Лекции по линейным алгоритмам классификации

К. В. Воронцов

19 января 2009 г.

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

Содержание

1 Линейные алгоритмы классификации

Линейный классификатор, называемый также *линейным решающим правилом* — это один из самых простых алгоритмов классификации. Теория распознавания, как и многие другие разделы математики и физики, начиналась с подробного изучения линейного случая. Затем находились различные способы его обобщения. Линейная теория чрезвычайно полезна как источник базовых концепций, однако практическая пригодность линейных моделей весьма ограничена.

В задачах с двумя классами линейный классификатор — это гиперплоскость, разделяющая n-мерное признаковое пространство на два класса — полупространства. В задачах со многими классами разделяющая поверхность кусочно-линейна.

Обучение линейного классификатора заключается в подборе такого положения гиперплоскости, при котором классы разделяются наилучшим образом. Что значит «наилучшим» — вопрос нетривиальный; существует несколько подходов, приводящих к различным решениям.

Простейшим обоснованием линейного классификатора служит его аналогия с нервной клеткой — нейроном. *Персептронные* принципы обучения, первоначально заимствованные из нейрофизиологии, затем нашли математическое обоснование с точки зрения градиентных методов минимизации эмпирического риска, см. §1.1. Дальнейшее обобщение шло по пути *коннективизма*, или соединения простых элементов в сложные композиции — *искусственные нейронные сети* (artificial neural network, ANN). Этот подход рассматривается отдельно в главе **??**.

Другое обоснование линейного классификатора даёт байесовская теория классификации. Известно, что *линейный дискриминант Фишера* является оптимальным байесовским классификатором, если плотности распределения классов нормальны и имеют одинаковые матрицы ковариации, то есть «одинаковую форму». Оказывается, требование нормальности избыточно, и линейный классификатор остаётся оптимальным при менее жёстких предположениях. Эти результаты лежат в основе метода *логистической регрессии*, см. §1.2. Внимательное рассмотрение выявляет значительную общность логистической регрессии с персептронными методами.

Ещё одно эвристическое обоснование линейного классификатора — принцип *оптимальной разделяющей гиперплоскости*. Гиперплоскость строится так, чтобы обучающие объекты были удалены от неё настолько далеко, насколько это возможно. Этот принцип лежит в основе *метода обобщённого портрета* [1], современная версия которого известна под названием *метода опорных векторов* (SVM), см. §1.3. Существует изящное обобщение SVM на нелинейный случай, благодаря которому он считается одним из наиболее продуктивных подходов в машинном обучении.

Несмотря на различия в исходных обоснованиях, все перечисленные методы допускают единое обобщение. Все они являются частными случаями минимизации эмпирического риска, и отличаются только видом функции потерь, либо видом регуляризатора, если таковой используется. При этом численные методы решения оптимизационных задач могут отличаться довольно сильно. В §1.4 описано соответствующее обобщение, позволяющее синтезировать новые процедуры обучения, обладающие нужными свойствами. Показано также ещё несколько общих приёмов, применимых и к нелинейным классификаторам.

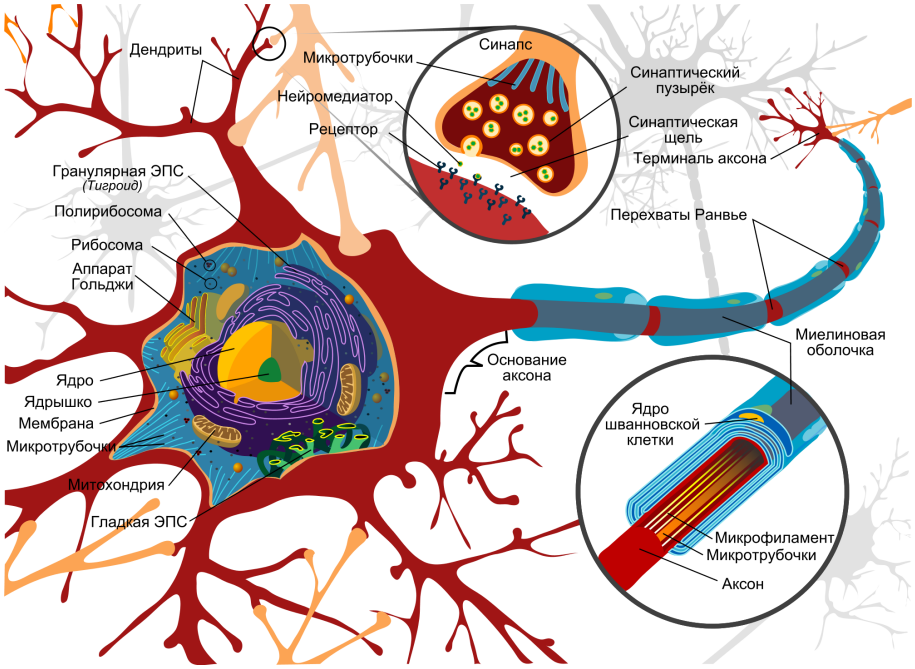


Рис. 1. Нервная клетка (иллюстрация из Википедии — свободной энциклопедии).

§1.1 Однослойный персептрон

1.1.1 Нейрофизиологическое обоснование линейного классификатора

Рассмотрим в общих чертах принципы работы нервной клетки — *нейрона*, Рис 1. Клетка имеет множество разветвлённых отростков — *дендритов*, и одно длинное тонкое волокно — *аксон*, на конце которого находятся *синапсы*, примыкающие к дендритам других нервных клеток. Каждая 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 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.

Таким образом, нервную клетку можно рассматривать как устройство, которое на каждом такте своей работы принимает заряды от множества входов-дендритов, и если суммарный накопленный заряд превосходит некоторый порог, передаёт импульс через множество выходов-синапсов.

Нервная система состоит из огромного числа связанных друг с другом нейронов, распространяющих направленные волны импульсов. Скорость распространения импульсов составляет приблизительно 100 м/с, что в миллион раз меньше скорости распространения электрического сигнала в медной проволоке. Тем не менее, сложные задачи распознавания человек решает за десятые доли секунды. Это означает, что нейровычисления требуют порядка 10^2 последовательных тактов и выполняются

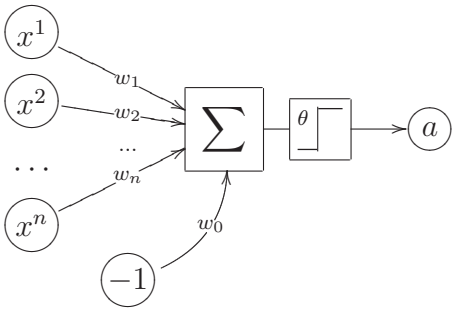


Рис. 2. Модель МакКаллока–Питтса.

с большой степенью параллелизма. Кора головного мозга человека содержит порядка 10^{11} нейронов, и каждый нейрон связан через синапсы с 10^3–10^4 других нейронов. Это обеспечивает высокую взаимозаменяемость нервных клеток и надежность системы в целом. Отказ части нейронов не нарушает нормального хода распространения нервного импульса.

Описанная схема сильно упрощена; на самом деле механизмы функционирования нервных клеток гораздо сложнее. Более того, известны десятки различных типов нейронов, которые функционируют совсем по-другому и не являются взаимозаменяемыми. Рассматриваемая далее математическая модель нейрона ещё сильнее упрощена. В теории искусственных нейронных сетей не ставится задача максимально точно воспроизвести функционирование биологической нейронной сети. Цель в том, чтобы подсмотреть некоторые принципы у живой природы и использовать их для интеллектуализации автоматических устройств.

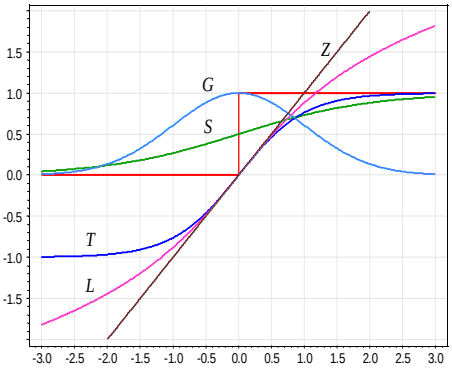
Модель МакКаллока–Питтса. Пусть X — пространство объектов; Y — множество допустимых ответов; объекты описываются n числовыми признаками $f_j\colon X \to \mathbb{R}$, $j = 1, \ldots, n$. Вектор $x = (x^1, \ldots, x^n) \in \mathbb{R}^n$, где $x^j = f_j(x)$, называется *признаковым описанием* объекта x.

В 1943 году МакКаллок и Питтс [16] предложили математическую модель нейрона, Рис. 2. Для простоты будем пока полагать, что все признаки бинарные, и ответы тоже бинарны, $Y = \{0, 1\}$. Значения признаков $x^j = f_j(x)$ будем трактовать как величины импульсов, поступающих на вход нейрона через n входных синапсов. Поступающие в нейрон импульсы складываются с весами $w_1, \ldots, w_n$. Если вес w_j положительный, то j-й синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный импульс превышает *порог активации* w_0, то нейрон возбуждается и выдаёт на выходе 1, иначе выдаётся 0. Таким образом, нейрон вычисляет n-арную булеву функцию вида

$$a(x) = \varphi\left(\sum_{j=1}^{n} w_j x^j - w_0\right),$$

где $\varphi(z) = [z \geqslant 0]$ — пороговая функция Хевисайда. Функцию φ, преобразующую значение суммарного импульса в выходное значение нейрона, принято называть *функцией активации*.

Введём дополнительный константный признак $x^0 \equiv -1$ и векторные обозначения $w = (w_0, w_1, \ldots, w_n)^\intercal$, $x = (x^0, x^1, \ldots, x^n)^\intercal$. Тогда линейную модель нейрона



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

Рис. 3. Стандартные функции активации $\varphi(z)$.

можно записать более компактно через скалярное произведение:

$$a(x, w) = \varphi\left(\sum_{j=0}^{n} w_j x^j\right) = \varphi\big(\langle w, x\rangle\big). \tag{1.1}$$

Таким образом, модель нейрона по МакКаллоку-Питтсу представляет собой линейный пороговый классификатор. Позже эта модель была обобщена на случай произвольных вещественных входов и выходов и произвольных функций активации. Чаще всего используются функции активации, показанные на Рис. 3.

1.1.2 Метод стохастического градиента

Рассмотрим задачу настройки (обучения) синаптических весов в модели МакКаллока-Питтса. Пусть $y^*\colon X \to Y$ — целевая зависимость, известная только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется найти вектор весов w, при котором алгоритм $a(x, w)$ аппроксимирует целевую зависимость $y^*(x)$. Согласно принципу *минимизации эмпирического риска* задача сводится к поиску вектора w, доставляющего минимум функционалу

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i, w), y_i\big) \to \min_w, \tag{1.2}$$

где $\mathscr{L}(a, y)$ — заданная функция потерь, характеризующая величину ошибки ответа a при правильном ответе y. Рассмотрим задачу минимизации в общем случае, пока не конкретизируя функцию $\mathscr{L}$.

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 $\nabla Q(w) = \big(\frac{\partial Q(w)}{\partial w_j}\big)_{j=1}^n$:

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

где $\eta > 0$ — величина шага в направлении антиградиента, называемая также *темпом обучения* (learning rate). Предполагая, что функция потерь $\mathscr{L}$ и функция активации φ дифференцируемы, распишем градиент:

$$w := w - \eta \sum_{i=1}^{\ell} \mathscr{L}'_a\big(a(x_i, w), y_i\big)\varphi'(\langle w, x_i\rangle)x_i. \tag{1.3}$$

Алгоритм 1.1. Обучение персептрона методом стохастического градиента.

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
λ — параметр сглаживания функционала Q;

Выход:
Синаптические веса $w_0, w_1, \ldots, w_n$;

1: инициализировать веса w_j, $j = 0, \ldots, n$;
2: инициализировать текущую оценку функционала:
$Q := \sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i, w), y_i\big)$;
3: **повторять**
4: выбрать объект x_i из X^ℓ (например, случайным образом);
5: вычислить выходное значение алгоритма $a(x_i, w)$ и ошибку:
$\varepsilon_i := \mathscr{L}\big(a(x_i, w), y_i\big)$;
6: сделать шаг градиентного спуска:
$w := w - \eta \mathscr{L}'_a\big(a(x_i, w), y_i\big)\varphi'(\langle w, x_i\rangle)x_i$;
7: оценить значение функционала:
$Q := (1 - \lambda)Q + \lambda\varepsilon_i$;
8: **пока** значение Q не стабилизируется и/или веса w не перестанут изменяться;

Каждый прецедент (x_i, y_i) вносит аддитивный вклад в изменение вектора w, но вектор w изменяется только после перебора всех ℓ объектов. Оказывается, незначительная модификация этого процесса способна существенно повысить скорость сходимости. Для этого надо выбирать прецеденты (x_i, y_i) по одному, для каждого делать градиентный шаг и сразу обновлять вектор весов:

$$w := w - \eta \mathscr{L}'_a\big(a(x_i, w), y_i\big)\varphi'(\langle w, x_i\rangle)x_i. \tag{1.4}$$

В методе *стохастического градиента* (stochastic gradient, SG) прецеденты перебираются в случайном порядке. Если же объекты предъявлять в некотором фиксированном порядке, процесс чаще оказывается расходящимся или зацикливается.

В Алгоритме 1.1 показана реализация метода SG при произвольных функциях $\mathscr{L}(a, y)$ и $\varphi(z)$. Обратим внимание на инициализацию весов и критерий останова.

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

$$w_j := \frac{\langle y, f_j\rangle}{\langle f_j, f_j\rangle}, \tag{1.5}$$

где $f_j = \big(f_j(x_i)\big)_{i=1}^{\ell}$ — вектор значений j-го признака. Эти оценки являются точными в одном нереалистичном частном случае — когда функция активации линейна, функция потерь квадратична и признаки статистически независимы. Доказательство данного факта вынесено в Упр. 1.1.

Критерий останова в Алгоритме 1.1 основан на приблизительной оценке функционала Q методом экспоненциальной скользящей средней. Точное значение потребовало бы вычисления ℓ скалярных произведений $\langle w, x_i \rangle$, что довольно накладно. Когда градиентный метод подходит к окрестности минимума, оценка скользящего среднего стабилизируется и приближается к точному значению функционала. Параметр λ можно положить равным $1/\ell$. В случае избыточно длинной выборки его рекомендуется увеличивать.

Преимущества метода SG

- Метод легко реализуется и легко легко обобщается на нейронные сети более сложных архитектур, см. **??**. Он считается классическим инструментом настройки нейронных сетей.
- Метод хорошо приспособлен для динамического обучения, когда обучающие объекты поступают потоком, и надо быстро обновлять вектор весов при появлении каждого нового объекта.
- Метод позволяет настраивать веса на избыточно больших выборках, за счёт того, что случайной подвыборки может оказаться достаточно для обучения.
- Допускаются различные стратегии обучения. В случае большой выборки или динамического потока можно вообще не сохранять обучающие объекты. В случае малой выборки можно повторно предъявлять для обучения одни и те же объекты, что способствует повышению качества классификации.

Недостатки метода SG

- Метод может не сходиться или сходиться очень медленно.
- Функционал Q, как правило, оказывается многоэкстремальным, и процесс градиентного спуска склонен «застревать» в локальных минимумах.
- 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 активации имеет горизонтальные асимптоты (например, сигмоидная или th), то процесс может попасть в состояние «паралича». Чем больше значение скалярного произведения $\langle w, x_i \rangle$ на входе функции активации φ, тем ближе значение производной φ' к нулю, тем меньше изменение синаптических весов, согласно (1.4). Если веса w_j попали в область больших значений, то у них практически не остаётся шансов выбраться из этой «мёртвой зоны».

В 1.1.4 будут перечислены некоторые приёмы, улучшающие сходимость и качество градиентного обучения. Но перед этим рассмотрим несколько частных случаев, которые были заметными вехами в развитии персептронных методов обучения.

1.1.3 Классические частные случаи

Адаптивный линейный элемент. Рассмотрим задачу регрессии, когда признаки и ответы вещественны, $X = \mathbb{R}^n$, $Y \subseteq \mathbb{R}$; функция активации линейна, $\varphi(z) = z$, функция потерь квадратична, $\mathscr{L}(a, y) = (a - y)^2$. Тогда правило обновления весов в методе стохастического градиента примет вид

$$w := w - \eta\big(\langle w, x_i\rangle - y_i\big)x_i. \tag{1.6}$$

Это правило обучения было предложено Видроу и Хоффом в 1960 году и называется *дельта-правилом* (delta-rule), а сам линейный нейрон — *адаптивным линейным элементом* или ADALINE [27].

Линейный классификатор и выбор функции активации. Формальное применение дельта-правила в задачах классификации оказывается не вполне адекватным. Действительно, положим для определённости $Y = \{0, 1\}$ и рассмотрим объект x класса $y = 0$. Если $\langle w, x\rangle = 1$, то на объекте x допускается ошибка, величина потери равна единице. Если же $\langle w, x\rangle = -1$, то это, напротив, говорит о надёжном отнесении объекта к классу 0, однако величина потери и в этом случае равна единице.

Чтобы использовать квадратичную функцию потерь в задачах классификации, нужно взять другую функцию активации, например, подойдёт сигмоида

$$\varphi(z) = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Перекодировка номеров классов $Y = \{-1, +1\}$ и применение нечётных функций потерь способствует увеличению скорости сходимости (примерно в полтора раза); например, можно взять гиперболический тангенс:

$$\varphi(z) = \operatorname{th}(z) = 2\sigma(2z) - 1.$$

Проблема в том, что функции $\sigma(z)$ и $\operatorname{th}(z)$ имеют горизонтальные асимптоты, ординаты которых совпадают с номерами классов. Итерационный процесс (1.4), стремясь уменьшить величину невязок $\big|\varphi\langle w, x_i\rangle - y_i\big|$, вынужден увеличивать норму вектора весов w, что может приводить к «параличу». Поэтому рекомендуется выбирать функцию активации так, чтобы метки классов попадали внутрь её области значений. В [15] предлагается брать функцию

$$\varphi(z) = 1.7159\ \operatorname{th}\big(\tfrac{2}{3}z\big),$$

у которой в точках с ординатами $\{-1, +1\}$ достигается максимум второй производной. Более радикальная рекомендация — добавлять линейный член:

$$\varphi(z) = \operatorname{th} z + \gamma z$$

с параметром γ, либо использовать медленно растущие функции. В [2] при решении задач прогнозирования применяется функция активации

$$\varphi(z) = \ln\big(z + \sqrt{z^2 + 1}\big).$$

Персептрон Розенблатта. В 1957 году Розенблатт предложил эвристический алгоритм обучения нейрона, основанный на принципах нейрофизиологии. Экспериментально было обнаружено, 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. Как признаки, так и ответы будем пока полагать бинарными. Перед началом обучения вектор весов некоторым способом инициализируется, например, заполняется нулевыми или случайными значениями. Затем обучающие объекты x_i по очереди подаются на вход модели МакКаллока–Питтса (1.1), и выданные ответы сравниваются с правильными.

Если ответ $a(x_i)$ совпадает с y_i, то вектор весов не изменяется.

Если $a(x_i) = 0$ и $y_i = 1$, то вектор весов w увеличивается. Увеличивать имеет смысл только те веса w_j, для которых $x_i^j \neq 0$; изменение других компонент не повлияет на результат. Положим $w := w + \eta x_i$, где $\eta > 0$ — темп обучения.

Если $a(x_i) = 1$ и $y_i = 0$, то вектор весов уменьшается: $w := w - \eta x_i$.

Поскольку все величины бинарные, три случая объединяются в одну формулу:

$$w := \begin{Bmatrix} w, & \text{при } a(x_i) = y_i \\ w + \eta x_i, & \text{при } a(x_i) = 0,\ y_i = 1 \\ w - \eta x_i, & \text{при } a(x_i) = 1,\ y_i = 0 \end{Bmatrix} = w - \eta\big(a(x_i) - y_i\big)x_i. \tag{1.7}$$

Обучающие объекты пропускаются через это правило многократно, пока веса изменяются. Этот алгоритм обучения и есть *однослойный персептрон Розенблатта*. Его можно рассматривать как частный случай Алгоритма 1.1, если взять пороговую функцию активации и заменить шаг 6 формулой (1.7).

Отметим, что формула (1.7), выведенная из чисто эвристических соображений, совпадает с дельта-правилом (1.6). То есть персептрон Розенблатта также можно рассматривать как частный случай градиентного метода.

Правило Хэбба. Иногда удобнее полагать, что классы помечены числами -1 и 1, а нейрон выдаёт знак скалярного произведения:

$$a(x, w) = \operatorname{sign}(\langle w, x\rangle).$$

Тогда несовпадение знаков $\langle w, x_i\rangle$ и y_i означает, что нейрон ошибается на объекте x_i, и выражение (1.7) преобразуется в следующее правило модификации весов:

$$\textbf{если } \langle w, x_i\rangle\, y_i < 0 \textbf{ то } w := w + \eta x_i y_i, \tag{1.8}$$

называемое *правилом Хэбба* [14]. Соответственно, в Алгоритме 1.1 заменяется шаг 6.

Для правила Хэбба докажем теорему сходимости. Она справедлива не только для бинарных, но и для произвольных действительных признаков.

Теорема 1.1 (Новиков, 1962 [18]). *Пусть* $X = \mathbb{R}^{n+1}$, $Y = \{-1, 1\}$, *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 1.1 с правилом Хэбба (1.8) находит вектор весов, разделяющий обучающую выборку без ошибок, за конечное число исправлений, из любого начального положения* 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 (1.8) в этом случае происходит модификация весов. В силу условия линейной разделимости, справедлива оценка снизу:

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

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

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

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 $\cos \leqslant 1$ следует $\sqrt{t}\delta/D \leqslant 1$, 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 вполне может оказаться расходящимся.

1.1.4 Эвристики для улучшения сходимости и обобщающей способности

В этом разделе рассматриваются эвристические приёмы и рекомендации, компенсирующие недостатки градиентных методов обучения. Все они в полной мере относятся к градиентным методам обучения нейронных сетей, включая широко известный метод *обратного распространения ошибок*, который будет рассмотрен позже, в **??**. Различных тонкостей настолько много, что применение градиентных методов по праву считается искусством, см. также обзор [15].

Нормализация данных. Градиентный метод чувствителен к масштабу измерения признаков. Если норма вектора объекта $\|x_i\|$ принимает большие значения, а функция активации 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 использовать метод стохастического градиента, то есть предъявлять 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 указать, какой из объектов максимально информативен на данном шаге обучения. Простая для реализации 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. Другой вариант предыдущей эвристики отличается простотой реализации и заключается в том, чтобы сравнить величину ошибки на предъявленном объекте с некоторым порогом. Если ошибка окажется меньше порога, вектор весов не модифицируется. Логика этой эвристики в точности та же, что у персептрона Розенблатта: если объект уже неплохо классифицируется, то менять веса не нужно. При этом увеличивается и скорость настройки.

Сокращение весов является частным случаем регуляризации некорректно поставленных задач по А. Н. Тихонову [5]. Аналогичный приём применяется в нормальном дискриминантном анализе (**??**) и в регрессионном анализе (**??**), где он получил название *гребневой регрессии*. Идея заключается в том, чтобы ограничить возможный рост абсолютных значений весов, добавив к минимизируемому функционалу $Q(w)$ штрафное слагаемое:

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

Смысл этой добавки в том, чтобы сделать решение более устойчивым. Если функционал $Q(w)$ достигает минимального (или близкого к минимальному) значения на некотором множестве векторов w, то из них разумнее всего выбрать вектор w с наименьшей нормой.

Добавление штрафного слагаемого приводит к аддитивной поправке градиента: $\nabla Q_\tau(w) = \nabla Q(w) + \tau w$. В результате правило обновления весов принимает вид

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

Таким образом, вся модификация алгоритма сводится к появлению неотрицательного множителя $(1-\eta\tau)$, приводящего к постоянному уменьшению весов. Отсюда название метода — *сокращение весов* (weights decay).

Преимущества метода в том, что он очень просто реализуется, сочетается со многими функционалами Q и многими градиентными методами оптимизации. Он предотвращает паралич и повышает устойчивость весов в случае мультиколлинеарности — когда имеются линейно зависимые или сильно коррелированные признаки. В конечном итоге сокращение весов способствует повышению обобщающей способности алгоритма и снижению риска переобучения. Дополнительный 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 затратами времени.

Можно показать, что метод сокращения весов уменьшает *эффективную размерность* вектора w, см. **??** и [13]. Однако количество параметров остаётся тем же. В некоторых случаях предпочтителен другой подход, при котором избыточные веса обнуляются, что фактически соответствует отбору признаков, см. **??** и **??**.

Выбор величины шага.

1. Известно, что градиентные методы сходятся при определённых условиях, 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. Метод скорейшего градиентного спуска приводит к выбору *адаптивного шага* η исходя из решения одномерной задачи минимизации $Q\big(w - \eta\nabla Q(w)\big) \to \min\limits_{\eta}$. Во многих случаях эту задачу удаётся решить аналитически [2]. В частности, для алгоритма ADALINE с линейной функцией активации

$$\eta = \|x_i\|^{-2}. \tag{1.9}$$

Формулы вычисления адаптивного шага получены и для более сложных случаев, в том числе для метода обратного распространения ошибок [2].

Выбивание из локальных минимумов необходимо для предотвращения сходимости к недостаточно хорошим локальным минимумам функционала $Q(w)$. 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. Слишком глубокая оптимизация может приводить к переобучению. Узкий глобальный минимум функционала $Q(w)$ менее предпочтителен, чем более пологий и устойчивый локальный минимум. Для предотвращения попадания в такие «расщелины» применяется техника *раннего останова* (early stopping). По ходу итераций вычисляется какой-нибудь *внешний критерий* (стр. **??**), и если он начинает возрастать, процесс настройки прекращается.

§1.2 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 вид функции активации (ею оказывается сигмоидная функция) и функции потерь. В-третьих, возникает интересная дополнительная возможность наряду с классификацией объекта получать численные оценки вероятности его принадлежности каждому из классов.

1.2.1 Обоснование логистической регрессии

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

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

Гипотеза 1.1. *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. 1.1. *Плотность распределения $p_y(x)$, $x \in \mathbb{R}^n$ называется экспонентной, если*

$$p_y(x) = \exp\bigl(c(\delta)\,\langle\theta, x\rangle + b(\delta, \theta) + d(x, \delta)\bigr),$$

где $\theta \in \mathbb{R}^n$ — векторный параметр, называемый сдвигом; δ — параметр, называемый 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 1.1. Многомерное нормальное распределение с вектором матожидания $\mu \in \mathbb{R}^n$ и ковариационной матрицей $\Sigma \in \mathbb{R}^{n\times n}$ является экспонентным с парамет-

ром сдвига $\theta = \Sigma^{-1}\mu$ и параметром разброса $\delta = \Sigma$:

$$\mathcal{N}(x;\mu,\Sigma) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\bigl(-\tfrac{1}{2}(x-\mu)^\intercal\Sigma^{-1}(x-\mu)\bigr) =$$
$$= \exp\bigl(\underbrace{\mu^\intercal\Sigma^{-1}x}_{\langle\theta,x\rangle} - \underbrace{\tfrac{1}{2}\mu^\intercal\Sigma^{-1}\Sigma\Sigma^{-1}\mu}_{b(\delta,\theta)} - \underbrace{\tfrac{1}{2}x^\intercal\Sigma^{-1}x - \tfrac{n}{2}\ln(2\pi) - \tfrac{1}{2}\ln|\Sigma|}_{d(x,\delta)}\bigr).$$

Гипотеза 1.2. *Функции правдоподобия классов $p_y(x)$ принадлежат экспонентному семейству плотностей, имеют равные значения параметров b, c, d, разброса δ, и отличаются только значениями параметра сдвига θ_y.*

Основная теорема. Напомним, что оптимальный байесовский классификатор имеет вид $a(x) = \arg\max\limits_{y\in Y}\lambda_y\,\mathsf{P}(y|x)$, где λ_y — штраф за ошибку на объектах класса y. В случае двух классов

$$a(x) = \operatorname{sign}\bigl(\lambda_+\,\mathsf{P}(+1|x) - \lambda_-\,\mathsf{P}(-1|x)\bigr) = \operatorname{sign}\left(\frac{\mathsf{P}(+1|x)}{\mathsf{P}(-1|x)} - \frac{\lambda_-}{\lambda_+}\right).$$

Теорема 1.2. *Если справедливы гипотезы 1.1, 1.2, и среди признаков $f_1(x),\ldots,f_n(x)$ есть константа, то байесовский классификатор линеен:*

$$a(x) = \operatorname{sign}\bigl(\langle w,x\rangle - w_0\bigr), \qquad w_0 = \ln(\lambda_-/\lambda_+),$$

причём апостериорная вероятность принадлежности произвольного объекта $x\in X$ классу $y\in\{-1,+1\}$ связана со значением дискриминантной функции:

$$\mathsf{P}(y|x) = \sigma\bigl(\langle w,x\rangle\,y\bigr),$$

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

Доказательство.

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

$$\frac{\mathsf{P}(+1|x)}{\mathsf{P}(-1|x)} = \frac{P_+p_+(x)}{P_-p_-(x)} = \exp\bigl(\underbrace{\langle c(\delta)(\theta_+-\theta_-),x\rangle}_{w=\text{const}(x)} + \underbrace{b(\delta,\theta_+) - b(\delta,\theta_-) + \ln\tfrac{P_+}{P_-}}_{\text{const}(x)}\bigr).$$

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

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

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

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

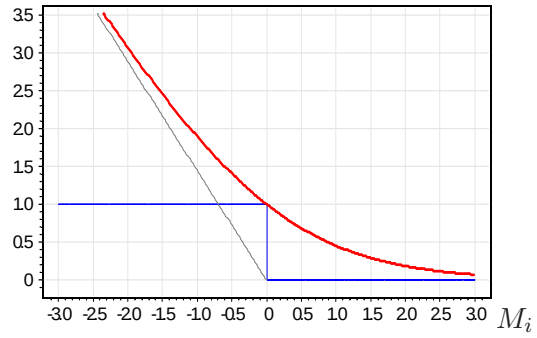


Рис. 4. Логарифмическая функция потерь $\log_2\bigl(1+e^{-M_i}\bigr)$ и её наклонная асимптота.

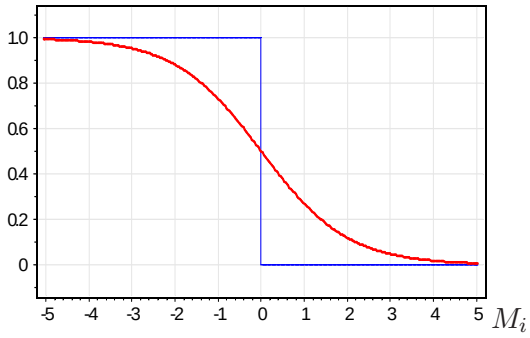


Рис. 5. Правило Хэбба: пороговое $[M_i < 0]$ и сглаженное $\sigma(-M_i)$.

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

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

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 1.2 через линейную дискриминантную функцию: $\mathsf{P}(y|x) = \sigma\bigl(\langle w,x\rangle\, y\bigr)$. Таким образом,

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

Максимизация $L(w)$ эквивалентна минимизации функционала $\widetilde{Q}(w)$:

$$\widetilde{Q}(w) = \sum_{i=1}^{\ell} \log_2\bigl(1+\exp(-\langle w,x_i\rangle\, y_i)\bigr) \to \min_w. \tag{1.10}$$

Опр. 1.2. *Величина $M_i(w) = \langle w,x_i\rangle\, y_i$ называется отступом (margin) объекта x_i относительно алгоритма классификации $a(x,w) = \operatorname{sign}\langle w,x\rangle$.*

Алгоритм $a(x,w)$ допускает ошибку на объекте x_i тогда и только тогда, когда его отступ $M_i(w)$ отрицателен. Чем больше отступ $M_i(w)$, тем надёжнее объект x_i относится к правильному классу y_i. Поэтому в задачах классификации имеет смысл применять функции потерь вида $\mathscr{L}\bigl(a(x_i,w),y_i\bigr) = g\bigl(M_i(w)\bigr)$, где $g(M)$ — некоторая монотонно невозрастающая функция отступа. В случае логистической регрессии, как следует из (1.10), оптимальным выбором функции g является $g(M) = \log_2\bigl(1+e^{-M}\bigr)$.

При любом $M \in \mathbb{R}$ справедливо неравенство $[M < 0] \leqslant g(M)$, см. Рис. 4. Поэтому $\widetilde{Q}(w)$ является верхней оценкой числа ошибок классификации $Q(w)$, а минимизация $\widetilde{Q}(w)$ ведёт к приближённой минимизации эмпирического риска:

$$Q(w) = \sum_{i=1}^{\ell} \big[a(x_i, w) \neq y_i\big] = \sum_{i=1}^{\ell} \big[\langle w, x_i\rangle\, y_i < 0\big] \leqslant \widetilde{Q}(w) \to \min_w .$$

1.2.2 Метод стохастического градиента для логистической регрессии

Гладкая функция потерь удобна тем, что позволяет применять градиентные методы для минимизации $\widetilde{Q}(w)$. Запишем градиент функционала $\widetilde{Q}(w)$, воспользовавшись определением отступа $M_i(w)$ и выражением для производной сигмоидной функции $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) = \sigma(z)\sigma(-z)$:

$$\nabla\widetilde{Q}(w) = -\sum_{i=1}^{\ell} y_i x_i \sigma\big(-M_i(w)\big).$$

Тогда градиентный шаг в методе стохастического градиента будет иметь вид:

$$w := w + \eta\, y_i x_i\, \sigma\big(-M_i(w)\big), \tag{1.11}$$

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

Связь с правилом Хэбба. Легко видеть, что логистическое правило пересчёта весов (1.11) есть ни что иное, как сглаженный вариант правила Хэбба (1.8), Рис. 5:

$$w := w + \eta\, y_i x_i\, \big[M_i(w) < 0\big].$$

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 повышению качества классификации и увеличению скорости сходимости [6, 23].

Об оценивании рисков Логистическая функция σ переводит значение линейной дискриминантной функции $\langle w, x\rangle$ в оценку вероятности того, что объект x принадлежит классу $+1$. Это свойство используется в тех приложениях, где наряду с классификацией объекта x требуется оценить связанный с ним *риск* как математическое ожидание потерь:

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

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

В практических ситуациях к оценкам риска следует относиться с осторожностью. Теорема 1.2 гарантирует, что $\mathsf{P}(y|x) = \sigma\big(\langle w, x\rangle\, y\big)$ 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.

О логарифмической функции потерь. Замена пороговой функции потерь её гладкой аппроксимацией может считаться ещё одним, хотя и менее строгим (эвристическим), обоснованием логистической регрессии. Логарифмическая функция потерь штрафует не только ошибки классификации, но и приближение объекта к границе классов (правая ветвь кривой, Рис. 4). В результате оптимизационная процедура ищет вектор w, увеличивающий зазор между классами, то есть более уверенно разделяющий классы.

В то же время, линейная асимптотика левой ветви кривой (Рис. 4) представляется на первый взгляд не вполне обоснованной эвристикой. Интуиция подсказывает, что решение w может оказаться неустойчивым, если обучающая выборка будет содержать *шумовые выбросы* — редкие объекты, попадающие в чужой класс из-за ошибок в исходных данных. Они имеют большой отрицательный отступ и вносят существенный вклад в функционал $\widetilde{Q}(w)$, хотя их следовало бы вовсе исключить из обучающей выборки. Возникает опасение, не будут ли веса w в значительной степени настраиваться по объектам-выбросам. Формула (1.11) показывает, что величина градиентного шага не чувствительна к величине отрицательного отступа, так что высказанное опасение не оправдано. Неустойчивость к шуму имела бы место, если бы левая ветвь функции потерь росла быстрее линейной функции.

О методах второго порядка. Для оптимизации весов w чаще используется метод Ньютона-Рафсона, описанный в главе о регрессионном анализе, см. §1.2. 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. Оба метода, 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 (вектора 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 *принципа Оккама* «не размножать 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.

- 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 $n \times n$-матрицы на каждом шаге и также не застрахован от плохой сходимости.

- Оценки вероятностей и рисков могут оказаться неадекватными, если не выполняются предположения Теоремы 1.2.

§1.3 Метод опорных векторов (SVM)

В 60–70-е годы коллективом советских математиков под руководством В. Н. Вапника был разработан метод *обобщённого портрета*, основанный на построении *оптимальной разделяющей гиперплоскости* [1]. 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 1.3.2, 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) [11]. В настоящее время он считается одним из лучших методов классификации. Его современное изложение можно найти в [9, 24].

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.

1.3.1 Линейно разделимая выборка

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

Будем строить линейный пороговый классификатор:

$$a(x) = \operatorname{sign}\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = \operatorname{sign}\big(\langle w, x\rangle - w_0\big), \tag{1.12}$$

где $x = (x^1, \ldots, x^n)$ — признаковое описание объекта x; вектор $w = (w^1, \ldots, w^n) \in \mathbb{R}^n$ и скалярный порог $w_0 \in \mathbb{R}$ являются параметрами алгоритма.

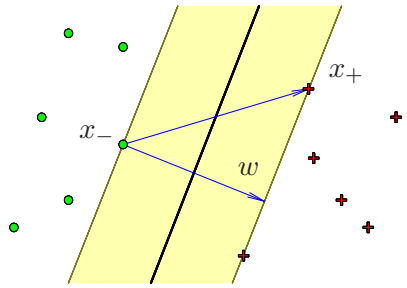


Рис. 6. Линейно разделимая выборка. Обучающие объекты x_- и x_+ находятся на границе разделяющей полосы. Вектор нормали w к разделяющей гиперплоскости определяет ширину полосы.

Уравнение $\langle w, x\rangle = w_0$ описывает гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^n$.

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

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

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 [7, 22, 26].

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

$$\min_{i=1,\ldots,\ell} y_i\big(\langle w, x_i\rangle - w_0\big) = 1. \tag{1.13}$$

Множество точек $\big\{x\colon\ -1 \leqslant \langle w, x\rangle - w_0 \leqslant 1\big\}$ описывает полосу, разделяющую классы, см. Рис. 6. 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 (1.13). В каждом из классов имеется хотя бы один такой объект, в противном случае разделяющую полосу можно было бы ещё немного расширить и нарушался бы принцип максимального зазора.

Ширина разделяющей полосы. Чтобы разделяющая гиперплоскость как можно дальше отстояла от точек выборки, ширина полосы должна быть максимальной. Пусть x_- и x_+ — два обучающих объекта классов -1 и $+1$ соответственно, лежащие на границе полосы. Тогда ширина полосы есть

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

Ширина полосы максимальна, когда норма вектора w минимальна.

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:

$$\begin{cases} \langle w, w\rangle \to \min; \\ y_i\big(\langle w, x_i\rangle - w_0\big) \geqslant 1, \quad i = 1, \ldots, \ell. \end{cases} \tag{1.14}$$

На практике линейно разделимые классы встречаются довольно редко. Поэтому постановку задачи (1.14) необходимо модифицировать так, чтобы система ограничений была совместна в любой ситуации.

1.3.2 Линейно неразделимая выборка

Чтобы обобщить постановку задачи на случай линейно неразделимой выборки, позволим алгоритму допускать ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше. Введём дополнительные переменные $\xi_i \geqslant 0$, характеризующие величину ошибки на объектах x_i, $i = 1, \ldots, \ell$. Ослабим в (1.14) ограничения-неравенства и одновременно введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell}\xi_i \to \min\limits_{w, w_0, \xi}; \\ y_i\big(\langle w, x_i\rangle - w_0\big) \geqslant 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \xi_i \geqslant 0, \quad i = 1, \ldots, \ell. \end{cases} \tag{1.15}$$

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

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

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

Алгоритм (1.12) 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.

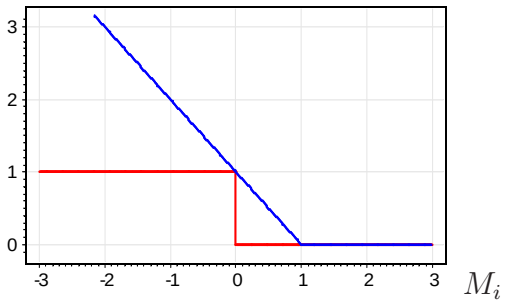


Рис. 7. Кусочно-линейная аппроксимация пороговой функции потерь: $[M_i < 0] \leqslant (1-M_i)_+$.

Согласно (1.15) ошибка ξ_i выражается через отступ M_i. Действительно, из ограничений-неравенств следует, что $\xi_i \geqslant 0$ и $\xi_i \geqslant 1-M_i$. 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 (1.15) оказывается эквивалентной безусловной минимизации функционала Q, не зависящего от переменных ξ_i:

$$Q(w, w_0) = \sum_{i=1}^{\ell} \big(1 - M_i(w, w_0)\big)_+ + \frac{1}{2C}\|w\|^2 \;\to\; \min_{w,w_0}. \tag{1.16}$$

В силу неравенства $[M_i < 0] \leqslant (1 - M_i)_+$, рис. 7, функционал (1.16) 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 $[M_i < 0]$ кусочно-линейной верхней оценкой $(1 - M_i)_+$ делает функцию потерь чувствительной к величине ошибки и 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 [5].

Двойственная задача. Запишем функцию Лагранжа задачи (1.15):

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

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 (1.15) эквивалентна двойственной задаче поиска седловой точки функции Лагранжа:

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

В последних двух строках записаны *условия дополняющей нежёсткости*.

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

$$\frac{\partial \mathscr{L}}{\partial w} = w - \sum_{i=1}^{\ell} \lambda_i y_i x_i = 0 \qquad \Longrightarrow \qquad w = \sum_{i=1}^{\ell} \lambda_i y_i x_i; \tag{1.17}$$

$$\frac{\partial \mathscr{L}}{\partial w_0} = -\sum_{i=1}^{\ell} \lambda_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^{\ell} \lambda_i y_i = 0; \tag{1.18}$$

$$\frac{\partial \mathscr{L}}{\partial \xi_i} = -\lambda_i - \eta_i + C = 0 \qquad \Longrightarrow \qquad \eta_i + \lambda_i = C, \quad i = 1, \ldots, \ell. \tag{1.19}$$

Из (1.17) следует, что искомый вектор весов w является линейной комбинацией векторов обучающей выборки x_i, причём только тех, для которых $\lambda_i > 0$.

Опр. 1.3. *Если $\lambda_i > 0$, то объект обучающей выборки x_i называется опорным вектором (support vector).*

Из третьего соотношения (1.19) и неравенства $\eta_i \geqslant 0$ следует $0 \leqslant \lambda_i \leqslant C$. 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. $\lambda_i = 0$; $\eta_i = C$; $\xi_i = 0$; $M_i \geqslant 1$.

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

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell}\lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell}\lambda_i\lambda_j y_i y_j \langle x_i, x_j\rangle \;\to\; \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell}\lambda_i y_i = 0. \end{cases} \tag{1.20}$$

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 (1.17). Для определения порога w_0 достаточно взять произвольный опорный граничный вектор x_i и выразить w_0 из равенства $w_0 = \langle w, x_i\rangle - y_i$. На практике для повышения численной устойчивости рекомендуется брать медиану множества значений w_0, вычисленных по всем граничным опорным векторам:

$$w_0 = \operatorname{med}\big\{\langle w, x_i\rangle - y_i \colon \lambda_i > 0,\ M_i = 1,\ i = 1, \ldots, \ell\big\}. \tag{1.21}$$

В итоге алгоритм классификации представляется в следующем виде:

$$a(x) = \operatorname{sign}\left(\sum_{i=1}^{\ell}\lambda_i y_i \langle x_i, x\rangle - w_0\right). \tag{1.22}$$

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 убрать из обучающей 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. В определённом смысле это недостаток SVM, поскольку нарушителями могут оказаться шумовые выбросы — ошибочные наблюдения, по каким-то причинам попавшие в обучающую выборку. Таким образом, классический SVM не является робастным (устойчивым к шуму) классификатором.

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.

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.

1.3.3 Ядра и спрямляющие пространства

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

Если предположить, что признаковыми описаниями объектов являются векторы $\psi(x_i)$, а не векторы x_i, то построение SVM проводится точно так же, как и ранее. Единственное отличие состоит в том, что скалярное произведение $\langle x, x'\rangle$ в пространстве X всюду заменяется скалярным произведением $\langle \psi(x), \psi(x')\rangle$ в пространстве H. Отсюда вытекает естественное требование: пространство H должно быть наделено скалярным произведением, в частности, подойдёт любое евклидово, а в общем случае и гильбертово, пространство.

Опр. 1.4. *Функция $K\colon X \times X \to \mathbb{R}$ называется ядром (kernel function), если она представима в виде $K(x,x') = \langle \psi(x), \psi(x')\rangle$ при некотором отображении $\psi\colon X \to H$, где H — пространство со скалярным произведением.*

Постановка двойственной задачи (1.20), и сам алгоритм классификации (1.22) зависят только от скалярных произведений объектов, но не от самих признаковых описаний. Это означает, что скалярное произведение $\langle x, x'\rangle$ можно формально заменить ядром $K(x,x')$. Поскольку ядро в общем случае нелинейно, такая замена приводит к существенному расширению множества реализуемых алгоритмов $a\colon X \to Y$.

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 1.3 (Мерсер, 1909 [17]). *Функция $K(x,x')$ является ядром тогда и только тогда, когда она симметрична, $K(x,x') = K(x',x)$, и неотрицательно определена: $\int_X \int_X K(x,x')g(x)g(x')dxdx' \geqslant 0$ для любой функции $g\colon X \to \mathbb{R}$.*

Существует эквивалентное определение неотрицательной определённости.

Опр. 1.5. *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: $z^\intercal K z \geqslant 0$ для любого $z \in \mathbb{R}^p$.*

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 [3, 4].

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 $\psi : X \to \mathbb{R}$ произведение $K(x, x') = \psi(x)\psi(x')$ является ядром.
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 $\varphi : X \to X$ и произвольного ядра K_0 является ядром: $K(x, x') = K_0(\varphi(x), \varphi(x'))$.
7. Если $s\colon X \times X \to \mathbb{R}$ — 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 является ядром.
10. Композиция произвольного ядра K_0 и произвольной функции $f\colon \mathbb{R} \to \mathbb{R}$, представимой в виде сходящегося степенного ряда с неотрицательными коэффициентами $K(x, x') = f\big(K_0(x, x')\big)$, является ядром. В частности, функции $f(z) = e^z$ и $f(z) = \frac{1}{1-z}$ от ядра являются ядрами.

Примеры ядер. Существует несколько «стандартных» 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 и плодотворных изобретений в машинном обучении. С другой стороны,

до сих пор не найдено эффективного общего подхода к их подбору в конкретных задачах.

Пример 1.2. Возьмём $X = \mathbb{R}^2$ и рассмотрим ядро $K(u,v) = \langle u,v\rangle^2$, где $u = (u_1,u_2)$, $v = (v_1,v_2)$, и попробуем понять, какое спрямляющее пространство и преобразование ψ ему соответствуют. Разложим квадрат скалярного произведения:

$$\begin{aligned} K(u,v) = \langle u,v\rangle^2 &= \langle (u_1,u_2),(v_1,v_2)\rangle^2 = \\ &= (u_1v_1 + u_2v_2)^2 = u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1u_2v_2 = \\ &= \Big\langle (u_1^2, u_2^2, \sqrt{2}u_1u_2), (v_1^2, v_2^2, \sqrt{2}v_1v_2)\Big\rangle. \end{aligned}$$

Ядро K представляется в виде скалярного произведения в пространстве $H = \mathbb{R}^3$. Преобразование $\psi\colon \mathbb{R}^2 \to \mathbb{R}^3$ имеет вид $\psi\colon (u_1,u_2) \mapsto (u_1^2, u_2^2, \sqrt{2}u_1u_2)$. 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 1.3. Усложним ситуацию. Пусть теперь $X = \mathbb{R}^n$,

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

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 1.4. Если $X = \mathbb{R}^n$,

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

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.

Связь SVM с двухслойными нейронными сетями. Рассмотрим структуру алгоритма $a(x)$ после замены в (1.22) 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

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

Если $X = \mathbb{R}^n$, то алгоритм $a(x)$ можно рассматривать как двухслойную нейронную сеть, имеющую n входных нейронов и h нейронов в скрытом слое, см. Рис. 8. Сеть, настроенная методом SVM, имеет несколько замечательных особенностей.

Во-первых, число нейронов скрытого слоя определяется автоматически.

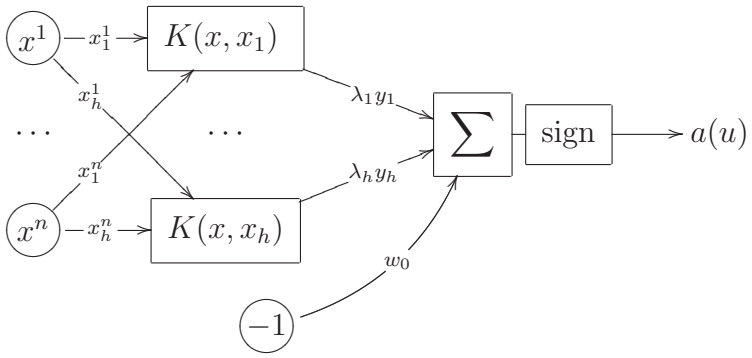


Рис. 8. Машина опорных векторов (SVM) как двухслойная нейросеть.

Во-вторых, векторы весов у нейронов скрытого слоя совпадают с признаковыми описаниями опорных объектов.

В-третьих, проясняется смысл двойственных переменных: λ_i — это вес, выражающий степень важности ядра $K(x_i, x)$.

Пример 1.5. Классическая нейронная сеть с сигмоидными функциями активации получится, если в качестве ядра взять функцию

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

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 $k_1 < 0$ [10]. Однако это не препятствует её успешному практическому применению. Вместо гиперболического тангенса $\operatorname{th} z$ 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 (1.20) 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 $0 \leqslant \lambda_i \leqslant C$ 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 1.6. Нейронная сеть с *радиальными базисными функциями* (radial basis functions, RBF) получится, если взять гауссовское ядро

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

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 -1 — с отрицательными. Выходной нейрон производит голосование, сравнивая суммарные близости распознаваемого объекта x к обоим классам.

В разделе ?? 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 «размытыми» границами классов.

Преимущества SVM перед методом стохастического градиента.

- Вместо многоэкстремальной задачи решается задача квадратичного программирования, имеющая единственное решение. Методы оптимизации в этом случае существенно более эффективны.
- Автоматически определяется число нейронов скрытого слоя. Оно равно числу опорных векторов.
- Принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы между классами, следовательно, к более уверенной классификации. Градиентные нейросетевые методы выбирают положение разделяющей гиперплоскости произвольным образом, «как придётся».

Недостатки SVM.

- Метод опорных векторов неустойчив к шуму в исходных данных. Если обучающая выборка содержит выбросы, они будут существенным образом учтены при построении разделяющей гиперплоскости. Этого недостатка лишён *метод релевантных векторов* (relevance vector machine, RVM), см. 1.4.2.
- До сих пор не разработаны общие методы построения спрямляющих пространств или ядер, наиболее подходящих для конкретной задачи. Построение адекватного ядра является искусством и, как правило, опирается на априорные знания о предметной области. На практике «вполне разумные» функции $K(x, x')$, выведенные из содержательных соображений, далеко не всегда оказываются положительно определёнными.
- В общем случае, когда линейная разделимость не гарантируется, приходится подбирать управляющий параметр алгоритма C.

1.3.4 Алгоритм настройки SVM

Двойственная задача (1.20) является задачей квадратичного программирования. Общие методы решения таких задач известны, но довольно трудоёмки, как в смысле реализации, так и по времени выполнения. Поэтому для обучения SVM применяются алгоритмы, учитывающие специфические особенности SVM. Специфика заключается в том, что число опорных векторов h, как правило, невелико, $h \ll \ell$, и эти векторы находятся поблизости от границы классов. Именно эти особенности

и позволяют ускорить поиск опорных объектов. Специализированные алгоритмы настройки SVM успешно справляются с выборками из десятков тысяч объектов [19, 20].

Здесь мы рассмотрим не самый известный, но также довольно эффективный алгоритм — *последовательный метод активных ограничений* (incremental active set method, INCAS), предложенный в [12, 21].

Перепишем двойственную задачу (1.20) в матричных обозначениях. Введём матрицу $Q = \big(y_i y_j K(x_i, x_j)\big)_{i=1,\ell}^{j=1,\ell}$ размера $\ell \times \ell$ и три вектор-столбца длины ℓ: вектор ответов $y = (y_i)_{i=1,\ell}$, вектор двойственных переменных $\lambda = (\lambda_i)_{i=1,\ell}$ и вектор единиц $e = (1)_{i=1,\ell}$. Тогда задачу (1.20) можно переписать в виде

$$\begin{cases} \frac{1}{2}\lambda^\intercal Q\lambda - e^\intercal\lambda \to \min\limits_{\lambda}; \\ y^\intercal\lambda = 0; \\ 0 \leqslant \lambda \leqslant Ce. \end{cases} \tag{1.23}$$

Допустим, что решение λ ещё не известно, но зато известно разбиение множества объектов на три непересекающихся подмножества $\{1, \ldots, \ell\} = I_O \cup I_C \cup I_S$:

$$\begin{aligned} I_O &= \{i\colon \lambda_i = 0\} && \text{— периферийные объекты, } M_i > 1; \\ I_S &= \{i\colon 0 < \lambda_i < C\} && \text{— опорные объекты, } M_i = 1; \\ I_C &= \{i\colon \lambda_i = C\} && \text{— объекты-нарушители, } M_i < 1; \end{aligned}$$

где $M_i = y_i\big(\langle w, x_i\rangle - w_0\big)$ — отступ объекта x_i от границы классов.

Ограничения-неравенства $0 \leqslant \lambda_i \leqslant C$ становятся *активными*, то есть обращаются в равенства, на периферийных объектах ($i \in I_O$, $\lambda_i = 0$) и объектах-нарушителях ($i \in I_C$, $\lambda_i = C$). Соответствующие значения λ_i можно подставить обратно в (1.23) и получить оптимизационную задачу, зависящую только от части переменных λ_i, $i \in I_S$. Чтобы выписать эту задачу в матричном виде, введём трёхблочные обозначения для матрицы Q и векторов y, e, λ:

$$Q = \begin{pmatrix} Q_{SS} & Q_{SO} & Q_{SC} \\ Q_{OS} & Q_{OO} & Q_{OC} \\ Q_{CS} & Q_{CO} & Q_{CC} \end{pmatrix}; \quad y = \begin{pmatrix} y_S \\ y_O \\ y_C \end{pmatrix}; \quad e = \begin{pmatrix} e_S \\ e_O \\ e_C \end{pmatrix}; \quad \lambda = \begin{pmatrix} \lambda_S \\ 0 \\ Ce_C \end{pmatrix}.$$

В этих обозначениях задача (1.23) принимает вид

$$\begin{cases} \frac{1}{2}\lambda_S^\intercal Q_{SS}\lambda_S + Ce_C^\intercal Q_{CS}\lambda_S - e_S^\intercal\lambda_S \to \min\limits_{\lambda_S}; \\ y_S^\intercal\lambda_S + Ce_C^\intercal y_C = 0; \end{cases} \tag{1.24}$$

Если не обращать внимания на ограничения-неравенства, то это задача минимизации квадратичного функционала от $h = |I_S|$ переменных с одним линейным ограничением типа равенства. Она легко решается стандартными методами линейной алгебры и сводится, фактически, к обращению симметричной положительно определённой матрицы Q_{SS} размера $h \times h$.

Решение этой задачи даёт весь вектор λ, что позволяет вычислить параметры алгоритма w и w_0 по формулам (1.17) и (1.21). Теперь можно классифицировать объекты выборки x_i, вычислить отступы M_i, и проверить, правильно ли множество объектов было разбито на подмножества $I_O \cup I_C \cup I_S$. Если условия Куна-Таккера

Алгоритм 1.2. Обучение SVM: последовательный метод активных ограничений

Вход:
X^ℓ — обучающая выборка;
C — параметр двойственной задачи;
Выход:
параметры линейного классификатора w, w_0;

1: начальное приближение:
$I_S =$ две ближайшие точки из разных классов;
$I_O =$ все остальные точки;
$I_C = \varnothing$;
2: **повторять**
3: **повторять**
4: решение оптимизационной задачи относительно λ_S:
$$\begin{cases} \frac{1}{2}\lambda_S^\intercal Q_{SS}\lambda_S + Ce_C^\intercal Q_{CS}\lambda_S - e_S^\intercal\lambda_S \to \min\limits_{\lambda_S}; \\ y_S^\intercal\lambda_S + Ce_C^\intercal y_C = 0; \end{cases}$$
5: **если** $|I_S| > 2$ и $\exists i$: $i \in I_S$ и $\lambda_i \leqslant 0$ **то** перевести i в I_O;
6: **если** $|I_S| > 2$ и $\exists i$: $i \in I_S$ и $\lambda_i \geqslant C$ **то** перевести i в I_C;
7: **пока** существует $i \in I_S$, который необходимо перевести в I_O или в I_C;
8: вычислить параметры алгоритма w и w_0 по формулам (1.17) и (1.21);
9: вычислить отступы M_i, $i \in I_O \cup I_C$;
10: **если** $\exists i$: $i \in I_O$ и $M_i \leqslant 1$ **то** перевести i в I_S;
11: **если** $\exists i$: $i \in I_C$ и $M_i \geqslant 1$ **то** перевести i в I_S;
12: **пока** существует $i \in I_O \cup I_C$, который необходимо перевести в I_S;

не нарушатся ни на одном объекте, значит, решение задачи (1.23) найдено. Если же обнаружится хотя бы одно противоречие, то соответствующий объект должен быть переведён из одного подмножества в другое.

Всего возможны четыре типа противоречий:

1. Если $i \in I_S$ и $\lambda_i \leqslant 0$, то объект x_i переводится из I_S в I_O.
2. Если $i \in I_S$ и $\lambda_i \geqslant C$, то объект x_i переводится из I_S в I_C.
3. Если $i \in I_O$ и $M_i \leqslant 1$, то объект x_i переводится из I_O в I_S.
4. Если $i \in I_C$ и $M_i \geqslant 1$, то объект x_i переводится из I_C в I_S.

После каждой модификации множеств I_O, I_C, I_S задача (1.24) решается заново. Итерационный процесс перевода объектов из одного множества в другое продолжается до тех пор, пока все объекты не будут удовлетворять условиям Куна-Таккера.

Описанный процесс является частным случаем *метода активных ограничений* (active sets method), который применяется для решения произвольных задач математического программирования с ограничениями-неравенствами. В линейном программировании он эквивалентен симплекс-методу. Сходимость данного метода в общем случае не гарантируется. Для предотвращения зацикливаний применяется рандомизированный выбор объекта x_i. Неплохая эвристика, способствующая увеличению скорости сходимости, заключается в том, чтобы с большей вероятностью выбирать те объекты x_i, для которых условия Куна-Таккера нарушаются сильнее.

Начальное приближение. Количество итераций существенно зависит от того, насколько удачным окажется начальное приближение. На практике применяются различные приёмы, чтобы сразу поточнее «угадать» множество опорных векторов I_S.

Приём 1. Выбирается произвольная точка выборки, и находится ближайшая к ней точка другого класса. Для неё, в свою очередь, находится ближайшая точка в первом классе, и т. д. Этот итерационный процесс, как правило, сходится очень быстро к некоторой паре пограничных точек. В Алгоритме 1.2 эта пара принимается за начальное приближение I_S, но можно и продолжить процесс, построив несколько или даже все такие пары.

Приём 2. Строится несколько достаточно грубых линейных классификаторов. Для этого можно использовать однослойные персептроны, проводя небольшое число итераций методом стохастического градиента. Затем для каждой линейной разделяющей поверхности находится несколько ближайших к ней точек, которые и принимаются за начальное приближение I_S.

Эффективность. Высокая эффективность Алгоритма 1.2 вытекает из двух фактов.

Во-первых, оптимизационная задача, решаемая на шаге 4, зависит только от матриц Q_{SS} и Q_{CS}. Значит, вычислять скалярные произведения $K(x, x')$ приходится только для пар объектов типа «опорный–опорный» и «опорный–нарушитель».

Во-вторых, множество I_S на каждом шаге изменяется только на один элемент. Это позволяет выполнять пересчёт обратной матрицы Q_{SS}^{-1} за $O(h^2)$ операций, тогда как обычное обращение потребовало бы $O(h^3)$ операций.

Преимущества метода INCAS.

- Метод позволяет решать задачи, в которых нет линейной разделимости, в том числе задачи с шумовыми выбросами.
- Метод особенно эффективен, когда число опорных векторов $h = |I_S|$ невелико.
- Метод хорошо приспособлен для решения таких задач, в которых обучающие объекты поступают по одному в режиме реального времени. Добавление нового объекта реализуется практически так же, как перевод объекта из одного подмножества в другое, и требует порядка $O(h^2)$ операций.

Недостатки метода INCAS.

- Метод становится неэффективен, когда число опорных векторов велико. В то же время, это означает, что либо выборка существенно неразделима, либо ядро $K(x, x')$ выбрано неудачно. В обоих случаях такую задачу, скорее всего, нет смысла решать — надо менять ядро или саму постановку.

§1.4 Обобщения

В данном параграфе рассматривается ряд обобщений описанных выше методов. Будет дана их общая вероятностная интерпретация, показана роль различных функций потерь и различных видов регуляризации. Несмотря на видимые различия,

рассмотренные выше методы классификации являются частными случаями одной общей конструкции. Её понимание даёт ключ к решению новых нестандартных задач обучения по прецедентам, а также позволяет судить о применимости тех или иных методов в конкретных практических задачах.

1.4.1 Непрерывные аппроксимации пороговой функции потерь

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*. В общем случае она не обязана быть линейной 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 $X^\ell = (x_i, y_i)_{i=1}^\ell$.

Принципы максимума отступов и минимума эмпирического риска. Следующее определение является обобщением определения 1.2.

Опр. 1.6. *Величина $M_i(w) = y_i f(x_i, w)$ называется отступом (margin) объекта x_i относительно алгоритма классификации $a(x, w) = \operatorname{sign} f(x, w)$.*

Алгоритм $a(x, w)$ допускает ошибку на объекте x_i тогда и только тогда, когда его отступ $M_i(w)$ отрицателен. Чем больше отступ $M_i(w)$, тем надёжнее объект x_i относится к правильному классу y_i. Поэтому в задачах классификации имеет смысл определять функцию потерь вида $\mathscr{L}\big(a(x_i, w), y_i\big) = g\big(M_i(w)\big)$, где $g(M)$ — монотонно невозрастающая функция отступа, и для настройки вектора параметров w по обучающей выборке X^ℓ минимизировать суммарные потери:

$$\widetilde{Q}(w) = \sum_{i=1}^{\ell} g\big(M_i(w)\big) \to \min_w .$$

Если $[M < 0] \leqslant g(M)$, то минимизацию суммарных потерь $\widetilde{Q}(w)$ можно рассматривать как приближённый метод минимизации эмпирического риска — числа ошибок на обучающей выборке $Q(w)$:

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \quad \leqslant \quad \widetilde{Q}(w) = \sum_{i=1}^{\ell} g\big(M_i(w)\big) \to \min_w . \tag{1.25}$$

Некоторые применяемые на практике функции потерь показаны на рис. 9. Квадратичная функция потерь не является монотонной, тем не менее, она соответствует линейному дискриминанту Фишера, который, хоть и редко, но применяется на практике. Кусочно-линейная функция потерь соответствует методу опорных векторов (SVM), сигмоидная используется в нейронных сетях, логарифмическая — 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.

Таким образом, исходя из чисто эвристического принципа максимизации отступов, невозможно ответить на ряд вопросов: какие ещё функции $g(M)$ допустимы, какие из них более предпочтительны, и в каких ситуациях?

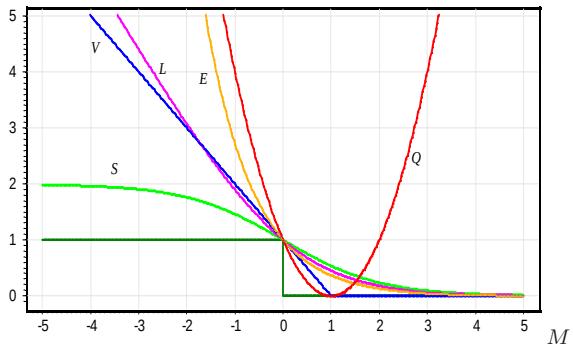


Рис. 9. Непрерывные аппроксимации пороговой функции потерь $[M < 0]$.

$Q(M) = (1 - M)^2$ — квадратичная;
$V(M) = (1 - M)_+$ — кусочно-линейная;
$S(M) = 2(1 + e^M)^{-1}$ — сигмоидная;
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая;
$E(M) = e^{-M}$ — экспоненциальная.

Принцип максимума правдоподобия позволяет по-другому взглянуть на задачу настройки параметров w. Будем исходить из того, что 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:

$$L(w, X^\ell) = \ln p(X^\ell|w) = \sum_{i=1}^{\ell} \ln p(x_i, y_i|w) \to \max_w. \tag{1.26}$$

Сопоставляя (1.26) с правой частью (1.25), легко заключить, что эти две задачи эквивалентны, если положить

$$-\ln p(x_i, y_i|w) = g\big(y_i f(x_i, w)\big).$$

Зная функциональный вид модели плотности $p(x, y|w)$, можно выписать вид разделяющей поверхности f и функцию g. Мы это уже не раз проделали: в **??** — для квадратичного и линейного дискриминанта; в §1.2 — для логистической регрессии. С другой стороны, задавая вид разделяющей поверхности и функцию потерь g, казалось бы, из чисто эвристических соображений, мы тем самым неявно принимаем некоторую вероятностную модель данных.

1.4.2 Принцип совместного правдоподобия данных и модели

Принцип минимума регуляризованного эмпирического риска. Вспомним, что в линейных алгоритмах классификации может возникать проблема мультиколлинеарности, которая приводит к неустойчивости вектора коэффициентов w и переобучению. Чтобы предотвратить эти нежелательные явления, в функционал вводится

аддитивный штраф за увеличение нормы вектора коэффициентов. В итоге приходим к задаче минимизации *регуляризованного эмпирического риска* $\widetilde{Q}_\tau$:

$$Q(w) \leqslant \widetilde{Q}(w) \leqslant \widetilde{Q}_\tau(w) = \sum_{i=1}^{\ell} g\big(M_i(w)\big) + \frac{\tau}{2}\|w\|^2 \to \min_w,$$

где константа τ называется *параметром регуляризации*. Этот приём используется для регуляризации ковариационной матрицы в линейном дискриминанте Фишера и в гребневой регрессии. В нейронных сетях и персептронах он приводит к методу сокращения весов (weight decay). На практике параметр регуляризации τ выбирается, как правило, по скользящему контролю.

Остаются вопросы: откуда берётся именно такой регуляризатор, не связан ли он исключительно с линейностью классификатора, и какие регуляризаторы могут возникать в других, нелинейных, моделях классификации?

Принцип максимума совместного правдоподобия данных и модели. Допустим, что наряду с параметрической моделью плотности распределения $p(x, y|w)$ имеется ещё априорная информация о плотности распределения параметров модели $p(w)$. Интерпретация этой плотности может быть следующей. Рассмотрим всевозможные задачи, которые встречаются на практике. Для каждой задачи существует хотя бы одно значение параметра w, при котором модель $p(x, y|w)$ описывает эту задачу наилучшим образом. Если полагать, что существует вероятностное распределение на множестве задач, то оно индуцирует некоторое распределение и в пространстве параметров модели $p(x, y|w)$. Проще говоря, некоторые модели чаще находят практическое применение, чем другие, и гипотезу о практической применимости моделей предлагается формулировать в виде функции $p(w)$ — *априорного распределения в пространстве параметров модели*.

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. Чуть позже мы увидим, как это работает на практике.

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*:

$$L(w, X^\ell) = \ln p(X^\ell, w;\gamma) = \sum_{i=1}^{\ell} \ln p(x_i, y_i|w) + \ln p(w;\gamma) \to \max_w. \tag{1.27}$$

Функционал $L(w, X^\ell)$ распадается на два слагаемых. Первое в точности равно логарифму правдоподобия (1.26), зависит только от данных и не зависит от значения гиперпараметра. Второе слагаемое, наоборот, не зависит от данных, и представляет собой логарифм априорного распределения параметров модели.

Сопоставляя (1.27) с принципом минимума регуляризованного эмпирического риска, нетрудно придти к выводу, что второе слагаемое в (1.27) имеет смысл аддитивного регуляризатора.

Пример 1.7. Пусть вектор $w \in \mathbb{R}^n$ имеет гауссовское (нормальное) распределение, все его компоненты независимы и имеют равные дисперсии σ. Будем считать σ гиперпараметром. Логарифмируя, получаем всё тот же квадратичный регуляризатор:

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

Отсюда становится понятен вероятностный смысл параметра регуляризации: он обратно пропорционален дисперсии вектора параметров, $\tau = \frac{1}{\sigma}$. Увеличивая параметр τ, мы уменьшаем дисперсию вектора параметров, следовательно, запрещаем коэффициентам w_j принимать слишком большие значения.

Метод опорных векторов (SVM). Попробуем разобраться, какой вероятностной модели соответствует метод опорных векторов. Напомним, что в SVM строится линейный классификатор вида $a(x;w,w_0) = \text{sign}\big(\langle w,x\rangle - w_0\big)$, а параметры $w \in \mathbb{R}^n$ и $w_0 \in \mathbb{R}$ обучаются по выборке X^ℓ путём минимизации функционала (1.16) с кусочно-линейной функцией потерь и квадратичным регуляризатором:

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

где $M_i(w,w_0) = \big(\langle w,x_i\rangle - w_0\big)y_i$ — отступы. Данная задача соответствует принципу максимума совместного правдоподобия (1.27), если принять модель плотности

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

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.

Сделанное наблюдение показывает, какие элементы постановки задачи за какие полезные свойства SVM отвечают. В частности, *свойство разреженности* связано с тем, что функция потерь недифференцируема, а свойство максимальности разделяющей полосы связано с квадратичным регуляризатором. Можно изобретать различные модификации метода, меняя вид регуляризатора, но свойство разреженности, присущее SVM, при этом будет сохранятся.

Лапласовский регуляризатор соответствует априорному распределению

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

называемому *распределением Лапласа*. Это распределение имеет более острый пик и более тяжёлые «хвосты», по сравнению с гауссовским распределением. Дисперсия распределения Лапласа равна $2C^2$. Наиболее интересное свойство этого регуляризатора заключается в том, что он приводит к отбору признаков, причём не только для линейных алгоритмов, но и для более широкого класса моделей. Покажем, что это происходит из-за негладкости регуляризатора Лапласа.

Запишем оптимизационную задачу настройки вектора параметров w:

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

где $\mathscr{L}_i(w) = \mathscr{L}(a(x_i, w), y_i)$ — некоторая дифференцируемая функция потерь.

Сделаем замену переменных. Каждой переменной w_j поставим в соответствие две новые неотрицательные переменные $u_j = \frac{1}{2}\big(|w_j| + w_j\big)$ и $v_j = \frac{1}{2}\big(|w_j| - w_j\big)$. Тогда $w_j = u_j - v_j$ и $|w_j| = u_j + v_j$. В новых переменных функционал становится гладким, но добавляются ограничения-неравенства:

$$\begin{aligned} Q(u, v) = &\sum_{i=1}^{\ell} \mathscr{L}_i(u - v) + \frac{1}{C}\sum_{j=1}^{n}(u_j + v_j) \to \min_{u,v} \\ &u_j \geqslant 0, \quad v_j \geqslant 0, \quad j = 1, \ldots, n. \end{aligned}$$

Для решения данной задачи можно применить метод множителей Лагранжа. Решение обладает следующим очевидным свойством: для любого j хотя бы одно из ограничений $u_j \geqslant 0$ и $v_j \geqslant 0$ является активным, то есть обращается в равенство. Возможны и такие ситуации, когда оба ограничения активны, $u_j = v_j = 0$; тогда параметр w_j также обнуляется. В линейных моделях это означает, что значения j-го признака игнорируются, и его можно исключить из модели. По мере уменьшения гиперпараметра C число нулевых параметров w_j возрастает.

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 регуляризации $\tau = \frac{1}{C}$ позволяет регулировать *селективность* метода. Чем больше τ, и, соответственно, меньше дисперсия распределения Лапласа, тем большее коэффициентов w_j окажутся нулевыми.

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

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

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

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

где $\mathscr{L}_i(w) = \mathscr{L}(a(x_i, w), y_i)$ — некоторая заданная функция потерь.

Результатом такой модификации является более мягкий отбор признаков, чем в случае Лапласовского регуляризатора. Если $C_j \to 0$, то параметр w_j также 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.

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 [25, 8].

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

$$w = \sum_{i=1}^{\ell} \lambda_i y_i x_i, \tag{1.28}$$

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 (1.28), 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 параметры 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 предыдущем случае, только теперь параметры априорного распределения связываются с объектами, а не с признаками.

1.4.3 Кривая ошибок и выбор порога в линейном классификаторе

Рассмотрим задачу классификации на два класса, $Y = \{-1, +1\}$, и модель алгоритмов $a(x, w) = \operatorname{sign}\big(f(x, w) - w_0\big)$, где $w_0 \in \mathbb{R}$ — аддитивный параметр дискриминантной функции (в терминах персептронов — порог активации).

Согласно Теореме 1.2 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 «+1» и «−1» соответственно.

На практике отношение потерь может многократно пересматриваться. Поэтому вводится специальная характеристика — ROC-кривая, которая показывает, что происходит с числом ошибок обоих типов, если изменяется отношение потерь.

Термин *операционная характеристика приёмника* (receiver operating characteristic, ROC curve) пришёл из теории обработки сигналов. Эту характеристику впервые ввели во время 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):

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

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

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

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

В случае логистической регрессии каждая точка ROC-кривой соответствует определённому значению параметра w_0. При этом 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 1.3 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 1.3.

1.4.4 Вероятностный выход, скоринг, и оценивание рисков

В случае бинарных признаков, $X = \{0, 1\}^n$, 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.

Алгоритм 1.3. Эффективный алгоритм построения ROC-кривой

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

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

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 задачу.

Во многих прикладных задачах исходные данные содержат разнотипные признаки. Если признак не бинарный, то его *бинаризуют*, разбивая множество его значений на подмножества, например, с помощью методов, описанных в разделе **??**. В результате один небинарный признак заменяется несколькими бинарными.

После бинаризации 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.

Пример 1.8. В задаче *кредитного скоринга* (credit scoring) признаковое описание заёмщика (физического лица) состоит из его ответов на вопросы анкеты. Среди признаков встречаются бинарные (пол, согласие дать телефон, наличие задолженностей), номинальные (место проживания, профессия, работодатель), порядковые (образование, должность) и количественные (сумма кредита, возраст, стаж, доход). Все они в конечном итоге приводятся к бинарному виду. Количество признаков при этом может существенно возрасти.

Резюме

1. *Однослойный персептрон* является простейшей моделью нервной клетки — нейрона. Персептрон с пороговой *функцией активации* представляет собой линейный классификатор. Персептроны с гладкими функциями активации способны решать задачи регрессии.

2. *Метод стохастического градиента* является стандартным подходом к обучению нейронных сетей. Он приводит к разным частным случаям, в зависимости от типа задачи, выбора функции потерь $\mathscr{L}$ и функции активации φ. Адаптивный линейный элемент ADALINE соответствует задаче регрессии при квадратичной $\mathscr{L}$ и линейной φ. Правило Хэбба соответствует задаче классификации на два класса $\{-1, +1\}$ при пороговых $\mathscr{L}$ и φ. Логистическая регрессия соответствует логарифмической $\mathscr{L}$ и сигмоидной φ.

3. *Логистическая регрессия* предназначена для решения задач классификации, однако оценивает она действительные переменные — апостериорные вероятности классов. При определённых (довольно сильных) вероятностных предположениях оказывается, что применение сигмоидной функции $\sigma(z) = \frac{1}{1+e^{-z}}$ к значению линейной дискриминантной функции даёт искомую апостериорную вероятность. Для настройки вектора коэффициентов применяется итерационный взвешенный МНК (метод IRLS), на каждом шаге которого, решается задача многомерной линейной регрессии.

4. *Метод опорных векторов* строит линейный классификатор, исходя из *принципа оптимальной разделяющей гиперплоскости*, согласно которому ширина зазора между выпуклыми оболочками классов должна быть как можно больше. Этот принцип эквивалентен минимизации регуляризованного эмпирического риска, если вместо пороговой функции потерь взять кусочно-линейную. Настройка весов по обучающей выборке сводится к решению задачи квадратичного программирования с ограничениями-неравенствами. Задача имеет единственное решение, которое зависит только от *опорных векторов* — объектов выборки, расположенных вдоль границы между классами. Более того, решение зависит не от самих опорных векторов, а только от их скалярных произведений, а сам классификатор зависит от скалярных произведений опорных векторов и классифицируемого вектора. Замена скалярного произведения произвольным *ядром* позволяет перейти в *спрямляющее пространство* и строить нелинейные разделяющие поверхности.

5. *Ядром* может быть произвольная симметричная неотрицательно определённая функция двух векторов. На практике бывает довольно трудно проверить, является ли заданная функция ядром. Гораздо проще строить ядра с помощью правил порождения. Вопрос о выборе ядра, оптимального для данной прикладной задачи, до сих пор остаётся открытой теоретической проблемой.

6. Известно несколько эффективных методов построения SVM. *Последовательный метод активных ограничений* INCAS основан на комбинаторном переборе различных способов разбиения множества объектов на опорные, периферийные

и нарушители. При каждом переходе от одного разбиения к другому делается попытка устранить наиболее сильные нарушения условий Куна-Таккера.

7. Оказывается, что все перечисленные выше методы построения линейных классификаторов могут быть получены как следствие принципа максимума совместного правдоподобия данных и модели, если фиксировать параметрическую модель плотности $p(x, y; w)$ и априорное распределение в пространстве параметров модели $p(w)$. Фиксация параметрической модели плотности эквивалентна заданию вида разделяющей поверхности и функции потерь. Фиксация априорного распределения моделей эквивалентна заданию регуляризатора. Существуют регуляризаторы, применение которых ведёт к отбору наиболее информативных признаков. В некоторых случаях такой регуляризатор имеет гиперпараметр селективности, с помощью которого можно управлять количеством отбираемых признаков.

8. *Кривая ошибок* или ROC-кривая позволяет наглядно отобразить зависимость числа ошибок I и II рода от значения порога в линейном классификаторе, в частности, в логистической регрессии. Оптимальное значение этого порога зависит только от соотношения цены ошибок I и II рода. Площадь под ROC-кривой (AUC) является инвариантной характеристикой качества классификатора, не зависящей от соотношения цены ошибок I и II рода.

Упражнения

Упр. 1.1. Доказать: формула 1.5 для весов w_j в однослойном персептроне является точной для адаптивного линейного элемента ADALINE (когда $\mathscr{L}(a, y) = (a - y)^2$, $\varphi(z) = z$), если признаки некоррелированы, $\langle f_j, f_k \rangle = 0$, $j \neq k$.

Упр. 1.2. Доказать: в задачах классификации с двумя классами $Y = \{0, 1\}$ персептронное правило (1.7) переходит в правило Хэбба (1.8), если изменить метку класса 0 на -1.

Упр. 1.3. Доказать, что в однослойном персептроне с линейной функцией активации при квадратичном функционале ошибки оптимальная величина темпа обучения η даётся формулой (1.9).

Список литературы

[1] *Вапник В. Н., Червоненкис А. Я.* Теория распознавания образов. — М.: Наука, 1974.

[2] *Головко В. А.* Нейронные сети: обучение, организация и применение. — М.: ИПРЖР, 2001.

[3] *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.

[4] *Мерков А. Б.* О статистическом обучении. — Лаборатория распознавания образов МЦНМО. — 2006.
http://www.recognition.mccme.ru/pub/RecognitionLab.html/slt.pdf.

[5] *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. — М.: Наука, 1986.

[6] *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.

[7] *Bartlett P., Shawe-Taylor J.* Generalization performance of support vector machines and other pattern classifiers // Advances in Kernel Methods. — MIT Press, Cambridge, USA, 1998.
http://citeseer.ist.psu.edu/bartlett98generalization.html.

[8] *Bishop C. M.* Pattern Recognition and Machine Learning. — Springer, Series: Information Science and Statistics, 2006. — P. 740.

[9] *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.

[10] *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.

[11] *Cortes C., Vapnik V.* Support-vector networks // *Machine Learning.* — 1995. — Vol. 20, no. 3. — Pp. 273–297.
http://citeseer.ist.psu.edu/cortes95supportvector.html.

[12] *Fine S., Scheinberg K.* INCAS: An incremental active set method for SVM: Tech. rep.: 2002.
http://citeseer.ist.psu.edu/fine02incas.html.

[13] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. — Springer, 2001.

[14] *Hebb D.* The organization of behavior. — New York: Wiley, 1949.

[15] *LeCun Y., Bottou L., Orr G. B., Muller K.-R.* Efficient BackProp // Neural Networks: tricks of the trade. — Springer, 1998.

[16] *McCulloch W. S., Pitts W.* A logical calculus of ideas immanent in nervous activity // *Bulletin of Mathematical Biophysics.* — 1943. — no. 5. — Pp. 115–133.

[17] *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.

[18] *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.

[19] *Osuna E., Freund R., Girosi F.* An improved training algorithm for support vector machines // Neural Networks for Signal Processing VII. IEEE Workshop. — 1997. — Pp. 276–285.
http://citeseer.ist.psu.edu/osuna97improved.html.

[20] *Platt J. C.* Fast training support vector machines using sequential minimal optimization // Advances in Kernel Methods / Ed. by B. Scholkopf, C. C. Burges, A. J. Smola. — MIT Press, 1999. — Pp. 185–208.

[21] *Scheinberg K.* An efficient implementation of an active set method for svms // *J. Mach. Learn. Res.* — 2006. — Vol. 7. — Pp. 2237–2257.

[22] *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.

[23] *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.

[24] *Smola A., Schoelkopf B.* A tutorial on support vector regression: Tech. Rep. NeuroCOLT2 NC2-TR-1998-030: 1998.
http://citeseer.ist.psu.edu/smola98tutorial.html.

[25] *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.

[26] *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.

[27] *Widrow B., Hoff M. E.* Adaptive switching circuits // IRE WESCON Convention Record. — DUNNO, 1960. — Pp. 96–104.