41]. Предположим, что ковариационные матрицы классов одинаковы и равны Σ. Оценим $\hat{\Sigma}$ по всем ℓ обучающим объектам. С учётом поправки на смещённость,

$$\hat{\Sigma} = \frac{1}{\ell - |Y|} \sum_{i=1}^{\ell} (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^{\mathsf{T}}$$

Согласно теореме 2.4, разделяющая поверхность линейна, если классов два, и кусочно-линейна, если классов больше. Запишем подстановочный алгоритм:

$$\begin{aligned} a(x) &= \arg\max_{y \in Y} \lambda_y P_y p_y(x) = \\ &= \arg\max_{y \in Y} \ln(\lambda_y P_y) - \tfrac{1}{2}\hat{\mu}_y^{\mathsf{T}} \hat{\Sigma}^{-1} \hat{\mu}_y + x^{\mathsf{T}} \hat{\Sigma}^{-1} \hat{\mu}_y = \\ &= \arg\max_{y \in Y} x^{\mathsf{T}} \underbrace{\hat{\Sigma}^{-1} \hat{\mu}_y}_{\alpha_y} + \underbrace{\ln(\lambda_y P_y) - \tfrac{1}{2}\hat{\mu}_y^{\mathsf{T}} \hat{\Sigma}^{-1} \hat{\mu}_y}_{\beta_y}. \end{aligned} \tag{2.16}$$

Этот алгоритм называется *линейным дискриминантом Фишера* (ЛДФ). Он неплохо работает, когда формы классов действительно близки к нормальным и не слишком сильно различаются. В этом случае линейное решающее правило близко к оптимальному байесовскому, но существенно более устойчиво, чем квадратичное, и часто обладает лучшей обобщающей способностью.

Вероятность ошибки линейного дискриминанта Фишера выражается через расстояние Махаланобиса между классами, в случае, когда классов два [28]:

$$R(a) = \Phi\left(-\tfrac{1}{2}\|\mu_1 - \mu_2\|_{\Sigma}\right),$$

где $\Phi(r) = \mathrm{N}(x; 0, 1)$ — функция стандартного нормального распределения.

Регуляризация ковариационной матрицы. Эвристика Фишера не является радикальным решением проблемы мультиколлинеарности: общая ковариационная матрица классов $\hat{\Sigma}$ также может оказаться плохо обусловленной (близкой к вырожденной). Признаком плохой обусловленности является наличие у матрицы собственных значений, близких к нулю. Поэтому один из способов решения проблемы — модифицировать матрицу $\hat{\Sigma}$ так, чтобы все её собственные значения λ увеличились на заданное число τ, а все собственные векторы v сохранились. Для этого достаточно прибавить к матрице единичную, умноженную на τ:

$$(\hat{\Sigma} + \tau I_n)v = \lambda v + \tau v = (\lambda + \tau)v.$$

Известны и другие способы решения проблемы плохой обусловленности. Можно пропорционально уменьшать недиагональные элементы — вместо $\hat{\Sigma}$ брать матрицу $(1 - \tau)\hat{\Sigma} + \tau \operatorname{diag} \hat{\Sigma}$ [28]. Можно занулять недиагональные элементы матрицы, соответствующие тем парам признаков, ковариации которых незначимо отличаются от нуля [2]; при этом матрица становится разреженной, и для её обращения могут применяться специальные, более эффективные, алгоритмы. Можно разбивать множество признаков на группы и полагать, что признаки из разных групп не коррелированы. Тогда матрица $\hat{\Sigma}$ приобретает блочно-диагональный вид. Для таких матриц также существуют специальные эффективные алгоритмы обращения.

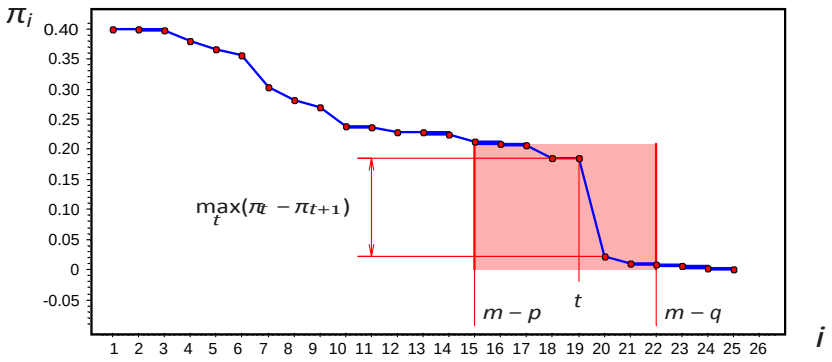


Рис. 4. Эксперт предположил, что в выборке длины $m = 25$ находится от $q = 3$ до $p = 10$ выбросов. Более точное число выбросов, равное 6, удалось определить по критерию «крутого склонас.

Отбор и преобразование признаков. Другой способ устранения мультиколлинеарности заключается в том, чтобы отбросить наименее значимые признаки. Различные методы *отбора признаков* (features selection) рассматриваются в разделе **??**. Обратим внимание на кажущийся парадокс: информация отбрасывается, но качество классификации повышается.

Существует простой «жадныйk метод отбора признаков, в котором исходная n-мерная задача сводится к серии двумерных задач. Это метод *редукции размерности* А. М. Шурыгина [28]. Сначала находятся два признака, в подпространстве которых линейный дискриминант Фишера наилучшим образом разделяет классы (т. е. байесовский риск $R(a)$ принимает наименьшее значение). Обозначим его параметры через $\{\alpha_y^{(2)}, \beta_y^{(2)} : y \in Y\}$, где в векторе $\alpha_y^{(2)} \in \mathrm{R}^n$ лишь два коэффициента не равны нулю. Функция проекции на нормаль разделяющей гиперплоскости принимается за новый признак: $\psi(x) = x^{\mathsf{T}}\alpha_y^{(2)}$. Из оставшихся $n-2$ признаков выбирается тот, который в паре с $\psi(x)$ наилучшим образом разделяет классы. Снова строится двумерный линейный дискриминант и определяются параметры $\{\alpha_y^{(3)}, \beta_y^{(3)} : y \in Y\}$, где в векторе $\alpha_y^{(3)} \in \mathrm{R}^n$ уже три коэффициента не равны нулю. Так продолжается до тех пор, пока присоединение новых признаков улучшает качество классификации (понижает байесовский риск). Достоинства метода редукции — возможность отбросить неинформативные признаки и обойтись без обращения ковариационных матриц размера более 2×2. В некоторых прикладных задачах он превосходит другие методы классификации [28]. Недостатком является отсутствие строгого теоретического обоснования. Как и всякая «жаднаяk стратегия, он находит неоптимальный набор признаков. По всей видимости, этот метод подходит для тех задач, в которых имеется небольшое число признаков, существенно более информативных, чем остальные.

Ещё один способ сокращения размерности заключается в том, чтобы из имеющегося набора признаков построить новый набор, состоящий из наименьшего числа максимально информативных признаков. Оптимальное *линейное* преобразование пространства признаков строится с помощью *метода главных компонент* (principal component analysis), который будет рассмотрен в разделе §5.4.

Робастные методы оценивания. Оценки, устойчивые относительно редких больших выбросов или других несоответствий реальных данных модельному распределению $\phi(x;\vartheta)$, называются *робастными* (robust — здравый).

Простейший метод фильтрации выбросов состоит в двукратном решении задачи. Сначала вычисляется оценка максимума правдоподобия для параметра $\hat{\vartheta}$ по всей выборке X^m. Для каждого объекта $x_i \in X^m$ вычисляется значение правдоподобия $\pi_i = \phi(x_i; \hat{\vartheta})$, и объекты сортируются по убыванию правдоподобий: $\pi_1 \geq \ldots \geq \pi_m$. Объекты, оказавшиеся в конце этого ряда, считаются нетипичными (выбросами) и удаляются из выборки. Для этого может применяться критерий «крутого склонак: задаются два параметра p и q, и находится значение $t \in \{p, \ldots, m-q-1\}$, для которого скачок правдоподобия $\pi_t - \pi_{t+1}$ максимален. Затем последние $(m-t)$ объектов удаляются из выборки, см. рис. 4. После этого оценка $\hat{\vartheta}$ вычисляется вторично по отфильтрованной выборке. Если выбросов много, то, возможно, придётся провести несколько таких фильтраций подряд.

Заметим, что удаление объекта может не требовать полного пересчёта оценки $\hat{\vartheta}$. В частности, оценки нормального распределения $\hat{\mu}$, $\hat{\Sigma}$ аддитивны по объектам, и для них достаточно вычесть слагаемое, соответствующее удаляемому объекту.

§2.4 Разделение смеси распределений

В тех случаях, когда «формук класса не удаётся описать каким-либо одним распределением, можно попробовать описать её смесью распределений.

Гипотеза 2.2. *Плотность распределения на X имеет вид смеси k распределений:*

$$p(x) = \sum_{j=1} w_j p_j(x), \qquad \sum_{j=1} w_j = 1, \qquad w_j \geq 0,$$

где $p_j(x)$ — функция правдоподобия j-й компоненты смеси, w_j — её априорная вероятность. Функции правдоподобия принадлежат параметрическому семейству распределений $\phi(x;\vartheta)$ и отличаются только значениями параметра, $p_j(x) = \phi(x;\vartheta_j)$.

Иными словами, «выбрать объект x из смеси $p(x)$k означает сначала выбрать j-ю компоненту смеси из дискретного распределения $w_1, \ldots, w_k$, затем выбрать объект x согласно плотности $p_j(x)$.

Задача разделения смеси заключается в том, чтобы, имея выборку X^m случайных и независимых наблюдений из смеси $p(x)$, зная число k и функцию ϕ, оценить вектор параметров $\Theta = (w_1, \ldots, w_k, \vartheta_1, \ldots, \vartheta_k)$.

EM-алгоритм

К сожалению, попытка разделить смесь, используя принцип максимума правдоподобия «в лобk, приводит к слишком громоздкой оптимизационной задаче. Обойти эту трудность позволяет алгоритм EM (expectation-maximization). Идея алгоритма заключается в следующем. Искусственно вводится вспомогательный вектор *скрытых* (hidden) переменных G, обладающий двумя замечательными свойствами. С одной стороны, он может быть вычислен, если известны значения вектора параметров Θ. С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

ЕМ-алгоритм состоит из итерационного повторения двух шагов. На Е-шаге вычисляется ожидаемое значение (expectation) вектора скрытых переменных G по текущему приближению вектора параметров Θ. На М-шаге решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ.

Алгоритм 2.1. Общая идея ЕМ-алгоритма

1: Вычислить начальное приближение вектора параметров Θ;
2: **повторять**
3: $\quad G := \mathtt{EStep}(\Theta)$;
4: $\quad \Theta := \mathtt{MStep}(\Theta, G)$;
5: **пока** Θ и G не стабилизируются.

Этот алгоритм был предложен и исследован М. И. Шлезингером как инструмент для *самопроизвольной классификации образов* [27]. Двенадцать лет спустя он был открыт заново в [39] под названием *ЕМ-алгоритма*. Область его применения чрезвычайно широка — дискриминантный анализ, кластеризация, восстановление пропусков в данных, обработка сигналов и изображений [26]. Здесь мы рассматриваем его как инструмент разделения смеси распределений.

Е-шаг (expectation). Обозначим через $p(x, \vartheta_j)$ плотность вероятности того, что объект x получен из j-й компоненты смеси. По формуле условной вероятности

$$p(x, \vartheta_j) = p(x)\,\mathsf{P}(\vartheta_j \mid x) = w_j p_j(x).$$

Введём обозначение $g_{ij} \equiv \mathsf{P}(\vartheta_j \mid x_i)$. Это неизвестная апостериорная вероятность того, что обучающий объект x_i получен из j-й компоненты смеси. Возьмём эти величины в качестве скрытых переменных. Обозначим $G = (g_{ij})_{m\times k} = (g_1, \ldots, g_j)$, где g_j — j-й столбец матрицы G. Каждый объект обязательно принадлежит какой-то компоненте, поэтому справедлива формула полной вероятности:

$$\sum_{j=1} g_{ij} = 1 \quad \text{для всех } i = 1, \ldots, \ell.$$

Зная параметры компонент w_j, ϑ_j, легко вычислить g_{ij} по формуле Байеса:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \quad \text{для всех } i, j. \tag{2.17}$$

В этом и заключается Е-шаг алгоритма ЕМ.

М-шаг (maximization). Покажем, что знание значений скрытых переменных g_{ij} и принцип максимума правдоподобия приводят к оптимизационной задаче, допускающей эффективное численное (или даже аналитическое) решение. Будем максимизировать логарифм правдоподобия

$$Q(\Theta) = \ln \prod_{i=1} p(x_i) = \sum_{i=1} \ln \sum_{j=1} w_j p_j(x_i) \to \max_{\Theta}.$$

при ограничении $\sum_{j=1}^{k} w_j = 1$. Запишем лагранжиан этой оптимизационной задачи:

$$L(\Theta; X^m) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i) \quad - \lambda \sum_{j=1}^{k} w_j - 1 \ .$$

Приравняем нулю производную лагранжиана по w_j:

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{m} \frac{p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} - \lambda = 0, \quad j = 1, \ldots, k. \tag{2.18}$$

Умножим левую и правую части на w_j, просуммируем все k этих равенств, и поменяем местами знаки суммирования по j и по i:

$$\sum_{i=1}^{m} \sum_{j=1}^{k} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} = \lambda \sum_{j=1}^{k} w_j,$$

откуда следует $\lambda = m$.

Теперь снова умножим левую и правую части (2.18) на w_j, подставим $\lambda = m$, и, замечая сходство с формулой (2.17), получим выражение весов компонент через скрытые переменные:

$$w_j = \frac{1}{m} \sum_{i=1}^{m} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} = \frac{1}{m} \sum_{i=1}^{m} g_{ij}, \quad j = 1, \ldots, k. \tag{2.19}$$

Легко проверить, что ограничения-неравенства $w_j \geq 0$ будут выполнены на каждой итерации, если они выполнены для начального приближения.

Приравняем нулю производную лагранжиана по ϑ_j, помня, что $p_j(x) \equiv \phi(x; \vartheta_j)$:

$$\frac{\partial L}{\partial \vartheta_j} = \sum_{i=1}^{m} \frac{w_j}{\sum_{s=1}^{k} w_s p_s(x_i)} \frac{\partial}{\partial \vartheta_j} p_j(x_i) = \sum_{i=1}^{m} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \frac{\partial}{\partial \vartheta_j} \ln p_j(x_i) =$$

$$= \sum_{i=1}^{m} g_{ij} \frac{\partial}{\partial \vartheta_j} \ln p_j(x_i) = \frac{\partial}{\partial \vartheta_j} \sum_{i=1}^{m} g_{ij} \ln p_j(x_i) = 0, \quad j = 1, \ldots, k.$$

Полученное условие совпадает с необходимым условием максимума в задаче максимизации взвешенного правдоподобия

$$\vartheta_j := \arg\max_{\vartheta} \sum_{i=1}^{m} g_{ij} \ln \phi(x_i; \vartheta), \quad j = 1, \ldots, k. \tag{2.20}$$

Таким образом, М-шаг сводится к вычислению весов компонент w_j как средних арифметических (2.19) и оцениванию параметров компонент ϑ_j путём решения k независимых оптимизационных задач (2.20). Отметим, что разделение переменных оказалось возможным благодаря удачному введению скрытых переменных.

Условия сходимости алгоритма EM рассматриваются в работах [39, 67, 47].

Алгоритм 2.2. EM-алгоритм с фиксированным числом компонент

Вход:
выборка $X^m = \{x_1, \ldots, x_m\}$;
k — число компонент смеси;
$\Theta = (w_j, \vartheta_j)_{j=1}^k$ — начальное приближение параметров смеси;
δ — параметр критерия останова;

Выход:
$\Theta = (w_j, \vartheta_j)_{j=1}^k$ — оптимизированный вектор параметров смеси;

1: **ПРОЦЕДУРА** EM (X^m, k, Θ, δ);
2: **повторять**
3: E-шаг (expectation):
для всех $i = 1, \ldots, m$, $j = 1, \ldots, k$
$$g_{ij}^0 := g_{ij}; \quad g_{ij} := \frac{w_j \phi(x_i; \vartheta_j)}{\sum_{s=1}^k w_s \phi(x_i; \vartheta_s)};$$
4: M-шаг (maximization):
для всех $j = 1, \ldots, k$
$$\vartheta_j := \arg\max_{\vartheta} \sum_{i=1}^m g_{ij} \ln \phi(x_i; \vartheta); \quad w_j := \frac{1}{m} \sum_{i=1}^m g_{ij};$$
5: **пока** $\max_{i,j} |g_{ij} - g_{ij}^0| > \delta$;
6: **вернуть** $(w_j, \vartheta_j)_{j=1}^k$;

Критерий останова. Итерации останавливаются, когда значения функционала $Q(\Theta)$ или скрытых переменных G перестают существенно изменяться. Удобнее контролировать скрытые переменные, так как они имеют смысл вероятностей и принимают значения из отрезка $[0, 1]$.

Реализация итерационного процесса показана в Алгоритме 2.2. На E-шаге вычисляется матрица скрытых переменных G по формуле (2.17). На M-шаге решается серия из k задач максимизации взвешенного правдоподобия (2.20), каждая из них — по полной выборке X^m с вектором весов g_j.

Обобщённый EM-алгоритм. Не обязательно добиваться высокой точности решения оптимизационной задачи (2.20) на каждом M-шаге алгоритма. Достаточно лишь сместиться в направлении максимума, сделав одну или несколько итераций, и затем выполнить E-шаг. Этот алгоритм также обладает неплохой сходимостью и называется *обобщённым EM-алгоритмом* (generalized EM-algorithm, GEM) [39].

Проблема выбора начального приближения. Алгоритм EM сходится при достаточно общих предположениях к локальному оптимуму. Качество этого решения и скорость сходимости существенно зависят от начального приближения. Сходимость ухудшается в тех случаях, когда делается попытка поместить несколько компонент в один фактический сгусток распределения, либо разместить компоненту посередине между сгустками. Стандартная (но далеко не самая лучшая) эвристика заключается в том, чтобы выбрать параметры начальных компонент случайным образом. Другая идея — взять в качестве центров компонент k объектов, максимально удалённых

Алгоритм 2.3. EM-алгоритм с последовательным добавлением компонент

Вход:

выборка $X^m = \{x_1, \ldots, x_m\}$

R — максимальный допустимый разброс правдоподобия объектов;

m_0 — минимальная длина выборки, по которой можно восстановить плотность;

δ — параметр критерия останова;

Выход:

k — число компонент смеси;

$\Theta = (w_j, \vartheta_j)_{j=1}^k$ — веса и параметры компонент;

1: начальное приближение — одна компонента:
$\vartheta_1 := \arg\max_{\vartheta} \sum_{i=1}^m \ln \phi(x_i; \vartheta)$; $w_1 := 1$; $k := 1$;

2: **для всех** $k := 2, 3, \ldots$

3: выделить объекты с низким правдоподобием:
$U := \{x_i \in X^m : p(x_i) < \max_j p(x_j)/R\}$;

4: **если** $|U| < m_0$ **то**

5: **выход** из цикла по k;

6: начальное приближение для k-й компоненты:
$\vartheta_k := \arg\max_{\vartheta} \sum_{x_i \in U} \ln \phi(x_i; \vartheta)$; $w_k := \frac{1}{m}|U|$;
$w_j := w_j(1 - w_k)$, $j = 1, \ldots, k-1$;

7: EM (X^m, k, Θ, δ);

друг от друга. Можно стартовать итерационный процесс много раз из различных начальных приближений и затем выбрать наилучшее решение.

EM-алгоритм с последовательным добавлением компонент позволяет решить две проблемы сразу — выбора начального приближения и выбора числа компонент. Идея заключается в следующем. Имея некоторый набор компонент, можно выделить объекты x_i, которые хуже всего описываются смесью — это объекты с наименьшими значениями правдоподобия $p(x_i)$. По этим объектам строится ещё одна компонента. Затем она добавляется в смесь и запускаются EM-итерации, чтобы новая компонента и старые «притёрлись друг к другуk. Так продолжается до тех пор, пока все объекты не окажутся покрыты компонентами.

Реализация этой идеи представлена в Алгоритме 2.3. На шаге 1 строится первая компонента и полагается $k = 1$. Затем в цикле последовательно добавляется по одной компоненте. Если значение правдоподобия $p(x_i)$ в R раз меньше максимального значения правдоподобия, значит объект x_i плохо описывается смесью. Заметим, что это лишь эвристика; совсем не обязательно сравнивать $p(x_i)$ именно с максимальным правдоподобием; можно брать среднее правдоподобие или фиксированное пороговое значение P_0. На шаге 3 формируется подвыборка U из объектов, которые не подходят ни к одной из компонент. Если длина этой подвыборки меньше порога m_0, то процесс добавления компонент на этом заканчивается, и оставшиеся объекты считаются выбросами. На шаге 6 снова применяется метод максимума правдоподобия для формирования новой компоненты, но теперь уже не по всей выборке, а только

по подвыборке U. Веса компонент пересчитываются таким образом, чтобы их сумма по-прежнему оставалась равной единице. На шаге 7 все предыдущие компоненты вместе с новой компонентой проходят через цикл итераций EM-алгоритма.

Стохастический EM-алгоритм. Максимизируемый функционал $Q(\Theta)$ в общем случае может иметь большое количество локальных экстремумов. Поэтому EM-алгоритму присущи обычные недостатки любого детерминированного процесса многоэкстремальной оптимизации: застревание в локальных максимумах, зависимость решения от начального приближения, медленная сходимость при неудачном выборе начального приближения. Обычно такого рода недостатки преодолеваются методами адаптивной стохастической оптимизации.

Описание одного из вариантов *стохастического EM-алгоритма* (stochastic EM-algorithm, SEM) можно найти в [1, стр. 207]. Основное отличие от Алгоритма 2.2 в том, что на M-шаге (шаг 4) вместо максимизации взвешенного правдоподобия

$$\vartheta_j := \arg\max_{\vartheta} \sum_{i=1}^{m} g_{ij} \ln \phi(x_i; \vartheta)$$

решается задача максимизации обычного, невзвешенного, правдоподобия

$$\vartheta_j := \arg\max_{\vartheta} \sum_{x_i \in X_j} \ln \phi(x_i; \vartheta),$$

где выборки X_j генерируются из X^m путём стохастического моделирования. Для каждого объекта $x_i \in X^m$ разыгрывается случайное значение $j(i)$ из дискретного распределения вероятностей $\{g_{ij} : j = 1, \ldots, k\}$, и объект $x_i \in X^m$ включается только в выборку $X_{j(i)}$.

В [1] описана другая стратегия — там число компонент k последовательно уменьшается, начиная с некоторого заведомо избыточного числа $k_{\max}$. Если компонента оказывается слишком малочисленной, $|X_j| \leq m_0$, то она вовсе удаляется. Возможно также совмещение обеих стратегий.

Преимущества SEM обусловлены тем, что рандомизация «выбивает» оптимизационный процесс из локальных максимумов. SEM работает быстрее обычного детерминированного EM, и его результаты меньше зависят от начального приближения. Как правило, SEM находит экстремум $Q(\Theta)$, более близкий к глобальному.

Смеси многомерных нормальных распределений

Рассмотрим решение задачи M-шага в частном случае, когда компоненты имеют нормальные (гауссовские) плотности. В этом случае функционал (2.20) является квадратичным и положительно определенным, поэтому решение выписывается в явном аналитическом виде.

Гауссовские смеси общего вида.

Теорема 2.6. *Пусть компоненты смеси имеют n-мерные нормальные распределения $\phi(x; \vartheta_j) = \mathrm{N}(x; \mu_j, \Sigma_j)$ с параметрами $\vartheta_j = (\mu_j, \Sigma_j)$, где $\mu_j \in \mathrm{R}^n$ — вектор математического ожидания, $\Sigma_j \in \mathrm{R}^{n \times n}$ — ковариационная матрица, $j = 1, \ldots, k$. Тогда стационарная*

точка оптимизационной задачи (2.20) имеет вид

$$\hat{\mu}_j = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} x_i, \quad j = 1, \ldots, k;$$

$$\hat{\Sigma}_j = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^{\mathsf{T}}, \quad j = 1, \ldots, k.$$

Таким образом, М-шаг сводится к вычислению выборочного среднего и выборочной ковариационной матрицы для каждой компоненты смеси со своим распределением весов объектов. Вес i-го объекта для j-й компоненты равен g_{ij} — оценке принадлежности данного объекта данной компоненте, вычисленной на Е-шаге.

Смеси многомерных нормальных распределений позволяют приближать любые непрерывные плотности вероятности. Они являются универсальными аппроксиматорами плотностей, подобно тому, как полиномы являются универсальными аппроксиматорами непрерывных функций. В практических задачах это позволяет восстанавливать функции правдоподобия классов даже в тех случаях, когда на первый взгляд для выполнения условий Теоремы 2.6 нет содержательных оснований.

Недостатком гауссовских смесей является необходимость обращать ковариационные матрицы. Это трудоёмкая операция. Кроме того, ковариационные матрицы нередко оказываются вырожденными или плохо обусловленными, что приводит к неустойчивости выборочных оценок плотности и самого классификатора. Стандартные приёмы (регуляризация, метод главных компонент) позволяют справиться с этой проблемой. Но есть и другой выход — использовать для описания компонент более простые распределения, например, сферические.

Гауссовские смеси с диагональными матрицами ковариации. Трудоёмкого обращения матриц можно избежать, если принять гипотезу, что в каждой компоненте смеси признаки некоррелированы. В этом случае гауссианы упрощаются, оставаясь, тем не менее, универсальными аппроксиматорами плотности.

Теорема 2.7. *Пусть компоненты смеси имеют n-мерные нормальные распределения с параметрами (μ_j, Σ_j), где $\mu_j = (\mu_{j1}, \ldots, \mu_{jn})$, $\Sigma_j = \operatorname{diag}(\sigma^2_{j1}, \ldots, \sigma^2_{jn})$ — диагональная матрица, $j = 1, \ldots, k$:*

$$\phi(x; \vartheta_j) = \mathrm{N}(x; \mu_j, \Sigma_j) = \prod_{d=1}^{n} \frac{1}{\sigma_{jd}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\xi_d - \mu_{jd}}{\sigma_{jd}}\right)^2\right), \quad x = (\xi_1, \ldots, \xi_n).$$

Тогда стационарная точка оптимизационной задачи (2.20) имеет вид

$$\hat{\mu}_{jd} = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} x_{id}, \quad d = 1, \ldots, n;$$

$$\hat{\sigma}^2_{jd} = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} (x_{id} - \hat{\mu}_{jd})^2, \quad d = 1, \ldots, n;$$

где $x_i = (x_{i1}, \ldots, x_{in})$ — объекты выборки X^m.

Доказательство. Продифференцируем логарифм нормальной плотности $\mathrm{N}(x; \mu_j, \Sigma_j)$ по параметрам μ_{jd}, σ_{jd} в точке $x_i = (x_{i1}, \ldots, x_{in})$:

$$\frac{\partial}{\partial \mu_{jd}} \ln \mathrm{N}(x_i; \mu_j, \Sigma_j) = \sigma_{jd}^{-2}(x_{id} - \mu_{jd});$$

$$\frac{\partial}{\partial \sigma_{jd}} \ln \mathrm{N}(x_i; \mu_j, \Sigma_j) = -\sigma_{jd}^{-1} + \sigma_{jd}^{-3}(x_{id} - \mu_{jd})^2.$$

Приравняем нулю производные взвешенного функционала правдоподобия по параметрам μ_{jd}, σ_{jd}:

$$-\sigma_{jd}^{-2} \sum_{i=1}^{\ell} g_{ij}(x_{id} - \mu_{jd}) = 0;$$

$$\sigma_{jd}^{-3} \sum_{i=1}^{\ell} g_{ij} \left(\sigma_{jd}^2 - (x_{id} - \mu_{jd})^2 \right) = 0.$$

Отсюда, вынося параметры μ_{jd}, σ_{jd} за знак суммирования по i, и применяя соотношение (2.19), получаем требуемое. □

Можно было бы пойти дальше и предположить, что компоненты имеют сферические плотности, $\Sigma_j = \sigma_j^2 I_n$. Однако такое предположение имеет очевидный недостаток: если признаки существенно различаются по порядку величины, то компоненты будут иметь сильно вытянутые формы, которые придётся аппроксимировать большим количеством сферических гауссианов. Предположение о неравных дисперсиях признаков приводит к алгоритму классификации, не чувствительному к различиям в масштабах измерения признаков.

Радиальные функции. Гауссиан $p_j(x) = \mathrm{N}(x; \mu_j, \Sigma_j)$ с диагональной матрицей Σ_j можно записать в виде

$$p_j(x) = \mathrm{N}_j \exp\left(-\tfrac{1}{2}\rho_j^2(x, \mu_j)\right),$$

где $\mathrm{N}_j = (2\pi)^{-\frac{n}{2}}(\sigma_{j1} \cdots \sigma_{jn})^{-1}$ — нормировочный множитель, $\rho_j(x, x')$ — *взвешенная евклидова метрика* в n-мерном пространстве X:

$$\rho_j^2(x, x') = \sum_{d=1}^{n} \sigma_{jd}^{-2} |\xi_d - \xi'_d|^2, \qquad x = (\xi_1, \ldots, \xi_n), \quad x' = (\xi'_1, \ldots, \xi'_n).$$

Чем меньше расстояние $\rho_j(x, \mu_j)$, тем выше значение плотности в точке x. Поэтому плотность $p_j(x)$ есть *функция близости* вектора x к центру μ_j.

Функции $f(x)$, зависящие только от расстояния между x и фиксированной точкой пространства X, принято называть *радиальными*.

Сеть радиальных базисных функций

Выше мы рассматривали задачу разделения смеси распределений, забыв на время об исходной задаче классификации.

Пусть теперь $Y = \{1, \ldots, M\}$, каждый класс $y \in Y$ имеет свою плотность распределения $p_y(x)$ и представлен частью выборки $X_y^\ell = \{(x_i, y_i) \in X^\ell \mid y_i = y\}$.

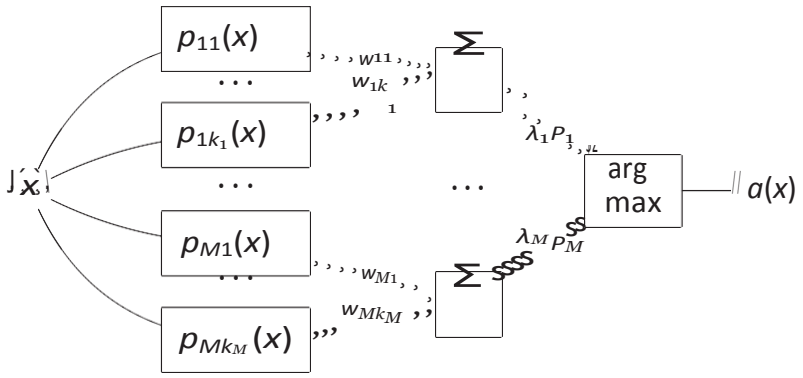


Рис. 5. Сеть радиальных базисных функций представляет собой трёхуровневую суперпозицию.

Пусть функции правдоподобия классов $p_y(x)$, $y \in Y$, представимы в виде смесей k_y компонент. Каждая компонента имеет n-мерную гауссовскую плотность с параметрами $\mu_{yj} = (\mu_{yj1}, \ldots, \mu_{yjn})$, $\Sigma_{yj} = \mathrm{diag}(\sigma^2_{yj1}, \ldots, \sigma^2_{yjn})$, $j = 1, \ldots, k_y$:

$$p_y(x) = \sum_{j=1} w_{yj} p_{yj}(x), \quad p_{yj}(x) = \mathrm{N}(x; \mu_{yj}, \Sigma_{yj}), \quad \sum_{j=1} w_{yj} = 1, \quad w_{yj} \geq 0.$$

Алгоритм классификации. Запишем байесовское решающее правило (2.2), выразив плотность каждой компоненты $p_{yj}(x)$ через взвешенное евклидово расстояние от объекта x до центра компоненты μ_{yj}:

$$a(x) = \arg\max_{y \in Y} \lambda_y P_y \sum_{j=1} w_{yj} \mathrm{N}_{yj} \exp\left(-\tfrac{1}{2}\rho^2(x, \mu_{yj})\right), \tag{2.21}$$

где $\mathrm{N}_{yj} = (2\pi)^{-\frac{n}{2}}(\sigma_{yj1} \ldots \sigma_{yjn})^{-1}$ — нормировочные множители. Алгоритм имеет вид суперпозиции, состоящей из трёх уровней или *слоёв*, Рис 5.

Первый слой образован $k_1 + \cdots + k_M$ гауссианами $p_{yj}(x)$, $y \in Y$, $j = 1, \ldots, k_y$. На входе они принимают описание объекта x, на выходе выдают оценки близости объекта x к центрам μ_{yj}, равные значениям плотностей компонент в точке x.

Второй слой состоит из M сумматоров, вычисляющих взвешенные средние этих оценок с весами w_{yj}. На выходе второго слоя появляются оценки принадлежности объекта x каждому из классов, равные значениям плотностей классов $p_{yj}(x)$.

Третий слой образуется единственным блоком arg max, принимающим окончательное решение об отнесении объекта x к одному из классов.

Таким образом, при классификации объекта x оценивается его близость к каждому из центров μ_{yj} по метрике $\rho_{yj}(x, \mu_{yj})$, $j = 1, \ldots, k_y$. Объект относится к тому классу, к чьим центрам он располагается ближе.

Описанный трёхуровневый алгоритм классификации называется *сетью с радиальными базисными функциями* или *RBF-сетью* (radial basis function network). Это одна из разновидностей *нейронных сетей*.

Обучение RBF-сети сводится к восстановлению плотностей классов $p_y(x)$ с помощью EM-алгоритма. Результатом обучения являются центры μ_{yj} и дисперсии Σ_{yj}

компонент $j = 1, \ldots, k_y$. Оценивая дисперсии, мы фактически подбираем веса признаков в метриках $\rho_{yj}(x, \mu_{yj})$ для каждого центра μ_{yj}. При использовании Алгоритма 2.3 для каждого класса определяется оптимальное число компонент смеси.

Преимущества EM-алгоритма. По сравнению с широко известными градиентными методами настройки нейронных сетей (см. главу 6), EM-алгоритм более устойчив к шуму и быстрее сходится. Кроме того, он описывает каждый класс как совокупность компонент или *кластеров*, что позволяет лучше понимать внутреннюю структуру данных. В частности, центры сферических гауссовских компонент μ_{yj} можно интерпретировать как виртуальные эталонные объекты, с которыми сравниваются классифицируемые объекты. Во многих прикладных задачах виртуальным эталонам удаётся подыскать содержательную интерпретацию. Например, в медицинской дифференциальной диагностике это может быть определённая (j-я) форма данного (y-го) заболевания. Информация о том, что классифицируемый объект близок именно к этому эталону, может быть полезной при принятии решений.

EM-алгоритм может также использоваться для решения задач *кластеризации*, о чём пойдёт речь в главе 7.

3 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 выравниваниемk, 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*[5]. Для формализации понятия «сходствak 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.

§3.1 Метод ближайшего соседа и его обобщения

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.

Обобщённый метрический классификатор

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

$$\rho(u, x_u^{(1)}) \le \rho(u, x_u^{(2)}) \le \cdots \le \rho(u, x_u^{(\ell)}),$$

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.

Опр. 3.1. *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:*

$$a(u; X^\ell) = \arg\max_{y \in Y} \Gamma_y(u, X^\ell); \qquad \Gamma_y(u, X^\ell) = \sum_{i=1}^{\ell} [y_u^{(i)} = y]\, w(i, u); \tag{3.1}$$

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

[5] В математике *компактными* принято называть ограниченные замкнутые множества. *Гипотеза компактности* не имеет ничего общего с этим понятием, и должна пониматься скорее в «бытовомс смысле этого слова.

Метрический классификатор определён с точностью до весовой функции $w(i, u)$. 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 естьk. Алгоритм $a(u; X^\ell)$ строит локальную аппроксимацию выборки X^ℓ, причём вычисления 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 «рассуждениик, так как на вопрос «почему объект u был отнесён к классу y?k алгоритм может дать понятное экспертам объяснение: «потому, что имеются схожие с ним прецеденты класса yk, и предъявить список этих прецедентов.

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

Метод ближайших соседей

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

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

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:

- 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.
- Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ.
- 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 $x^{(i)}$, $i = 1, \ldots, k$:

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

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.

$$\mathrm{LOO}(k, X^\ell) = \sum_{i=1}^{\ell} \left[a\left(x_i; X^\ell \setminus \{x_i\}, k\right) \neq y_i \right] \to \min_k .$$

Заметим, что 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 $\mathrm{LOO}(k)$ 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 w_i, задающих вклад i-го соседа в классификацию:

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

Выбор последовательности w_i является эвристикой. Если взять линейно убывающие веса $w_i = \frac{k+1-i}{k}$, 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: $w_i = q^i$, 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.

- 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.

- 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.

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

Метод парзеновского окна

Ещё один способ задать веса соседям — определить w_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 $w(i, u) = K\left(\frac{1}{h}\rho(u, x_u^{(i)})\right)$ в общей формуле (3.1), получим алгоритм

$$a(u; X^\ell, h) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_u^{(i)} = y] K\left(\frac{\rho(u, x_u^{(i)})}{h}\right). \qquad (3.2)$$

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 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: $h(u) = \rho(u, x_u^{(k+1)})$. Тогда алгоритм принимает вид

$$a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_u^{(i)} = y] K\left(\frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})}\right). \qquad (3.3)$$

Заметим, что 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 обсуждается также в разделе 2.2.2.

Метод потенциальных функций

В методе парзеновского окна центр радиального ядра $K_h(u, x) = K\left(\frac{1}{h}\rho(u, x)\right)$ 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:

$$a(u; X^\ell) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_i = y] \gamma_i K\left(\frac{\rho(u, x_i)}{h_i}\right), \qquad \gamma_i \geq 0,\ h_i > 0. \qquad (3.4)$$

По сути, эта формула отличается от (3.3) только тем, что здесь ширина окна h_i зависит от обучающего объекта x_i, а не от классифицируемого объекта u.

Алгоритм 3.1. Метод потенциальных функций

Вход:
X^ℓ — обучающая выборка;
Выход:
Коэффициенты γ_i, $i = 1, \ldots, \ell$ в (3.4);

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

Данная идея лежит в основе *метода потенциальных функций* [3] 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 $\frac{1}{z+a}$, однако для наших целей совершенно не обязательно брать именно такое ядро.

Алгоритм (3.4) имеет достаточно богатый набор из 2ℓ параметров γ_i, h_i. Простейший и исторически самый первый метод их настройки представлен в Алгоритме 3.1. 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. Условия его сходимости и многочисленные вариации исследованы в [3].

Достоинство Алгоритма 3.1 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 3.1 представляет скорее исторический интерес.

Недостатков у Алгоритма 3.1 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.

Более современный метод настройки линейных комбинаций потенциальных функций, основанный на EM-алгоритме, рассматривается в §2.4. 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 (3.1), 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.

§3.2 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 толщek 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.

Понятие отступа объекта

Общая формула (3.1) позволяет ввести характеристику объектов, показывающую, насколько глубоко объект погружён в свой класс.

Опр. 3.2. *Отступом (margin) объекта* $x_i \in X^\ell$ *относительно алгоритма классификации, имеющего вид* $a(u) = \arg\max_{y \in Y} \Gamma_y(u)$, *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, Рис. 6.

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

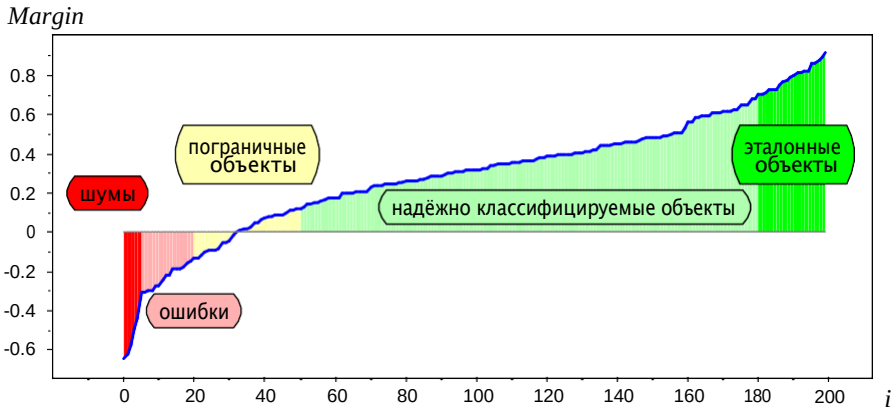


Рис. 6. Упорядоченные по возрастанию отступов M_i объекты выборки, $i = 1, \ldots, 200$. Условное деление объектов на пять типов.

- *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.

- *Ошибочные* объекты имеют отрицательные отступы и классифицируются неверно. Возможной причиной может быть неадекватность алгоритмической модели (3.1), в частности, неудачная конструкция метрики ρ.

- *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».

Алгоритм STOLP для отбора эталонных объектов

Идея отбора эталонов реализована в алгоритме STOLP [11]. Мы рассмотрим его обобщённый вариант с произвольной весовой функцией $w(i, u)$. Будем строить метрический алгоритм $a(u; \Omega)$ вида (3.1), где $\Omega \subseteq X^\ell$ — множество эталонов.

Алгоритм 3.2. Отбор эталонных объектов STOLP

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:
$\Omega := \{\arg\max_{x_i \in X^\ell_y} M(x_i, X^\ell) : y \in Y\}$;

5: **пока** $\Omega = X^\ell$;

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:
$x_i := \arg\min_{x \in E} M(x, \Omega)$; $\Omega := \Omega \cup \{x_i\}$;

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.

Алгоритм 3.2 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 «жадногоk наращивания множества Ω. 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 «центральнымk эталонным объектом и массой «приграничныхk 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 «приграничныеk эталоны, и тем более простой, менее «изрезаннойk будет граница между классами.

В описанном варианте алгоритм STOLP имеет относительно низкую эффективность. Для присоединения очередного эталона необходимо перебрать множество объектов $X^\ell \setminus \Omega$, 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. Реализация этих идей не показана в Алгоритме 3.2, чтобы не загромождать его техническими подробностями.

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.

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 $X^\ell = (x_i, y_i)_{i=1}^\ell$.

Опр. 4.1. *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 $\mathsf{L}\ M_i(w)$, где $\mathsf{L}(M)$ — монотонно невозрастающая функция отступа, мажорирующая пороговую функцию потерь: $[M < 0] \le \mathsf{L}(M)$. Тогда минимизацию суммарных потерь можно рассматривать как приближённый метод минимизации эмпирического риска — числа ошибок на обучающей выборке:

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

Некоторые применяемые на практике функции потерь показаны на рис. 7. 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, см. **??**. Каждый из перечисленных методов имеет свою разумную мотивацию, однако все они приводят к разным функциям потерь. Все эти методы будут рассмотрены далее.

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

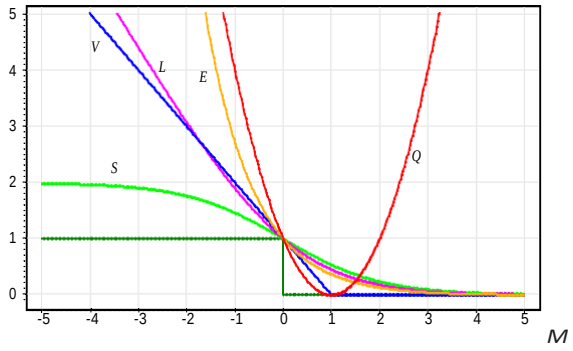


Рис. 7. Непрерывные аппроксимации пороговой функции потерь [$M < 0$].

$Q(M) = (1 - M)^2$ — 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.

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

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

Зная модель плотности $p(x, y|w)$, можно выписать вид разделяющей поверхности f и функцию L. С другой стороны, задавая вид разделяющей поверхности и функцию потерь 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*:

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

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

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

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:

$$Q(w) = \sum_{i=1}^{\ell} L_i(w) + \frac{1}{C}\sum_{j=1}^{n} |w_j| \to \min_w,$$

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

$$Q(u, v) = \sum_{i=1}^{\ell} \mathsf{L}_i(u - v) + \frac{1}{C}\sum_{j=1}^{n}(u_j + v_j) \to \min_{u,v};$$
$$u_j \geq 0, \quad v_j \geq 0, \quad j = 1, \ldots, n.$$

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 $p(w)$, $w \in \mathrm{R}^n$ с независимыми параметрами w_j, но теперь не будем предполагать, что дисперсии C_j параметров w_j одинаковы:

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

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

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

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 $f_j : X \to \mathrm{R}$, $j = 1, \ldots, n$. Вектор $x = (x^1, \ldots, x^n) \in \mathrm{R}^n$, где $x^j = f_j(x)$, называется *признаковым описанием* объекта x.

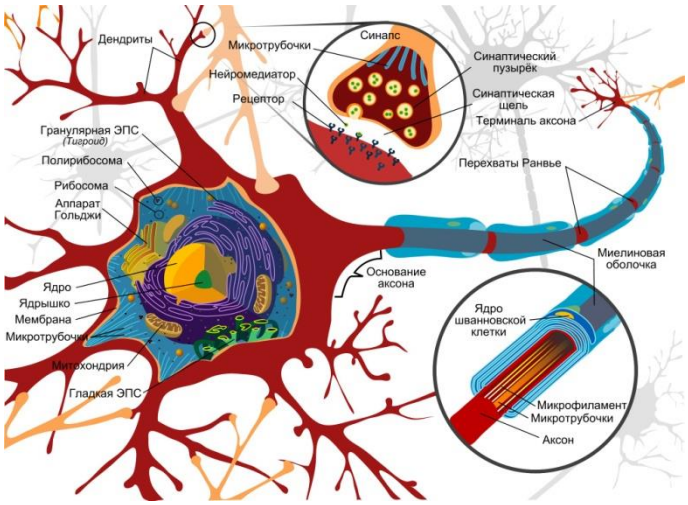


Рис. 8. Нервная клетка (рисунок из Википедии — свободной энциклопедии).

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

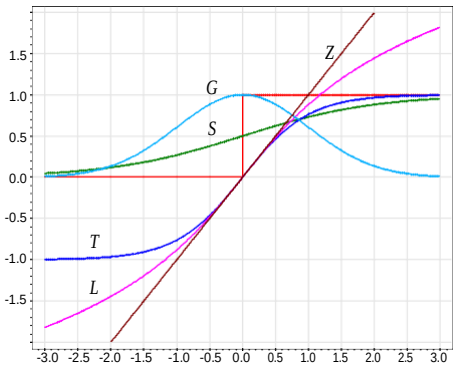
$$a(x, w) = \operatorname{sign}\left(\langle w, x\rangle - w_0\right) = \operatorname{sign}\left(\sum_{j=1} w_j f_j(x) - w_0\right). \qquad (4.5)$$

Уравнение $\langle w, x\rangle = 0$ задаёт гиперплоскость, разделяющую классы в пространстве 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*, Рис 8. 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.

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,



$\vartheta(z) = [z \geq 0]$	пороговая функция Хевисайда;
$\sigma(z) = (1 + e^{-z})^{-1}$	сигмоидная функция (S);
$\mathrm{th}(z) = \sqrt{2\sigma(2z)} - 1$	гиперболический тангенс (T);
$\ln(z + \sqrt{z^2 + 1})$	логарифмическая функция (L);
$\exp(-z^2/2)$	гауссовская функция (G);
z	линейная функция (Z);

Рис. 9. Стандартные функции активации $\phi(z)$.

то тормозящий. Если суммарный заряд превышает *порог активации* w_0, то нейрон возбуждается и выдаёт на выходе +1, иначе выдаётся −1.

Функцию $\phi(z) = \mathrm{sign}(z)$, преобразующую значение суммарного импульса в выходное значение нейрона, называют *функцией активации*. В общем случае это не обязательно пороговая функция. Используют также «сглаженнуюk пороговую функцию — гиперболический тангенс $\phi(z) = \mathrm{th}(z)$ и другие, см. Рис. 9.

Таким образом, линейный классификатор (4.5) является математической моделью нейрона. Эту модель предложили в 1943 году МакКаллок и Питтс [52].

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 10^2 последовательных тактов. Кора головного мозга человека содержит порядка 10^{11} нейронов, и каждый нейрон имеет синаптические связи с 10^3–10^4 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.

Мы начнём с линейных классификаторов, а к задачам обучения искусственных нейронных сетей вернёмся позже, через несколько лекций.

§4.3 Метод стохастического градиента

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

$$Q(w, X^\ell) = \sum_{i=1}^{\ell} \mathsf{L}\big\langle w, x_i\big\rangle_{y_i} \to \min_w. \tag{4.6}$$

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 $Q'(w) = \left\{\frac{\partial Q(w)}{\partial w_j}\right\}_{j=1}^{n}$:

$$w := w - \eta Q'(w),$$

где $\eta > 0$ — величина шага в направлении антиградиента, называемая также *темпом обучения* (learning rate). Предполагая, что функция потерь L дифференцируема, распишем градиент:

$$w := w - \eta \sum_{i=1}^{\ell} \mathsf{L}'\big\langle w, x_i\big\rangle_{y_i} x_i y_i. \tag{4.7}$$

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:

$$w := w - \eta \mathsf{L}'_a\big\langle w, x_i\big\rangle_{y_i} x_i y_i. \tag{4.8}$$

В методе *стохастического градиента* (stochastic gradient, SG) прецеденты перебираются в случайном порядке, см. Алгоритм 4.1. Если же объекты предъявлять в некотором фиксированном порядке, процесс может зациклиться или разойтись.

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

$$w_j := \frac{\langle y, f_j\rangle}{\langle f_j, f_j\rangle}, \quad j = 1, \ldots, n, \tag{4.9}$$

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 4.1 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.

Алгоритм 4.1. Метод стохастического градиента.

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

Преимущества метода SG.

- Метод легко реализуется и легко обобщается на нелинейные классификаторы и на нейронные сети — суперпозиции линейных классификаторов.
- Метод подходит для динамического обучения, когда обучающие объекты поступают потоком, и вектор весов обновляется при появлении каждого объекта.
- Метод позволяет настраивать веса на избыточно больших выборках, за счёт того, что случайной подвыборки может оказаться достаточно для обучения.

Недостатки метода SG.

- Функционал Q, как правило, многоэкстремальный, и процесс может сходиться к локальному минимуму, сходиться очень медленно или не сходиться вовсе.
- 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.
- Если функция потерь имеет горизонтальные асимптоты, то процесс может попасть в состояние «паралич». Чем больше значение скалярного произведения $\langle w, x_i$, тем ближе значение производной 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».

Классические частные случаи

Адаптивный линейный элемент. Рассмотрим случай, когда функция потерь квадратична, $\mathsf{L}(M) = (M-1)^2$. Тогда правило обновления весов на каждой итерации метода стохастического градиента примет вид

$$w := w - \eta \underbrace{(\langle w, x_i\rangle - y_i)}_{} x_i. \tag{4.10}$$

Это правило предложено Видроу и Хоффом в 1960 году и называется *дельта-правилом* (delta-rule), а сам линейный нейрон — *адаптивным линейным элементом* или ADALINE [66]. Это правило подходит также и для решения задач линейной регрессии, когда $Y = \mathrm{R}$, $a(x) = \langle w, x\rangle$ и функция потерь имеет вид $(\langle w, x_i\rangle - y_i)^2$.

Персептрон Розенблатта. В 1957 году Розенблатт предложил эвристический алгоритм обучения нейрона, основанный на принципах нейрофизиологии. Экспериментально было установлено, что при синхронном возбуждении двух связанных нервных клеток синаптическая связь между ними усиливается. Чем чаще синапс угадывает правильный ответ, тем сильнее становится связь. Своеобразная тренировка связи приводит к постепенному запоминанию информации. Если же синапс начинает часто ошибаться или вообще перестаёт использоваться, связь ослабевает, информация начинается забываться. Таким образом, память реализуется в синапсах. В математической модели нейрона роль памяти играет вектор синаптических весов w.

Данное правило обучения нетрудно формализовать. Признаки будем пока полагать бинарными, $f_j(x) \in \{0, 1\}$. Ответы также принимают только два значения, $y_i \in \{-1, 1\}$. Допустим, что сразу после получения классификации объекта $a(x_i)$ становится известен правильный ответ y_i. Возможны три случая.

1. Если ответ $a(x_i)$ совпадает с y_i, то вектор весов изменять не нужно.
2. Если $a(x_i) = -1$ и $y_i = 1$, то вектор весов w увеличивается. Увеличивать имеет смысл только те веса w_j, для которых $f_j(x_i) = 0$; изменение других компонент не повлияет на результат. Положим $w := w + \eta x_i$, где $\eta > 0$ — темп обучения.
3. Если $a(x_i) = 1$ и $y_i = -1$, то вектор весов уменьшается: $w := w - \eta x_i$.

Эти три случая объединяются в так называемое *правило Хэбба* [45]:

$$\textbf{если } \langle w, x_i\rangle\, y_i < 0 \textbf{ то } w := w + \eta x_i y_i. \tag{4.11}$$

Легко проверить, что оно в точности соответствует градиентному шагу (4.8), если взять кусочно-линейную функцию потерь $\mathsf{L}(M) = (-M)_+$. Однако формула (4.8) верна для произвольных признаков, не обязательно бинарных.

Для правила Хэбба доказана теорема сходимости, которая также справедлива для произвольных действительных признаков.

Теорема 4.1 (Новиков, 1962 [55]). *Пусть $X = \mathrm{R}^n$, $Y = \{-1, 1\}$, и выборка X^ℓ линейно разделима — существует вектор $\tilde{w}$ и положительное число δ такие, что $\langle \tilde{w}, x_i\rangle\, y_i > \delta$ для всех $i = 1, \ldots, \ell$. Тогда Алгоритм 4.1 с правилом Хэбба (4.11) за конечное число исправлений находит вектор весов, разделяющий обучающую выборку без ошибок, причём из любого начального приближения w^0, при любом $\eta > 0$, независимо от порядка предъявления объектов. Если $w^0 = 0$, то достаточное число исправлений вектора весов не превосходит*

$$t_{\max} = \left(\frac{D}{\delta}\right)^2, \quad \text{где } D = \max_{x \in X^\ell} \|x\|.$$

Доказательство. Запишем выражение для косинуса угла между вектором $\tilde{w}$ и вектором весов после t-го исправления w^t, полагая без ограничения общности $\|\tilde{w}\| = 1$:

$$\cos(\widehat{\tilde{w}, w^t}) = \frac{\langle \tilde{w}, w^t \rangle}{\|w^t\|}.$$

При t-м исправлении нейрону с вектором весов w^{t-1} предъявляется обучающий объект x, правильный ответ y, и при этом нейрон совершает ошибку $\langle x, w^{t-1} \rangle y < 0$. Согласно правилу Хэбба (4.11) в этом случае происходит модификация весов. В силу условия линейной разделимости, справедлива оценка снизу:

$$\langle \tilde{w}, w^t \rangle = \langle \tilde{w}, w^{t-1} \rangle + \eta \langle \tilde{w}, x \rangle y > \langle \tilde{w}, w^{t-1} \rangle + \eta\delta > \langle \tilde{w}, w^0 \rangle + t\eta\delta.$$

В силу ограниченности выборки, $\|x\| < D$, справедлива оценка сверху:

$$\|w^t\|^2 = \|w^{t-1}\|^2 + \eta^2\|x\|^2 + 2\eta \langle x, w^{t-1} \rangle y < \|w^{t-1}\|^2 + \eta^2 D^2 < \|w^0\|^2 + t\eta^2 D^2.$$

Подставим полученные соотношения в выражение для косинуса:

$$\cos(\widehat{\tilde{w}, w^t}) > \frac{\langle \tilde{w}, w^0 \rangle + t\eta\delta}{\sqrt{\|w^0\|^2 + t\eta^2 D^2}} \to \infty \text{ при } t \to \infty.$$

Косинус не может превышать единицы. Следовательно, при некотором достаточно большом t не найдётся ни одного $x \in X^\ell$ такого, что $\langle x, w^t \rangle y < 0$, то есть выборка окажется поделенной безошибочно.

Если $w^0 = 0$, то нетрудно оценить сверху достаточное число исправлений. Из условия $\cos \leq 1$ следует $t\delta/D \leq 1$, откуда $t_{\max} = (D/\delta)^2$. □

На практике линейная разделимость выборки является скорее исключением, чем правилом. Если условия теоремы Новикова не выполнены, то процесс обучения может оказаться расходящимся.

Эвристики для улучшения градиентных методов обучения

В этом разделе рассматриваются эвристические приёмы и рекомендации, компенсирующие недостатки градиентных методов обучения. Все они в полной мере относятся к обучению нейронных сетей, включая широко известный метод *обратного распространения ошибок* , который будет рассмотрен в §6.2. Различных тонкостей настолько много, что применение градиентного обучения по праву считается искусством, см. также обзор [50].

Нормализация данных. Градиентный метод чувствителен к масштабу измерения признаков. Если норма вектора объекта $\|x_i\|$ принимает большие значения, а функция потерь имеет горизонтальные асимптоты, то итерационный процесс может оказаться «парализованным». Поэтому общей практикой является предварительная *нормализация* признаков:

$$x^j := \frac{x^j - x^j_{\min}}{x^j_{\max} - x^j_{\min}}, \quad \text{либо} \quad x^j := \frac{x^j - x^j_{\text{ср}}}{x^j_{\text{ско}}}, \quad j = 1, \ldots, n,$$

где $x^j_{\min}$, $x^j_{\max}$, $x^j_{\text{ср}}$, $x^j_{\text{ско}}$ — соответственно минимальное, максимальное, среднее значения и среднеквадратичное отклонение j-го признака.

Порядок предъявления объектов. Кроме стандартной рекомендации брать объекты в случайном порядке, имеются ещё следующие соображения.

1. Наибольшее смещение весов ожидается для того объекта, который наименее похож на объекты, предъявленные до него. В общем случае довольно трудно найти объект, максимально информативный на данном шаге обучения. Простая эвристика заключается в том, чтобы попеременно предъявлять объекты из разных классов, поскольку объекты одного класса с большей вероятностью содержат схожую информацию. Эта техника называется *перетасовкой объектов* (shuffling).

2. Ещё одна эвристика состоит в том, чтобы чаще предъявлять те объекты, на которых была допущена ошибка. Для этого вероятность появления каждого объекта устанавливается пропорционально величине ошибки на данном объекте. Эту эвристику рекомендуется применять только в тех случаях, когда исходные данные не содержат выбросов, иначе процесс обучения может сосредоточиться на шумовых объектах, которые вообще следовало бы исключить из обучающей выборки.

3. Простая для реализации эвристика заключается в том, чтобы сравнить величину ошибки на предъявленном объекте с некоторым порогом. Если ошибка окажется меньше порога, вектор весов не модифицируется. Логика та же, что у персептрона Розенблатта: если объект неплохо классифицируется, то менять веса не нужно. При этом увеличивается и скорость настройки.

Квадратичная регуляризация в теории нейронных сетей называется также *сокращением весов* (weights decay). Чтобы ограничить рост абсолютных значений весов, к минимизируемому функционалу $Q(w)$ добавляется штрафное слагаемое:

$$Q_\tau(w) = Q(w) + \frac{\tau}{2}\|w\|^2.$$

Это приводит к появлению аддитивной поправки в градиенте: $Q'_\tau(w) = Q'(w) + \tau w$. В результате правило обновления весов принимает вид

$$w := w(1 - \eta\tau) - \eta Q'(w).$$

Таким образом, вся модификация сводится к появлению неотрицательного множителя $(1 - \eta\tau)$, приводящего к постоянному уменьшению весов. Регуляризация предотвращает паралич, повышает устойчивость весов в случае мультиколлинеарности, способствует повышению обобщающей способности алгоритма и снижению риска переобучения. Управляющий параметр τ позволяет найти компромисс между точностью настройки на конкретную выборку и устойчивостью весов.

Недостаток метода в том, что параметр τ приходится подбирать в режиме скользящего контроля, что связано с большими вычислительными затратами.

Выбор величины шага.

1. Известно, что градиентные методы сходятся к локальному минимуму, если величину шага η уменьшать с числом итераций t. Точнее, сходимость гарантируется при $\eta_t \to 0$, $\sum_{t=1}^{\infty} \eta_t = \infty$, $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, в частности можно положить $\eta_t = 1/t$.

2. В методе скорейшего градиентного спуска выбирается *адаптивный шаг* η, который является решением одномерной задачи $Q\ w - \eta Q'(w) \to \min_{\eta}$. Во многих случаях эту задачу удаётся решить аналитически [8]. В частности, для алгоритма ADALINE с квадратичной функцией потерь $\eta = \|x_i\|^{-2}$.

Выбивание из локальных минимумов необходимо для предотвращения сходимости к недостаточно хорошим локальным решениям. Один из простейших способов заключается в том, чтобы при каждой стабилизации функционала производить случайные модификации вектора весов в довольно большой окрестности текущего значения и запускать процесс градиентного спуска из новых точек. Этот приём называют *встряхиванием коэффициентов* (jog of weights). По сути дела, он является симбиозом градиентного метода и *случайного локального поиска* (stochastic local search).

Ранний останов. Чрезмерная оптимизация может вести к переобучению. Узкий глобальный минимум функционала $Q(w, X^\ell)$ хуже более широкого, но устойчивого локального минимума. Для предотвращения попадания в такие «расщелины» применяется *ранний останова* (early stopping): в ходе итераций вычисляется какой-нибудь *внешний критерий* (стр. **??**), например, средняя потеря на независимой контрольной выборке, и если он начинает возрастать, процесс настройки прекращается.

§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.

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 $f_j : X \to \mathrm{R}$, $j = 1, \ldots, n$. Будем полагать $X = \mathrm{R}^n$, 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 $p(x, y) = P_y p_y(x) = \mathsf{P}(y|x)p(x)$, где P_y — априорные вероятности, $p_y(x)$ — функции правдоподобия, $\mathsf{P}(y|x)$ — апостериорные вероятности классов $y \in Y$.*

Опр. 4.2. *Плотность распределения $p(x)$, $x \in \mathrm{R}^n$ называется экспонентной, если $p(x) = \exp\big(c(\delta)\langle \vartheta, x\rangle + b(\delta, \vartheta) + d(x, \delta)\big)$, где параметр $\vartheta \in \mathrm{R}^n$ 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.

Пример 4.1. Многомерное нормальное распределение с вектором матожидания $\mu \in \mathbb{R}^n$ и ковариационной матрицей $\Sigma \in \mathbb{R}^{n\times n}$ является экспонентным с параметром сдвига $\vartheta = \Sigma^{-1}\mu$ и параметром разброса $\delta = \Sigma$:

$$\mathrm{N}(x;\mu,\Sigma) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\Big(-\tfrac{1}{2}(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu)\Big) =$$
$$= \exp\Big(\underbrace{\mu^{\mathsf{T}}\Sigma^{-1}}_{\langle\vartheta,\,\cdot\,\rangle}x - \underbrace{\tfrac{1}{2}\mu^{\mathsf{T}}\Sigma^{-1}\Sigma\Sigma^{-1}\mu}_{b(\delta,\vartheta)} - \underbrace{\tfrac{1}{2}x^{\mathsf{T}}\Sigma^{-1}x - \tfrac{n}{2}\ln(2\pi) - \tfrac{1}{2}\ln|\Sigma|}_{d(x,\delta)}\Big).$$

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. Напомним, что оптимальный байесовский классификатор имеет вид $a(x) = \arg\max_{y\in Y}\lambda_y \mathrm{P}(y\,|\,x)$, где λ_y — штраф за ошибку на объектах класса y.

В случае двух классов

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

Теорема 4.2. *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: $a(x) = \operatorname{sign}(\langle w, x\rangle - w_0)$, где $w_0 = \ln(\lambda_-/\lambda_+)$, а вектор w не зависит от штрафов λ_-, λ_+;

2) апостериорная вероятность принадлежности произвольного объекта $x \in X$ классу $y \in \{-1, +1\}$ может быть вычислена по значению дискриминантной функции: $\mathrm{P}(y|x) = \sigma(y\langle w, x\rangle)$, 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:

$$\frac{\mathrm{P}(+1|x)}{\mathrm{P}(-1|x)} = \frac{P_+ p_+(x)}{P_- p_-(x)} = \exp\Big(\underbrace{(\vartheta_+ - \vartheta_-)}_{w=\mathrm{const}(x)}x + \underbrace{b(\delta,\vartheta_+) - b(\delta,\vartheta_-) + \ln\frac{P_+}{P_-}}_{\mathrm{const}(x)}\Big).$$

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

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

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

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

Объединяя эти два равенства в одно, получаем требуемое: $\mathrm{P}(y|x) = \sigma(y\langle w, x\rangle)$.

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

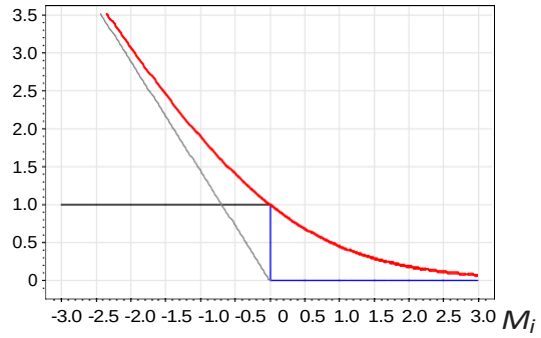


Рис. 10. Логарифмическая функция потерь $\log_2(1+e^{-M_i})$ и её наклонная асимптота.

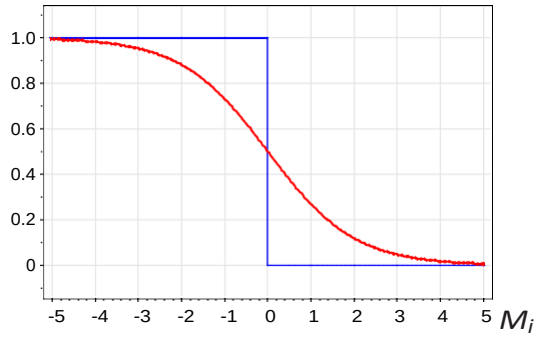


Рис. 11. Правило Хэбба: пороговое $[M_i < 0]$ и сглаженное $\sigma(-M_i)$.

Метод стохастического градиента для логистической регрессии

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

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

Согласно определению условной вероятности, $p(x, y) = \mathsf{P}(y|x)p(x)$, где плотности распределения объектов $p(x)$ не зависят от вектора параметров w. Апостериорные вероятности выражаются согласно Теореме 4.2 через линейную дискриминантную функцию: $\mathsf{P}(y|x) = \sigma(\langle w, x\rangle y)$. Таким образом,

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

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

$$\tilde{Q}(w, X^\ell) = \sum_{i=1}^{\ell} \log_2(1 + \exp(-\langle w, x_i\rangle y_i)) \to \min_w. \tag{4.12}$$

Таким образом, логистическая функция потерь $\mathsf{L}(M) = \log_2(1+e^{-M})$ 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:

$$w := w + \eta\, y_i x_i\, \sigma(-\langle w, x_i\rangle y_i), \tag{4.13}$$

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, Рис. 10. Поэтому и логистическое правило (4.13) оказывается, в свою очередь, сглаженным вариантом правила Хэбба (4.11), Рис. 11:

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

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 [31, 60].

О методах второго порядка. Для оптимизации весов w чаще используется метод Ньютона-Рафсона, описанный в главе о регрессионном анализе, см. 5.5.5. 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 $P(+1|x)$.

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). Почему? Дело в том, 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.

- 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).
- Возможность оценивать апостериорные вероятности и риски.

Недостатки логистической регрессии.

- Оценки вероятностей и рисков могут оказаться неадекватными, если не выполняются предположения Теоремы 4.2.

- 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.

Скоринг и оценивание апостериорных вероятностей

Скоринг. В случае бинарных признаков, $X = \{0, 1\}^n$, можно полагать, что функции правдоподобия классов $p_y(x)$ описываются биномиальными распределениями, следовательно, являются экспонентными. Это соображение служит дополнительным обоснованием *бинаризации признаков*, когда каждый небинарный исходный признак заменяется одним или несколькими бинарными.

В бинарном случае вычисление линейной дискриминантной функции удобно рассматривать как подсчёт *баллов* (score): если $f_j(x) = 1$, то есть признак f_j наблюдается у объекта x, то к сумме баллов добавляется вес w_j. Классификация производится путём сравнения набранной суммы баллов с пороговым значением w_0.

Возраст	до 25	5
	25 - 40	10
	40 - 50	15
	50 и больше	10
Собственность	владелец	20
	совладелец	15
	съемщик	10
	другое	5
Работа	руководитель	15
	менеджер среднего звена	10
	служащий	5
	другое	0
Стаж	1/безработный	0
	1..3	5
	3..10	10
	10 и больше	15
Работа_мужа /жены	нет/домохозяйка	0
	руководитель	10
	менеджер среднего звена	5
	служащий	1

Рис. 12. Фрагмент скоринговой карты для задачи принятия кредитных решений.

Благодаря свой простоте подсчёт баллов или *скоринг* (scoring) пользуется большой популярностью в таких областях, как медицина, геология, банковское дело, социология, маркетинг, и др. Абсолютное значение веса w_j можно интерпретировать как степень важности признака f_j, а знак $\mathrm{sign}(w_j)$ показывает, в пользу какого класса свидетельствует наличие данного признака. Это важная дополнительная информация о признаках, помогающая экспертам лучше понимать и задачу, и классификатор.

После бинаризации признаков классификатор представляется в виде так называемой *скоринговой карты* (scorecard), в которой перечисляются все исходные признаки, для каждого исходного — все построенные по нему бинарные признаки, для каждого бинарного — его вес. Имея такую карту, классификацию можно проводить с помощью стандартной электронной таблицы или даже вручную. Рис. 12.

Вероятностный выход и оценивание рисков Логистическая функция σ переводит значение линейной дискриминантной функции $\langle w, x \rangle$ в оценку апостериорной вероятности того, что объект x принадлежит классу $+1$: $\mathrm{P}(+1|x) = \sigma\langle w, x \rangle$. Это свойство используется в тех приложениях, где наряду с классификацией объекта x требуется оценить связанный с ним *риск* как математическое ожидание потерь:

$$R(x) = \sum_{y \in Y} \lambda_y \mathrm{P}(y|x) = \sum_{y \in Y} \lambda_y \sigma\langle w, x \rangle y,$$

где λ_y — величина потери при ошибочной классификации объекта класса y.

В практических ситуациях к оценкам апостериорной вероятности следует относиться с осторожностью. Теорема 4.2 гарантирует, что $\mathsf{P}(y|x) = \sigma\,\langle\, w, x\rangle\; y$ только для экспонентных классов с равными параметрами разброса. В остальных случаях оценка вероятности носит эвристический характер. На практике экспонентность редко когда проверяется, а гарантировать равенство разброса вообще не представляется возможным.

Вероятностная калибровка позволяет пересчитать значение дискриминантной функции в оценку апостериорной вероятности, когда условия теоремы 4.2 не выполняются, и даже когда классификатор $a(x) = \operatorname{sign} f\,(x,\, w)$ не является линейным. Предполагается, что апостериорная вероятность $\mathsf{P}(+1|x)$ монотонно зависит от значения дискриминантной функции $f(x, w)$. Существует много способов ввести модель этой зависимости, но мы рассмотрим только один — *калибровку Платта*, основанную на линейно-сигмоидальной модели:

$$\mathsf{P}(+1|x) = \sigma\;\alpha f(x, w) + \beta\;.$$

Для настройки неизвестных параметров $(\alpha,\ \beta) \in \mathbb{R}^2$ решается задача максимизации правдоподобия

$$\sum_{i=1} \log \mathsf{P}(y_i|x_i) = \sum_{i=1} \log \sigma\;\alpha y_i f(x_i, w) + \beta y_i \;\;\to \max_{\alpha,\beta}$$

любым стандартными численными методами оптимизации.

§4.5 Метод опорных векторов

В 60–70-е годы коллективом советских математиков под руководством В. Н. Вапника был разработан метод *обобщённого портрета*, основанный на построении *оптимальной разделяющей гиперплоскости* [7]. 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. Однако, как станет видно в 4.5.2, различия на самом деле не столь велики.

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) [38]. В настоящее время он считается одним из лучших методов классификации. Его современное изложение можно найти в [35, 61].

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.

Линейно разделимая выборка

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

Будем строить линейный пороговый классификатор:

$$a(x) = \operatorname{sign}\Big(\sum_{j=1}^{n} w_j x^j - w_0\Big) = \operatorname{sign}\big(\langle w, x\rangle - w_0\big), \tag{4.14}$$

где $x = (x^1, \ldots, x^n)$ — признаковое описание объекта x; вектор $w = (w^1, \ldots, w^n) \in \mathrm{R}^n$ и скалярный порог $w_0 \in \mathrm{R}$ являются параметрами алгоритма. Уравнение $\langle w, x\rangle = w_0$ описывает гиперплоскость, разделяющую классы в пространстве 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 [32, 59, 65].

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

$$\min_{i=1,\ldots,\ell} y_i \big(\langle w, x_i\rangle - w_0\big) = 1. \tag{4.15}$$

Множество точек x: $-1 \le \langle w, x\rangle - w_0 \le 1$ описывает полосу, разделяющую классы, см. Рис. 13. Ни один из объектов обучающей выборки не попадает внутрь этой 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 (4.15). 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.

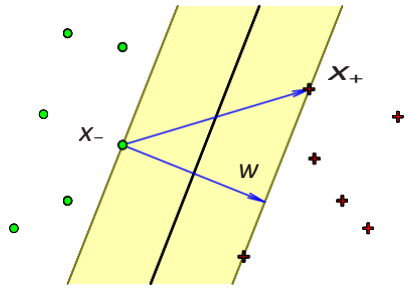


Рис. 13. Линейно разделимая выборка. Обучающие объекты x_- и x_+ находятся на границе разделяющей полосы. Вектор нормали w к разделяющей гиперплоскости определяет ширину полосы.

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

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

Ширина полосы максимальна, когда норма вектора w минимальна.

Итак, 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:

$$\begin{cases} \langle w, w\rangle \to \min; \\ y_i \langle w, x_i\rangle - w_0 \geq 1, \quad i = 1, \ldots, \ell. \end{cases} \tag{4.16}$$

На практике линейно разделимые классы встречаются довольно редко. Поэтому постановку задачи (4.16) необходимо модифицировать так, чтобы система ограничений была совместна в любой ситуации.

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 $\xi_i \geq 0$, характеризующие величину ошибки на объектах x_i, $i = 1, \ldots, \ell$. Ослабим в (4.16) ограничения-неравенства и одновременно введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{\ell}\xi_i \to \min\limits_{w, w_0, \xi}; \\ y_i \langle w, x_i\rangle - w_0 \geq 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \xi_i \geq 0, \quad i = 1, \ldots, \ell. \end{cases} \tag{4.17}$$

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

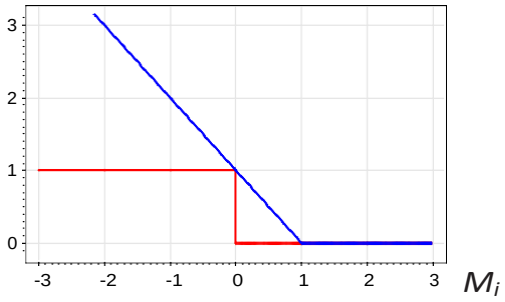


Рис. 14. Кусочно-линейная аппроксимация пороговой функции потерь: $[M_i < 0] \le (1 - M_i)_+$.

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 (4.14) 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 (4.17) ошибка ξ_i выражается через отступ M_i. Действительно, из ограничений-неравенств следует, что $\xi_i \ge 0$ и $\xi_i \ge 1 - M_i$. 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 (4.17) оказывается эквивалентной безусловной минимизации функционала Q, не зависящего от переменных ξ_i:

$$Q(w, w_0) = \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}. \qquad (4.18)$$

В силу неравенства $[M_i < 0] \le (1 - M_i)_+$, рис. 14, функционал (4.18) можно рассматривать как верхнюю оценку *эмпирического риска* (числа ошибочных классификаций объектов обучающей выборки), к которому добавлен *регуляризатор* $\|w\|^2$, умноженный на *параметр регуляризации* $\frac{1}{2C}$.

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 [23].

Задача (4.18) соответствует принципу максимума совместного правдоподобия (4.3), если принять модель плотности

$$p(x_i, y_i | w) = z_1 \exp\left(-(1 - M_i(w, w_0))_+\right),$$

гауссовскую модель априорного распределения вектора параметров w

$$p(w; C) = z_2 \exp\left(-\frac{\|w\|^2}{2C}\right),$$

и не накладывать никаких ограничений на параметр w_0. Здесь z_1, z_2 — нормировочные константы, C — гиперпараметр.

Понимание роли функции потерь и регуляризатора необходимо для того, чтобы более вольно обращаться с постановкой задачи, при необходимости модифицировать её, получая методы, похожие на SVM, но обладающие требуемыми свойствами.

Двойственная задача. Запишем функцию Лагранжа задачи (4.17):

$$\mathsf{L}(w, w_0, \xi; \lambda, \eta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\lambda_i\,M_i(w, w_0) - 1 + \xi_i \;- \sum_{i=1}^{\ell}\xi_i\eta_i =$$
$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell}\lambda_i\,M_i(w, w_0) - 1 \;- \sum_{i=1}^{\ell}\xi_i\,\lambda_i + \eta_i - C\,,$$

где $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ — вектор переменных, двойственных к w; $\eta = (\eta_1, \ldots, \eta_\ell)$ — вектор переменных, двойственных к $\xi = (\xi_1, \ldots, \xi_\ell)$.

Согласно теореме Куна-Таккера задача (4.17) эквивалентна двойственной задаче поиска седловой точки функции Лагранжа:

$$\begin{cases}
\mathsf{L}(w, w_0, \xi; \lambda, \eta) \;\to\; \min\limits_{w, w_0, \xi}\ \max\limits_{\lambda, \eta}; \\
\xi_i \geq 0, \quad \lambda_i \geq 0, \quad \eta_i \geq 0, \quad i = 1, \ldots, \ell; \\
\lambda_i = 0 \;\text{ либо }\; M_i(w, w_0) = 1 - \xi_i, \quad i = 1, \ldots, \ell; \\
\eta_i = 0 \;\text{ либо }\; \xi_i = 0, \quad i = 1, \ldots, \ell;
\end{cases}$$

В последних двух строках записаны *условия дополняющей нежёсткости*.

Необходимым условием седловой точки функции Лагранжа является равенство нулю её производных. Отсюда получаются три полезных соотношения:

$$\frac{\partial \mathsf{L}}{\partial w} = w - \sum_{i=1}^{\ell}\lambda_i y_i x_i = 0 \qquad \Longrightarrow \qquad w = \sum_{i=1}^{\ell}\lambda_i y_i x_i; \tag{4.19}$$

$$\frac{\partial \mathsf{L}}{\partial w_0} = -\sum_{i=1}^{\ell}\lambda_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^{\ell}\lambda_i y_i = 0; \tag{4.20}$$

$$\frac{\partial \mathsf{L}}{\partial \xi^i} = -\lambda_i - \eta_i + C = 0 \qquad \Longrightarrow \qquad \eta_i + \lambda_i = C, \quad i = 1, \ldots, \ell. \tag{4.21}$$

Из (4.19) следует, что искомый вектор весов w является линейной комбинацией векторов обучающей выборки x_i, причём только тех, для которых $\lambda_i > 0$.

Опр. 4.3. *Если $\lambda_i > 0$, то объект обучающей выборки x_i называется опорным вектором (support vector).*

Из третьего соотношения (4.21) и неравенства $\eta_i \geq 0$ следует $0 \leq \lambda_i \leq C$. Отсюда, и из условий дополняющей нежёсткости вытекает, что возможны только три допустимых сочетания значений переменных ξ_i, λ_i, η_i и отступов M_i.

Соответственно, все объекты x_i, $i = 1, \ldots, \ell$ делятся на следующие три типа:

1. $\lambda_i = 0$; $\eta_i = C$; $\xi_i = 0$; $M_i \geq 1$.

Объект x_i классифицируется правильно и не влияет на решение w. Такие объекты будем называть *периферийными* или *неинформативными*.

2. $0 < \lambda_i < C$; $0 < \eta_i < C$; $\xi_i = 0$; $M_i = 1$.

Объект x_i классифицируется правильно и лежит в точности на границе разделяющей полосы. Такие объекты будем называть *опорными граничными*.

3. $\lambda_i = C$; $\eta_i = 0$; $\xi_i > 0$; $M_i < 1$.

Объект x_i либо лежит внутри разделяющей полосы, но классифицируется правильно ($0 < \xi_i < 1$, $0 < M_i < 1$), либо попадает на границу классов ($\xi_i = 1$, $M_i = 0$), либо вообще относится к чужому классу ($\xi_i > 1$, $M_i < 0$). Во всех этих случаях объект x_i будем называть *опорным нарушителем*.

В силу соотношения (4.21) в лагранжиане обнуляются все члены, содержащие переменные ξ_i и η_i, и он выражается только через двойственные переменные λ_i.

$$\begin{cases} -L(\lambda) = -\sum\limits_{i=1}^{\ell}\lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell}\lambda_i\lambda_j y_i y_j \langle x_i, x_j\rangle \;\to\; \min\limits_{\lambda}; \\ 0 \le \lambda_i \le C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell}\lambda_i y_i = 0. \end{cases} \tag{4.22}$$

Здесь минимизируется квадратичный функционал, имеющий неотрицательно определённую квадратичную форму, следовательно, выпуклый. Область, определяемая ограничениями неравенствами и одним равенством, также выпуклая. Следовательно, данная двойственная задача имеет единственное решение.

Допустим, мы решили эту задачу. Тогда вектор w вычисляется по формуле (4.19). Для определения порога w_0 достаточно взять произвольный опорный граничный вектор x_i и выразить w_0 из равенства $w_0 \leftarrow \langle w, x_i\rangle - y_i$. На практике для повышения численной устойчивости рекомендуется брать медиану множества значений w_0, вычисленных по всем граничным опорным векторам:

$$w_0 = \operatorname{med}\{\langle w, x_i\rangle - y_i : \lambda_i > 0,\ M_i = 1,\ i = 1, \ldots, \ell\}. \tag{4.23}$$

В итоге алгоритм классификации представляется в следующем виде:

$$a(x) = \operatorname{sign}\left(\sum_{i=1}^{\ell}\lambda_i y_i \langle x_i, x\rangle - w_0\right). \tag{4.24}$$

Обратим внимание, что суммирование идёт не по всей выборке, а только по опорным векторам, для которых $\lambda_i \neq 0$. Классификатор $a(x)$ не изменится, если все остальные объекты исключить из выборки. Это свойство называют *разреженностью* (sparsity); именно оно и отличает SVM от других линейных классификаторов — дискриминанта Фишера, логистической регрессии и однослойного персептрона.

Ненулевыми λ_i обладают не только граничные опорные объекты, но и объекты-нарушители. Это говорит о недостаточной робастности (устойчивости к шуму) SVM. Нарушителями могут быть объекты, появившиеся в результате ошибочных наблюдений; их надо было бы исключить из выборки, а не строить по ним решение.

О подборе параметра регуляризации. Константу C обычно выбирают по критерию скользящего контроля. Это трудоёмкий способ, так как задачу приходится решать заново при каждом значении C. Хорошо то, что решение, как правило, не очень чувствительно к выбору C, и слишком точная его оптимизация не требуется.

Если есть основания полагать, что выборка почти линейно разделима, и лишь объекты-выбросы классифицируются неверно, то можно применить фильтрацию выбросов. Сначала задача решается при некотором C, и из выборки удаляется небольшая доля объектов, имеющих наибольшую величину ошибки ξ_i. После этого задача решается заново по усечённой выборке. Возможно, придётся проделать несколько таких итераций, пока оставшиеся объекты не окажутся линейно разделимыми.

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 (x, x') в пространстве X всюду заменяется скалярным произведением $(\psi(x), \psi(x'))$ 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.

Опр. 4.4. *Функция $K : X \times X \to \mathbb{R}$ называется ядром (kernel function), если она представима в виде $K(x, x') = \langle \psi(x), \psi(x') \rangle$ при некотором отображении $\psi : X \to H$, где H — пространство со скалярным произведением.*

Постановка двойственной задачи (4.22), и сам алгоритм классификации (4.24) зависят только от скалярных произведений объектов, но не от самих признаковых описаний. Это означает, что скалярное произведение (x, x') можно формально заменить ядром $K(x, x')$. Поскольку ядро в общем случае нелинейно, такая замена приводит к существенному расширению множества реализуемых алгоритмов $a : X \to Y$.

Более того, можно вообще не строить спрямляющее пространство H в явном виде, и вместо подбора отображения ψ заниматься непосредственно подбором ядра.

Можно пойти ещё дальше, и вовсе отказаться от признаковых описаний объектов. Во многих практических задачах объекты изначально задаются информацией об их попарном взаимоотношении, например, отношении сходства. Если эта информация допускает представление в виде двуместной функции $K(x, x')$, удовлетворяющей аксиомам скалярного произведения, то задача может решаться методом SVM. Для такого подхода недавно был придуман термин *беспризнаковое распознавание* (featureless recognition), хотя многие давно известные метрические алгоритмы классификации (kNN, RBF и др.) также не требуют задания признаковых описаний.

Теорема Мерсера. Любая ли функция двух аргументов $K(x, x')$ может исполнять роль ядра? Следующая теорема даёт исчерпывающий ответ на этот вопрос и показывает, что класс допустимых ядер достаточно широк.

Теорема 4.3 (Мерсер, 1909 [53]). *Функция $K(x, x')$ является ядром тогда и только тогда, когда она симметрична, $K(x, x') = K(x', x)$, и неотрицательно определена: $\int_X \int_X K(x, x')g(x)g(x')dxdx' \geq 0$ для любой функции $g : X \to \mathrm{R}$.*

Существует эквивалентное определение неотрицательной определённости.

Опр. 4.5. *Функция $K(x, x')$ неотрицательно определена, если для любой конечной выборки $X^p = (x_1, \ldots, x_p)$ из X матрица $K = \|K(x_i, x_j)\|$ размера $p \times p$ неотрицательно определена: $z^\intercal Kz \geq 0$ для любого $z \in \mathrm{R}^p$.*

Проверка неотрицательной определённости функции в практических ситуациях может оказаться делом нетривиальным. Часто ограничиваются перебором конечного числа функций, про которые известно, что они являются ядрами. Среди них выбирается лучшая, как правило, по критерию скользящего контроля. Очевидно, что это не оптимальное решение. На сегодняшний день проблема выбора ядра, оптимального для данной конкретной задачи, остаётся открытой.

Конструктивные способы построения ядер. Следующие правила порождения позволяют строить ядра в практических задачах [19, 20].

1. Произвольное скалярное произведение $K(x, x') = \langle x, x' \rangle$ является ядром.
2. Константа $K(x, x') = 1$ является ядром.
3. Произведение ядер $K(x, x') = K_1(x, x')K_2(x, x')$ является ядром.
4. Для любой функции $\psi : X \to \mathrm{R}$ произведение $K(x, x') = \psi(x)\psi(x')$ — ядро.
5. Линейная комбинация ядер с неотрицательными коэффициентами $K(x, x') = \alpha_1 K_1(x, x') + \alpha_2 K_2(x, x')$ является ядром.
6. Композиция произвольной функции $\phi : X \to X$ и произвольного ядра K_0 является ядром: $K(x, x') = K_0(\phi(x), \phi(x'))$.
7. Если $s : X \times X \to \mathrm{R}$ — произвольная симметричная интегрируемая функция, то $K(x, x') = \int_X s(x, z)s(x', z)\, dz$ является ядром.
8. Функция вида $K(x, x') = k(x - x')$ является ядром тогда и только тогда, когда Фурье-образ $F[k](\omega) = (2\pi)^{\frac{n}{2}} \int_X e^{-i\langle \omega, x \rangle} k(x)\, dx$ неотрицателен.
9. Предел локально-равномерно сходящейся последовательности ядер — ядро.
10. Композиция произвольного ядра K_0 и произвольной функции $f : \mathrm{R} \to \mathrm{R}$, представимой в виде сходящегося степенного ряда с неотрицательными коэффициентами $K(x, x') = f\, K_0(x, x')$, является ядром. В частности, функции $f(z) = e^z$ и $f(z) = \frac{1}{1-z}$ от ядра являются ядрами.

Примеры ядер. Существует несколько «стандартныхk ядер, которые при ближайшем рассмотрении приводят к уже известным алгоритмам: полиномиальным разделяющим поверхностям, двухслойным нейронным сетям, потенциальным функциям (RBF-сетям), и другим. Таким образом, ядра претендуют на роль универсального языка для описания широкого класса алгоритмов обучения по прецедентам.

Наблюдается парадоксальная ситуация. С одной стороны, ядра — одно из самых красивых изобретений в машинном обучении. С другой стороны, до сих пор не найдено эффективного общего подхода к их подбору в конкретных задачах.

Пример 4.2. Возьмём $X = \mathrm{R}^2$ и рассмотрим ядро $K(u, v) = \langle u, v \rangle^2$, где $u = (u_1, u_2)$, $v = (v_1, v_2)$. Попробуем понять, какое спрямляющее пространство и преобразование ψ ему соответствуют. Разложим квадрат скалярного произведения:

$$K(u, v) = \langle u, v \rangle^2 = \langle (u_1, u_2), (v_1, v_2) \rangle^2 =$$
$$= (u_1 v_1 + u_2 v_2)^2 = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 u_2 v_2 =$$
$$= \left(u_1^2,\ u_2^2,\ \sqrt{2}\,u_1 u_2\right)\left(v_1^2,\ v_2^2,\ \sqrt{2}\,v_1 v_2\right).$$

Ядро K представляется в виде скалярного произведения в пространстве $H = \mathrm{R}^3$. Преобразование $\psi : \mathrm{R}^2 \to \mathrm{R}^3$ имеет вид $\psi : (u_1, u_2) \mapsto (u_1^2, u_2^2, \sqrt{2}u_1 u_2)$. Линейной поверхности в пространстве H соответствует квадратичная поверхность в исходном пространстве X. Данное ядро позволяет разделить внутреннюю и наружную часть произвольного эллипса, что невозможно в исходном двумерном пространстве.

Пример 4.3. Усложним ситуацию. Пусть теперь $X = \mathrm{R}^n$,

$$K(u, v) = \langle u, v \rangle^d.$$

Тогда компонентами вектора $\psi(u)$ являются различные произведения $(u_1)^{d_1} \cdots (u_n)^{d_n}$ при всевозможных целых неотрицательных $d_1, \ldots, d_n$, удовлетворяющих условию $d_1 + \cdots + d_n = d$. Число таких мономов, а следовательно и размерность пространства H, равно C^d_{n+d-1}. Пространство H изоморфно пространству всех полиномов, состоящих из мономов степени d от переменных $u_1, \ldots, u_n$.

Пример 4.4. Если $X = \mathrm{R}^n$,

$$K(u, v) = \langle u, v \rangle + 1^d,$$

то H — пространство всех мономов степени *не выше* d от переменных $u_1, \ldots, u_n$. В этом случае пространство H изоморфно пространству всех полиномов степени d. Линейная разделимость множеств в этом пространстве эквивалентна полиномиальной разделимости множеств в исходном пространстве X.

SVM как двухслойная нейронная сеть. Рассмотрим структуру алгоритма $a(x)$ после замены в (4.24) скалярного произведения $\langle x_i, x \rangle$ ядром $K(x_i, x)$. Перенумеруем объекты так, чтобы первые h объектов оказались опорными. Поскольку $\lambda_i = 0$ для всех неопорных объектов, $i = h + 1, \ldots, \ell$, алгоритм $a(x)$ примет вид

$$a(x) = \operatorname{sign} \sum_{i=1}^{h} \lambda_i y_i K(x_i, x) - w_0 \quad .$$

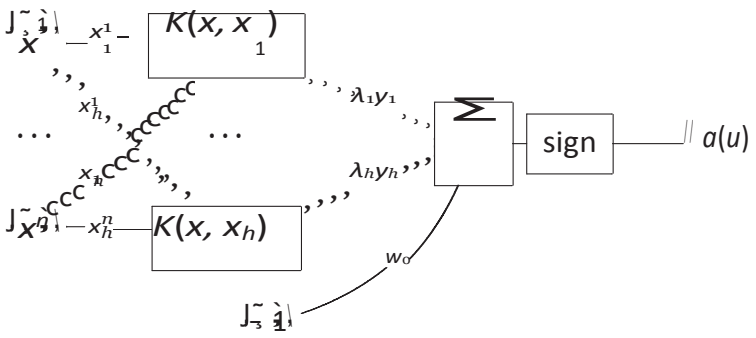


Рис. 15. Машина опорных векторов (SVM) как двухслойная нейросеть.

Если $X = \mathrm{R}^n$, то алгоритм $a(x)$ можно рассматривать как суперпозицию, называемую *двухслойной нейронной сетью*. Первый слой образуют ядра, второй слой — собственно линейный классификатор. Такая суперпозиция, построенная методом SVM, имеет несколько замечательных особенностей.

Во-первых, число нейронов первого слоя определяется автоматически, тогда как в нейронных сетях определение числа нейронов является отдельной проблемой.

Во-вторых, проясняется смысл двойственных переменных: λ_i — это степень важности ядра $K(x, x_i)$, фактически, важности или «опорностиk объекта x_i.

Пример 4.5. Классическая нейронная сеть с сигмоидными функциями активации получится, если в качестве ядра взять функцию

$$K(u, v) = \mathrm{th}\ k_0 + k_1 \langle u, v \rangle \ .$$

Данная функция удовлетворяет условиям Мерсера не при всех значениях параметров k_0 и k_1. В частности, она им не удовлетворяет при $k_0 < 0$ или $k_1 < 0$ [36]. Однако это не препятствует её успешному практическому применению. Вместо гиперболического тангенса $\mathrm{th}\, z$ часто используют также логистическую функцию $\sigma(z) = \frac{1}{1+e^{-z}}$

Что плохого произойдёт, если функция $K(u, v)$ не будет удовлетворять условиям Мерсера? Постановка задачи квадратичного программирования (4.22) останется той же и в этом случае. Однако квадратичная форма утратит свойство неотрицательной определённости, минимизируемый функционал уже не будет выпуклым, и решение может оказаться не единственным. Самое неприятное то, что на границах гиперпараллелепипеда $0 \le \lambda_i \le C$ возникнет огромное количество локальных минимумов, и поиск решения среди них в общем случае потребует полного перебора. В этой ситуации многие методы квадратичного программирования будут выдавать какой-то локальный минимум, совсем не обязательно хороший.

Пример 4.6. Нейронная сеть с *радиальными базисными функциями* (radial basis functions, RBF) получится, если взять гауссовское ядро

$$K(u, v) = \exp\ -\beta \|u - v\|^2 \ ,$$

где β — параметр. Ядро $K(x_i, x)$ вычисляет оценку близости объекта x к опорному объекту x_i. Чем ближе объекты, тем больше значение ядра. Выходной нейрон

складывает все эти оценки, умножая их на коэффициенты $\lambda_i y_i$. При этом близости к опорным объектам класса +1 суммируются с положительными весами, а к объектам класса $\mathtt{1}$ — с отрицательными. Выходной нейрон производит голосование, сравнивая суммарные близости распознаваемого объекта x к обоим классам.

В разделе 2.4.3 рассматривался альтернативный метод обучения RBF-сетей, основанный на EM-алгоритме. Тогда гауссовские ядра играли роль компонент смеси вероятностных распределений. Центры ядер размещались не в опорных объектах, а в местах локальных сгущений плотности объектов. В этом и заключается основное отличие SVM-RBF от EM-RBF. Метод SVM сдвигает центры гауссианов ближе к границе классов, в результате форма разделяющей поверхности описывается более чётко. Таким образом, SVM-RBF лучше подходит для описания классов с границами сложной формы. С другой стороны, EM-RBF более устойчив к выбросам и предпочтителен в задачах с «размытымиk границами классов.

Преимущества SVM.

- Задача квадратичного программирования имеет единственное решение, для нахождения которого разработаны достаточно эффективные методы.
- Автоматически определяется сложность суперпозиции — число нейронов первого слоя, равное числу опорных векторов.
- Максимизация зазора между классами улучшает обобщающую способность.

Недостатки SVM.

- Неустойчивость к шуму в исходных данных. Объекты-выбросы являются опорными и существенно влияют на результат обучения.
- До сих пор не разработаны общие методы подбора ядер под конкретную задачу. На практике «вполне разумныеk ядра, построенные с учётом специфики задачи, могут и не обладать свойством положительной определённости.
- Подбор параметра C требует многократного решения задачи.

Метод релевантных векторов (RVM). Ещё одна нетривиальная идея регуляризации заключается в том, что может быть указан вид функциональной зависимости вектора параметров модели w от обучающей выборки и каких-то новых параметров λ. Тогда априорное распределение можно задавать не для w, а для λ. Поясним эту конструкцию на примере *метода релевантных векторов* (relevance vector machine, RVM). Приведём только основную идею метода; за подробностями надо обращаться к работам Типпинга и Бишопа [64, 33].

Напомним, что в методе опорных векторов (SVM) вектор параметров w является линейной комбинацией опорных векторов x_i:

$$w = \sum_{i=1} \lambda_i y_i x_i, \qquad (4.25)$$

где λ_i — неотрицательные двойственные переменные, не равные нулю только для опорных векторов x_i. Один из недостатков SVM состоит в том, что опорными векторами становятся не только пограничные объекты, но и объекты-нарушители, в том числе шумовые выбросы. Метод RVM был предложен как альтернатива, призванная устранить данный дефект.

За основу в RVM берётся формула (4.25), и ставится задача определить, какие из коэффициентов λ_i можно положить равными нулю. Иными словами, делается попытка оптимизировать множество опорных объектов, сохранив свойство разреженности SVM. Для этого предполагается, что λ_i — независимые нормально распределённые случайные величины с неравными дисперсиями α_i:

$$p(\lambda) = \frac{1}{(2\pi)^{\ell/2}\sqrt{\alpha_1 \cdots \alpha_\ell}} \exp\left(-\sum_{i=1}^{\ell} \frac{\lambda_i^2}{2\alpha_i}\right)$$

То есть идея та же, что и в регуляризаторе (4.4), только теперь параметры априорного распределения связываются с объектами, а не с признаками. В результате вместо отбора признаков получаем отбор объектов, однако не такой, как в SVM, поэтому здесь *опорные* объекты называют *релевантными*. Эксперименты показали, что решение получается ещё более разреженным, чем в SVM, то есть релевантных объектов, как правило, существенно меньше, чем опорных. К сожалению, далеко не во всех задачах это действительно приводит к улучшению качества классификации.

§4.6 ROC-кривая и оптимизация порога решающего правила

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

Согласно Теореме 4.2 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 «+1k и « -1k 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 II 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):

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

Алгоритм 4.2. Эффективный алгоритм построения ROC-кривой

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.

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

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-специфичностьk.

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

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

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 4.2 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.

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 4.2.

5 Методы восстановления регрессии

Задачу обучения по прецедентам при $Y = \mathrm{R}$ 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 $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. 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:

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

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:

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

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:

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

§5.2 Непараметрическая регрессия: ядерное сглаживание

Непараметрическое восстановление регрессии основано на той же идее, что и непараметрическое восстановление плотности распределения, рассмотренное в 2.2.2. 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 $\rho(x, x')$.

Формула Надарая-Ватсона

Возьмём самую простую модель регрессии, какая только возможна — константу $g(x, \alpha) = \alpha$, $\alpha \in \mathrm{R}$. 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:

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

Зададим веса w_i 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 $\frac{\partial Q}{\partial \alpha} = 0$, получим формулу *ядерного сглаживания* Надарая–Ватсона:

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

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

В одномерном случае $X = \mathrm{R}^1$ метрика задаётся как $\rho(x, x_i) = |x - x_i|$. При этом строгим обоснованием формулы (5.4) служит следующая теорема, аналогичная Теореме 2.3 о непараметрическом восстановлении плотности.

Теорема 5.1 ([25]). *Пусть выполнены следующие условия:*

1) выборка $X^\ell = (x_i, y_i)_{i=1}^\ell$ простая, получена из распределения $p(x, y)$;

2) ядро $K(r)$ удовлетворяет ограничениям $\int_0^\infty K(r)\,dr < \infty$ и $\lim_{r\to\infty} rK(r) = 0$;

3) восстанавливаемая зависимость, определяемая плотностью $p(y|x)$, удовлетворяет при любом $x \in X$ ограничению $\mathrm{E}(y^2|x) = \int_Y y^2 p(y|x)\,dy < \infty$;

4) последовательность h_ℓ такова, что $\lim_{\ell\to\infty} h_\ell = 0$ и $\lim_{\ell\to\infty} \ell h_\ell = \infty$.

Тогда имеет место сходимость по вероятности: $a_{h_\ell}(x; X^\ell) \xrightarrow{P} \mathrm{E}(y|x)$ в любой точке $x \in X$, в которой $\mathrm{E}(y|x)$, $p(x)$ и $\mathrm{D}(y|x)$ непрерывны и $p(x) > 0$.

Таким образом, для широкого класса ядер оценка Надарая–Ватсона сходится к ожидаемому значению восстанавливаемой зависимости при неограниченном увеличении длины выборки ℓ и одновременном уменьшении ширины окна h.

Выбор ядра и ширины окна

Ядерное сглаживание — это довольно простой метод с точки зрения реализации. Обучение алгоритма $a_h(x; X^\ell)$ 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 $K(r)$. Часто используемые ядра показаны на Рис. 3. Для ядерного сглаживания чаще всего бе рут гауссовское ядро $K_G(r) = \exp\left(-\frac{1}{2}r^2\right)$ или квартическое $K_Q(r) = (1 - r^2)^2 \quad |r| < 1$.

Если ядро $K(r)$ финитно, то есть $K(r) = 0$ при $r \geq 1$, то ненулевые веса получат только те объекты x_i, для которых $\rho(x, x_i) < h$. Тогда в формуле (5.4) достаточно суммировать только по ближайшим соседям объекта x. В одномерном случае $X = \mathrm{R}^1$ 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.

Проблема локальных сгущений возникает, когда объекты выборки распределены неравномерно в пространстве X. В областях локальных сгущений оптимальна меньшая ширина окна, чем в областях разреженности. В таких случаях используется окно *переменной ширины* $h(x)$, зависящей от объекта x. Соответственно, веса вычисляются по формуле $w_i(x) = K\left(\frac{\rho(x,x_i)}{h(x)}\right)$.

Самый простой способ — взять в качестве ширины окна $h(x)$ расстояние от объекта x до его $k+1$-го соседа: $h_k(x) = \rho(x, x_x^{(k+1)})$. Недостаток этого способа в том, что функция $h_k(x)$ является непрерывной, но не гладкой, поэтому у функций $w_i(x)$ и $a_{h_k}(x)$ будут разрывные первые производные, даже если ядро гладкое. Для устранения этого недостатка можно сгладить саму функцию $h_k(x)$ по узлам равномерной сетки, при постоянной ширине окна и каком-либо гладком ядре, скажем, K_Q.

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):

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

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

Проблема выбросов: робастная непараметрическая регрессия

Оценка Надарайя–Ватсона крайне чувствительна к большим одиночным выбросам. Идея обнаружения выбросов заключается в том, что чем больше величина

Алгоритм 5.1. LOWESS — локально взвешенное сглаживание.

Вход:
X^ℓ — обучающая выборка;

Выход:
коэффициенты γ_i, $i = 1, \ldots, \ell$;

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

$$a_i := a_h\left(x_i; X^\ell \setminus \{x_i\}\right) = \frac{\sum_{j=1,\, j\neq i} \gamma_j \gamma_j K\left(\frac{\rho(x_i,x_j)}{h(x_i)}\right)}{\sum_{j=1,\, j\neq i} \gamma_j K\left(\frac{\rho(x_i,x_j)}{h(x_i)}\right)}, \quad i = 1, \ldots, \ell$$

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

ошибки $\varepsilon_i = \cdot a_h\left(x_i; X^\ell \setminus \{x_i\}\right) - y_i \cdot$, 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 $\gamma_i = \tilde{K}(\varepsilon_i)$, где $\tilde{K}$ — ещё одно ядро, вообще говоря, отличное от $K(r)$.

Коэффициенты γ_i, как и ошибки ε_i, зависят от функции a_h, которая, в свою очередь, зависит от γ_i. Разумеется, это не «порочный круг», а хороший повод для организации итерационного процесса, см. Алгоритм 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) [37].

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

Возможны различные варианты задания ядра $\tilde{K}(\varepsilon)$.

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 $\tilde{K}(\varepsilon) = [\varepsilon \leq \varepsilon^{(\ell-t)}]$.

Мягкая фильтрация [37]: используется квартическое ядро $\tilde{K}(\varepsilon) = K_Q\left(\frac{\varepsilon}{6\,\mathrm{med}\{\varepsilon_i\}}\right)$, где $\mathrm{med}\{\varepsilon_i\}$ — медиана вариационного ряда ошибок.

Проблема краевых эффектов

В одномерном случае $X = \mathrm{R}^1$ часто наблюдается значительное смещение аппроксимирующей функции $a_h(x)$ от истинной зависимости $y^*(x)$ вблизи минимальных и максимальных значений x_i, см. Рис **??**. 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:

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

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 $X = R^n$ для вычисления коэффициентов в линейной форме $a(u) = \alpha^\intercal (u - x) + \beta$ приходится решать задачу многомерной линейной регрессии (см. ниже). 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 $f_j : X \to R$ — числовые признаки, $j = 1, \ldots, n$. Линейной моделью регрессии называется линейная комбинация признаков с коэффициентами $\alpha \in R^n$:

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

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:

$$\frac{\partial Q}{\partial \alpha}(\alpha) = 2F^\intercal (F\alpha - y) = 0,$$

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

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

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

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

где $P_F = FF^+ = F(F^\top F)^{-1}F^\top$ — *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.

Сингулярное разложение

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

$$F = V D U^\top,$$

обладающего рядом замечательных свойств (позже мы докажем Теорему 5.2, из которой эти свойства будут вытекать как следствия):

1) $n \times n$-матрица D диагональна, $D = \operatorname{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\right)$, где $\lambda_1, \ldots, \lambda_n$ — общие ненулевые собственные значения матриц $F^\top F$ и $FF^\top$.

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

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

Имея сингулярное разложение, легко записать псевдообратную матрицу:

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

вектор МНК-решения:

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

вектор $F\alpha^*$ — МНК-аппроксимацию целевого вектора y:

$$F\alpha^* = P_F y = (V DU^\top) UD^{-1}V^\top y = V V^\top y = \sum_{j=1}^{n} v_j (v_j^\top y); \tag{5.6}$$

и норму вектора коэффициентов:

$$\|\alpha^*\|^2 = y^\top V D^{-1} U^\top U D^{-1} V^\top y = y^\top V D^{-2} V^\top y = \sum_{j=1}^{n} \frac{1}{\lambda_j} (v_j^\top y)^2. \tag{5.7}$$

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.

Проблема мультиколлинеарности

Если ковариационная матрица $\Sigma = F^\intercal F$ 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 $\mu(\Sigma)$ 4 10^2 . . . 10^4. 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 α_j становится невозможно судить о степени важности признака f_j.

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:

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

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 разделе 2.3.3 в связи с обращением ковариационной матрицы Σ в линейном дискриминанте Фишера.

Выразим регуляризованное МНК-решение через сингулярное разложение:

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

Теперь найдём регуляризованную МНК-аппроксимацию целевого вектора y:

$$F\alpha_\tau^* = V DU^\intercal \alpha_\tau^* = V \operatorname{diag}\left(\frac{\lambda_j}{\lambda_j + \tau}\right) V^\intercal y = \sum_{j=1} \frac{\lambda_j}{\lambda_j + \tau} v_j (v_j^\intercal y). \tag{5.8}$$

Как и прежде в (5.6), МНК-аппроксимация представляется в виде разложения целевого вектора y по базису собственных векторов матрицы $FF^\intercal$. 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:

$$\|\alpha_\tau^*\|^2 = \|D^2(D^2 + \tau I_n)^{-1}D^{-1}V^\intercal y\|^2 = \sum_{j=1} \frac{1}{\lambda_j + \tau}(v_j^\intercal y)^2 < \sum_{j=1} \frac{1}{\lambda_j}(v_j^\intercal y)^2 = \|\alpha^*\|^2.$$

Отсюда ещё одно название метода — *сжатие* (shrinkage) или *сокращение весов* (weight decay) [44].

Понятие эффективной размерности. Из формул видно, что по мере увеличения параметра τ вектор коэффициентов α_τ^* становится всё более устойчивым и жёстко определённым. Фактически, происходит понижение *эффективной размерности* решения — это второй смысл термина «сжатие».

Можно показать, что роль размерности играет след проекционной матрицы. Действительно, в нерегуляризованном случае имеем

$$\operatorname{tr} F(F^\intercal F)^{-1}F^\intercal = \operatorname{tr}(F^\intercal F)^{-1}F^\intercal F = \operatorname{tr} I_n = n.$$

При использовании регуляризации эффективная размерность принимает значение от 0 до n, не обязательно целое, и убывает при возрастании τ:

$$n_{\text{эфф}} = \operatorname{tr} F(F^\intercal F + \tau I_n)^{-1}F^\intercal = \operatorname{tr} \operatorname{diag}\left(\frac{\lambda_j}{\lambda_j + \tau}\right) = \sum_{j=1} \frac{\lambda_j}{\lambda_j + \tau} < n.$$

Проблема выбора константы регуляризации. При $\tau \to 0$ регуляризованное решение стремится к МНК-решению: $\alpha_\tau^* \to \alpha^*$. При $\tau \to \infty$ чрезмерная регуляризации приводит к вырожденному решению: $\alpha_\tau^* \to 0$. Оба крайних случая нежелательны,

поэтому оптимальным является некоторое промежуточное значение τ^*. Для его нахождения можно применять скользящий контроль, см. раздел 1.1.6 или, более подробно, **??**. Зависимость оценки скользящего контроля от параметра τ, как правило, имеет характерный минимум.

Скользящий контроль — вычислительно трудоёмкая процедура. Известна практическая рекомендация брать τ в отрезке $[0.1,\ 0.4]$, если столбцы матрицы F заранее стандартизованы (центрированы и нормированы). Ещё одна эвристика — выбрать τ так, чтобы число обусловленности приняло заданное не слишком большое значение: $M_0 = \mu(F^\intercal F + \tau I_n) = \frac{\lambda_{\max}+\tau}{\lambda_{\min}+\tau}$, откуда следует рекомендация $\tau^* \approx \lambda_{\max}/M_0$.

Лассо Тибширани

Ещё один метод регуляризации внешне похож на гребневую регрессию, но приводит к качественно иному поведению вектора коэффициентов. Вместо добавления штрафного слагаемого к функционалу качества вводится ограничение-неравенство, запрещающее слишком большие абсолютные значения коэффициентов:

$$Q(\alpha) = \|F\alpha - y\|^2 \to \min_{\alpha};$$

$$\sum_{j=1}^{n} |\alpha_j| \le \kappa; \qquad (5.9)$$

где κ — параметр регуляризации. При больших значениях κ ограничение (5.9) становится строгим неравенством, и решение совпадает с МНК-решением. Чем меньше κ, тем больше коэффициентов α_j обнуляются. Происходит отбор (селекция) признаков, поэтому параметр κ называют ещё *селективностью*. Образно говоря, параметр κ зажимает вектор коэффициентов, лишая его избыточных степеней свободы. Отсюда и название метода — *лассо* (LASSO, least absolute shrinkage and selection operator) [63].

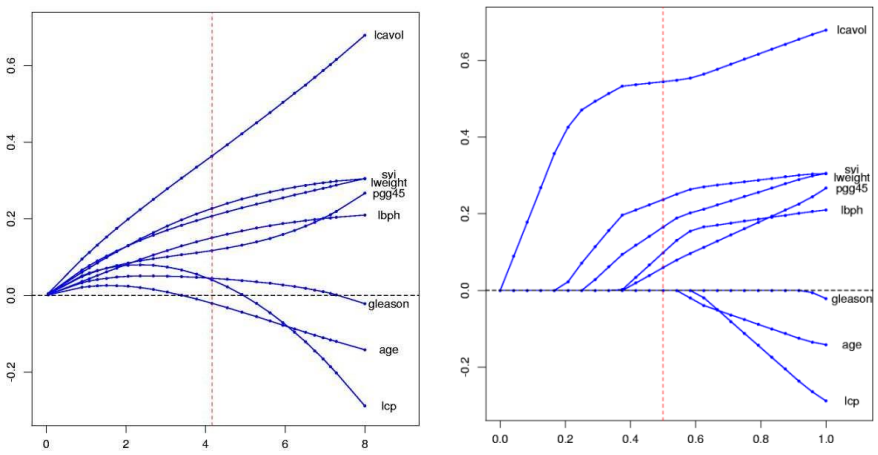
Чтобы понять, почему лассо осуществляет отбор признаков, приведём задачу квадратичного программирования (5.9) к каноническому виду. Заменим каждую переменную α_j разностью двух новых неотрицательных переменных: $\alpha_j = \alpha_j^+ - \alpha_j^-$. Функционал Q останется квадратичным по новым переменным, ограничение (5.9) примет линейный вид, и добавится $2n$ ограничений-неравенств:

$$\sum_{j=1}^{n} \alpha_j^+ + \alpha_j^- \le \kappa; \qquad \alpha_j^+ \ge 0; \qquad \alpha_j^- \ge 0.$$

Чем меньше κ, тем больше ограничений обращаются в равенства $\alpha_j^+ = \alpha_j^- = 0$, что соответствует обнулению коэффициента α_j и исключению j-го признака.

Сравнение лассо и гребневой регрессии. Оба метода успешно решают проблему мультиколлинеарности. Гребневая регрессия использует все признаки, стараясь «выжать максимум» из имеющейся информации. Лассо производит отбор признаков, что предпочтительнее, если среди признаков есть шумовые или измерения признаков связаны с ощутимыми затратами. На Рис. 16, взятом из [44], левый ряд графиков соответствует гребневой регрессии, правый ряд — лассо.

Ослабление регуляризации ведёт к уменьшению ошибки на обучении и увеличению нормы вектора коэффициентов. При этом ошибка на контроле в какой-то момент проходит через минимум, и далее только возрастает — это и есть переобучение.



(а) зависимость $\{\alpha_j\}$ от $\sigma = 1/\tau$ (б) зависимость $\{\alpha_j\}$ от к

Рис. 16. Зависимость коэффициентов линейной модели от параметра $\sigma = 1/\tau$ для гребневой регрессии и от параметра к для лассо Тибширани, по реальным данным задачи UCI.cancer [44].

Линейная монотонная регрессия

В некоторых приложениях возникает линейная модель регрессии с неотрицательными коэффициентами. Например, заранее может быть известно, что чем больше значение признака f_j, тем больше должен быть отклик y. При построении линейной композиции алгоритмов регрессии или прогнозирования роль признаков играют базовые алгоритмы (см. главу **??**). Естественно полагать, что если базовые алгоритмы настраиваются на один и тот же целевой вектор y, то они должны учитываться в композиции с положительными весами.

Возникает задача минимизации квадратичного функционала $Q(\alpha, X^\ell)$ с n ограничениями типа неравенств:

$$\begin{cases} Q(\alpha) = \|F\alpha - y\|^2 \to \min\limits_{\alpha}; \\ \alpha_j \geq 0; \quad j = 1, \ldots, n. \end{cases}$$

Это опять-таки, задача квадратичного программирования с линейными ограничениями. Когда ограничение $\alpha_j \geq 0$ становится активным, то есть обращается в равенство, признак f_j, фактически, исключается из модели регрессии. В линейной композиции это соответствует исключению j-го базового алгоритма из композиции.

§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–признакик 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 $f_j(x)$, $j = 1, \ldots, n$. Как обычно, будем отождествлять объекты обучающей выборки и их признаковые описания: $x_i \equiv (f_1(x_i), \ldots, f_n(x_i))$, $i = 1, \ldots, \ell$. Рассмотрим матрицу F, строки которой соответствуют признаковым описаниям обучающих объектов:

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

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

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

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

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

или в векторной записи: $\hat{x} = zU^\mathsf{T}$. 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:

$$\Delta^2(G, U) = \sum_{i=1}^{\ell} \|\hat{x}_i - x_i\|_2 = \sum_{i=1}^{\ell} \|z_i U^\mathsf{T} - x_i\|_2 = \|GU^\mathsf{T} - F\|_2^2 \to \min_{G,U}, \tag{5.10}$$

где все нормы евклидовы. Напомним, что $\|A\|_2^2 = \mathrm{tr}\, AA^\mathsf{T} = \mathrm{tr}\, A^\mathsf{T}A$, где tr — операция следа матрицы.

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

Исчерпывающее решение задачи (5.10) даёт следующая теорема.

Теорема 5.2. *Если $m \leq \mathrm{rk}\, F$, то минимум $\Delta^2(G, U)$ достигается, когда столбцы матрицы U есть собственные векторы $F^\mathsf{T}F$, соответствующие m максимальным собственным значениям. При этом $G = FU$, матрицы U и G ортогональны.*

Доказательство. Запишем необходимые условия минимума:

$$\begin{cases} \partial\Delta^2/\partial G = (GU^\intercal - F)U = 0; \\ \partial\Delta^2/\partial U = G^\intercal(GU^\intercal - F) = 0. \end{cases}$$

Поскольку искомые матрицы G и U невырождены, отсюда следует

$$\begin{cases} G = FU\,(U^\intercal U)^{-1}; \\ U = F^\intercal G(G^\intercal G)^{-1}. \end{cases} \tag{5.11}$$

Функционал $\Delta^2(G, U)$ зависит только от произведения матриц $GU^\intercal$, поэтому решение задачи (5.10) определено с точностью до произвольного невырожденного преобразования R: $GU^\intercal = (GR)(R^{-1}U^\intercal)$. Распорядимся свободой выбора R так, чтобы матрицы $U^\intercal U$ и $G^\intercal G$ оказались диагональными. Покажем, что это всегда возможно.

Пусть $\tilde{G}\tilde{U}^\intercal$ — произвольное решение задачи (5.10).

Матрица $\tilde{U}^\intercal\tilde{U}$ симметричная, невырожденная, положительно определенная, поэтому существует невырожденная матрица $S_{m\times m}$ такая, что $S^{-1}\tilde{U}^\intercal\tilde{U}S^{-1\intercal} = I_m$.

Матрица $S^\intercal\tilde{G}^\intercal\tilde{G}S$ симметричная и невырожденная, поэтому существует ортогональная матрица $T_{m\times m}$ такая, что $T^\intercal(S^\intercal\tilde{G}^\intercal\tilde{G}S)T = \operatorname{diag}(\lambda_1, \ldots, \lambda_m) \equiv \Lambda$ — диагональная матрица. По определению ортогональности $T^\intercal T = I_m$.

Преобразование $R = ST$ невырождено. Положим $G = \tilde{G}R$, $U^\intercal = R^{-1}\tilde{U}^\intercal$. Тогда

$$G^\intercal G = T^\intercal(S^\intercal\tilde{G}^\intercal\tilde{G}S)T = \Lambda;$$

$$U^\intercal U = T^{-1}(S^{-1}\tilde{U}^\intercal\tilde{U}S^{-1\intercal})T^{-1\intercal} = (T^\intercal T)^{-1} = I_m.$$

В силу $GU^\intercal = \tilde{G}\tilde{U}^\intercal$ матрицы G и U являются решением задачи (5.10) и удовлетворяют необходимому условию минимума. Подставим матрицы G и U в (5.11). Благодаря диагональности $G^\intercal G$ и $U^\intercal U$ соотношения существенно упростятся:

$$\begin{cases} G = FU; \\ U\Lambda = F^\intercal G. \end{cases}$$

Подставим первое соотношение во второе, получим $U\Lambda = F^\intercal FU$. Это означает, что столбцы матрицы U обязаны быть собственными векторами матрицы $F^\intercal F$, а диагональные элементы $\lambda_1, \ldots, \lambda_m$ — соответствующими им собственными значениями.

Аналогично, подставив второе соотношение в первое, получим $G\Lambda = FF^\intercal G$, то есть столбцы матрицы G являются собственными векторами $FF^\intercal$, соответствующими тем же самым собственным значениям.

Подставляя G и U в функционал $\Delta^2(G, U)$, находим:

$$\begin{aligned} \Delta^2(G, U) &= \|F - GU^\intercal\|^2 = \operatorname{tr}(F^\intercal - UG^\intercal)(F - GU^\intercal) = \operatorname{tr} F^\intercal(F - GU^\intercal) = \\ &= \operatorname{tr} F^\intercal F - \operatorname{tr} F^\intercal GU^\intercal = \|F\|^2 - \operatorname{tr} U\Lambda U^\intercal = \\ &= \|F\|^2 - \operatorname{tr}\Lambda = \sum_{j=1}^{n}\lambda_j - \sum_{j=1}^{m}\lambda_j = \sum_{j=m+1}^{n}\lambda_j, \end{aligned}$$

где $\lambda_1, \ldots, \lambda_n$ — все собственные значения матрицы $F^\intercal F$. Минимум Δ^2 достигается, когда $\lambda_1, \ldots, \lambda_m$ — наибольшие m из n собственных значений. □

Собственные векторы $u_1, \ldots, u_m$, отвечающие максимальным собственным значениям, называют *главными компонентами*.

Из Теоремы 5.2 вытекают следующие свойства метода главных компонент.

Связь с сингулярным разложением. Если $m = n$, то $\Delta^2(G, U) = 0$. В этом случае представление $F = GU^\top$ является точным и совпадает с сингулярным разложением: $F = GU^\top = VDU^\top$, если положить $G = VD$ и $\Lambda = D^2$. При этом матрица V ортогональна: $V^\top V = I_m$. Остальные свойства сингулярного разложения, перечисленные на стр. 85, непосредственно вытекают из Теоремы 5.2.

Если $m < n$, то представление $F \approx GU^\top$ является приближённым. Сингулярное разложение матрицы $GU^\top$ получается из сингулярного разложения матрицы F путём отбрасывания (обнуления) $n - m$ минимальных собственных значений.

Преобразование Карунена-Лоэва. Диагональность матрицы $G^\top G = \Lambda$ означает, что новые признаки $g_1, \ldots, g_m$ не коррелируют на обучающих объектах. Ортогональное преобразование U называют *декоррелирующим* или преобразованием *Карунена–Лоэва*. Если $m = n$, то прямое и обратное преобразование вычисляются с помощью одной и той же матрицы U: $F = GU^\top$ и $G = FU$.

Задача наименьших квадратов в новом признаковом пространстве имеет вид

$$\|G\beta - y\|^2 \to \min_{\beta}.$$

Поскольку U ортогональна, $G\beta = GU^\top U\beta = GU^\top\alpha \approx F\alpha$, где $\alpha = U\beta$. Это означает, что задача наименьших квадратов в новом пространстве соответствует замене матрицы F на её приближение $GU^\top$ в исходной задаче наименьших квадратов.

Интересно отметить, что новый вектор коэффициентов β связан со старым α тем же линейным преобразованием U: $\beta = U^\top U\beta = U^\top\alpha$.

В новом пространстве МНК-решение не требует явного обращения матрицы, поскольку $G^\top G$ диагональна:

$$\beta^* = \Lambda^{-1}G^\top y = D^{-1}V^\top y;$$
$$G\beta^* = VD\beta^* = VV^\top y.$$

Для вектора $\alpha^* = U\beta^*$ МНК-решение выглядит так же, как и раньше, с той лишь разницей, что в суммах (5.5)–(5.7) надо взять первые $m \leq n$ слагаемых, а оставшиеся $n - m$ просто отбросить.

Интересно сравнить метод главных компонент и гребневую регрессию. Оба сводятся к модификации сумм (5.5)–(5.7). Гребневая регрессия сокращает коэффициенты при всех слагаемых, а метод главных компонент обнуляет коэффициенты при последних слагаемых. Возможно, имеют смысл и комбинации этих двух методов.

Эффективная размерность. Главные компоненты содержат основную информацию о матрице F. Число главных компонент m называют также *эффективной размерностью* задачи. На практике её определяют следующим образом. Все собственные значения матрицы $F^\top F$ упорядочиваются по убыванию: $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$. Задаётся пороговое значение $\varepsilon \in [0, 1]$, достаточно близкое к нулю, и определяется наименьшее целое m, при котором относительная погрешность приближения матрицы F не превышает ε:

$$E(m) = \frac{\|GU^\top - F\|^2}{\|F\|^2} = \frac{\lambda_{m+1} + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_n} \leq \varepsilon.$$

Величина $E(m)$ показывает, какая доля информации теряется при замене исходных признаковых описаний длины n на более короткие описания длины m. Метод главных компонент особенно эффективен в тех случаях, когда $E(m)$ оказывается малым уже при малых значениях m.

Если задать число ε из априорных соображений не представляется возможным, прибегают к *критерию «крутого обрыва»*. На графике $E(m)$ отмечается то значение m, при котором происходит резкий скачок: $E(m-1) \gg E(m)$, при условии, что $E(m)$ уже достаточно мало́.

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 (см. **??**) и карты Кохонена (см. 7.2.2).

§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.

Нелинейная модель регрессии

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

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

Для выполнения численной минимизации функционала Q воспользуемся методом Ньютона–Рафсона. Выберем начальное приближение $\alpha^0 = (\alpha^0_1, \ldots, \alpha^0_p)$ и организуем итерационный процесс

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

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:

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

Запишем компоненты гессиана:

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

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:

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

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; $f_t = \left(f(x_i, \alpha^t)\right)_{i=1,\ell}$ — вектор значений аппроксимирующей функции на t-й итерации. Тогда формула t-й итерации метода Ньютона–Гаусса в матричной записи примет вид:

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

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.

Нелинейные одномерные преобразования признаков

На практике встречаются ситуации, когда линейная модель регрессии представляется необоснованной, но предложить адекватную нелинейную модель $f(x, \alpha)$

Алгоритм 5.2. Метод настройки с возвращениями (backfitting).

Вход:

F, y — матрица «объекты–признаки» k и вектор ответов;

Выход:

$\phi_j(x)$ — функции преобразования признаков, в общем случае нелинейные.

1: нулевое приближение:
   $\alpha :=$ решение задачи МЛР с признаками $f_j(x)$;
   $\phi_j(x) := \alpha_j f_j(x), \quad j = 1, \ldots, n$;
2: **повторять**
3: **для** $j = 1, \ldots, n$
4: $z_i := y_i - \sum_{k=1, k \neq j} \phi_k(f_k(x_i)), \quad i = 1, \ldots, \ell$;
5: $\phi_j := \arg\min_{\phi} \sum_{i=1}^{\ell} \phi(f_j(x)) - z_i)^2$;
6: $Q_j := \sum_{i=1}^{\ell} \phi_j(f_j(x)) - z_i)^2$;
7: **пока** значения Q_j не стабилизируются

также не удаётся. Тогда в качестве компромисса строится модель вида

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

где $\phi_j : R \to R$ — 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 [43]. Схема реализации показана в Алгоритме 5.2.

Метод основан на итерационном уточнении функций ϕ_j. На первом шаге они полагаются линейными, $\phi_j(x) = \alpha_j f_j(x)$, 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

$$Q(\phi_j, X^\ell) = \sum_{i=1}^{\ell} \phi_j(f_j(x_i)) - \underbrace{\Big(y_i - \sum_{k=1, k \neq j}^{n} \phi_k(f_k(x_i))\Big)}_{z_i = \text{const}(\phi_j)}\Bigg)^2 \to \min_{\phi_j}$$

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.

Обобщённые линейные модели

Рассмотрим другую ситуацию, когда модель регрессии $f(x, \alpha)$ линейна, но известна нелинейная функция связи $g(f)$ между выходом модели f и целевым признаком y. Задача аппроксимации ставится, исходя из принципа наименьших квадратов:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} g\Big(\underbrace{\textstyle\sum_{j=1}^{n} \alpha_j f_j(x_i)}_{z_i}\Big) - y_i\Big)^2 \to \min_{\alpha \in \mathbb{R}^n},$$

где $g(f)$ — заданная непрерывно дифференцируемая функция.

Допустим, имеется некоторое приближение вектора коэффициентов α. Линеаризуем функцию $g(z)$ в окрестности каждого из ℓ значений z_i:

$$g(z) = g(z_i) + g'(z_i)(z - z_i).$$

Тогда функционал Q аппроксимируется функционалом $\tilde{Q}$, квадратичным по вектору коэффициентов α:

$$\tilde{Q}(\alpha, X^\ell) = \sum_{i=1}^{\ell} \Big(g(z_i) + g'(z_i)\Big(\sum_{j=1}^{n} \alpha_j f_j(x_i) - z_i\Big) - y_i\Big)^2 =$$

$$= \sum_{i=1}^{\ell} \underbrace{g'(z_i)^2}_{w_i} \Big(\sum_{j=1}^{n} \alpha_j f_j(x_i) - \underbrace{\Big(z_i + \frac{y_i - g(z_i)}{g'(z_i)}\Big)}_{\tilde{y}_i}\Big)^2 \to \min_{\alpha \in \mathbb{R}^n}.$$

Линеаризованная задача сводится к стандартной многомерной линейной регрессии с весами объектов w_i и модифицированным целевым признаком $\tilde{y}$. Решение этой задачи принимается за следующее приближение вектора коэффициентов α. Итерации повторяются до тех пор, пока вектор коэффициентов α или значение функционала $Q(\alpha)$ не перестанет существенно изменяться.

Неквадратичные функции потерь

Функция потерь $\mathsf{L}(a, y)$ характеризует величину потери от ответа $a \in Y$ при точном ответе $y \in Y$. Она задаётся априори, и благодаря ей задача обучения алгоритма $a(x)$ по выборке $X^\ell = (x_i, y_i)_{i=1}^{\ell}$ сводится к минимизации суммарных потерь:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \mathsf{L}\big(a(x_i), y(x_i)\big) \to \min_{a:\, X \to Y}.$$

Если функция потерь квадратична, $\mathsf{L}(a, y) = (a - y)^2$, то минимизация Q соответствует методу наименьших квадратов, который был рассмотрен выше. При неквадратичных функциях потерь применяются численные методы оптимизации. Мы не будем подробно останавливаться на методах, а ограничимся перечислением ситуаций, в которых возникают функции потерь, отличные от квадратичных.

Ненормальный шум. Как было показано в Примере 1.4, вид функции потерь связан с априорными предположениями о распределении шума. В частности, квадратичная функция потерь соответствует гауссовскому шуму. Если распределение шума не гауссовское, то функция потерь окажется неквадратичной.

Проблемно-зависимые функции потерь. Во многих прикладных задачах минимизация ошибки предсказания $|a - |y$ или максимизация правдоподобия являются не самыми естественными критериями качества алгоритма.

Пример 5.1. При планировании закупок в розничной сети решается регрессионная задача прогнозирования потребительского спроса. Строится алгоритм $a(x)$, который отвечает на вопрос, сколько единиц данного товара купят в данном магазине в ближайшее время (скажем, в течение следующей недели). Квадрат отклонения $(a\ y)^2$ прогноза a от реального спроса y экономического смысла не имеет. Гораздо удобнее измерять потери в рублях. Потери от заниженного прогноза $a < y$ связаны с недополученной прибылью и прямо пропорциональны величине отклонения: $\mathsf{L}(a, y) = c_1|a - |y$, где c_1 — коэффициент торговой наценки. Потери от завышенного прогноза $a > y$ связаны с замораживанием средств, затовариванием склада, а в худшем случае — с истечением срока годности и списанием товара. В первом приближении эти потери также прямо пропорциональны отклонению, но с другим коэффициентом: $\mathsf{L}(a, y) = c_2|a - y|$. Коэффициенты c_1 и c_2 известны для каждого магазина и каждого товара. Таким образом, в данной задаче более обоснованной оказывается не квадратичная, а кусочно-линейная несимметричная функция потерь.

Пример 5.2. При создании автоматических систем биржевой торговли строится алгоритм $a(x)$, прогнозирующий в момент времени x_i цену финансового инструмента на следующий момент x_{i+1}. В данном случае квадрат отклонения $(a\ y)^2$ особого интереса не представляет. Экономический смысл имеет величина прибыли, которую можно получить, играя на бирже с применением алгоритма $a(x)$. Допустим, мы покупаем 1 акцию, если алгоритм предсказывает повышение, и продаём 1 акцию, если он предсказывает понижение. В следующий момент времени совершаем противоположную операцию (на языке трейдеров «закрываем позициюk), и тут же принимаем следующее решение согласно алгоритму $a(x)$, и так далее. Суммарная прибыль, заработанная в течение ℓ последовательных моментов времени $x_1, \ldots, x_\ell$, равна

$$Q(a) = \sum_{i=1}^{\ell} \operatorname{sign}\ a(x_i) - y(x_i)\quad y(x_{i+1}) - y(x_i)\ .$$

Обучение алгоритма a сводится к максимизации функционала $Q(a)$ на обучающей последовательности цен $y(x_1), \ldots, y(x_{\ell+1})$. Разности цен $w_i = |y(x_{i+1}) - y(x_i)|$ играют роль весов объектов — чем больше величина скачка w_i, тем важнее правильно спрогнозировать направление скачка. Итак, в данной задаче содержательно обоснованной оказалась взвешенная кусочно-постоянная функция потерь.

Робастная регрессия. Чтобы функционал $Q(a, X^\ell)$ был нечувствителен к выбросам, вводится огранич енная сверху функция потерь, например, функция Мешалкина $\mathsf{L}(a, y) = 1 - \exp\left(-\frac{1}{2\sigma}(a - y)^2\right)$, где σ — параметр, равный дисперсии «обычногk шума, не связанного с большими выбросами. Задача минимизации функционала $Q(a, X^\ell)$ с такой функцией потерь уже не может быть решена средствами линейной алгебры; приходится применять численные методы оптимизации, например, метод сопряжённых градиентов.

Логистическая регрессия и итерационный взвешенный МНК

Напомним, что неквадратичная функция потерь используется также в *логистической регрессии*, см. §4.4. Там минимизируемый функционал имеет вид

$$Q(w) = \sum_{i=1}^{\ell} \ln\left(1 + \exp\left(-w^{\mathsf{T}} x_i y_i\right)\right) = -\sum_{i=1}^{\ell} \ln \sigma\left(w^{\mathsf{T}} x_i y_i\right) \to \min_w,$$

где $\sigma(z) = (1 + e^{-z})^{-1}$ — сигмоидная функция.

Стандартная техника настройки параметров w заключается в применении метода Ньютона-Рафсона для минимизации нелинейного функционала $Q(w)$. В качестве нулевого приближения можно взять «наивноek решение задачи классификации как задачи многомерной линейной регрессии, в которой ответы принимают только два значения, $y_i \in \{-1, +1\}$. Затем начинается итерационный процесс, на t-м шаге которого уточняется вектор коэффициентов w^{t+1}:

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

где $Q'(w^t)$ — вектор первых производных (градиент) функционала $Q(w)$ в точке w^t, $Q''(w^t)$ — матрица вторых производных (гессиан) функционала $Q(w)$ в точке w^t, h_t — величина шага, который можно положить равным 1, но более тщательный его подбор способен увеличить скорость сходимости.

Найдём выражения для градиента и гессиана.

Обозначим $\sigma_i = \sigma(y_i w^{\mathsf{T}} x_i)$ и заметим, что производная логистической функции есть $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.

Элементы градиента (вектора первых производных) функционала $Q(w)$:

$$\frac{\partial Q(w)}{\partial w_j} = -\sum_{i=1}^{\ell} (1 - \sigma_i) y_i f_j(x_i), \qquad j = 1, \ldots, n.$$

Элементы гессиана (матрицы вторых производных) функционала $Q(w)$:

$$\frac{\partial^2 Q(w)}{\partial w_j \partial w_k} = -\frac{\partial}{\partial w_k} \sum_{i=1}^{\ell} (1 - \sigma_i) y_i f_j(x_i) =$$

$$= \sum_{i=1}^{\ell} (1 - \sigma_i)\sigma_i f_j(x_i) f_k(x_i), \qquad j = 1, \ldots, n, \quad k = 1, \ldots, n.$$

Введём матричные обозначения:

$F_{\ell \times n} = \left(f_j(x_i)\right)$ — матрица признаковых описаний объектов;

$\Gamma_{\ell \times \ell} = \operatorname{diag}\left(\sqrt{(1 - \sigma_i)\sigma_i}\right)$ — диагональная матрица весов объектов;

$\tilde{F} = \Gamma F$ — взвешенная матрица признаковых описаний объектов;

$\tilde{y}_i = y_i \sqrt{(1 - \sigma_i)/\sigma_i}$, $\tilde{y} = (\tilde{y}_i)_{i=1}^{\ell}$ — взвешенный вектор ответов.

В этих обозначениях произведение матрицы, обратной к гессиану, на вектор градиента принимает следующий вид:

$$Q''(w)^{-1} Q'(w) = -(F^{\mathsf{T}} \Gamma^2 F)^{-1} F^{\mathsf{T}} \Gamma \tilde{y} = -(\tilde{F}^{\mathsf{T}} \tilde{F})^{-1} \tilde{F}^{\mathsf{T}} \tilde{y} = -\tilde{F}^{+} \tilde{y}.$$

Алгоритм 5.3. IRLS — итерационный взвешенный метод наименьших квадратов

Вход:
F, y — матрица «объекты–признаки» и вектор ответов;
Выход:
w — вектор коэффициентов линейной комбинации.

1: нулевое приближение — обычный МНК:
$w := (F^\top F)^{-1} F^\top y$;
2: **для** $t := 1, 2, 3, \ldots$
3: $z := Fw$;
4: $\gamma_i := \sqrt{(1 - \sigma(z_i))\sigma(z_i)}$ для всех $i = 1, \ldots, \ell$;
5: $\tilde{F} := \operatorname{diag}(\gamma_1, \ldots, \gamma_\ell) F$;
6: $\tilde{y}_i := y_i \sqrt{(1 - \sigma(z_i))/\sigma(z_i)}$ для всех $i = 1, \ldots, \ell$;
7: выбрать градиентный шаг h_t;
8: $w := w + h_t (\tilde{F}^\top \tilde{F})^{-1} \tilde{F}^\top \tilde{y}$;
9: **если** $\sigma(z_i)$ мало изменились относительно предыдущей итерации **то**
10: прервать итерации, выйти из цикла;
11: **конец** цикла по t.

Полученное выражение совпадает с решением задачи наименьших квадратов для многомерной линейной регрессии со взвешенными объектами и модифицированными ответами:

$$Q(w) = \|\tilde{F}w - \tilde{y}\|^2 = \sum_{i=1}^{\ell} \underbrace{(1 - \sigma_i)\sigma_i}_{\gamma_i^2} \Big(w^\top x_i - y_i \underbrace{\sqrt{(1 - \sigma_i)/\sigma_i}}_{\tilde{y}_i} \Big)^2 \to \min_w.$$

Таким образом, решение задачи классификации сводится к последовательности регрессионных задач, для каждой из которых веса объектов и ответы пересчитываются заново. Отсюда и название — метод *наименьших квадратов с итерационным перевзвешиванием* (iteratively reweighted least squares, IRLS)

Понять смысл этого пересчёта совсем нетрудно. Во-первых, заметим, что величина σ_i равна вероятности правильного ответа алгоритма w^t на объекте x_i. Поэтому вес γ_i максимален для пограничных объектов, у которых эта вероятность близка к $\frac{1}{2}$. Увеличение точности настройки на этих объектах способствует уменьшению неопределённости классификации. Во-вторых, по мере увеличения вероятности ошибки алгоритма w^t на объекте x_i модифицированный ответ $\tilde{y}_i$ возрастает по модулю. Это приводит к повышению точности настройки алгоритма w^{t+1} на тех объектах, которые оказались «наиболее трудными» для алгоритма w^t на предыдущей итерации.

§5.6 Метод опорных векторов в задачах регрессии

В главе 5 мы уже рассматривали задачи многомерной линейной регрессии, предполагая, что $X = \mathrm{R}^n$, $Y = \mathrm{R}$, алгоритм имеет вид $a(x) = \langle w, x \rangle - w_0$, и для настройки параметров $w \in \mathrm{R}^n$ и $w_0 \in \mathrm{R}$ минимизируется квадратичный функционал. В случае гребневой регрессии (см. раздел 5.3.3) вводится ещё и штрафное слагаемое,

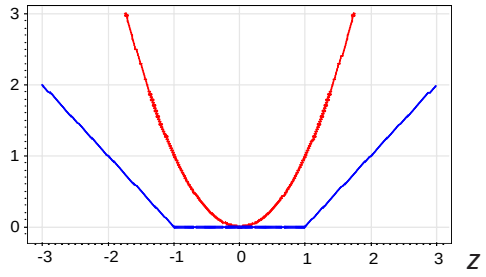


Рис. 17. Функции потерь в задачах регрессии: кусочно-линейная $|z|_\varepsilon$ при $\varepsilon = 1$ и квадратичная z^2.

предотвращающее бесконтрольное увеличение коэффициентов w:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \big(\langle w, x_i \rangle - w_0 - y_i \big)^2 + \tau \|w\|^2 \to \min_{w, w_0},$$

где τ — параметр регуляризации. Выбор именно квадратичной функции потерь обусловлен удобством решения задачи наименьших квадратов.

Однако в некоторых случаях более естественно использовать кусочно-линейную *функцию ε-чувствительности*, показанную на Рис 17: $|z|_\varepsilon = (|z| - \varepsilon)_+$, которая не считает за ошибки отклонения $a(x_i)$ от y_i, меньшие ε. Предполагается, что значение параметра ε задаёт эксперт, исходя из априорных соображений.

С этой функцией потерь функционал принимает вид

$$Q_\varepsilon(a, X^\ell) = \sum_{i=1}^{\ell} \big(|\langle w, x_i \rangle - w_0 - y_i| - \varepsilon \big)_+ + \tau \langle w, w \rangle^2 \to \min_{w, w_0}. \qquad (5.12)$$

Легко обнаруживается сходство данной задачи с задачей классификации (4.18). Покажем, что минимизация (5.12) эквивалентна некоторой задаче квадратичного программирования с линейными ограничениями типа неравенств. При этом также возникает двойственная задача, зависящая только от двойственных переменных; также достаточно оставить в выборке только опорные объекты; также решение выражается через скалярные произведения объектов, а не сами объекты; и также можно использовать ядра. Иными словами, SVM-регрессия отличается от SVM-классификации только в технических деталях, основные идеи остаются теми же.

Положим $C = \frac{1}{2\tau}$. Введём дополнительные переменные ξ_i^+ и ξ_i^-, значения которых равны потере при завышенном и заниженном ответе $a(x_i)$ соответственно:

$$\xi_i^+ = (a(x_i) - y_i - \varepsilon)_+, \quad \xi_i^- = (-a(x_i) + y_i - \varepsilon)_+, \quad i = 1, \ldots, \ell.$$

Тогда задача минимизации (5.12) может быть переписана в эквивалентной форме как задача квадратичного программирования с линейными ограничениями-неравенствами относительно переменных w_i, w_0, ξ_i^+ и ξ_i^-:

$$\begin{cases} \frac{1}{2}\langle w, w \rangle + C \sum_{i=1}^{\ell} (\xi_i^+ + \xi_i^-) \to \min_{w, w_0, \xi^+, \xi^-}; \\ y_i - \varepsilon - \xi_i^- \le \langle w, x_i \rangle - w_0 \le y_i + \varepsilon + \xi_i^+, \quad i = 1, \ldots, \ell; \\ \xi_i^- \ge 0, \quad \xi_i^+ \ge 0, \quad i = 1, \ldots, \ell. \end{cases} \qquad (5.13)$$

Как и в предыдущих случаях, лагранжиан этой задачи выражается через двойственные переменные λ_i^+, λ_i^-, $i = 1, \ldots, \ell$, а скалярные произведения $\langle x_i, x_j \rangle$ можно

заменить ядром $K(x_i, x_j)$. Опуская выкладки, представим результат:

$$\begin{cases} L(\lambda^+, \lambda^-) = -\varepsilon \sum\limits_{i=1}^{\ell} (\lambda_i^- + \lambda_i^+) + \sum\limits_{i=1}^{\ell} (\lambda_i^- - \lambda_i^+) y_i - \\ \qquad -\frac{1}{2} \sum\limits_{i,j=1}^{\ell} (\lambda_i^- - \lambda_i^+)(\lambda_j^- - \lambda_j^+) K(x_i, x_j) \to \max\limits_{\lambda^+, \lambda^-}; \\ 0 \le \lambda_i^+ \le C, \quad 0 \le \lambda_i^- \le C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} (\lambda_i^- + \lambda_i^+) = 0. \end{cases}$$

Все объекты x_i, $i = 1, \ldots, \ell$ делятся на следующие пять типов:

1. $|a(x_i) - y_i| < \varepsilon$; $\lambda_i^+ = \lambda_i^- = \xi_i^+ = \xi_i^- = 0$.

Ответ алгоритма $a(x_i)$ находится внутри отрезка $[y_i - \varepsilon, y_i + \varepsilon]$ и считается верным. Объект x_i не является опорным — вектор весов w не изменился бы, если бы этого объекта изначально не было в выборке.

2. $a(x_i) = y_i + \varepsilon$; $0 < \lambda_i^+ < C$; $\lambda_i^- = 0$; $\xi_i^+ = \xi_i^- = 0$.

3. $a(x_i) = y_i - \varepsilon$; $0 < \lambda_i^- < C$; $\lambda_i^+ = 0$; $\xi_i^+ = \xi_i^- = 0$.

4. $a(x_i) > y_i + \varepsilon$; $\lambda_i^+ = C$; $\lambda_i^- = 0$; $\xi_i^+ = a(x_i) - y_i - \varepsilon > 0$; $\xi_i^- = 0$.

5. $a(x_i) < y_i - \varepsilon$; $\lambda_i^- = C$; $\lambda_i^+ = 0$; $\xi_i^- = y_i - a(x_i) - \varepsilon > 0$; $\xi_i^+ = 0$.

Объекты типов 2–5 являются опорными и учитываются при определении вектора весов. При этом только на объектах типов 4 и 5 возникает ненулевая ошибка.

Уравнение регрессии также выражается через двойственные переменные:

$$a(x) = \sum_{i=1}^{\ell} (\lambda_i^- - \lambda_i^+) K(x_i, x) - w_0;$$

где параметр w_0 определяется из ограничений-неравенств, которые становятся равенствами на опорных объектах типа 2 и 3:

$$\langle w, x_i \rangle - w_0 = \begin{cases} y_i + \varepsilon, & \text{если } x_i \text{ — объект типа 2;} \\ y_i - \varepsilon, & \text{если } x_i \text{ — объект типа 3.} \end{cases}$$

Как и раньше, чтобы избежать численной неустойчивости, имеет смысл взять медиану множества значений w_0, вычисленных по всем опорным векторам.

В этом методе есть два управляющих параметра. Параметр точности ε задаётся из априорных соображений. Параметр регуляризации C подбирается, как правило, по скользящему контролю, что является вычислительно трудоёмкой процедурой.

Более обстоятельное изложение многочисленных особенностей SVM-регрессии можно найти в руководстве [61].

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 «подсмотретьk 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 вида (**??**) 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 «Персептронык [54]. Следующий классический контрпример иллюстрирует невозможность нейронной реализации даже очень простых функций.

Задача «исключающего ИЛИ»

Легко построить нейроны, реализующие логические функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2, см. Рис. 18:

$$x^1 \vee x^2 = x^1 + x^2 - \tfrac{1}{2} > 0\ ;$$

$$x^1 \wedge x^2 = x^1 + x^2 - \tfrac{3}{2} > 0\ ;$$

$$\neg x^1 = -x^1 + \tfrac{1}{2} > 0\ ;$$

Однако функцию $x^1 \oplus x^2 = [x^1 \quad x^2]$ — *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.

Возможны два пути решения этой проблемы, см. Рис 20.

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:

$$x^1 \oplus x^2 = x^1 + x^2 - 2x^1x^2 - \tfrac{1}{2} \geq 0\ .$$

Расширенные пространства признаков, в которых линейный классификатор безошибочно разделяет обучающую выборку, называют *спрямляющими*. Проблема

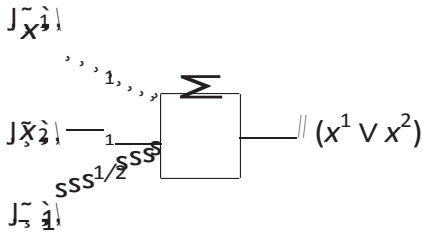


Рис. 18. Однослойный персептрон, реализующий операцию ИЛИ.

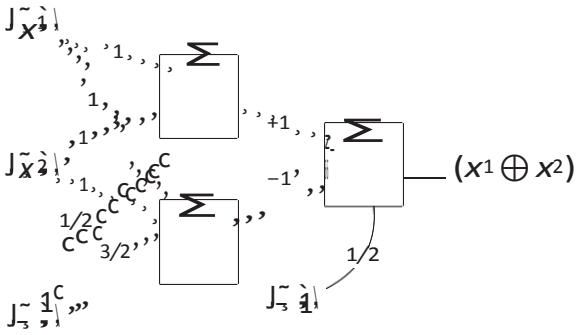


Рис. 19. Двухслойная сеть, реализующая операцию исключающего ИЛИ.

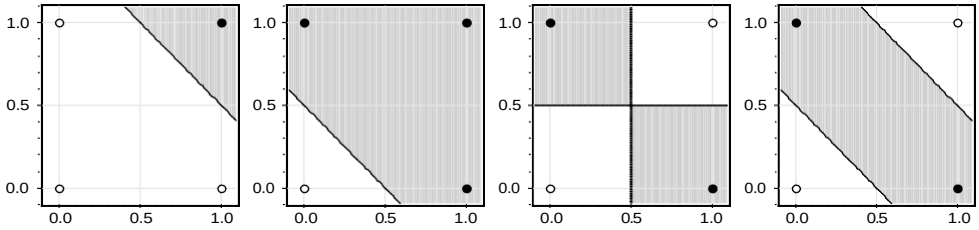


Рис. 20. Персептронная реализация элементарных логических функций. Слева направо: И, ИЛИ, XOR с помощью добавления признака x^1x^2, XOR с помощью двухслойной сети.

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, Рис 19:

$$x^1 \oplus x^2 = (x^1 \vee x^2) - (x^1 \wedge x^2) - \tfrac{1}{2} \geqslant 0 \ .$$

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 Рис. 21. 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*.[6] В общем случае сеть может содержать произвольное число слоёв. Все слои, за исключением последнего, называются *скрытыми* (hidden layers).

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]Существует терминологическая путаница с подсчётом числа слоёв. Иногда такую сеть (видимо, глядя на картинку) называют трёхслойной, считая входы $x^0, x^1, \ldots, x^n$ особым, «распределительнымс слоем. По делу, в счёт должны идти только слои, состоящие из суммирующих нейронов.

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 [29].

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 в [14].

Теорема 6.1 (Колмогоров, 1957). *Любая непрерывная функция n аргументов на единичном кубе $[0, 1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции сложения:*

$$f(x^1, x^2, \ldots, x^n) = \sum_{k=1}^{2n+1} h_k \sum_{i=1} \phi_{ik}(x^i) \ ,$$

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 [62].

Опр. 6.1. *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 6.2 (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 [21]. Этот результат имеет прямое отношение к нейронным сетям, поскольку они строятся из операций сложения, умножения и нелинейных функций активации.

Опр. 6.2. *Набор функций $F \subseteq C(X)$ называется замкнутым относительно функции $\phi : \mathrm{R} \to \mathrm{R}$, если для любого $f \in F$ выполнено $\phi(f) \in F$.*

Теорема 6.3 (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) [58].

Метод обратного распространения ошибок

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, Рис. 21. Такая сеть называется *полносвязной*. Для большей общности положим $X = \mathrm{R}^n$, $Y = \mathrm{R}^M$.

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

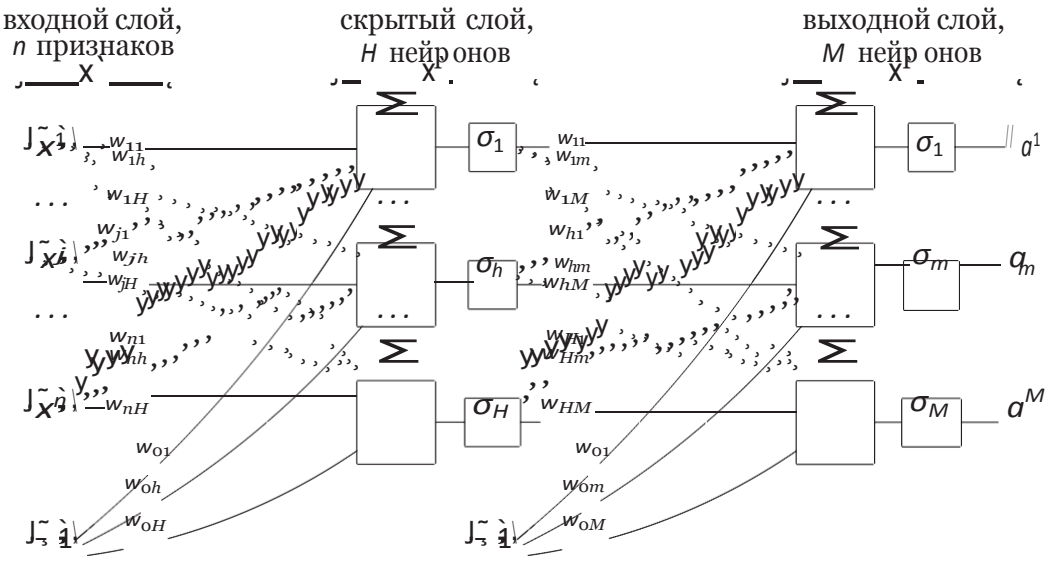


Рис. 21. Многослойная сеть с одним скрытым слоем.

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, $j = 1, \ldots, J$ и синаптическими весами w_{jh}. В общем случае число слоёв может быть произвольным. Если сеть двухслойная, то v^j есть просто j-й признак: $v^j(x) \equiv f_j(x)$ $x^j \equiv$ и $J = n$. Обозначим через w вектор всех синаптических весов сети.

Выходные значения сети на объекте x_i вычисляются как суперпозиция:

$$a^m(x_i) = \sigma_m \sum_{h=0} w_{hm} u^h(x_i) \;; \qquad u^h(x_i) = \sigma_h \sum_{j=0} w_{jh} v^j(x_i) \;. \tag{6.1}$$

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

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

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

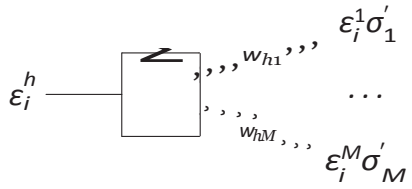
$$\frac{\partial Q(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m.$$

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

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

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 наперёдk, 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:

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

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

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, см. Алгоритм 6.1.

Достоинства метода обратного распространения.

- Достаточно высокая эффективность. В случае двухслойной сети прямой ход, обратный ход и вычисления градиента требуют порядка $O(Hn+HM)$ операций.
- Через каждый нейрон проходит информация только о связных с ним нейронах. Поэтому back-propagation легко реализуется на вычислительных устройствах с параллельной архитектурой.
- 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.

Недостатки метода обратного распространения.

- Метод наследует известные недостатки градиентной настройки весов в однослойном персептроне. Здесь также возникают проблемы медленной сходимости

Алгоритм 6.1. Обучение двухслойной сети методом back-propagation — обратного распространения ошибки

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;

или расходимости, «застреванияк в локальных минимумах функционала Q, переобучения и паралича. Причём парализоваться могут отдельные связи, нейроны, или вся сеть в целом.

- 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.

Эвристики для улучшения сходимости

Для улучшения сходимости и качества градиентного обучения применяются все те же приёмы, которые рекомендовались в 4.3.2 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 6.1 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 зонуk функций активации, представленных на Рис. 9.

Существует и более тонкий способ формирования начального приближения. Идея заключается в том, чтобы сначала настроить нейроны первого слоя по-отдельности, 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[7]. 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. В обзоре [50] даются 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.

[7] Фактически, такая двухслойная сеть является простейшей алгоритмической композицией. Нейроны первого скрытого слоя играют роль *базовых алгоритмов*, нейроны второго слоя реализуют *корректирующую операцию*. Обучение первого слоя по случайным подвыборкам с последующим взвешенным усреднением во втором слое в точности соответствует методу *бэггинга* (bagging), см. раздел **??**. Композиции общего вида рассматриваются в главе **??**.

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.

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) [51, 42] 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:

$$Q(w + \delta) = Q(w) + \tfrac{1}{2}\delta^{\mathsf{T}} H(w)\delta + o(\|\delta\|^2),$$

где $H(w) = \frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}$ — 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

$$\delta^{\mathsf{T}} H(w)\delta = \sum_{j=0}\sum_{h=1} \delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}.$$

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: $S_{jh} = w_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}$.

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_{hm} 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 $S_h = \sum_{m=1}^{M} S_{hm}$.

7 Кластеризация и визуализация

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, две строки с описанием молекул ДНК или протеинов гораздо легче сравнить непосредственно друг с другом, чем преобразовывать каждый из них в вектор признаков, и затем сравнивать эти векторы. То же самое можно сказать про тексты, временные ряды или растровые изображения.

Задача классификации объектов на основе их сходства друг с другом, когда принадлежность обучающих объектов каким-либо классам не задаётся, называется задачей *кластеризации*. В §7.1 рассматриваются статистические, иерархические и графовые алгоритмы кластеризации. В §7.3 рассматриваются методы *многомерного шкалирования*, позволяющие восстанавливать признаковые описания объектов по матрице попарных расстояний между ними.

§7.1 Алгоритмы кластеризации

Задача *кластеризации* (или обучения без учителя) заключается в следующем. Имеется обучающая выборка $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые *кластерами*, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^\ell$ приписывается метка (номер) кластера y_i.

Алгоритм кластеризации — это функция $a : X \to Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин. Во-первых, не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд достаточно разумных критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построениюk. Все они могут давать разные результаты. Во-вторых, число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием. В-третьих, результат кластеризации существенно зависит от метрики ρ, выбор которой, как правило, также субъективен и определяется экспертом.

Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки исходных объектов y_i изначально не заданы, и даже может быть неизвестно само множество Y. В этом смысле задача кластеризации ещё в большей степени некорректно поставленная, чем задача классификации.

Цели кластеризации могут быть различными в зависимости от особенностей конкретной прикладной задачи:

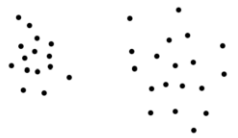
- Упростить дальнейшую обработку данных, разбить множество X^ℓ на группы схожих объектов чтобы работать с каждой группой в отдельности (задачи классификации, регрессии, прогнозирования).

- Сократить объём хранимых данных, оставив по одному представителю от каждого кластера (задачи сжатия данных).
- Выделить нетипичные объекты, которые не подходят ни к одному из кластеров (задачи одноклассовой классификации).
- Построить иерархию множества объектов (задачи таксономии).

В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

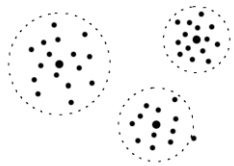
Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются *задачами таксономии* (taxonomy). Результатом таксономии является не простое разбиение множества объектов на кластеры, а древообразная иерархическая структура. Вместо номера кластера объект характеризуется перечислением всех кластеров, которым он принадлежит, от крупного к мелкому. Классическим примером таксономии на основе сходства является систематизация живых существ, предложенная Карлом Линнеем в середине XVIII века. В современном представлении биологическая иерархия имеет около 30 уровней, 7 из них считаются основными: царство, тип, класс, отряд, семейство, род, вид. Таксономии строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов. Мы будем рассматривать алгоритмы *иерархической кластеризации*, позволяющие автоматизировать процесс построения таксономий.

Типы кластерных структур. Попытаемся составить реестр различных типов кластерных структур, которые могут возникать в практических задачах.



Сгущения: внутрикластерные расстояния, как правило, меньше межкластерных.

Ленты: для любого объекта найдётся близкий к нему объект того же кластера, в то же время существуют объекты одного кластера, которые не являются близкими.



Кластеры с центром: в каждом кластере найдётся объект, такой, что почти все объекты кластера лежат внутри шара с центром в этом объекте.

Кластеры могут соединяться перемычками, что затрудняет работу многих алгоритмов кластеризации.

Кластеры могут накладываться на разреженный фон из редких нетипичных объектов.

Кластеры могут перекрываться.

Кластеры могут образовываться не по принципу сходства, а по каким-либо иным, заранее неизвестным, свойствам объектов. Стандартные методы кластеризации здесь бессильны.

Кластеры могут вообще отсутствовать. В этом случае надо применять не кластеризацию, а иные методы анализа данных.

Различные алгоритмы кластеризации могут быть более или менее успешны в этих ситуациях. Простые алгоритмы, как правило, узко специализированы и дают адекватные результаты только в одной-двух ситуациях. Более сложные алгоритмы, такие как FOREL или агломеративная процедура Ланса-Вильямса, справляются с несколькими типами ситуаций. Однако создание алгоритма, успешно работающего во всех ситуациях без исключения, представляется трудной и едва ли разрешимой задачей [18]. Неплохой обзор по методам кластеризации можно найти также в [46].

Эвристические графовые алгоритмы

Обширный класс алгоритмов кластеризации основан на представлении выборки в виде графа. Вершинам графа соответствуют объекты выборки, а рёбрам — попарные расстояния между объектами $\rho_{ij} = \rho(x_i, x_j)$.

Достоинством графовых алгоритмов кластеризации является наглядность, относительная простота реализации, возможность вносить различные усовершенствования, опираясь на простые геометрические соображения.

Алгоритм выделения связных компонент. Задаётся параметр R и в графе удаляются все рёбра (i, j), для которых $\rho_{ij} > R$. Соединёнными остаются только наиболее близкие пары объектов. Идея алгоритма заключается в том, чтобы подобрать такое значение $R \in [\min \rho_{ij}, \max \rho_{ij}]$, при котором граф развалится на несколько связных компонент. Найденные связные компоненты — и есть кластеры.

Связной компонентой графа называется подмножество его вершин, в котором любые две вершины можно соединить путём, целиком лежащим в этом подмножестве. Для поиска связных компонент можно использовать стандартные алгоритмы поиска в ширину (алгоритм Дейкстры) или поиска в глубину.

Алгоритм 7.1. Алгоритм кратчайшего незамкнутого пути (КНП)

1: Найти пару точек (i, j) с наименьшим ρ_{ij} и соединить их ребром;
2: **пока** в выборке остаются изолированные точки
3: найти изолированную точку, ближайшую к некоторой неизолированной;
4: соединить эти две точки ребром;
5: удалить $K - 1$ самых длинных рёбер;

Для подбора параметра R обычно рекомендуется построить гистограмму распределения попарных расстояний ρ_{ij}. В задачах с выраженной кластерной структурой эта гистограмма имеет два чётких пика: зона небольших внутриклассовых расстояний и зона больших межклассовых расстояний. Параметр R задаётся как расстояние, соответствующее точке минимума между этими пиками [16].

Отметим два недостатка этого алгоритма.

- Ограниченная применимость. Алгоритм выделения связных компонент наиболее подходит для выделения кластеров типа сгущений или лент. Наличие разреженного фона или «узких перемычекk между кластерами приводит к неадекватной кластеризации.

- Плохая управляемость числом кластеров. Для многих приложений удобнее задавать не параметр R, а число кластеров или некоторый порог «чёткости кластеризациик. Управлять числом кластеров с помощью параметра R довольно затруднительно. Приходится многократно решать задачу при разных R, что отрицательно сказывается на временных затратах.

Алгоритм кратчайшего незамкнутого пути строит граф из $\ell - 1$ рёбер так, чтобы они соединяли все ℓ точек и обладали минимальной суммарной длиной. Такой граф называется *кратчайшим незамкнутым путём* (КНП), минимальным покрывающим деревом или каркасом. Доказано, что этот граф строится с помощью несложной процедуры, соответствующей шагам 1–4 Алгоритма 7.1. На шаге 5 удаляются K 1 самых длинных рёбер, и связный граф распадается на K кластеров.

В отличие от предыдущего алгоритма, число кластеров K задаётся как входной параметр. Его можно также определять графически, если упорядочить все расстояния, образующие каркас, в порядке убывания и отложить их на графике. Резкий скачок вниз где-то на начальном (левом) участке графика покажет количество наиболее чётко выделяемых кластеров.

Этот алгоритм, как и предыдущий, очень прост и также имеет ограниченную применимость. Наличие разреженного фона или перемычек приводит к неадекватной кластеризации. Другим недостатком КНП является высокая трудоёмкость — для построения кратчайшего незамкнутого пути требуется $O(\ell^3)$ операций.

Алгоритм FOREL (ФОРмальный ЭЛемент) предложен Загоруйко и Ёлкиной в 1967 году при решении одной прикладной задачи в области палеонтологии. Алгоритм имеет многочисленные вариации, подробно описанные в [12, 11]. В основе всех этих вариаций лежит следующая базовая процедура.

Пусть задана некоторая точка $x_0 \in X$ и параметр R. Выделяются все точки выборки $x_i \in X^\ell$, попадающие внутрь сферы $\rho(x_i, x_0) \le R$, и точка x_0 переносится в центр тяжести выделенных точек. Эта процедура повторяется до тех пор, пока состав выделенных точек, а значит и положение центра, не перестанет меняться. Доказано, что эта процедура сходится за конечное число шагов. При этом сфера перемещается в место локального сгущения точек. Центр сферы x_0 в общем случае не является объектом выборки, потому и называется *формальным элементом*.

Для вычисления центра необходимо, чтобы множество объектов X было не только метрическим, но и линейным векторным пространством. Это требование естественным образом выполняется, когда объекты описываются числовыми признаками. Однако существуют задачи, в которых изначально задана только метрика, а сложение и умножение на число не определены на X. Тогда в качестве центра сферы можно взять тот объект обучающей выборки, для которого среднее расстояние до других объектов кластера минимально. Соответственно, шаг 6 заменяется на

$$x_0 := \arg\min_{x \in K_0} \sum_{x' \in K_0} \rho(x, x').$$

При этом заметно увеличивается трудоёмкость алгоритма. Если в линейном пространстве для вычисления центра требуется $O(к)$ операций, то в метрическом — $O(к^2)$, где к — чисто точек в кластере. Алгоритм можно несколько ускорить, если заметить, что пересчёт центра при добавлении или удалении отдельной точки кластера требует лишь $O(к)$ операций, а в линейном пространстве — $O(1)$.

Различные варианты алгоритма FOREL отличаются способами объединения сфер в кластеры, способами варьирования параметра R, способами выбора начального приближения для точек x_0. В Алгоритме 7.2 представлен один из вариантов, в котором сферы строятся последовательно. На шаге 9 к центрам этих сфер применяется алгоритм КНП. С одной стороны, это решает проблему низкой эффективности КНП, так как сфер гораздо меньше, чем исходных объектов. С другой стороны, мы получаем более тонкую, двухуровневую, структуру кластеров: каждый кластер верхнего уровня распадается на более мелкие подкластеры нижнего уровня.

Другое преимущество этого алгоритма — возможность описывать кластеры произвольной геометрической формы. Варьируя параметр R, можно получать кластеризации различной степени детальности. Если кластеры близки по форме к шарам, можно сделать R достаточно большим. Для описания кластеров более сложной формы следует уменьшать R.

Алгоритм 7.2 довольно чувствителен к выбору начального положения точки x_0 для каждого нового кластера. Для устранения этого недостатка в [11] предлагается генерировать несколько (порядка 10..20) кластеризаций. Поскольку начальное положение центров выбирается случайным образом, эти кластеризации будут довольно сильно отличаться. Окончательно выбирается та кластеризация, которая доставляет наилучшее значение заданному функционалу качества.

Различные виды функционалов качества рассматриваются далее.

Функционалы качества кластеризации

Задачу кластеризации можно ставить как задачу дискретной оптимизации: необходимо так приписать номера кластеров y_i объектам x_i, чтобы значение выбранного функционала качества приняло наилучшее значение. Существует много

Алгоритм 7.2. Алгоритм FOREL

1: Инициализировать множество некластеризованных точек:
$U := X^\ell$;
2: **пока** в выборке есть некластеризованные точки, $U \neq \varnothing$:
3: взять произвольную точку $x_0 \in U$ случайным образом;
4: **повторять**
5: образовать кластер — сферу с центром в x_0 и радиусом R:
$K_0 := \{x_i \in U \mid \rho(x_i, x_0) \leq R\}$;
6: поместить центр сферы в центр масс кластера:
$x_0 := \frac{1}{|K_0|} \sum_{x_i \in K_0} x_i$;
7: **пока** центр x_0 не стабилизируется;
8: пометить все точки K_0 как кластеризованные:
$U := U \setminus K_0$;
9: применить алгоритм КНП к множеству центров всех найденных кластеров;
10: каждый объект $x_i \in X^\ell$ приписать кластеру с ближайшим центром;

разновидностей функционалов качества кластеризации, но нет «самого правильного» функционала. По сути дела, каждый метод кластеризации можно рассматривать как точный или приближённый алгоритм поиска оптимума некоторого функционала.

Среднее внутрикластерное расстояние должно быть как можно меньше:

$$F_0 = \frac{\sum_{i<j} [y_i = y_j]\rho(x_i, x_j)}{\sum_{i<j} [y_i = y_j]} \to \min.$$

Среднее межкластерное расстояние должно быть как можно больше:

$$F_1 = \frac{\sum_{i<j} [y_i \neq y_j]\rho(x_i, x_j)}{\sum_{i<j} [y_i \neq y_j]} \to \max.$$

Если алгоритм кластеризации вычисляет центры кластеров μ_y, $y \in Y$, то можно определить функционалы, вычислительно более эффективные.

Сумма средних внутрикластерных расстояний должна быть как можно меньше:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i:\, y_i = y} \rho^2(x_i, \mu_y) \to \min,$$

где $K_y = \{x_i \in X^\ell \mid y_i = y\}$ — кластер с номером y. В этой формуле можно было бы взять не квадраты расстояний, а сами расстояния. Однако, если ρ — евклидова метрика, то внутренняя сумма в Φ_0 приобретает физический смысл момента инерции кластера K_y относительно его центра масс, если рассматривать кластер как материальное тело, состоящее из $|K_y|$ точек одинаковой массы.

Сумма межкластерных расстояний должна быть как можно больше:

$$\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \to \max,$$

где μ — центр масс всей выборки.

На практике вычисляют отношение пары функционалов, чтобы учесть как межкластерные, так и внутрикластерные расстояния:

$$F_0/F_1 \to \min, \quad \text{либо} \quad \Phi_0/\Phi_1 \to \min.$$

Алгоритм 7.3. Кластеризация с помощью EM-алгоритма

1: начальное приближение для всех кластеров $y \in Y$:
$w_y := 1/|Y|$;
$\mu_y :=$ случайный объект выборки;
$\sigma^2_{yj} := \frac{1}{\ell|Y|} \sum_{i=1}^{\ell} (f_j(x_i) - \mu_{yj})^2$, $j = 1, \ldots, n$;
2: **повторять**
3: E-шаг (expectation):
$g_{iy} := \frac{w_y p_y(x_i)}{\sum_{z \in Y} w_z p_z(x_i)}$, $y \in Y$, $i = 1, \ldots, \ell$;
4: M-шаг (maximization):
$w_y := \frac{1}{\ell} \sum_{i=1}^{\ell} g_{iy}$, $y \in Y$;
$\mu_{yj} := \frac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} f_j(x_i)$, $y \in Y$, $j = 1, \ldots, n$;
$\sigma^2_{yj} := \frac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} (f_j(x_i) - \mu_{yj})^2$, $y \in Y$, $j = 1, \ldots, n$;
5: Отнести объекты к кластерам по байесовскому решающему правилу:
$y_i := \arg\max_{y \in Y} g_{iy}$, $i = 1, \ldots, \ell$;
6: **пока** y_i не перестанут изменяться;

Статистические алгоритмы

Статистические алгоритмы основаны на предположении, что кластеры неплохо описываются некоторым семейством вероятностных распределений. Тогда задача кластеризации сводится к разделению смеси распределений по конечной выборке.

Снова EM-алгоритм. Напомним основные гипотезы байесовского подхода к разделению смесей вероятностных распределений, см. также §2.4.

Гипотеза 7.1 (о вероятностной природе данных). *Объекты выборки X^ℓ появляются случайно и независимо согласно вероятностному распределению, представляющему собой смесь распределений*

$$p(x) = \sum_{y \in Y} w_y p_y(x), \qquad \sum_{y \in Y} w_y = 1,$$

где $p_y(x)$ — функция плотности распределения кластера y, w_y — неизвестная априорная вероятность появления объектов из кластера y.

Конкретизируя вид распределений $p_y(x)$, чаще всего берут сферические гауссовские плотности. Мы будем считать, что кластеры похожи скорее не на шары, а на эллипсоиды, оси которых направлены вдоль осей координат. Преимущество эллиптических гауссианов в том, что они обходят проблему выбора нормировки признаков. Нормировать можно немного по-разному, причём результат кластеризации существенно зависит от нормировки. Пока не произведена кластеризация, трудно понять, какая нормировка лучше. При использовании эллиптических гауссианов оптимальная нормировка подбирается самим алгоритмом кластеризации, индивидуально для каждого кластера.

Гипотеза 7.2 (о пространстве объектов и форме кластеров). *Каждый объект x из $X = \mathbb{R}^n$ описывается n числовыми признаками: $x \equiv f_1(x), \ldots, f_n(x)$. Каждый кластер $y \in Y$ описывается n-мерной гауссовской плотностью $p_y(x)$ с центром $\mu_y = (\mu_{y1}, \ldots, \mu_{yn})$ и диагональной матрицей ковариаций $\Sigma_y = \operatorname{diag}(\sigma_{y1}^2, \ldots, \sigma_{yn}^2)$:*

$$p_y(x) = (2\pi)^{-n/2}(\sigma_{y1} \cdots \sigma_{yn})^{-1} \exp\left(-\frac{1}{2}\rho_y^2(x, \mu_y)\right),$$

где $\rho_y^2(x, x') = \sum_{j=1}^{n} \sigma_{yj}^{-2} |f_j(x) - f_j(x')|^2$ *— взвешенное евклидово расстояние с весами* σ_{yj}^{-2}.

При этих предположениях задача кластеризации совпадает с задачей разделения смеси вероятностных распределений, и для её решения можно применить EM-алгоритм 2.3. Для оценивания параметров кластеров воспользуемся формулами, полученными в Теореме 2.7 как раз для случая эллиптических гауссианов. Реализация этой идеи представлена в Алгоритме 7.3.

Напомним, что EM-алгоритм заключается в итерационном повторении двух шагов. На E-шаге по формуле Байеса вычисляются скрытые переменные g_{iy}. Значение g_{iy} равно апостериорной вероятности того, что объект $x_i \in X^\ell$ принадлежит кластеру $y \in Y$. На M-шаге уточняются параметры каждого кластера (μ_y, Σ_y), при этом существенно используются скрытые переменные g_{iy}.

В Алгоритме 7.3 для простоты предполагается, что число кластеров известно заранее. Однако в большинстве практических случаев его лучше определять автоматически, как это было сделано в Алгоритме 2.3.

Метод *k*-средних, представленный в Алгоритме 7.4, является упрощением EM-алгоритма. Главное отличие в том, что в EM-алгоритме каждый объект x_i распределяется по всем кластерам с вероятностями $g_{iy} = \mathsf{P}\{y_i = y\}$. В алгоритме *k-средних* (*k*-means) каждый объект жёстко приписывается только к одному кластеру.

Второе отличие в том, что в *k*-means форма кластеров не настраивается. Однако это отличие не столь принципиально. Можно предложить упрощённый вариант EM, в котором форма кластеров также не будет настраиваться — для этого достаточно взять сферические гауссианы с ковариационными матрицами $\Sigma_y = \sigma_y I_n$. С другой стороны, возможен и обобщённый вариант *k*-means, в котором будут определяться размеры кластеров вдоль координатных осей. Для этого в Алгоритме 7.3 достаточно убрать шаг 5 и заменить E-шаг жёстким приписыванием объектов кластерам:

$$y_i := \arg\min_{y \in Y} \rho_y(x_i, \mu_y), \quad j = 1, \ldots, n;$$

$$g_{iy} := [y_i = y], \quad j = 1, \ldots, n, \quad y \in Y.$$

Таким образом, EM и *k*-means довольно плавно «перетекают» друг в друга, позволяя строить различные «промежуточные» варианты этих двух алгоритмов.

Заметим, что *k*-means похож также на поиск центра кластера в алгоритме FOREL. Отличие в том, что в FOREL кластер — это шар заданного радиуса R, тогда как в *k*-means объекты относятся к кластерам по принципу ближайшего соседа.

Существует два «канонических» варианта алгоритма *k*-means. Вариант Болла-Холла [18, стр. 110] представлен в Алгоритме 7.4. Вариант МакКина [18, стр. 98] отличается тем, что всякий раз, когда некоторый объект x_i переходит из одного

Алгоритм 7.4. Кластеризация с помощью алгоритма k-средних

1: сформировать начальное приближение центров всех кластеров $y \in Y$:
μ_y — наиболее удалённые друг от друга объекты выборки;
2: **повторять**
3: отнести каждый объект к ближайшему центру (аналог Е-шага):
$y_i := \arg\min_{y \in Y} \rho(x_i, \mu_y)$, $i = 1, \ldots, \ell$;
4: вычислить новое положение центров (аналог М-шага):
$$\mu_{yj} := \frac{\sum_{i=1}^{\ell} [y_i = y] f_j(x_i)}{\sum_{i=1}^{\ell} [y_i = y]}, \quad y \in Y, \quad j = 1, \ldots, n;$$
5: **пока** y_i не перестанут изменяться;

кластера в другой, центры обоих кластеров пересчитываются. Для этого шаг 4 надо перенести внутрь цикла по i, выполняемого на шаге 3. МакКин показал в 1967 году, что этот вариант алгоритма приводит к локальному минимуму функционала Φ_0.

Алгоритм k-means крайне чувствителен к выбору начальных приближений центров. Случайная инициализация центров на шаге 1 может приводить к плохим кластеризациям. Для формирования начального приближения можно выделить k наиболее удалённых точек выборки: первые две точки выделяются по максимуму всех попарных расстояний; каждая следующая точка выбирается так, чтобы расстояние от неё до ближайшей уже выделенной было максимально.

Другая рекомендация — выполнить кластеризацию несколько раз, из различных случайных начальных приближений и выбрать кластеризацию с наилучшим значением заданного функционала качества.

Кластеризация может оказаться неадекватной и в том случае, если число кластеров будет изначально неверно угадано. Стандартная рекомендация — провести кластеризацию при различных значениях k и выбрать то, при котором достигается резкое улучшение качества кластеризации по заданному функционалу.

Кластеризация с частичным обучением. Алгоритмы EM и k-means легко приспособить для решения задач кластеризации *с частичным обучением* (semi-supervised learning), когда для некоторых объектов x_i известны правильные классификации $y^*(x_i)$. Обозначим через U подмножество таких объектов, $U \subset X^\ell$.

Примером такой задачи является рубрикация текстовых документов, в частности, страниц в Интернете. Типична ситуация, когда имеется относительно небольшое множество документов, вручную классифицированных по тематике. Требуется определить тематику большого числа неклассифицированных документов. Сходство документов $\rho(x, x')$ может оцениваться по-разному в зависимости от целей рубрикации и специфики самих документов: по частоте встречаемости ключевых слов, по частоте посещаемости заданным множеством пользователей, по количеству взаимных гипертекстовых ссылок, или другими способами.

Модификация обоих алгоритмов довольно проста: на Е-шаге (шаг 3) для всех $x_i \in U$ полагаем $g_{iy} := [y = y^*(x_i)]$, для всех остальных $x_i \in X^\ell \setminus U$ скрытые переменные g_{iy} вычисляются как прежде. На практике частичная классификация даже небольшого количества объектов существенно улучшает качество кластеризации.

Иерархическая кластеризация

Иерархические алгоритмы кластеризации, называемые также алгоритмами *таксономии*, строят не одно разбиение выборки на непересекающиеся классы, а систему вложенных разбиений. Результат таксономии обычно представляется в виде таксономического дерева — *дендрограммы*. Классическим примером такого дерева является иерархическая классификация животных и растений.

Среди алгоритмов иерархической кластеризации различаются два основных типа. *Дивизимные* или нисходящие алгоритмы разбивают выборку на всё более и более мелкие кластеры. Более распространены *агломеративные* или восходящие алгоритмы, в которых объекты объединяются во всё более и более крупные кластеры. Реализация этой идеи представлена в Алгоритме 7.5.

Сначала каждый объект считается отдельным кластером. Для одноэлементных кластеров естественным образом определяется функция расстояния

$$R(\{x\}, \{x'\}) = \rho(x, x').$$

Затем запускается процесс слияний. На каждой итерации вместо пары самых близких кластеров U и V образуется новый кластер $W = U \cup V$. Расстояние от нового кластера W до любого другого кластера S вычисляется по расстояниям $R(U, V)$, $R(U, S)$ и $R(V, S)$, которые к этому моменту уже должны быть известны:

$$R(U \cup V, S) = \alpha_U R(U, S) + \alpha_V R(V, S) + \beta R(U, V) + \gamma |R(U, S) - R(V, S)|,$$

где α_U, α_U, β, γ — числовые параметры. Эта универсальная формула обобщает практически все разумные способы определить расстояние между кластерами. Она была предложена Лансом и Уильямсом в 1967 году [49, 24].

На практике используются следующие способы вычисления расстояний $R(W, S)$ между кластерами W и S. Для каждого из них доказано соответствие формуле Ланса-Вильямса при определённых сочетаниях параметров [18]:

Расстояние ближнего соседа: $\alpha_U = \alpha_V = \frac{1}{2}$, $\beta = 0$, $\gamma = -\frac{1}{2}$;

$$R^{б}(W, S) = \min_{w \in W, s \in S} \rho(w, s);$$

Расстояние дальнего соседа: $\alpha_U = \alpha_V = \frac{1}{2}$, $\beta = 0$, $\gamma = \frac{1}{2}$;

$$R^{д}(W, S) = \max_{w \in W, s \in S} \rho(w, s);$$

Среднее расстояние: $\alpha_U = \frac{|U|}{|W|}$, $\alpha_V = \frac{|V|}{|W|}$, $\beta = \gamma = 0$;

$$R^{с}(W, S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w, s);$$

Расстояние между центрами: $\alpha_U = \frac{|U|}{|W|}$, $\alpha_V = \frac{|V|}{|W|}$, $\beta = -\alpha_U \alpha_V$, $\gamma = 0$;

$$R^{ц}(W, S) = \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right);$$

Расстояние Уорда: $\alpha_U = \frac{|S|+|U|}{|S|+|W|}$, $\alpha_V = \frac{|S|+|V|}{|S|+|W|}$, $\beta = \frac{-|S|}{|S|+|W|}$, $\gamma = 0$.

$$R^{у}(W, S) = \frac{|S||W|}{|S|+|W|} \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right);$$

Возможных вариантов слишком много, и на первый взгляд все они кажутся достаточно разумными. Возникает вопрос: какой из них предпочесть? Рассмотрим несколько дополнительных свойств, характеризующих качество кластеризации.

Алгоритм 7.5. Агломеративная кластеризация Ланса-Уильямса

1: инициализиро вать множеств о кластеров C_1:
$t := 1; \quad C_t = \{x_1\}, \ldots, \{x_\ell\}$;
2: **для всех** $t = 2, \ldots, \ell$ (t — номер итерации):
3: найти в C_{t-1} два ближайших кластера:
$(U, V) := \arg\min_{U\ V} R(U, V)$;
$R_t := R(U, V)$;
4: изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}$;
5: **для всех** $S \in C_t$
6: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;

Свойство монотонности. Обозначим через R_t расстояние между ближайшими кластерами, выбранными на t-м шаге для слияния. Говорят, что функция расстояния R обладает свойством *монотонности*, если при каждом слиянии расстояние между объединяемыми кластерами только увеличивается: $R_2 \le R_3 \le \ldots \le R_\ell$.

Свойство монотонности позволяет изобразить процесс кластеризации в виде специального графика, называемого *дендрограммой*. По вертикальной оси откладываются объекты, по горизонтальной — расстояния R_t. Нетрудно доказать, что если кластеризация обладает свойством монотонности, то дендрограмму можно построить так, чтобы она не имела самопересечений. При этом любой кластер из множества C_t представляется сплошной последовательностью точек на вертикальной оси. Если же процесс кластеризации идёт не монотонно, то вместо дендрограммы получается запутанный клубок линий, на котором трудно что-либо разобрать.

Дендрограмма позволяет представить кластерную структуру в виде плоского графика независимо от того, какова размерность исходного пространства. Существуют и другие способы двумерной визуализации многомерных данных, такие как многомерное шкалирование или карты Кохонена, но они привносят в картину искусственные искажения, влияние которых довольно трудно оценить.

Оказывается, не любое сочетание коэффициентов в формуле Ланса-Вильямса приводит к монотонной кластеризации.

Теорема 7.1 (Миллиган, 1979). *Если выполняются следующие три условия, то кластеризация является монотонной:*

1) $\alpha_U \ge 0, \alpha_V \ge 0$;
2) $\alpha_U + \alpha_V + \beta \ge 1$;
3) $\min\{\alpha_U, \alpha_V\} + \gamma \ge 0$.

Из перечисленных выше расстояний только $R^{\text{ц}}$ не является монотонным. Расстояние Уорда отличается от него мультипликативной поправкой, которая и делает его монотонным.

Свойства растяжения и сжатия. Некоторые расстояния обладают *свойством растяжения*. По мере того, как кластер укрупняется, расстояния от него до других кластеров увеличиваются, как будто пространство вокруг кластера растягивается.

На дендрограмме растягивающие расстояния характеризуются повышением плотности линий слева, в области наименьших значений R_t. Свойство растяжения считается желательным, так как оно способствует более чёткому отделению кластеров. С другой стороны, при слишком сильном растяжении возможно найти кластеры там, где их изначально не было. Растягивающими являются расстояния $R^{д}$ и $R^{у}$.

Некоторые расстояния, наоборот, обладают *свойством сжатия*. По мере роста кластера расстояния от него до других кластеров уменьшается, и кажется, что пространство вокруг кластера сжимается. Естественная кластеризация при этом может исчезнуть. Для сжимающих расстояний характерно уплотнение дендрограммы справа, в области наибольших значений R_t. Расстояние ближнего соседа $R^{б}$ является сильно сжимающим.

Свойства сжатия и растяжения определяются через отношение $R_t/\rho(\mu_U, \mu_V)$, где $R_t = R(U, V)$ — расстояние между ближайшими кластерами, объединяемыми на t-м шаге, μ_U и μ_V — центры этих кластеров. Если это отношение на каждом шаге больше единицы, то расстояние R является растягивающим; если оно всегда меньше единицы, то сжимающим. Есть и такие расстояния, которые не являются ни сжимающими, ни растягивающими, например, $R^{с}$ и $R^{ц}$. О них говорят, что они *сохраняют метрику пространства*.

На практике часто применяют *гибкое расстояние*, которое представляет собой компромисс между методами ближнего соседа, дальнего соседа и среднего расстояния. Оно определяется одним параметром β вместо четырёх:

$$\alpha_U = \alpha_V = (1 - \beta)/2, \quad \gamma = 0, \quad \beta < 1.$$

Гибкое расстояние является сжимающим при $\beta > 0$ и растягивающим при $\beta < 0$. Стандартная рекомендация: $\beta = -0{,}25$ [24].

Свойство редуктивности. Самой трудоёмкой операцией в Алгоритме 7.5 является поиск пары ближайших кластеров на шаге 3. Он требует $O(\ell^2)$ операций внутри основного цикла. Соответственно, построение всего таксономического дерева требует $O(\ell^3)$ операций. Это ограничивает применимость алгоритма выборками длины в несколько сотен объектов.

Идея ускорения алгоритма заключается в том, чтобы перебирать лишь наиболее близкие пары. Задаётся параметр δ, и перебор ограничивается сокращённым множеством пар $\{(U, V)\colon R(U, V) \le \delta\}$. Когда все такие пары будут исчерпаны, параметр δ увеличивается, и формируется новое сокращённое множество пар. И так далее, до полного слияния всех объектов в один кластер. Реализация представлена в Алгоритме 7.6.

Доказано, что этот алгоритм приводит к той же кластеризации, что и Алгоритм 7.5, если расстояние R обладает свойством редуктивности:

Опр. 7.1 (Брюинош, 1978). *Расстояние R называется редуктивным, если для любого $\delta > 0$ и любых δ-близких кластеров U и V объединение δ-окрестностей U и V содержит в себе δ-окрестность кластера $W = U \cup V$:*

$$\{S \cdot R(U \cup V, S) < \delta,\ R(U, V) \le \delta\} \subseteq \{S \cdot R(S, U) < \delta \text{ или } R(S, V) < \delta\}.$$

Алгоритм 7.6. Быстрая агломеративная кластеризация на основе редуктивности

1: инициализиро вать множеств о кластеров C_1:
$t := 1; \quad C_t = \{x_1\}, \ldots, \{x_\ell\}$;
2: выбрать начальное значение параметра δ;
3: $P(\delta) := (U, V) \cdot U, V \in C_t,\ R(U, V) \le \delta$;
4: **для всех** $t = 2, \ldots, \ell$ (t — номер итерации):
5: **если** $P(\delta) = \emptyset$ **то**
6: увеличить δ так, чтобы $P(\delta) \quad \emptyset$;
7: найти в $P(\delta)$ пару ближайших кластеров:
$(U, V) := \arg\min_{(U,V)\in P(\delta)} R(U, V)$;
$R_t := R(U, V)$;
8: изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}$;
9: **для всех** $S \in C_t$
10: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;
11: **если** $R(W, S) \le \delta$ **то**
12: $P(\delta) := P(\delta) \cup (W, S)$;

Теорема 7.2 (Диде и Моро, 1984). *Если выполняются следующие три условия, то расстояние R является редуктивным:*

1) $\alpha_U \ge 0,\ \alpha_V \ge 0$;
2) $\alpha_U + \alpha_V + \min\{\beta, 0\} \ge 1$;
3) $\min\{\alpha_U, \alpha_V\} + \gamma \ge 0$.

Сравнение условий теорем 7.2 и 7.1, показывает, что всякое редуктивное расстояние является монотонным, следовательно, позволяет отобразить процесс кластеризации в виде дендрограммы. Из перечисленных выше расстояний только $R^{ц}$ не является редуктивным.

В Алгоритме 7.6 возможны различные эвристические стратегии выбора параметра δ на шагах 2 и 6. Общие соображения таковы: если δ настолько велико, что $P(\delta)$ содержит практически все пары кластеров, то мы фактически возвращаемся к неэффективному Алгоритму 7.5; если же δ мало́, то приходится слишком часто формировать множество $P(\delta)$. На практике поступают следующим образом. Если число кластеров в C_t не превышает порог n_1, то в качестве $P(\delta)$ берут множество всех возможных пар (U, V) из C_t. В противном случае выбирается случайным образом n_2 расстояний $R(U, V)$, и δ полагается равным наименьшему из них. В случае редуктивности параметры алгоритма n_1 и n_2 влияют только на время выполнения алгоритма, но не на результат кластеризации. Оптимальные значения для них подбираются с помощью калибровочных тестов и, вообще говоря, зависят от компьютера. В качестве начального выбора можно предложить $n_1 = n_2 = 20$.

Определение числа кластеров проще всего производить путём отсечения правого участка дендрограммы. На горизонтальной оси находится интервал максимальной длины $|R_{t+1} - R_t|$, и в качестве результирующей кластеризации выдаётся множество

кластеров C_t. Число кластеров равно $K = \ell - t + 1$. При необходимости можно задать ограничение на минимальное и максимальное число кластеров $K_0 \leq K \leq K_1$ и выбирать t, удовлетворяющие ограничениям $\ell - K_1 + 1 \leq t \leq \ell - K_0 + 1$.

Во многих прикладных задачах интерес представляет таксономическое дерево целиком, и определять оптимальное число кластеров не имеет особого смысла.

Достоинства и недостатки агломеративной кластеризации. Точного ответа на вопрос, какой алгоритм кластеризации лучше, не существует. Каждое из расстояний, перечисленных выше, имеет свои недостатки и подходит не для всех задач.

Метод ближнего соседа обладает *цепочечным эффектом*, когда независимо от формы кластера к нему присоединяются ближайшие к границе объекты. В некоторых случаях это приводит к тому, что кластеры «отращивают щупальцаk. В зависимости от задачи это свойство может быть как полезным, так и мешающим. Метод ближнего соседа хорошо подходит для выделения кластеров ленточной формы.

Метод дальнего соседа цепочечного эффекта не имеет, но на раннем этапе может объединять довольно несхожие группы.

Расстояние между центрами масс не монотонно и не редуктивно, поэтому редко используется на практике, хотя интуитивно кажется «золотой серединойk.

Метод Уорда оказался наилучшим по результатам экспериментального сравнения на представительном наборе модельных задач [18]. Он чаще других методов строит интуитивно лучшую таксономию.

§7.2 Сети Кохонена

До сих пор мы рассматривали нейронные сети, предназначенные для обучения с учителем, когда для каждого объекта x_i задан соответствующий ему ответ y_i. Сети Кохонена решают задачи обучения без учителя, когда задаются только сами объекты x_i, и требуется выделить обособленные «плотные сгусткиk объектов — кластеры, и научиться относить новые объекты к этим кластерам. Кластеризация основывается на предположении, что в пространстве X введена метрика $\rho : X \times X \to \mathrm{R}$, адекватно оценивающая степень сходства любой пары объектов.

Модели конкурентного обучения

Пусть $X = \mathrm{R}^n$ — пространство объектов, $Y = \{1, \ldots, M\}$ — множество кластеров, число M фиксировано. Задана обучающая выборка объектов $X^\ell = \{x_i\}_{i=1}^{\ell}$. Требуется выработать правило отнесения объектов к кластерам $a : X \to Y$.

Правило жёсткой конкуренции WTA. Наиболее очевидный подход заключается в том, чтобы ввести векторы $w_m \in \mathrm{R}^n$, $m = 1, \ldots, M$, описывающие центры кластеров, и относить произвольный объект $x \in X$ к ближайшему кластеру:

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m). \tag{7.1}$$

Образно говоря, кластеры соревнуются за право присоединить к себе объект x. Кластер, ближайший к x, называется кластером-победителем, а выражение (7.1) — *правилом WTA* (winner takes all).

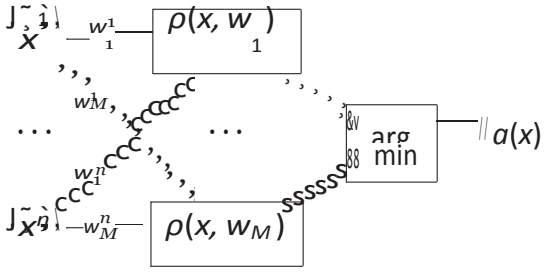


Рис. 22. Представление алгоритма кластеризации (7.1) в виде двухслойной нейронной сети.

Настройка алгоритма кластеризации $a(x)$ сводится к оптимизации расположения центров w_m. Для этого минимизируется функционал качества кластеризации, равный среднему квадрату расстояния между объектами и центрами кластеров:

$$Q(w_1, \ldots, w_M) = \frac{1}{2} \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)}) \to \min_{\{w_m\}}.$$

Допустим, что метрика евклидова: $\rho(x, w) = \|x - w\|$. Тогда можно выписать градиент функционала Q по вектору w_m:

$$\frac{\partial Q}{\partial w_m} = \sum_{i=1}^{\ell} (w_m - x_i)\,[a(x_i) = m]\,.$$

Для поиска центров кластеров w_m можно применить метод стохастического градиента — Алгоритм 4.1, практически без модификаций. Единственное отличие заключается в том, что теперь правило обновления весов на шаге 6 будет иметь вид

$$w_m := w_m + \eta(x_i - w_m)\,[a(x_i) = m]\,, \tag{7.2}$$

где $x_i \in X^\ell$ — случайным образом выбранный обучающий объект, η — градиентный шаг, он же *темп обучения*. Смысл данного правила очевиден: если объект x_i относится к кластеру m, то центр этого кластера w_m немного сдвигается в направлении объекта x_i, остальные центры не изменяются.

Формальное сходство формулы (7.2) с персептронным правилом наводит на мысль, что кластеризация осуществляется алгоритмом, аналогичным нейронной сети. Выражение (7.1) и в самом деле представимо в виде двухслойной суперпозиции, только вместо привычных скалярных произведений $\langle x, w_m \rangle$ вычисляются функции расстояния $\rho(x, w_m)$, а на выходе вместо суммирования применяется операция минимума, см. Рис. 22. При этом центры кластеров w_m взаимно однозначно соответствуют нейронам скрытого стоя, которые называются *нейронами Кохонена*. Нейрон, выход которого минимален, называется *нейроном-победителем*.

Рассмотренная разновидность нейронных сетей называется *сетью с конкурентным обучением*. Исходно она была предложена как чисто математическая модель. Позже нейрофизиологам удалось найти некоторые её аналоги в биологических нейронных сетях [40].

Альтернативное название — *обучающееся векторное квантование* (learning vector quantization, LVQ) — связано с тем, что по исходной выборке из ℓ объектов x_i строятся M новых объектов w_m, похожих на исходные, — это центры ячеек, по которым распределяются («квантуются») исходные объекты. Как правило, $M \ll \ell$, поэтому замена объектов на ближайшие к ним центры позволяет эффективно сжимать данные при незначительной потере информации. Объём сохраняемой информации регулируется единственным параметром M, что достаточно удобно в приложениях.

Правило справедливой конкуренции CWTA. Недостаток конкурентного обучения по правилу WTA заключается в том, что при случайной инициализации весов нейрон Кохонена может попасть в такую область, где он никогда не станет победителем. В результате появится неинформативный пустой кластер.

Для преодоления этого недостатка алгоритм (7.1) немного модифицируется. Вводится «механизм утомленияk победителей — *правило CWTA* (conscience WTA):

$$a(x) = \arg\min_{m \in Y} C_m \rho(x, w_m), \tag{7.3}$$

где C_m — количество побед m-го нейрона в ходе обучения. Таким образом, кластеры штрафуются за слишком частое присоединение объектов.

Правило мягкой конкуренции WTM. Другим недостатком правила WTA является медленная скорость сходимости, связанная с тем, что на каждой итерации модифицируется только один нейрон-победитель w_m. Для ускорения сходимости, особенно на начальных итерациях, можно подстраивать сразу несколько нейронов, близких к объекту x_i. Для этого вводится ядро — неотрицательная монотонно убывающая на $[0, +\infty)$ функция расстояния $K(\rho)$. Иногда накладывается дополнительное требование нормировки $K(0) = 1$. Часто берут гауссовское ядро $K(\rho) = \exp(-\beta\rho^2)$ при некотором $\beta > 0$. Вместо правила жёсткой конкуренции WTA вводится мягкая конкуренция — *правило WTM* (winner takes most):

$$w_m := w_m + \eta(x_i - w_m)\, K(\rho(x_i, w_m)), \qquad m = 1, \ldots, M. \tag{7.4}$$

Теперь на каждой итерации центры *всех* кластеров смещаются в сторону x_i, но чем дальше центр находится от x_i, тем меньше величина смещения. Заметим, что (7.2) является частным случаем (7.4), если положить $K(\rho(x_i, w_m)) = [a(x_i) = m]$.

На начальных итерациях имеет смысл выбрать небольшое значение коэффициента β, чтобы все весовые векторы успели переместиться ближе к области входных векторов. Затем β можно увеличивать, делая конкуренцию всё более жёсткой, и постепенно переходя к коррекции только одного нейрона-победителя.

Благодаря способности к сглаживанию, правило WTM имеет многочисленные применения. Одно из важнейших — самоорганизующиеся карты Кохонена.

Самоорганизующиеся карты Кохонена

Самоорганизующиеся карты Кохонена (self-organizing maps, SOM) применяются для визуализации многомерных данных. Они дают лишь общую картину, довольно размытую и подверженную искажениям, поскольку спроецировать многомерную выборку на плоскость без искажений в общем случае невозможно. Тем не менее, карты Кохонена позволяют увидеть ключевые особенности кластерной структуры выборки. Они используются на стадии *разведочного анализа данных* , скорее для общего понимания задачи, чем для получения каких-либо точных результатов.

Идея заключается в том, чтобы спроецировать все объекты выборки на плоскую карту, точнее, на множество узлов прямоугольной сетки заранее заданного размера $M \times H$. На практике M и H имеют порядок десятков или сотен. Каждому узлу сетки приписывается нейрон Кохонена с вектором весов $w_{mh} \in \mathbb{R}^n$, $m = 1, \ldots, M$,

Алгоритм 7.7. Обучение карты Кохонена методом стохастического градиента

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
Выход:
Векторы синаптических весов w_{mh}, $m = 1, \ldots, M$, $h = 1, \ldots, H$;

1: инициализировать веса:
$w_{mh} := \text{random}\left(-\frac{1}{2MH}, \frac{1}{2MH}\right)$;
2: **повторять**
3: выбрать объект x_i из X^ℓ случайным образом;
4: WTA: вычислить координаты узла, в который проецируется объект x_i:
$(m_i, h_i) := a(x_i) \equiv \arg\min_{(m,h)\in Y} \rho(x_i, w_{mh})$;
5: **для всех** $(m, h) \in Y$, достаточно близких к (m_i, h_i)
6: WTM: сделать шаг градиентного спуска:
$w_{mh} := w_{mh} + \eta(x_i - w_{mh})\, K\, r((m_i, h_i), (m, h))$;
7: **пока** размещение всех объектов в узлах сетки не стабилизируется;

$h = 1, \ldots, H$. Таким образом, множество Y совпадает с множеством узлов сетки, $Y = \{1, \ldots, M\} \times \{1, \ldots, H\}$.

Алгоритм кластеризации $a(x)$ выдаёт пару индексов $(m, h) \in Y$, показывающих, в какой узел сетки проецируется объект x. Чтобы карта отражала кластерную структуру выборки, близкие объекты должны попадать в близкие узлы сетки.

Обучение нейронов производится методом стохастического градиента, см. Алгоритм 7.7. После случайного выбора объекта x_i на шаге 3 определяется нейрон-победитель согласно правилу WTA. Соответствующий ему узел сетки обозначается в алгоритме через (m_i, h_i). Затем, согласно правилу WTM, этот нейрон и нейроны, расположенные в ближайших узлах сетки, сдвигаются в сторону вектора x_i. Правило обучения, применяемое на шаге 6, схоже с (7.4), только вместо метрики $\rho(x, x')$, определённой на пространстве объектов, используется евклидова метрика на множестве узлов сетки Y:

$$r\left((m_i, h_i), (m, h)\right) = \sqrt{(m - m_i)^2 + (h - h_i)^2}.$$

По прежнему, $K(\rho)$ — ядро сглаживания, например, $K(\rho) = \exp(-\beta\rho^2)$. Параметр β задаёт степень сглаженности карты: чем меньше β, тем мягче конкуренция нейронов, и тем более сглаженными будут выглядеть границы кластеров на карте. Имеет смысл увеличивать значение параметра β от итерации к итерации, чтобы сеть Кохонена сначала обучилась кластерной структуре «в общих чертаxk, а затем сосредоточилась на деталях.

Алгоритм останавливается, когда проекции всех, или хотя бы большинства, объектов выборки $(m_i, h_i) = a(x_i)$ перестанут меняться от итерации к итерации.

Искусство интерпретации карт Кохонена. Если через настроенную карту Кохонена (алгоритм $a(x)$) пропустить все объекты обучающей выборки X^ℓ, то кластеры исходного пространства отобразятся в сгустки точек на карте. При этом векторы

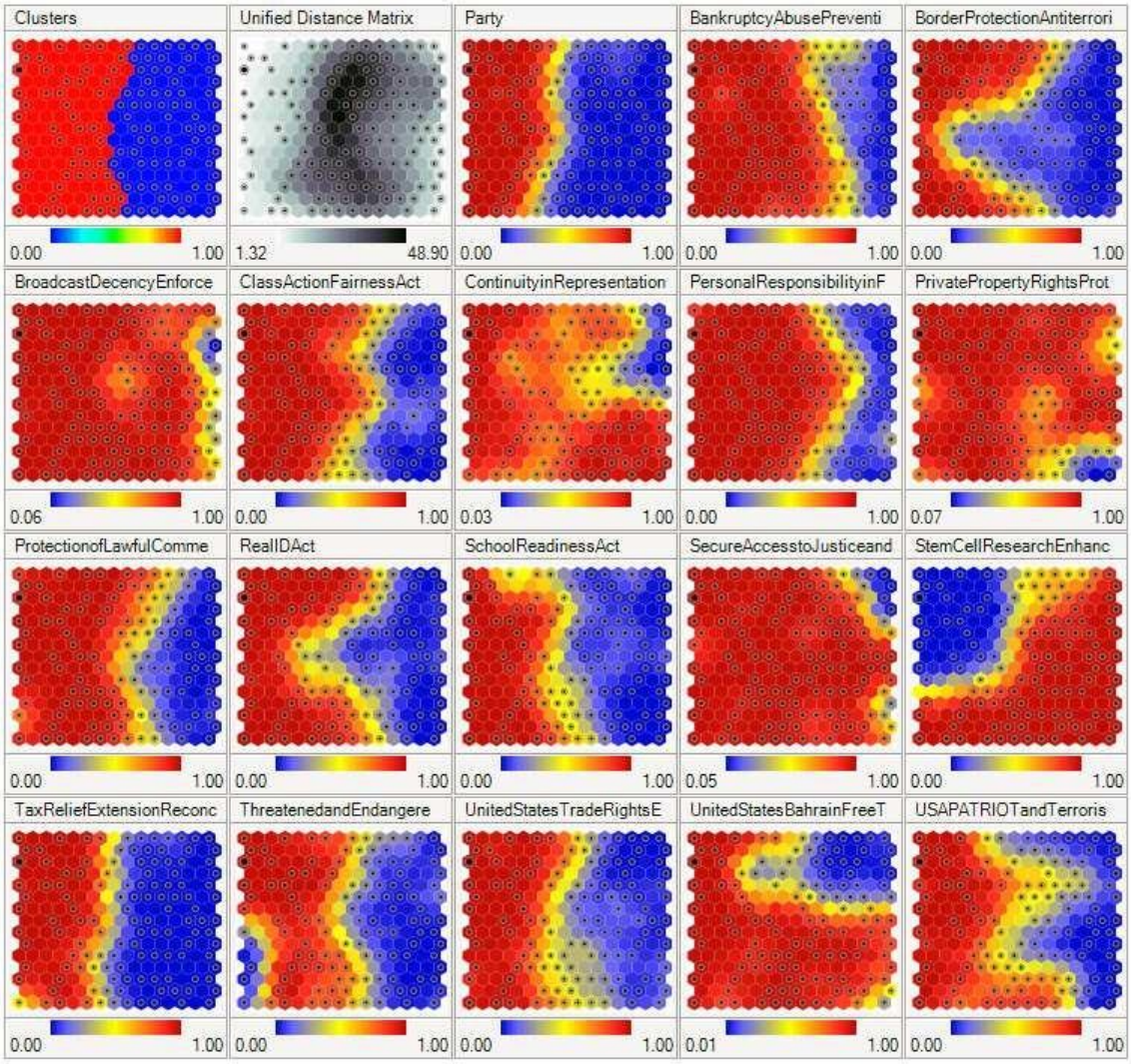


Рис. 23. Карты Кохонена для задачи UCI.house-votes (объекты — конгрессмены, признаки — результаты голосования по 17 вопросам). Первые три графика в верхнем ряду: (1) разделение выборки на два кластера, (2) двухцветная полутоновая карта, (3) цветная карта по признаку «партияс {демократ, республиканец}. Голосование по большинству вопросов хорошо согласуется с партийной принадлежностью конгрессменов.

весов в узлах-сгустках должны быть одновременно похожи на все объекты соответствующих кластеров.

Обычно для каждой точки карты вычисляют среднее расстояние до k ближайших точек выборки, и полученное распределение средних расстояний отображают в виде двухцветной полутоновой карты. Чем меньше среднее расстояние, тем выше локальная плотность точек на карте. На той же карте отмечают и сами точки обучающей выборки. На карте хорошо видно, сколько кластеров выделено, в каких местах они расположились, и какие объекты выборки в них попадают, Рис. 23.

Следующий шаг — поиск интерпретации кластеров. Для этого строятся ещё n карт, по одной карте на каждый признак. Теперь цвет узла (m, h) соответствует значению j-й компоненты вектора весов $w_{m,h}$. Обычно эти карты раскрашивают в геодезические цвета от синего до коричневого. Синие участки карты соответствуют наименьшим значениям j-го признака, красные — наибольшим. Сравнивая карты

признаков с общей картой кластеров, можно найти кластеры, которым свойственны повышенные или пониженные значения отдельных признаков.

В результате содержательная интерпретация кластеров формулируется путём перечисления всех признаков, имеющих либо повышенные, либо пониженные значения в каждом из кластеров.

Ещё один способ интерпретации — выделение на карте всех узлов, в которые попадает выбранное подмножество объектов. Если объекты сгруппировались в одном месте, значит мы имеем дело с кластером или его частью. Выделяя различные подмножества объектов, имеющие заведомо известную содержательную интерпретацию, можно находить интерпретации различных зон на карте.

Недостатки карт Кохонена.

- Субъективность. Не всегда ясно, какие особенности карты обусловлены кластерной структурой данных, а какие — свойствами сглаживающего ядра. От выбора параметра β существенно зависит сглаженность границ кластеров и степень детализации карты.

- Наличие искажений. Близкие объекты исходного пространства могут переходить в далёкие точки на карте. В частности, объективно существующие кластеры могут разрываться на фрагменты [44]. И наоборот, далёкие объекты могут случайно оказаться на карте рядом, особенно, если они были одинаково далеки от всех кластеров. Искажения неизбежны при проецировании многомерной выборки на плоскость. Распределение точек на карте позволяет судить лишь о локальной структуре многомерных данных, и то не всегда.

- Зависимость от инициализации. Начальное распределение весов существенно влияет на процесс обучения и может сказываться не только на расположении кластеров, но даже на их количестве. Иногда рекомендуется построить несколько карт и выбрать из них наиболее «удачнуюk.

Не секрет, что популярность карт Кохонена обусловлена в значительной степени их эстетической привлекательностью и впечатлением наукообразной загадочности, которое они производят на неискушённого пользователя. На практике они применяются в основном как вспомогательное средство для предварительного визуального анализа и выявления неочевидных закономерностей в многомерных данных.

Гибридные сети встречного распространения

Ещё одно важное применение нейронов Кохонена с их способностью кластеризовать исходные векторы — кусочная аппроксимация функций.

Рассмотрим задачу восстановления регрессии, когда на элементах обучающей выборки $X^\ell = \{x_i\}_{i=1}^\ell$ заданы вещественные ответы $y_i = y^*(x_i)$. Идея заключается в том, чтобы сначала разбить обучающие объекты (и всё пространство X) на кластеры, не пользуясь ответами y_i, затем для каждого кластера построить свою локальную аппроксимацию целевой зависимости y^*. При использовании жёсткой конкуренции WTA эти аппроксимации образуют кусочно-непрерывную функцию. Мягкая конкуренция WTM «склеиваетk из локальных кусков гладкую аппроксимацию.

Кусочно-постоянная аппроксимация. Чтобы алгоритм кластеризации (7.1) превратить в алгоритм кусочной аппроксимации, к нему добавляется ещё один нейрон с линейной функцией активации, образующий третий слой с весами $v_1, \ldots, v_M$:

$$a(x) = v_{m^*(x)} = \sum_{m=1}^{M} v_m \, [m^*(x) = m] \; ; \tag{7.5}$$

$$m^*(x) = \arg\min_{m=1,\ldots,M} \rho(x, w_m);$$

где $m^*(x)$ — номер нейрона-победителя на объекте x, вычисляемый по правилу WTA.

При квадратичной функции потерь задача настройки весов v_m легко решается аналитически:

$$Q(v) = \frac{1}{2} \sum_{i=1}^{\ell} \left(a(x_i) - y_i \right)^2 \to \min_v;$$

$$\frac{\partial Q}{\partial v_m} = \sum_{i=1}^{\ell} \left(a(x_i) - y_i \right) [m^*(x_i) = m] = 0.$$

Подставляя сюда выражение для $a(x_i)$ из (7.5), получаем

$$v_m = \frac{\sum_{i=1}^{\ell} y^i \, [m^*(x_i) = m]}{\sum_{i=1}^{\ell} [m^*(x_i) = m]}.$$

Проще говоря, оптимальное значение весового коэффициента v_m есть среднее y_i по всем объектам x_i, попавшим в m-й кластер. Ответ алгоритма $a(x)$ для всех объектов, попадающих в m-й кластер, будет одинаков и равен v_m. Это означает, что $a(x)$ реализует кусочно-постоянную функцию.

Гладкая аппроксимация строится с помощью правила мягкой конкуренции WTM при выбранном гладком ядре $K(\rho)$. Алгоритм $a(x)$ представляет собой формулу Надарая-Ватсона для сглаживания ответов v_m по M точкам — центрам кластеров:

$$a(x) = \sum_{m=1}^{M} v_m \frac{K \, \rho(x, w_m)}{\sum_{s=1}^{M} K \, \rho(x, w_s)}.$$

В этом случае задача минимизации $Q(v)$ сводится к решению системы линейных уравнений размера $M \times M$. Вместо явного решения системы можно снова воспользоваться методом стохастического градиента. На каждой итерации обновляются и веса слоя Кохонена w_m, и веса третьего (выходного) слоя v_m:

-
- $w_m := w_m - \eta (w_m - x_i) K \quad \rho(x_i, w_m) \; ;$
- $v_m := v_m \quad - \eta (a(x_i) - y_i) \dfrac{K \, \rho(x_i, w_m)}{\sum_{s=1}^{M} K \, \rho(x_i, w_s)};$

Заметим, что обновление весов w_m влияет на веса v_m, а обратного влияния нет.

Данный метод получил название *встречного распространения ошибки*, поскольку второй слой настраивается по правилу Кохонена, а третий слой — так же, как в методе back-propagation.

§7.3 Многомерное шкалирование

Визуализация кластерной структуры заданной выборки объектов X^ℓ — непростая проблема, особенно если выборка содержит тысячи объектов, а пространство объектов существенно многомерно.

Задача *многомерного шкалирования* (multidimensional scaling, MDS) заключается в следующем. Имеется обучающая выборка объектов $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$. Заданы расстояния $R_{ij} = \rho(x_i, x_j)$ для некоторых пар обучающих объектов $(i, j) \in D$. Требуется для каждого объекта $x_i \in X^\ell$ построить его признаковое описание — вектор $x_i = (x_i^1, \ldots, x_i^n)$ в евклидовом пространстве R^n так, чтобы евклидовы расстояния d_{ij} между объектами x_i и x_j

$$d_{ij}^2 = \sum_{d=1}^{n} (x_i^d - x_j^d)^2$$

как можно точнее приближали исходные расстояния R_{ij} для всех $(i, j) \in D$. Данное требование можно формализовать по-разному; один из наиболее распространённых способов — через минимизацию функционала, называемого *стрессом*:

$$S(X^\ell) = \sum_{(i,j)\in D} w_{ij}(d_{ij} - R_{ij})^2 \to \min,$$

где минимум берётся по совокупности ℓn переменных $(x_i^d)_{i=1,\ell}^{d=1,n}$.

Размерность n обычно задаётся небольшая. В частности, при $n = 2$ многомерное шкалирование позволяет отобразить выборку в виде множества точек на плоскости (scatter plot). Плоское представление, как правило, искажено ($S > 0$), но в целом отражает основные структурные особенности многомерной выборки, в частности, её кластерную структуру. Поэтому двумерное шкалирование часто используют как инструмент предварительного анализа и понимания данных.

На практике расстояния чаще бывают известны для всех пар объектов, но мы будем рассматривать более общий случай, когда D — произвольное заданное множество пар индексов объектов.

Веса w_{ij} задаются исходя из целей шкалирования. Обычно берут $w_{ij} = (R_{ij})^\gamma$. При $\gamma < 0$ приоритет отдаётся более точному приближению малых расстояний; при $\gamma > 0$ — больших расстояний. Наиболее адекватным считается значение $\gamma = -2$, когда функционал стресса приобретает физический смысл потенциальной энергии системы ℓ материальных точек, связанных упругими связями; требуется найти равновесное состояние системы, в котором потенциальная энергия минимальна.

Функционал стресса $S(X^\ell)$ сложным образом зависит от ℓn переменных, имеет огромное количество локальных минимумов, и его вычисление довольно трудоёмко. Поэтому многие алгоритмы многомерного шкалирования основаны на итерационном размещении объектов по одному. На каждой итерации оптимизируются евклидовы координаты только одного из объектов при фиксированных остальных объектах.

Размещение одного объекта методом Ньютона-Рафсона. Рассмотрим аддитивную составляющую функционала стресса, зависящую только от одного объекта $x \in X^\ell$ с координатами $x \equiv (x^1, \ldots, x^n)$, которые и требуется найти:

$$S(x) = \sum_{x_i \in U} w_i \left(d_i(x) - R_i\right)^2 \to \min_x,$$

где $U \subseteq X^\ell$ — все объекты с известными расстояниями $R_i = \rho(x, x_i)$; $d_i(x)$ — евклидово расстояние между векторами $x \equiv (x^1, \ldots, x^n)$ и $x_i \equiv (x^1_i, \ldots, x^n_i)$.

Применим метод Ньютона–Рафсона для минимизации $S(x)$.

Поскольку функционал многоэкстремальный, очень важно выбрать удачное начальное приближение $x^{(0)}$. Вектор x должен быть похож на те векторы $x_i \in U$, для которых расстояния R_i малы. Неплохая эвристика — взять в качестве $x^{(0)}$ взвешенное среднее всех векторов из U, например, с весами $c_i = R_i^{-2}$:

$$x^{(0)} = \frac{\sum_{x_i \in U} c_i x_i}{\sum_{x_i \in U} c_i}.$$

Затем запускается итерационный процесс

$$x^{(t+1)} := x^{(t)} - h_t S''(x^{(t)})^{-1} S'(x^{(t)}),$$

где $S'(x^{(t)})$ — вектор первых производных (градиент) функционала $S(x)$ в точке $x^{(t)}$, $S''(x^{(t)})$ — матрица вторых производных (гессиан) функционала $S(x)$ в точке $x^{(t)}$, h_t — величина шага, который можно положить равным 1, но более тщательный его подбор способен увеличить скорость сходимости.

Найдём выражения для градиента и гессиана.

$$\frac{\partial d_i}{\partial x^a} = \frac{x^a - x_i^a}{d_i};$$

$$\frac{\partial S}{\partial x^a} = 2 \sum_{x_i \in U} w_i \left(1 - \frac{R_i}{d_i}\right)(x^a - x_i^a);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^a} = 2 \sum_{x_i \in U} w_i \left(\frac{R_i}{d_i} \frac{(x^a - x_i^a)^2}{d_i^2} - \frac{R_i}{d_i} + 1\right);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^b} = 2 \sum_{x_i \in U} w_i \frac{R_i}{d_i} \frac{x^a - x_i^a}{d_i} \frac{x^b - x_i^b}{d_i}.$$

Заметим, что в пространствах малой размерности $n \le 3$ обращение гессиана — довольно простая операция, однако с ростом размерности вычислительные затраты растут как $O(n^3)$.

Итерации Ньютона–Рафсона прекращаются, когда значение стресса $S(x)$ стабилизируется или вектор x перестаёт существенно изменяться, то есть стабилизируется норма разности $\|x^{(t+1)} - x^{(t)}\|$.

Будем полагать, что размещение объекта x относительно множества уже размещённых объектов $U \subseteq X^\ell$ реализуется процедурой НьютонРафсон(x, U).

Субквадратичный алгоритм многомерного шкалирования. Алгоритм 7.8 начинает с того, что находит две самые удалённые точки выборки x_i, x_j. Достаточно решить эту задачу приближённо. В частности, можно взять произвольную точку, найти самую удалённую от неё, затем для этой точки найти самую удалённую, и так далее. Обычно 3–4 итераций хватает, чтобы найти пару достаточно далёких точек. Найденным точкам приписываются (в двумерном случае) евклидовы координаты $(0, 0)$ и $(0, R_{ij})$. Затем находится третья точка x_k, наиболее удалённая от первых

Алгоритм 7.8. Субквадратичный алгоритм многомерного шкалирования

Вход:

R_{ij} — матрица попарных расстояний между объектами, возможно, разреженная;

K — размер скелета;

Выход:

евклидовы координаты всех объектов выборки $x_i \equiv (x^1, \ldots, x^n)$, $i = 1, \ldots, \ell$;

1: Инициализировать скелет:
   U := три достаточно далёкие друг от друга точки;
2: **пока** $|U| < K$ — наращивать скелет:
3: $x := \arg\max_{x_i \in X^\ell \setminus U} \min_{x_j \in U} R_{ij}$;
4: НьютонРафсон(x, U);
5: $U := U \cup \{x\}$;
6: **пока** координаты точек скелета не стабилизируются:
7: найти наиболее напряжённую точку в скелете:
   $x := \arg\max_{x_i \in U} S(x)$;
8: НьютонРафсон$(x, U \setminus \{x\})$;
9: **для** $x \in X^\ell \setminus U$
10: НьютонРафсон(x, U);

двух, то есть для которой значение $\min\{R_{ki}, R_{kj}\}$ максимально. Евклидовы координаты x_k определяются (в двумерном случае) исходя из того, что треугольник ijk жёстко задан длинами своих сторон.

Затем начинается поочерёдное добавление точек и их размещение относительно уже имеющихся. После размещения первых K точек их положение уточняется друг относительно друга. Эти точки, размещённые с особой тщательностью, становятся «скелетом», относительно которого размещаются все остальные точки. Расстояния между «не-скелетными» точками в алгоритме вообще не задействуются.

Если $K = \ell$, то все точки будут «скелетными»; в этом случае размещение является наиболее точным, но и наиболее долгим, требуя $O(\ell^2)$ операций. В общем случае число операций алгоритма $O(K^2) + O(K\ell)$. Выбирая размер «скелета» K, можно находить компромисс между точностью и временем построения решения.

Карта сходства отображает результат многомерного шкалирования при $n = 2$ в виде плоского точечного графика. Евклидова метрика и функционал стресса инвариантны относительно произвольных ортогональных преобразований карты сходства — сдвигов, поворотов и зеркальных отражений. Поэтому оси на карте не имеют интерпретации. Для понимания карты на ней обозначают *ориентиры* — те объекты выборки, интерпретации которых хорошо известны. Если искажения расстояний на карте не велики, то объекты, близкие к ориентирам, должны иметь схожие интерпретации.

Карта сходства даёт лишь общее представление о взаимном расположении объектов выборки. Делать на её основе какие-либо количественные выводы, как правило, нельзя.

Диаграмма Шепарда позволяет сказать, насколько сильно искажены расстояния на карте сходства. Это точечный график; по горизонтальной оси откладываются исходные расстояния R_{ij}; по вертикальной оси откладываются евклидовы расстояния d_{ij}; каждая точка на графике соответствует некоторой паре объектов $(i, j) \in D$. Если число пар превышает несколько тысяч, отображается случайное подмножество пар. Иногда на диаграмме изображается сглаженная зависимость $d_{ij}(R_{ij})$, а также сглаженные границы верхних и нижних доверительных интервалов, в которых $d_{ij}(R)$ находится с высокой вероятностью (например, 90%) при каждом значении R.

Идеальной диаграммой Шепарда является наклонная прямая — биссектриса первой четверти. Чем «толщеk облако точек, представленное на диаграмме, тем сильнее искажения, и тем меньшего доверия заслуживает карта сходства.

Пример 7.1. Один из надёжных способов тестирования методов многомерного шкалирования, предложенный в [15] — размещение изначально двумерных (или близких к двумерным) выборок. Например, можно взять множество городов Российской Федерации, и задать R_{ij} как евклидовы расстояния между их географическими координатами (долгота, широта). Хороший алгоритм шкалирования должен построить карту сходства, похожую на географическую карту (с точностью до поворотов и отражений), а диаграмма Шепарда должна оказаться практически диагональной.

Список литературы

[1] *Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д.* Прикладная статистика: классификация и снижение размерности. — М.: Финансы и статистика, 1989.

[2] *Айвазян С. А., Енюков И. С., Мешалкин Л. Д.* Прикладная статистика: исследование зависимостей. — М.: Финансы и статистика, 1985.

[3] *Айзерман М. А., Браверман Э. М., Розоноэр Л. И.* Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — 320 pp.

[4] *Вапник В. Н.* Восстановление зависимостей по эмпирическим данным. — М.: Наука, 1979.

[5] *Вапник В. Н., Червоненкис А. Я.* О равномерной сходимости частот появления событий к их вероятностям // *ДАН СССР*. — 1968. — Т. 181, № 4. — С. 781–784.

[6] *Вапник В. Н., Червоненкис А. Я.* О равномерной сходимости частот появления событий к их вероятностям // *Теория вероятностей и ее применения*. — 1971. — Т. 16, № 2. — С. 264–280.

[7] *Вапник В. Н., Червоненкис А. Я.* Теория распознавания образов. — М.: Наука, 1974.

[8] *Головко В. А.* Нейронные сети: обучение, организация и применение. — М.: ИПРЖР, 2001.

[9] *Епанечников В. А.* Непараметрическая оценка многомерной плотности вероятности // *Теория вероятностей и её применения*. — 1969. — Т. 14, № 1. — С. 156–161.

[10] *Ермаков С. М., Михайлов Г. А.* Курс статистического моделирования. — М.: Наука, 1976.

[11] *Загоруйко Н. Г.* Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.

[12] *Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С.* Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985.

[13] *Закс Ш.* Теория статистических выводов. — М.: Мир, 1975.

[14] *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного // *Докл. АН СССР*. — 1958. — Т. 114, № 5. — С. 953–956.

[15] *Кулаичев А. П.* Методы и средства комплексного анализа данных. — М: ИНФРА-М, 2006.

[16] *Лагутин М. Б.* Наглядная математическая статистика. — М.: П-центр, 2003.

[17] *Лапко А. В., Ченцов С. В., Крохов С. И., Фельдман Л. А.* Обучающиеся системы обработки информации и принятия решений. Непараметрический подход. — Новосибирск: Наука, 1996.

[18] *Мандель И. Д.* Кластерный анализ. — М.: Финансы и Статистика, 1988.

[19] *Мерков А. Б.* Основные методы, применяемые для распознавания рукописного текста. — Лаборатория распознавания образов МЦНМО. — 2005. http://www.recognition.mccme.ru/pub/RecognitionLab.html/methods.html.

[20] *Мерков А. Б.* О статистическом обучении. — Лаборатория распознавания образов МЦНМО. — 2006. http://www.recognition.mccme.ru/pub/RecognitionLab.html/slt.pdf.

[21] Нейроинформатика / А. Н. Горбань, В. Л. Дунин-Барковский, А. Н. Кирдин, Е. М. Миркес, А. Ю. Новоходько, Д. А. Россиев, С. А. Терехов и др. — Новосибирск: Наука, 1998. — 296 с.

[22] *Орлов А. И.* Нечисловая статистика. — М.: МЗ-Пресс, 2004.

[23] *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. — М.: Наука, 1986.

[24] *Уиллиамс У. Т., Ланс Д. Н.* Методы иерархической классификации // Статистические методы для ЭВМ / Под ред. М. Б. Малютов. — М.: Наука, 1986. — С. 269–301.

[25] *Хардле В.* Прикладная непараметрическая регрессия. — М.: Мир, 1993.

[26] *Шлезингер М., Главач В.* Десять лекций по статистическому и структурному распознаванию. — Киев: Наукова думка, 2004.

[27] *Шлезингер М. И.* О самопроизвольном различении образов // Читающие автоматы. — Киев, Наукова думка, 1965. — Pp. 38–45.

[28] *Шурыгин А. М.* Прикладная стохастика: робастность, оценивание, прогноз. — М.: Финансы и статистика, 2000.

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

[30] *Asuncion A., Newman D.* UCI machine learning repository: Tech. rep.: University of California, Irvine, School of Information and Computer Sciences, 2007.
http://www.ics.uci.edu/~mlearn/MLRepository.html.

[31] *Bartlett P.* The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network // *IEEE Transactions on Information Theory*. — 1998. — Vol. 44, no. 2. — Pp. 525–536.
http://discus.anu.edu.au/~bartlett.

[32] *Bartlett P., Shawe-Taylor J.* Generalization performance of support vector machines and other pattern classifiers // Advances in Kernel Methods. — MIT Press, Cambridge, USA, 1999. — Pp. 43–54.
http://citeseer.ist.psu.edu/bartlett98generalization.html.

[33] *Bishop C. M.* Pattern Recognition and Machine Learning. — Springer, Series: Information Science and Statistics, 2006. — 740 pp.

[34] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // *ESAIM: Probability and Statistics*. — 2005. — no. 9. — Pp. 323–375.
http://www.econ.upf.edu/~lugosi/esaimsurvey.pdf.

[35] *Burges C. J. C.* A tutorial on support vector machines for pattern recognition // *Data Mining and Knowledge Discovery*. — 1998. — Vol. 2, no. 2. — Pp. 121–167.
http://citeseer.ist.psu.edu/burges98tutorial.html.

[36] *Burges C. J. C.* Geometry and invariance in kernel based methods // Advances in Kernel Methods / Ed. by B. Scholkopf, C. C. Burges, A. J. Smola. — MIT Press, 1999. — Pp. 89 – 116.

[37] *Cleveland W. S.* Robust locally weighted regression and smoothing scatter plots // *Journal of the American Statistical Association*. — 1979. — Vol. 74, no. 368. — Pp. 829–836.

[38] *Cortes C., Vapnik V.* Support-vector networks // *Machine Learning*. — 1995. — Vol. 20, no. 3. — Pp. 273–297.
http://citeseer.ist.psu.edu/cortes95supportvector.html.

[39] *Dempster A. P., Laird N. M., Rubin D. B.* Maximum likelihood from incomplete data via the EM algorithm // *J. of the Royal Statistical Society, Series B*. — 1977. — no. 34. — Pp. 1–38.

[40] *Durbin R., Rummelhart D. E.* Product units: A computationally powerful and biologically plausible extension to backpropagation networks // *Neural Computation*. — 1989. — Vol. 1, no. 4. — Pp. 133–142.

[41] *Fisher R. A.* The use of multiple measurements in taxonomic problem // *Ann. Eugen.* — 1936. — no. 7. — Pp. 179–188.

[42] *Hassibi B., Stork D. G.* Second order derivatives for network pruning: Optimal brain surgeon // Advances in Neural Information Processing Systems / Ed. by S. J. Hanson, J. D. Cowan, C. L. Giles. — Vol. 5. — Morgan Kaufmann, San Mateo, CA, 1993. — Pp. 164–171.
http://citeseer.ist.psu.edu/hassibi93second.html.

[43] *Hastie T., Tibshirani R.* Generalized additive models // *Statistical Science*. — 1986. — Vol. 1. — Pp. 297–318.
http://citeseer.ist.psu.edu/hastie95generalized.html.

[44] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. — Springer, 2001. — 533 pp.
http://http://www-stat.stanford.edu/~tibs/ElemStatLearn.

[45] *Hebb D.* The organization of behavior. — New York: Wiley, 1949.

[46] *Jain A., Murty M., Flynn P.* Data clustering: A review // *ACM Computing Surveys*. — 1999. — Vol. 31, no. 3. — Pp. 264–323.
http://citeseer.ifi.unizh.ch/jain99data.html.

[47] *Jordan M. I., Xu L.* Convergence results for the EM algorithm to mixtures of experts architectures: Tech. Rep. A.I. Memo No. 1458: MIT, Cambridge, MA, 1993.

[48] *Kohavi R.* A study of cross-validation and bootstrap for accuracy estimation and model selection // 14th International Joint Conference on Artificial Intelligence, Palais de Congres Montreal, Quebec, Canada. — 1995. — Pp. 1137–1145.
http://citeseer.ist.psu.edu/kohavi95study.html.

[49] *Lance G. N., Willams W. T.* A general theory of classification sorting strategies. 1. hierarchical systems // *Comp. J.* — 1967. — no. 9. — Pp. 373–380.

[50] *LeCun Y., Bottou L., Orr G. B., Muller K.-R.* Effi cient BackProp // Neural Networks: tricks of the trade. — Springer, 1998.

[51] *LeCun Y., Denker J., Solla S., Howard R. E., Jackel L. D.* Optimal brain damage // Advances in Neural Information Processing Systems II / Ed. by D. S. Touretzky. — San Mateo, CA: Morgan Kauff man, 1990.
http://citeseer.ist.psu.edu/lecun90optimal.html.

[52] *McCulloch W. S., Pitts W.* A logical calculus of ideas immanent in nervous activity // *Bulletin of Mathematical Biophysics*. — 1943. — no. 5. — Pp. 115–133.

[53] *Mercer J.* Functions of positive and negative type and their connection with the theory of integral equations // *Philos. Trans. Roy. Soc. London*. — 1909. — Vol. A, no. 209. — Pp. 415–446.

[54] *Minsky M., Papert S.* Perceptrons: an Introduction to Computational Geometry. — MIT Press, 1968.

[55] *Novikoff A. B. J.* On convergence proofs on perceptrons // Proceedings of the Symposium on the Mathematical Theory of Automata. — Vol. 12. — Polytechnic Institute of Brooklyn, 1962. — Pp. 615–622.

[56] *Parzen E.* On the estimation of a probability density function and mode // *Annals of Mathematical Statistics*. — 1962. — Vol. 33. — Pp. 1065–1076.
http://citeseer.ist.psu.edu/parzen62estimation.html.

[57] *Rosenblatt M.* Remarks on some nonparametric estimates of a density function // *Annals of Mathematical Statistics*. — 1956. — Vol. 27, no. 3. — Pp. 832–837.

[58] *Rummelhart D. E., Hinton G. E., Williams R. J.* Learning internal representations by error propagation // Vol. 1 of Computational models of cognition and perception, chap. 8. — Cambridge, MA: MIT Press, 1986. — Pp. 319–362.

[59] *Shawe-Taylor J., Cristianini N.* Robust bounds on generalization from the margin distribution: Tech. Rep. NC2–TR–1998–029: Royal Holloway, University of London, 1998.
http://citeseer.ist.psu.edu/shawe-taylor98robust.html.

[60] *Smola A., Bartlett P., Scholkopf B., Schuurmans D.* Advances in large margin classifiers. — MIT Press, Cambridge, MA. — 2000.
http://citeseer.ist.psu.edu/article/smola00advances.html.

[61] *Smola A., Schoelkopf B.* A tutorial on support vector regression: Tech. Rep. NeuroCOLT2 NC2-TR-1998-030: Royal Holloway College, London, UK, 1998.
http://citeseer.ist.psu.edu/smola98tutorial.html.

[62] *Stone M. N.* The generalized Weierstrass approximation theorem // *Math. Mag.* — 1948. — Vol. 21. — Pp. 167–183, 237–254.

[63] *Tibshirani R. J.* Regression shrinkage and selection via the lasso // *Journal of the Royal Statistical Society. Series B (Methodological).* — 1996. — Vol. 58, no. 1. — Pp. 267–288.
http://citeseer.ist.psu.edu/tibshirani94regression.html.

[64] *Tipping M.* The relevance vector machine // Advances in Neural Information Processing Systems, San Mateo, CA. — Morgan Kaufmann, 2000.
http://citeseer.ist.psu.edu/tipping00relevance.html.

[65] *Vapnik V., Chapelle O.* Bounds on error expectation for support vector machines // *Neural Computation*. — 2000. — Vol. 12, no. 9. — Pp. 2013–2036.
http://citeseer.ist.psu.edu/vapnik99bounds.html.

[66] *Widrow B., Hoff M. E.* Adaptive switching circuits // IRE WESCON Convention Record. — DUNNO, 1960. — Pp. 96–104.

[67] *Wu C. F. G.* On the convergence properties of the EM algorithm // *The Annals of Statistics*. — 1983. — no. 11. — Pp. 95–103.
http://citeseer.ist.psu.edu/78906.html.