Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Институт фундаментального образования

О.В. ЛИМАНОВСКАЯ
Т.И. АЛФЕРЬЕВА

ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Министерство науки и высшего образования
Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

О. В. Лимановская, Т. И. Алферьева

ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся
по направлениям подготовки:
09.03.04 — Программная инженерия;
09.03.03 — Прикладная информатика;
02.04.02 — Фундаментальная информатика
и информационные технологии;
09.04.02 — Информационные системы и технологии

Екатеринбург
Издательство Уральского университета
2020

УДК 004.94(075.8)
ББК 32.973-018я73
Л58

Рецензенты:
канд. физ.-мат. наук А. П. Сергеев (завлабораторией физики и экологии Института промышленной экологии УрО РАН);
канд. техн. наук, доц. Т. Я. Ткаченко (заместитель завотделом цифровых образовательных технологий ФГБОУ ВО «Уральский государственный медицинский университет»)

Научный редактор — канд. техн. наук, доц. И. Н. Обабков

Лимановская, О. В.

Л58 Основы машинного обучения : учебное пособие / О. В. Лимановская, Т. И. Алферьева ; Мин-во науки и высш. образования РФ. — Екатеринбург : Изд-во Урал. ун-та, 2020. — 88 с.

ISBN 978-5-7996-3015-7

Изложены основы машинного обучения, а также история его появления. Даны определения основным понятиям: выборка, объекты выборки, параметры, функционал ошибки и прочее. Описаны основы градиентного спуска и его модификаций, основные алгоритмы обучения с учителем и обучения без учителя.

Рис. 30. Табл. 4.

УДК 004.94(075.8)
ББК 32.973-018я73

ISBN 978-5-7996-3015-7

Data Science — что это такое и зачем она нужна?

Что такое data science?

История вопроса

Человечество всегда мечтало приподнять занавес с грядущего и желало знать что будет. И если для сбора данных использовались точные методы (таблицы, архивы, летописи), то для предсказания все шло в ход — шаманы впадали в транс и общались с потусторонним миром, сообщая новости оттуда; пифии, будучи опять же в трансе, делали малосвязные предсказания, которые потом трактовались жрецами в нужном смысле; астрологи пытались применить околонаучный подход и рассчитывали гороскопы для мероприятий и людей. Многое из этого набора до сих пор живо используется, но этим прогнозы не обоснованы и к ним нет доверия у научного сообщества.

Сбор данных можно смело считать началом статистики. Первая статистическая информация — глиняные таблички шумерского царства (III–II тысячелетие до н. э.). В них содержалась экономическая информация — сделки, количество собранного урожая, налоги и пр.

В Римской республике, а затем и в империи, была развитая финансовая и налоговая система, которая требовала ведения точного учета и сбора данных по сделкам, земельным владениям, товарам, услугам и т. д. Официальная отчетность наносилась на доски: мраморные, бронзовые, медные, свинцовые и побеленные деревянные. Текущие записи велись на деревянных та-

бличках, скрепленных вместе с одного края по две, три и больше — кодексы (лат. *code* — дерево). После завоевания Римом Египта появился папирус. Около 180 г. до н. э. был изобретен пергамент (изготовлялся из телячьей кожи, был дорог, но прочен). На развитие учета влияли техника письма и система счета. Для вычислений использовался абак, заимствованный древними греками у египтян.

Бухгалтерский учет велся в Памятных книгах, или Мемориалах, куда записывались ежедневные факты хозяйственной деятельности. Также велась кассовая книга — первый кодекс и книга системной записи — второй кодекс.

Бюджетный учет велся в государственных масштабах. В отдельных провинциях велась книга Бревариум, в которой отражались как сметные ассигнования, так и их исполнение. Такой регистр получил название Книги имперских счетов, которую можно рассматривать как первый баланс государственного бюджета.

Развивался и налоговый учет, который требовал классификации и оценки имущества для начисления налога.

И хотя учет в Древнем Риме носил контрольный характер, уже тогда, по мнению древнеримского ученого Колумеллы, важнейшей функцией учета становилось умение предвидеть результат хозяйствования.

В Средневековье функции сбора данных остались те же — контрольный учет для сбора налогов и ведения хозяйственной деятельности.

С возникновением теории вероятностей в XVII веке были совершены первые попытки обработки накопленных данных и построения первых моделей для прогнозирования. Например, изучалась частота рождения мальчиков и девочек. Своим появлением теория вероятностей обязана азартным играм. Исследуя вероятность выигрыша, Пьер Ферми и Блез Паскаль открыли первые вероятностные закономерности. Независимо от них, но под влиянием их работ, Христиан Гюйгенс в 1657 г. опубликовал работу, в которой дал основные понятия теории

вероятностей (понятие вероятности как величины шанса; математическое ожидание для дискретных случаев, в виде цены шанса) и теоремы сложения и умножения вероятностей.

В 1794 г. (по другим данным — в 1795 г.) немецкий математик формализовал один из методов современной математической статистики. Данный метод стал основой для построения регрессионных моделей, цель которых — предсказание заданной величины. В XIX веке получил развитие анализ больших данных, который дал новый толчок к развитию статистических моделей.

В XX веке пошло быстрое развитие статистики и математической статистики как науки. В начале XX века была развита параметрическая статистика, созданы методы сравнения групп данных, оценки параметров групп и т. д.

Цель сбора данных кардинально изменилась к XX веку и перешла от контрольного учета к созданию математических предсказательных моделей.

Теперь перейдем ближе к современности и к науке о данных.

1974 г. впервые введен термин data science датским ученым в области информатики и компьютерной науки Питером Науром. Он считал, что наука о данных — дисциплина, изучающая жизненный цикл цифровых данных от появления до преобразования для представления в других областях знаний.

В начале 2000-х гг. data science выделяется как отдельная дисциплина.

Определения

Определение науки о данных вполне точно приведено в Wikipedia.

Наука о данных (*data science*; иногда *даталогия — datalogy*) — раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме*.

* С сайта http:www.ru.wikipedia.org.

В принципе такое определение достаточно полно описывает цели и суть науки о данных. Основной целью науки о данных является вывод новых знаний из имеющегося набора данных и получение новых зависимостей, часто неявных. Кроме того, одним из важных разделов науки о данных является визуализация больших данных.

Суть и цели

Остановимся подробнее на сути науки о данных. Исходной точкой в науке о данных являются собственно данные, и чем их больше — тем лучше. Далее нужно на основе этих данных найти взаимосвязи в них или убедиться, что их нет.

Для чего все это нужно? А целей — множество.

Во-первых, на основе полученных закономерностей можно построить прогноз для заданной величины. Например, на основе данных об урожае пшеницы за последние 10 лет в заданном регионе можно построить прогноз урожайности на следующий год.

Во-вторых, можно провести классификацию объектов на основе данных о них. Например, можно на основе клинических данных классифицировать методику лечения как эффективную или неэффективную.

В-третьих, можно визуализировать данные. Визуализация помогает выбрать стратегию анализа данных, а иногда она сама является целью анализа. Например, визуализация данных по движению городского транспорта в режиме on line ценно само по себе.

В-четвертых, можно провести анализ текстовой информации и, например, понять тональность отзыва о компании.

И наконец, в-пятых, можно найти новые зависимости в данных и на их основе прийти к новым знаниям о предмете анализа.

Data Science — зачем она нужна?

Спасаем Мир

Эпидемия Эбола в 2014 г. унесла более 11 000 жизней, и каждый день приносил новые смерти. Для data scientist задача по анализу данных и построению модели стала вызовом. И в 2014 г. the Leiden Centre of Data Science (LCDS) принял этот вызов. В результате разработана комплексная модель симуляции лихорадки Эбола, включающая диагностику распространения и испытание лекарств. Врачи, используя эту модель, остановили эпидемию Эбола.

Немного о модели

Стандартную модель симуляции распространения эпидемии дополнили картой, составленной на основе SMS-сообщений, звонков и другой активности с мобильных телефонов и добавили в нее все источники масс-медиа. На основе данной карты построили модель по пересечению и вычленению реальных данных. Использование данных с мобильных телефонов позволило установить направление распространения эпидемии и уже на основе этих данных получить оптимальные места для развертывания медицинских центров. Комплексная сеть данных (мобильные, масс-медиа и правительственные данные) и социальная сеть контактов дали возможность спрогнозировать скорость и направление развития эпидемии.

Модели, полученные из анализа данных пациентов, позволяют оценивать эффективность лекарств и проводить быстро множество тестов.

Познаем вселенную

The Center for Computational Astrophysics разрабатывает новый фреймворк (каркас программной системы), который предназначен для анализа астрономических данных. Он используется для построения модели Вселенной и оценки космологических констант. В XXI веке на основе нейронных сетей разработана 3D-модель Вселенной, в которой учтено распространение темной материи и есть возможность предсказания космологических констант.

Контрольные вопросы

1. Что является основной целью науки о данных?
2. Приведите примеры задач, которые можно решать с помощью науки о данных.

Основы обучения с учителем

Основные понятия

Итак, во введении мы рассмотрели области применения машинного обучения и его возможности на примерах. Результаты применения поражают воображение. Теперь настала пора разобраться в деталях и понять, как же это становится возможным.

В машинном обучении выделяют 2 основных подхода — обучение с учителем и обучение без учителя. В этой главе рассмотрим первый подход — с учителем.

Начнем с жизненной ситуации. У девушки — день рождения, и Пете нужно подарить ей цветы. Известно, что она не любит экзотические цветы, но какие нравятся — не известно. Петя подошел к проблеме с точки зрения машинного обучения и собрал данные о том, какие цветы больше всего любят девушки, и выбрал девушек близкого к имениннице возраста и внешности. Оказалось, что в предпочтениях лидируют два самых популярных цветка — роза и гербера. Причем 80 % девушек отдают предпочтение розам, а 20 % — герберам.

Теперь разберемся, что же сделал Петя. Все девушки, о которых Петя собрал информацию о предпочтениях в цветах, являются обучающей выборкой. Параметры, по которым Петя выбирал девушек, а именно возраст и цвет волос, являются признаками или факторами выборки. Информация о каждой отдельной девушке (цвет волос, возраст и любимый цветок) является объектом выборки. Причем цвет волос и возраст являются параметрами объекта, которые обычно обозначаются как x_1 и x_2,

а любимый цветок — ответ на данном объекте выборки (обозначается как y_i, где i — номер девушки в списке). Информация о возрастах и цвете волос всех девушек в выборке является пространством объектов X. Информация о любимых цветках всех девушек из выборки является пространством ответов Y.

Далее Петя должен найти зависимость между цветом волос и возрастом девушек и любимым цветком, то есть Петя должен найти коэффициенты w_1 и w_2 в уравнении $y = w_1 x_1 + w_2 x_2$. Зная их, он подставит возраст и цвет волос именинницы и получит ответ. Уравнение, которое определит Петя, — это алгоритм обучения. Строго говоря, алгоритм обучения $a(X)$ — функция перехода из пространства X в пространство Y.

Однако, возможно, Петя ошибется и ответ получит неверный, поэтому всегда нужно иметь инструмент оценки качества алгоритма. Таким инструментом является функционал ошибки. Функционал ошибки $Q(a, X)$ — ошибка алгоритма a на выборке X.

Обучение с учителем имеет два основных применения: классификация и регрессия. Классификация дает возможность предсказать по значениям признаков, к какому классу будет относиться новый объект. Кстати, пример с Петей — это типичная задача бинарной классификации. Регрессионный анализ дает возможность предсказать значение целевой переменной (ответа) по имеющейся модели, построенной на обучающей выборке.

Рассмотрим подробнее задачу регрессии. Допустим, у нас есть данные по продажам рожков мороженого и данные по температуре воздуха в дни продаж. Построим график, где по оси X отложим температуру воздуха, а по оси Y — количество проданных рожков мороженого, и отложим точки на нем. Далее, выведем зависимость количества проданных рожков мороженого от температуры воздуха и построим ее на этом же графике. Итак, сами данные о продажах нам уже не нужны. Теперь, зная температуру и пользуясь прямой зависимости, можно узнать, сколько рожков мороженого может быть продано.

Рассмотрим математическое описание линейного алгоритма модели. Алгоритм линейной модели $a(x)$ выражается в виде взвешенной суммы

$$a(x) = w_0 + \sum_{j=1}^{d} w_i x_j, \quad (1)$$

где w_0, w_i — веса модели; j — номер объекта в выборке; d — количество объектов в выборке; i — номер параметра объекта.

Перейдем к векторной форме записи выражения (1). Для этого внесем w_0 в сумму, добавив еще один признак, всегда равный 1:

$$a(x) = \sum_{j=1}^{d+1} w_j x_j = \langle W, X \rangle, \quad (2)$$

где W — вектор весовых коэффициентов; X — вектор признаков.

Вернемся к вышеописанному примеру модели, предсказывающей количество проданных рожков мороженого в зависимости от температуры воздуха. Алгоритм данной модели $a(t)$ запишется в виде следующей суммы:

$$a(t) = w_0 + \sum_{j=1}^{d} w_1 t_j, \quad (3)$$

где t — температура воздуха; w_0 и w_1 — веса модели; j — номер дня, для которого известна температура воздуха и количество проданных рожков мороженого; d — количество дней.

Можно выражение (3) представить также в векторной форме:

$$a(t) = \langle W, X \rangle.$$

Рассмотрим функционал ошибки, используемый в задаче регрессии. В качестве функционала ошибки в задаче регрессии используется среднеквадратичная ошибка. По сути своей она отображает квадрат разности между значением, полученным алгоритмом ($a(x)$) и ответом на обучающей выборке (y):

$$Q(a,x)=\frac{1}{d}\sum_{i=1}^{d}(a(x_i)-y_i)^2, \tag{4}$$

где y_i — ответ на i-м объекте обучающей выборки.

В векторной форме среднеквадратичная ошибка представляется в виде выражения

$$Q(a,x)=\frac{1}{d}\sum_{i=1}^{d}(\langle W,X\rangle-y_i)^2. \tag{5}$$

Теперь мы выяснили, как выглядит алгоритм и как оценить его ошибку. Тем не менее, как видно из формул (1)–(2), в алгоритме известны X и Y, а вот веса модели нужно найти. Процесс вычисления значений весов модели называется обучением модели. Естественно, нужно найти такие веса модели, при которых ошибка модели будет минимальной. Поэтому цель обучения модели — найти минимум функционала модели:

$$Q(x)\to\min.$$

Выражение (4) можно представить в полной форме:

$$Q(a,x)=\frac{1}{d}\sum_{i=1}^{d}(a(x)-y_i)^2\to\min. \tag{6}$$

Перепишем выражение (6) в матричной форме:

$$Q(W,X)=\frac{1}{d}\|XW-Y\|^2\to\min_w, \tag{7}$$

где X и Y — матрицы параметров и ответов соответственно:

$$X=\begin{pmatrix} x_{11} & x_{12} & x_{1n} \\ & & \\ x_{m1} & & x_{mn} \end{pmatrix},\quad Y=\begin{pmatrix} y_1 \\ \\ y_m \end{pmatrix}.$$

Матричное выражение (7) имеет аналитическое решение относительно вектора W

$$w=\left(X^TX\right)^{-1}X^Ty. \tag{8}$$

Однако, как видно из выражения (8), в аналитическом решении необходимо определить обратную матрицу, что не всегда возможно и очень долго. Поэтому на практике для расчета коэффициентов модели используется градиентный спуск.

Суть градиентного спуска заключается в пошаговом изменении значения весов и пересчете ошибки модели, причем изменение шага предпринимается в направлении антиградиента функции ошибки. В конечном итоге достигается минимум функции ошибки, что и дает искомый набор значений весов модели. Подробнее о методе градиентного спуска и его модификациях будет описано в следующей главе. А теперь рассмотрим метрики качества регрессионной модели.

Первая метрика, на которой и происходит обучение модели, — это уже рассмотренная выше среднеквадратичная ошибка. Обратите внимание на ее график, который принимает вид параболы (рис. 1). Парабола является гладкой, непрерывной функцией с 1 глобальным минимумом и поэтому может быть использована для дифференцирования.

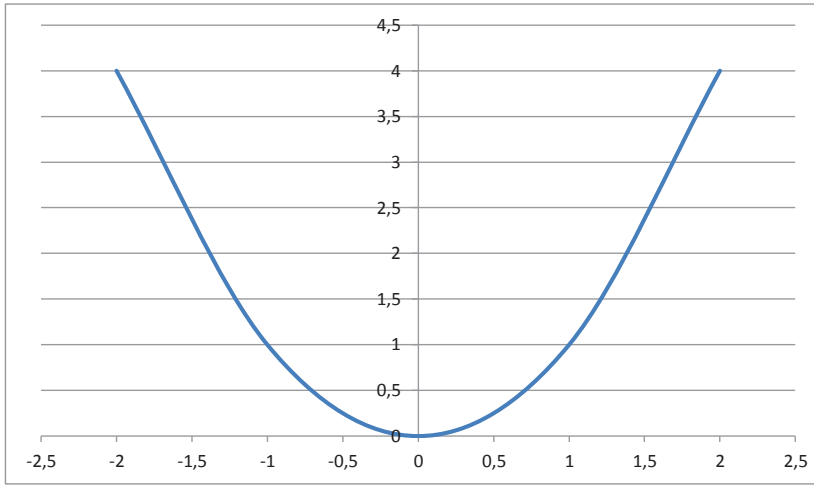

Рис. 1. График среднеквадратичной ошибки

Следующая метрика — средняя абсолютная ошибка ($MAE(a, X)$). Она отображает модуль разности между результатом, полученным алгоритмом модели и ответами на выборке:

$$MAE(a, X) = \frac{1}{d}\sum_{i=1}^{d}\left|a(x_i) - y_i\right|.$$

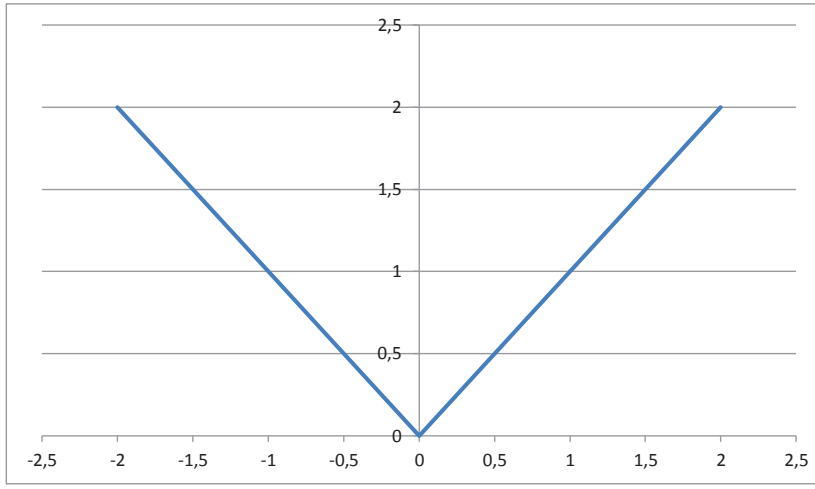

Рис. 2. График абсолютной ошибки

Как видно из рис. 2, абсолютную ошибку уже нельзя дифференцировать.

Следующая метрика качества регрессии — это коэффициент детерминации R^2:

$$R^2(a, X) = 1 - \frac{\sum_{i=1}^{d}\left(a(x_i) - y_i\right)^2}{\sum_{i=1}^{d}\left(y_i - \bar{y}\right)},$$

где $\bar{y}$ — среднее значение ответов на выборке.

Коэффициент детерминации показывает, какую долю дисперсии ответов алгоритма модель может объяснить.

Контрольные вопросы

1. Запишите выражение для алгоритма линейного регрессора.
2. Что рассчитывается при обучении модели?
3. В чем смысл коэффициента детерминации?

Градиентный спуск

Итак, мы переходим к рассмотрению вопроса: как вычислять веса модели? Как уже было сказано ранее, для вычисления весов модели используется функционал ошибки, а именно среднеквадратичная ошибка. Значения весов модели, соответствующие точке минимума среднеквадратичной ошибки, являются искомыми весами модели.

Суть метода заключается в пошаговом приближении к минимуму функции среднеквадратичной ошибки. Сначала назначаются случайным образом значения весов модели, затем вычисляются градиенты для каждого веса модели. Значения весов изменяются в направлении антиградиента — вычисляется среднее значение ошибки на всей выборке. Сравнивается полученное значение со значением с предыдущего шага. Если изменение ошибки становится незначительным, значит, функция зашла в свой минимум и искомые значения весов найдены.

Рассмотрим несколько алгоритмов метода.

Пакетный метод градиентного спуска

Алгоритм метода пакетного градиентного спуска для модели с одним параметром при постоянном шаге

Сперва назначаются начальные значения весов модели w_0 и w_1.

Далее вычисляется градиент w_0:

$$\frac{\partial Q}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_i - y_i\right), \tag{9}$$

где n — число объектов в выборке,

а также градиент w_1:

$$\frac{\partial Q}{\partial w_1} = \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_i - y_i\right)x_i. \tag{10}$$

Вычисляются новые значения w_0 и w_1 при постоянном шаге (h = const):

$$\begin{aligned} w_0^t &= w_0^{t-1} - h\cdot\frac{\partial Q}{\partial w_0}; \\ w_1^t &= w_1^{t-1} - h\cdot\frac{\partial Q}{\partial w_1}. \end{aligned} \tag{11}$$

Вычисляется значение среднеквадратичной ошибки:

$$Q^t = \frac{1}{n}\sum_{i=1}^{n}\left(w_0^t + w_1^t x_i - y_i\right)^2 \tag{12}$$

Вычисляется изменение ошибки и сравнивается с заданной точностью ε:

$$\left|Q^t - Q^{t-1}\right| < \varepsilon. \tag{13}$$

Если условие (13) выполняется, то значения весов найдены, если нет, то расчет повторить по формулам (9)–(13).

Теперь рассмотрим алгоритм метода пакетного градиентного спуска для двух и более параметров модели.

Алгоритм метода пакетного градиентного спуска для модели с двумя и более параметрами при постоянном шаге

Назначаются начальные значения весов модели w_0, w_1, w_2, w_3, …, w_n.

Вычисляется градиент w_0:

$$\frac{\partial Q}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{1i} + w_2 x_{2i} + \ldots + w_n x_{ni} - y_i\right). \tag{14}$$

Вычисляются градиенты для каждого из всех весов по формуле

$$\frac{\partial Q}{\partial w_n} = \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{1i} + \ldots + w_n x_{ni} - y_i\right)x_{ni}. \qquad (15)$$

Вычисляются новые значения весов при постоянном шаге ($h = \text{const}$):

$$w_0^t = w_0^{t-1} - h\cdot\frac{\partial Q}{\partial w_0}, \quad w_1^t = w_1^{t-1} - h\cdot\frac{\partial Q}{\partial w_1}, \ \ldots, \ w_n^t = w_n^{t-1} - h\cdot\frac{\partial Q}{\partial w_n}. \qquad (16)$$

Вычисляется среднее на выборке значение среднеквадратичной ошибки:

$$Q^t = \frac{1}{n}\sum_{i=1}^{n}\left(w_0^t + w_1^t x_{1i} + w_2^t x_{2i} + \ldots + w_n^t x_{ni} - y_i\right)^2. \qquad (17)$$

Вычисляется изменение ошибки и сравнивается с заданной точностью ε по формуле (13). Если условие выполняется, то значения весов найдены, если нет, то расчет повторить по формулам (13)–(17).

А теперь зададимся вопросом: а сколько итераций требуется, чтобы ошибка стала минимальной? Для этого нужно посмотреть на сходимость метода, которая показывает, как изменяется ошибка с каждой итерацией (рис. 3).

Как видно из рис. 3, на первых итерациях идет быстрое снижение ошибки, а вот после 40 итераций снижение ошибки уже незначительно.

Поскольку пакетный градиентный спуск считает ошибку метода, усредняя ее на всей выборке, то у него есть ряд недостатков из-за этого. Во-первых, просто долго считать, поскольку нужно вычислять градиент для каждого параметра на всей выборке. Во-вторых, и это более существенно, если расчеты привели в локальный минимум, то выйти из него уже нельзя, а истинное значение весов соответствует глобальному минимуму.

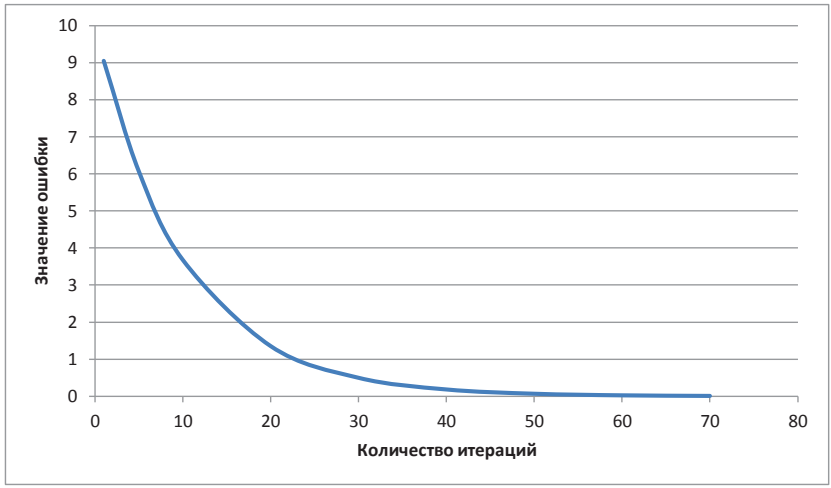

Рис. 3. Сходимость пакетного градиентного спуска

Стохастический градиентный спуск

Для устранения вышеуказанных недостатков есть два пути. Первый путь — брать не весь объем выборки, а его часть. Второй путь — изменить шаг по мере приближения к минимуму.

Для начала рассмотрим первый путь. Допустим, будем брать не всю выборку, а лишь один случайный объект из нее, а на следующей итерации возьмем другой случайный объект. Таким образом, даже если на первом объекте попали в локальный минимум, то на втором объекте мы можем оказаться где угодно, но вряд ли снова в локальном минимуме. Этот подход используется в стохастическом градиентном спуске.

Рассмотрим алгоритм стохастического градиентного спуска с постоянным шагом.

Алгоритм стохастического градиентного спуска с постоянным шагом

Назначаются начальные значения весов и шаг.

Затем случайным образом выбирается объект из выборки.

Вычисляется градиент веса w_0 для выбранного объекта:

$$\frac{\partial Q}{\partial w_0} = w_0 + w_1 x_{1i} + w_{2i} x_i + \ldots + w_{ni} x_i - y_i,$$

а также градиенты весов w_1, w_2, ..., w_n для выбранного i-го объекта:

$$\frac{\partial Q}{\partial w_j} = (w_0 + w_1 x_{1i} + w_{2i} x_i + \ldots + w_{ni} x_i - y_i) x_{ji}.$$

Изменяются веса в сторону антиградиентов, вычисляется ошибка и сравнивается с полученной на предыдущей итерации.

Если разница между ошибками меньше заданной точности, то процесс останавливается, если нет, то снова выбирается объект и все действия повторяются.

На рис. 4 показаны сходимости пакетного и стохастического градиентного спуска. Как видно из рисунка, стохастический спуск сходится быстрее, но при этом есть нестабильность сходимости, которая вызвана тем, что идет постоянная смена объектов выборки.

Рассмотрим особенности стохастического градиентного спуска. Во-первых, очевидно, что при постоянной смене объектов выборки можно легко выйти из локального минимума. Во-вторых, легко добавить новый объект в выборку и не нужно будет заново все пересчитывать, поскольку есть вероятность, что он попадет в расчет как случайный объект. Но расплачиваться за эти достоинства приходится нестабильной сходимостью.

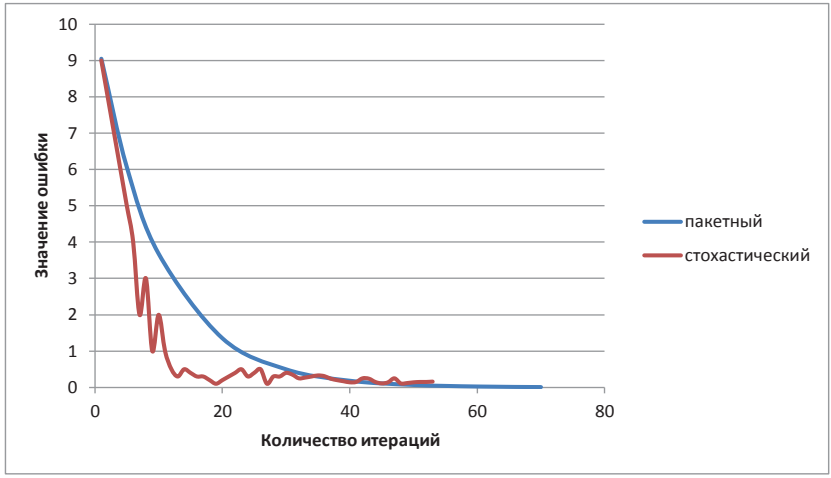

Рис. 4. Сходимость пакетного и стохастического градиентного спуска

Mini-batch

Стохастический градиентный спуск — метод безусловно хороший. Но если выборка очень большая, то существует вероятность, что в случайные выборки попадет слишком мало объектов и модель обучится только на них. Поэтому есть еще одна модификация градиентного спуска, в которой случайным образом уже выбирают не один объект, а несколько объектов из выборки — mini-batch.

Рассмотрим алгоритм mini-batch метода.

Алгоритм mini-batch градиентного спуска с постоянным шагом

Сперва назначаются начальные значения весов и шаг.

Случайным образом выбирается n объектов из выборки, где n — размер mini-batch.

Вычисляется градиент веса w_0 для mini-batch:

$$\frac{\partial Q}{\partial w_0} = \sum_{i=0}^{n} (w_0 + w_1 x_{1i} + w_{2i} x_i + \ldots + w_{ni} x_i - y_i).$$

Вычисляются градиенты весов w_1, w_2, ..., w_n для mini-batch:

$$\frac{\partial Q}{\partial w_j} = \frac{2}{n} \sum_{i=1}^{n} (w_0 + w_1 x_{1i} + w_{2i} x_i + \ldots + w_{ni} x_i - y_i) x_{ji}.$$

Изменяются веса в сторону антиградиента, вычисляется ошибка и сравнивается с полученной на предыдущей итерации.

Если разница между ошибками меньше заданной точности, то процесс останавливается, если нет, то снова выбирается n объектов и все действия повторяются.

Адаптивный градиентный спуск

А теперь рассмотрим варианты изменения шага на каждой итерации. Очевидно, что на первых итерациях шаг нужен большой, чтобы быстрее прийти в минимум, но дальше нужно его уменьшать. В адаптивном методе шаг уменьшается вполовину на каждой итерации.

Рассмотрим алгоритм метода.

Алгоритм метода адаптивного градиентного спуска для модели с одним параметром при адаптивном шаге

Назначаются начальные значения весов модели w_0 и w_1 и h.

Вычисляется градиент w_0 по формуле (9).

Вычисляется градиент w_1 по формуле (10).

Вычисляются новые значения w_0 и w_1 при текущем значении h по формуле (11).

Вычисляется новое значение $h = h/2$.

Вычисляется значение среднеквадратичной ошибки по формуле (12).

Вычисляется изменение ошибки и сравнивается с заданной точностью ε по формуле (13).

Если условие формулы (13) выполняется, то значения весов найдены, если нет, то снова действия повторяются по формулам (9)–(13).

В заключение темы сравним различные модификации метода градиентного спуска. Пакетный метод градиентного спуска лучше применять для гладкой выпуклой функции, так как быстро идет в минимум. Стохастический градиентный спуск хорошо применим для сложного вида функций с большим количеством локальных минимумов. Mini-batch хорошо векторизуется и поэтому хорошо применим для облачных вычислений.

Итак, мы рассмотрели, как идет обучение модели, то есть как рассчитываются веса модели. Но модель можно как переобучить, так и недоучить. В следующем разделе мы рассмотрим проблему переобучения модели и способы ее решения.

Контрольные вопросы

1. Как вычисляется изменение весов на каждой итерации градиентного спуска?
2. Что является условием остановки расчета при градиентном спуске?
3. Как считается ошибка на каждой итерации пакетного градиентного спуска?
4. Сколько объектов из выборки используется в методе стохастического градиентного спуска?

Переобучение модели и методы борьбы с ним

Суть проблемы переобучения

Как уже было описано ранее, обучение модели идет на выборке с ответами. В результате обучения может получиться модель, которая великолепно описывает именно ту выборку, на которой она обучена, но стоит дать ей объект, не входящий в эту выборку, модель дает ответ с большой ошибкой. Этот факт свидетельствует о переобучении модели.

Например, у нас есть выборка объектов с параметрами X и ответами Y. Нанесем точки на график и построим на этом же графике линейную модель алгоритма обучения (рис. 5).

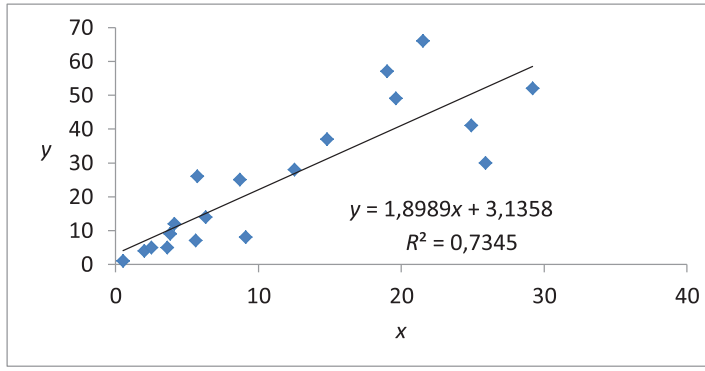

Рис. 5. Линейная модель

Как видно из рис. 5, модель дает относительно большую ошибку на выборке ($R^2 = 0{,}73$). Улучшим модель и повысим точность до $R^2 = 0{,}93$ (рис. 6).

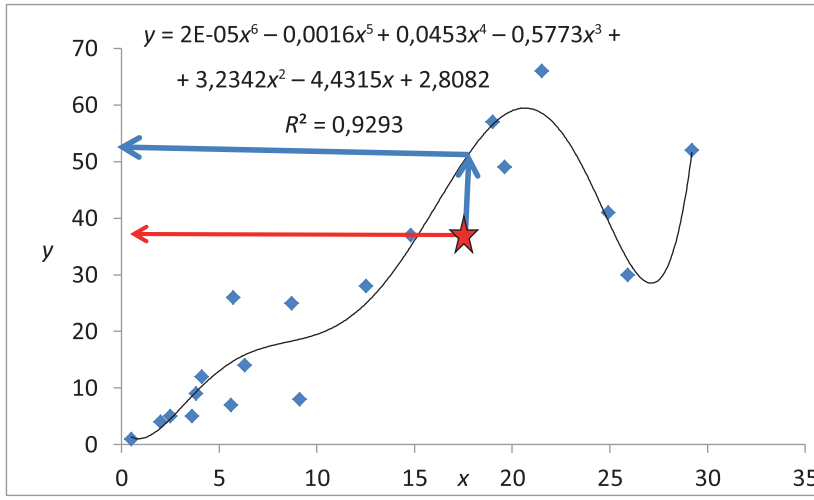

Рис. 6. Полиномная модель

Для данных выборки она дает хорошую точность. Но вот мы даем ей новый объект (помечен звездочкой на рис. 6), и наша модель существенно завышает ответ для него. Поднимая точность модели, мы слишком хорошо настроили ее под заданный набор данных и сделали непригодной для другого набора, то есть мы переобучили модель.

В примере переобучение модели показано графически, на практике построить график модели не всегда представляется возможным. Поэтому нужны какие-то признаки переобученности модели. И они есть. Первый признак — это малая ошибка на обучающей выборке и высокая на реальном объекте. Второй признак — аномально большие веса модели. Например, в модели (см. ниже), прогнозирующей спрос на прокат велосипедов в зависимости от погодных условий и дня недели, видна переобученность:

признак	вес
season	503,102
yr	2059,82
mnth	−29,80
holiday	−308,03
weekday	80,44
workingday	95,70
wearthersit	−579,49
temp	35,82
atemp	82,27
windspeed (mph)	33 160 096 171,02
windspeed (ms)	−74 179 135 208,08

На нее указывают аномально высокие веса при признаках windspeed.

Рассмотрим причины переобучения модели. Как можно видеть из примера, признаки windspeed на самом деле выражают одно и то же — скорость ветра, только в разных единицах измерения. Поэтому эти признаки очень сильно зависят друг от друга. Такие признаки называются *коллинеарными*. Выявить взаимосвязь признаков можно на основе коэффициентов корреляции. Для вещественных чисел используются коэффициенты корреляции Спирмена или Пирсона. Данные коэффициенты показывают силу связи между признаками и меняются от −1 до 1. При значении коэффициента близком к 0 связь слабая (при 0 вообще отсутствует). При значении коэффициента близком к 1 есть сильная положительная связь, то есть при увеличении значения одного признака значение второго увеличивается. При значении коэффициента близком к −1 есть сильная обратная связь, то есть при увеличении значения одного признака значение второго уменьшается. При значении коэффициента равном 1 признаки полностью взаимозаменяемы.

Рассмотрим значения коэффициента Пирсона для 6 параметров вышеописанной модели (табл. 1).

Как видно из табл. 1, значения коэффициентов Пирсона для признаков windspeed равны 1, то есть они не только коллинеарны, но и полностью взаимозаменяемы.

Таблица 1

Значения коэффициентов Пирсона

Признак	temp	atemp	hum	windspeed (mph)	windspeed (ms)	cnt
temp	1,00	0,99	0,13	–0,16	–0,16	0,63
atemp	0,99	1,00	0,14	–0,18	–0,18	0,63
hum	0,13	0,14	1,00	–0,25	–0,25	–0,10
windspeed (mph)	–0,16	–0,18	–0,25	1,00	1,00	0,23
windspeed (ms)	–0,16	–0,18	–0,25	1,00	1,00	0,23
cnt	0,63	0,63	–0,10	–0,23	–0,23	1,00

Рассмотрим методы борьбы с переобучением.

Отложенная выборка

В этом методе предлагается имеющуюся для обучения выборку разбить на две неравные части — обучающую выборку и тестовую. В обучающую часть выборки вносится более 70 %, в тестовую — оставшаяся часть. Обучение модели проводится на обучающей части, а проверка ее качества — на тестовой. Тут же возникает проблема разбиения выборки: если просто взять первые 70 % выборки и отнести их в обучающую часть, то существует опасность, что некоторые уникальные объекты не попадут в обучающую часть выборки и модель не обучится на них. Поэтому когда модель их встретит в тестовой части, то даст большую ошибку. Но можно выбирать в тестовую выборку случайные объекты из выборки, а оставшиеся считать обучающей выборкой. Тогда шанс, что уникальные объекты не попадут в обучающую выборку, снижается.

Кросс-валидация

Есть другой подход в разделении выборки, при котором все объекты точно попадут в обучающую часть выборки. Для этого выборку делят на n равных частей — фолдов (обычно 3) и обучают модель n раз, причем каждый раз выделяется одна часть как тестовая, а остальные становятся обучающими. Например, делим выборку на три части. В первый раз обучаем модель на первых двух частях, а тестируем на третьей. Во второй раз обучаем модель на первой и третьей части, а тестируем на второй, в третий раз — обучаем на второй и третьей части, а тестируем на первой.

В результате модель обучается n раз, и соответственно рассчитывается n раз ошибка модели (каждый раз при обучении модели рассчитывается своя ошибка). Итоговой ошибкой модели становится средняя ошибка по всем оценкам.

Достоинством данного метода бесспорно является невозможность пропустить объект мимо обучающей выборки. Недостаток тоже очевиден — нужно много раз обучать модель, а это — и время и деньги.

Рассмотрим пример применения кросс-валидации. Например, мы разбили выборку на 5 частей и получили на них оценки качества модели (пусть это будет R^2): 0,75; 0,75; 0,77; 0,79; 0,83. Тогда итоговая оценка модели составит $(0{,}75 + 0{,}75 + 0{,}77 + 0{,}79 + 0{,}83)/5 = 0{,}78$.

Регуляризация

Еще одним способом борьбы с переобучением модели является регуляризация. При обучении переобученной модели мы добились слишком маленькой ошибки на обучающей выборке,

то есть слишком плотно подошли к данным. Поэтому простой способ избежать данной ситуации — это добавить нечто к функционалу ошибки, чтобы было невозможно получить слишком маленькую ошибку. В этом и состоит регуляризация. В ходе регуляризации к функционалу ошибки, а именно к среднеквадратичной ошибке, добавляют дополнительный член и ищут минимум уже общей суммы:

$$Q(a, X) + \lambda \|w\| \to \min,$$

где λ — коэффициент регуляризатора; w — веса модели.

Используется два типа регуляризатора:

- L2-регуляризатор (квадратичный регуляризатор) — Ridge, который, как правило, не обнуляет веса, а только снижает их значение:

$$\|w\|^2 = \sum_{i=1}^{d} w_i^2;$$

- L1-регуляризатор — Lasso, который может обнулить веса малозначимых параметров модели:

$$\|w\|_1 = \sum_{i=1}^{d} |w_i|.$$

Применим регуляризаторы Lasso и Ridge к модели, прогнозирующей спрос на прокат велосипедов в зависимости от погоды и типа дня. Полученные после применения регуляризаторов веса модели приведены в табл. 2.

Таблица 2

Веса модели после применения к модели регуляризаторов

Признак	Значения веса при признаке после регуляризатора	
	Lasso	Ridge
season	499,78	500,24
yr	2055,19	2042,84
mnth	–28,77	–28,94
holiday	–268,46	–283,53

Окончание табл. 2

Признак	Значения веса при признаке после регуляризатора	
	Lasso	Ridge
weekday	80,46	80,57
workingday	93,54	96,02
wearthersit	–572,12	–570,48
temp	37,22	35,56
atemp	81,12	82,71
windspeed (mph)	–33,01	–27,58
windspeed (ms)	0,00	–12,33

Как видно из табл. 2, веса при признаках windspeed стали небольшими, а регуляризатор Lasso один из них обнулил. Остальные же веса не сильно изменились после применения регуляризаторов.

Контрольные вопросы

1. В чем смысл процесса регуляризации?
2. Приведите формулу L1-регуляризатора.
3. Приведите формулу L2-регуляризатора.

Бинарная классификация

Основные понятия и задачи классификации

Задачей бинарной классификации является отнесение объекта к одному из двух классов. Точнее, выделить объекты одного класса из выборки, а все остальные объекты по умолчанию попадают в другой класс, который обозначается как –1 или 0. Например, ответ верный или нет, банан спелый или нет, машина исправна или нет и т. д. Для решения поставленной задачи нужна обученная модель классификатора. Для обучения модели нужна обучающая выборка. Вспомним все, что мы уже знаем о выборке и объектах и дадим им определения. Объект выборки (x) — то, для чего нужно предсказание. Ответ на объекте выборки (y) — заданное значение для объекта выборки x. Пространство объектов выборки (X) — множество объектов выборки. Признаковое описание объекта — набор всех его признаков ($x_1, x_2, x_3, ...$). Обучающая выборка — набор объектов с ответами (x, y). Алгоритм обучения — $a(X)$ — функция перехода из пространства X в пространство Y. Функционал ошибки $Q(a, X)$ — ошибка алгоритма a на выборке X.

Рассмотрим на примере работу бинарного классификатора. Допустим, у нас есть анкета из 4 пунктов (любимый напиток чай или кофе, собаки или кошки, уровень оптимизма и уровень пессимизма) и мы провели опрос по ней среди «жаворонков» и «сов». Данные опроса представлены в табл. 3. Теперь по имеющейся выборке мы хотим провести анкетирование у нового человека и угадать по его ответам кто он — «жаворонок» или «сова». Все содержимое таблицы — это обучающая выборка,

названия столбцов — параметры выборки (или факторы), столбец «Жаворонок» — ответы на выборке.

Таблица 3

Результаты анкетирования

Чай или кофе?	Кошка или собака?	Оптимизм	Пессимизм	Жаворонок
Кофе	Кошка	3	8	Нет (–1)
Кофе	Кошка	5	5	Нет (–1)
Чай	Кошка	6	4	Да (+1)
Чай	Собака	8	3	Да (+1)
Кофе	Собака	7	3	Да (+1)
Чай	Собака	6	4	Да (+1)
Чай	Кошка	4	6	Нет (–1)

Допустим, мы опросили нового человека и его ответы следующие: чай, собака, 3, 8. Мы хотим предсказать кто он — «жаворонок» или «сова». Для этого нам нужна модель классификатора. Модель классификатора представляет собой алгоритм, дающий минимальную ошибку предсказания.

Линейный классификатор

Рассмотрим математическую модель линейного классификатора. Суть ее практически не отличается от линейной модели регрессии, рассмотренной ранее, только в качестве ответа здесь идет уже не вещественное число, а 0 или 1. Поэтому полученный вещественный ответ нужно перевести в 0 или 1. Для этого используется либо знаковая функция, либо пороговая соответственно:

$$a(x) = \operatorname{sign}\left(\sum_{j=1}^{d+1} w_j x_j\right) = \operatorname{sign}\langle w, x\rangle,$$

$$a(x)=\begin{cases}1, \sum_{j=1}^{d+1} w_j x_j > t,\\ 0, \sum_{j=1}^{d+1} w_j x_j \le t,\end{cases}$$

где t — порог.

Рассмотрим геометрический смысл классификатора. Если построение регрессионной модели сводилось к построению прямой, заменяющей набор точек выборки, то построение классификатора сводится к построению плоскости $<W, X> = 0$, оптимально разделяющую объекты выборки на 2 класса (рис. 7).

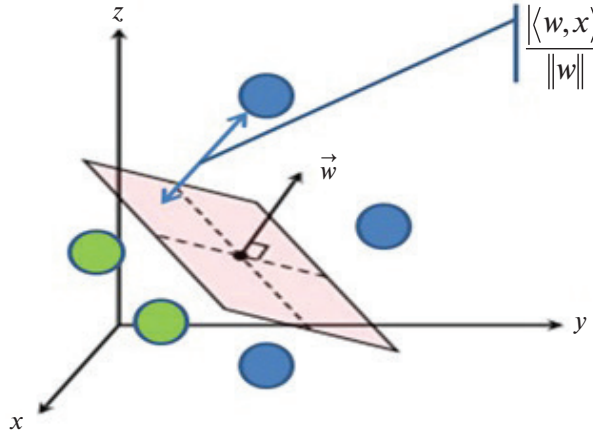

Рис. 7. Геометрический смысл бинарного классификатора

Причем скалярное произведение $<w, x>$ с одной стороны плоскости будет положительным, а с другой — отрицательным.

Тогда расстояние r от объекта, имеющего признаковое описание x, до плоскости $<w, x> = 0$ рассчитывается по формуле

$$r=\frac{|\langle w,x\rangle|}{\|w\|}.$$

Введем понятие отступа M:

$$M_i = y_i \langle w, x_i \rangle.$$

Отступ является мерой правильности ответа алгоритма классификатора. Он служит мерой уверенности в правильном выборе класса для объекта: чем он больше, тем надежнее объект отнесен к этому классу. Если объект попал в свой класс, то отступ будет положительным, если не в свой класс, то отступ будет отрицательным. Таким образом, количество положительных отступов показывает количество правильных ответов, количество отрицательных — количество ошибок алгоритма.

На основе отступа можно сформировать функционал ошибки бинарной классификации Q:

$$Q(a, X) = \frac{1}{n} \sum_{i=1}^{n} [M_i < 0].$$

В такой трактовке отступ становится функцией потерь, поскольку показывает, сколько неправильных ответов дает алгоритм a на выборке X.

Функция потерь (отступ) является пороговой функцией и имеет разрыв в точке 0 (рис. 8).

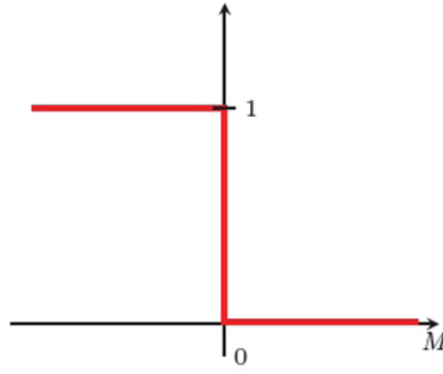

Рис. 8. График функции потерь

Для нахождения весов алгоритма необходимо дифференцировать функционал ошибки, а в случае бинарной классификации — функцию потерь. Поскольку данная функция не гладкая и имеет разрыв в 0, это сделать невозможно (рис. 8). Поэтому для нахождения минимума функционала ошибки в задаче классификации дифференцируют *оценку функции потерь*.

Для построения оценки функции потерь введем некую гладкую и неразрывную функцию $L(M_i)$, которая покрывает функцию потерь:

$$L(M_i) \geq [M_i < 0].$$

Тогда можно построить оценку функционала ошибки $\bar{Q}(a, X)$:

$$\bar{Q}(a, X) \geq Q(a, x),$$

$$\bar{Q}(a, X) = \frac{1}{n} \sum_{i=1}^{n} L(M_i).$$

Поскольку функция $L(M_i)$ гладкая и неразрывная, то оценку функционала ошибки уже можно дифференцировать и нужно будет минимизировать для нахождения значения весов бинарного классификатора.

В качестве гладкой функции $L(M_i)$ могут быть использованы разные функции. Рассмотрим 3 из них:

- логистическая функция потерь (используется в логистической регрессии)

$$L(M) = \log_2(\exp(-M));$$

- экспоненциальная функция потерь (выражается через экспоненту)

$$L(M) = \exp(-M);$$

- кусочно-линейная функция потерь (используется в методе опорных векторов)

$$L(M) = \max(0, 1 - M).$$

На рис. 9 приведен вид всех 3 оценок функционала ошибки. Синей линией изображена экспоненциальная оценка функции потерь, оранжевой линией — логистическая оценка функции потерь, серой — кусочно-линейная оценка функции потерь.

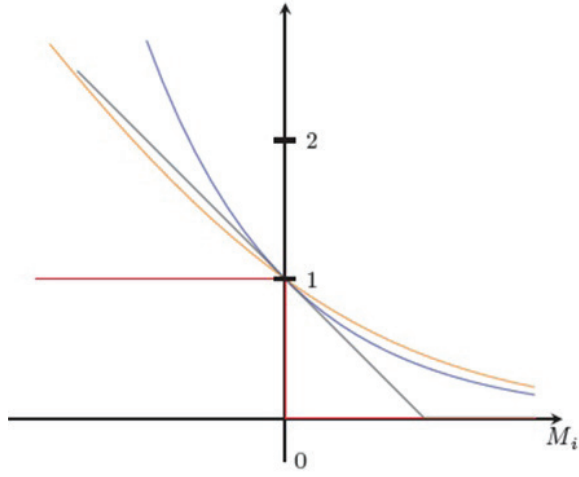

Рис. 9. График оценок функционала ошибки

Логистическая регрессия

Пусть название не вводит вас в заблуждение. Результатом работы логистического регрессора является 0 или 1, то есть это бинарный классификатор. Логистический регрессор оценивает вероятность попадания в класс 1 через отношение шансов $OR(x)$:

$$OR(x) = \frac{P(x)}{1 - P(x)},$$

где $P(x)$ — вероятность события x.

Отношение шансов меняется в диапазоне от $-\infty$ до $+\infty$, а нам нужен диапазон от 0 до 1. Поэтому прологарифмируем полученное выражение $OR(x)$ и выразим из него $P(X)$:

$$P_+ = \frac{e^{\langle w,x\rangle}}{1+e^{\langle w,x\rangle}} = \frac{1}{1+e^{-\langle w,x\rangle}}.$$

Полученное выражение является формулой логистического регрессора.

На рис. 10 приведен график логистического регрессора.

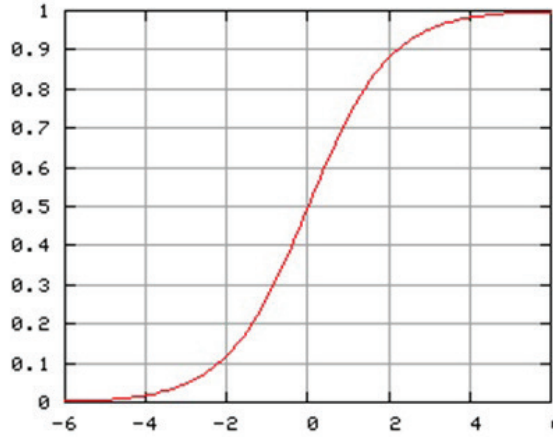

Рис. 10. График логистической регрессии

В логистической регрессии, как и в линейной модели, необходим функционал ошибки, минимизируя который, мы будем искать веса модели. В качестве функционала ошибки здесь используется уже описанная нами ранее логистическая функция потерь с некоторыми изменениями:

$$\bar{Q} = \frac{1}{n}\sum_{i=1}^{n}\ln(\exp(-M_i)) = \frac{1}{n}\sum_{i=1}^{n}\ln(\exp(-y_i < w,x >)).$$

А если внести $1/n$ под знак суммы, то получим

$$\bar{Q} = \sum_{i=1}^{n} \ln(1 + \exp(-y_i < w, x >)).$$

Для поиска минимума функционала ошибки в логистической регрессии используются так же, как и в линейной модели, градиентный спуск или его модификации. Результатом обучения модели логистической регрессии является так же, как и в линейной модели, набор весов модели. А вот ответами в логистической регрессии будут метки класса для каждого объекта с указанием вероятности отнесения объекта к этому классу.

Обучение модели проходит на обучающей части выборки, а оценка качества модели — на тестовой части выборки. Вот и пришло время обсудить метрики качества классификации.

Метрики качества классификации

Доля правильных ответов

Доля правильных ответов (accuracy (a, X)) — наиболее популярная метрика качества классификационных моделей. Она указывает в долях или процентах количество правильных ответов, то есть тех ответов алгоритма, которые совпали с истинными значениями в обучающей или тестовой части выборки:

$$\text{accuracy}(a, X) = \frac{1}{n} \sum_{i=1}^{n} [a(x_i) = y_i].$$

Но с этой метрикой надо быть осторожными.

Допустим, в выборке много объектов класса 1, но мало класса 0. Тогда модель будет отлично находить объекты класса 1, а вот с объектами класса 0 будет ошибаться. И если поменять разметку классов на обратную, то доля правильных ответов рез-

ко снизится. Кроме того, при построении классификационной модели всегда нужно помнить о цене ошибки. Например, модель оценивает эффективность методики диагностирования туберкулеза в открытой форме, когда человек крайне заразен для окружающих. Тогда класс 1 (методика эффективна) становится очень критичным. И если модель ошибается и признает человека здоровым (то есть относит объект к классу 0, а не к 1), а он болен, то вред от такой оценки очень велик. Как этого избежать? Ответ прост: в обучающей выборке должно быть больше случаев с эффективной методикой, чем с неэффективной.

Матрица ошибок

Кстати, об ошибках классификации. Они бывают двух типов — False Negative (FN) и False Positive (FP).

Матрица ошибок

	Y=1	Y=−1
a(x) =1	True Positive (TP)	False Positive (FT)
a(x) =−1	False Negative (FN)	True Negative (TN)

False Negative (FN) — количество объектов, которые на самом деле являются классом 0 или −1, но алгоритм отнес их к классу 1. False Positive (FP) — количество объектов, которые на самом деле являются классом 1, но алгоритм их отнес к классам −1 или 0.

Матрица ошибок содержит и правильные ответы, которые также делятся на 2 группы: True Positive (TP) и True Negative (TN). True Positive (TP) — количество объектов класса 1, которые нашла модель. True Negative (TN) — количество объектов классов −1 или 0, которые нашла модель.

Рассмотрим пример с двумя матрицами ошибок.

Пример 1 с матрицей ошибок

	Y=1	Y=−1
a1(x) =1	80	20
a1(x) =−1	20	80

Пример 2 с матрицей ошибок

	Y=1	Y=−1
a2(x) =1	48	2
a2(x) =−1	52	98

Первый алгоритм (пример 1) хорошо опознает и класс 1 и класс −1, причем ошибается тоже одинаково на обоих классах. Второй алгоритм (пример 2) очень точно выделяет класс 1: почти все объекты, попавшие в класс 1, действительно ими и являются. Но при этом алгоритм отбрасывает много объектов класса 1 в класс −1.

Точность и полнота

Зачем же все-таки нужны матрицы ошибок, ведь информация в них избыточна? Вроде бы достаточно просто показать, насколько хорош алгоритм и сколько он делает ошибок при классификации. Матрицы ошибок позволяют понять не только, сколько ошибок делает алгоритм, но понять, насколько он точен и как полно находит заданный класс. Для этого вводятся понятия точность и полнота.

Точность показывает, насколько можно доверять классификатору. Классификатор с высокой точностью выловит в класс 1 только объекты класса 1 и не прихватит в него объекты другого класса. Это такой очень аккуратный сборщик ягод: собирает только ягоды, без мусора (листвы и прочего). Вычисляется точность precision (a, X) по формуле

$$\text{precision}(a, X) = \frac{TP}{TP + FP}.$$

Как видно из вышеприведенной формулы, точность — это отношение количества правильно отнесенных к классу 1 объектов ко всем объектам класса 1 в выборке.

Полнота показывает долю истинных ответов, на которых срабатывает классификатор. Полнота показывает, как хорошо классификатор находит объекты заданного класса. Классификатор с высокой полнотой найдет все объекты класса 1 в выборке, но при этом прихватит в класс 1 и много объектов из класса −1. Это жадный сборщик ягод, соберет все ягоды в лесу без остатка вместе со всем, что попадется. Вычисляется полнота (recall (a, X)) по формуле

$$\text{recall}(a, X) = \frac{TP}{TP + FN}.$$

Полнота, как видно из вышеприведенной формулы, — это отношение количества правильно отнесенных к классу 1 объектов ко всем объектам, отнесенным классификатором к классу 1.

Вернемся к примерам матриц ошибок (см. с. 39). Первый алгоритм имеет высокую и точность (0,8) и полноту (0,8). Очень хороший алгоритм — можно использовать. Второй алгоритм имеет высокую точность (0,96), а вот полнота у него низкая — 0,48. Такой алгоритм можно использовать, если необходимо найти объекты только класса 1, не взяв случайно объект класса −1.

Как правило, настроить алгоритм на высокую и точность, и полноту невозможно, поэтому чем-то из них приходится жертвовать. Чем именно можно пожертвовать, определяется ценой ошибки, то есть конкретной задачей, в которой нужно понять, что важнее — найти все объекты класса или найти объекты только одного класса.

Компромиссом между этими мерами является F-мера.

F-мера

F-Мера сочетает в себе и точность, и полноту и рассчитывается по формуле

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Если нужно учесть предпочтения точности или полноте, то вводят коэффициент β:

$$F = \left(1 + \beta^2\right) \frac{2 \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}.$$

ROC-AUC кривая

Еще одной популярной метрикой качества классификации является ROC-AUC кривая. ROC-AUC кривая строится в координатах *TPR* и *FPR* соответственно:

$$TPR = \frac{TP}{TP + FN},$$

$$FPR = \frac{FP}{FP + TN}.$$

Обычный вид ROC-AUC кривой представлен на рис. 11.

Качество классификатора оценивается площадью под ROC-AUC кривой: чем она ближе к 1, тем лучше классификатор. Для идеального классификатора ROC-AUC кривая выглядит, как показано на рис. 12.

Если площадь под ROC-AUC кривой близка к 0,5, то этот классификатор выдает случайные ответы. График ROC-AUC кривой для случайного классификатора представлен на рис. 13.

Поэтому на практике часто на графике отображают еще и график случайного классификатора, чтобы показать, насколько полученный классификатор отличается от случайного (рис. 14).

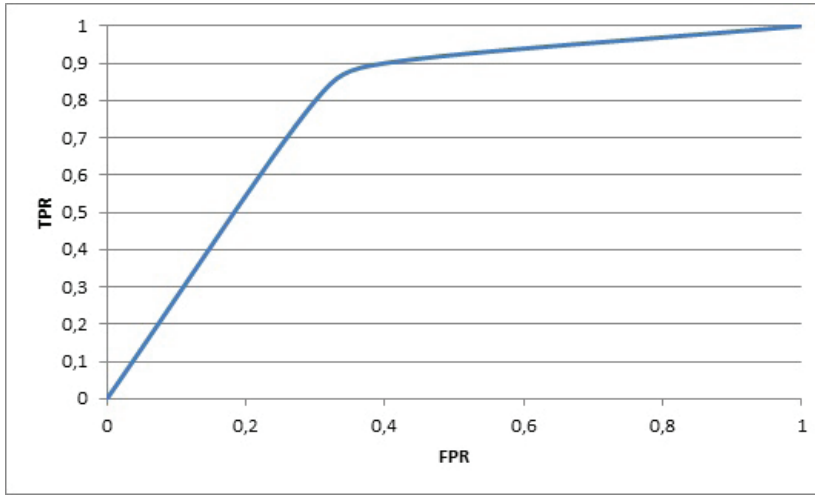

Рис. 11. ROC-AUC кривая

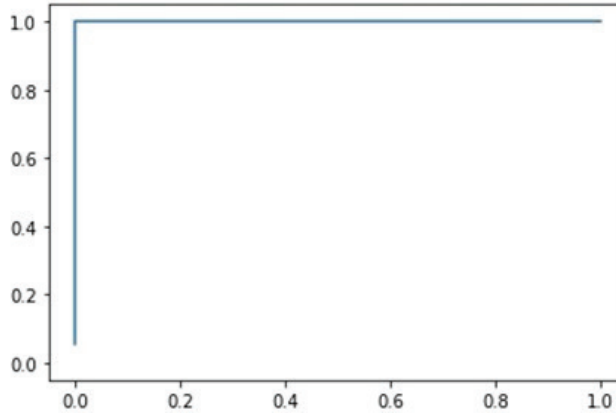

Рис. 12. ROC-AUC кривая идеального классификатора

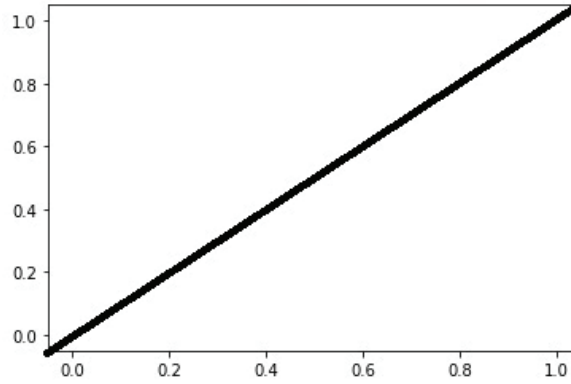

Рис. 13. ROC-AUC кривая случайного классификатора

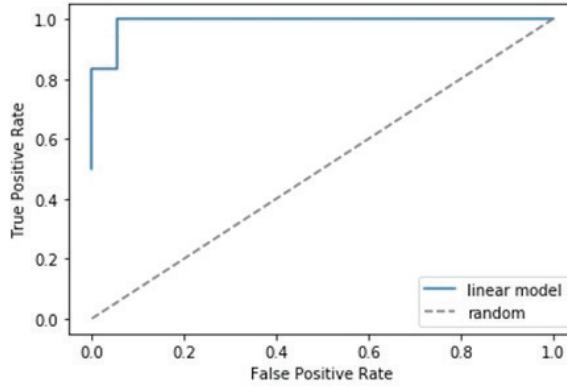

Рис. 14. Пример ROC-AUC кривой

Контрольные вопросы

1. Опишите задачу бинарной классификации.
2. Какая стандартная функция используется в работе линейного классификатора?
3. Что оценивает логистический регрессор?

Решающие деревья и случайный лес

Решающие деревья

Общие понятия

В этой главе рассмотрим новый способ классификации и регрессии — решающие деревья (дерево принятия решений). Решающие деревья представляют собой направленный иерархический граф. В его узлах стоят признаки, по которым идет разделение выборки, а в листьях — части выборки. Причем в левом листе-потомке узла находится та часть выборки, для которой признак есть (или для которого ответом на вопрос в узле было да), а в правом листе-потомке — часть выборки, которая не имеет признака в узле (ответ на вопрос в узле — нет). Количество уровней иерархии в решающем дереве называется его глубиной. Например, дерево принятия решений по симптомам болезни (рис. 15).

Как видно из рис. 15, дерево решений имеет глубину 3 и принятие решений начинается с параметра «температура тела». Если температура превышает 37 °С, то проверяется скорость роста температуры; если меньше, то также идет проверка наличия кашля. Если температура растет быстро, то проверяется наличие ломоты в теле; если нет быстрого роста, то предполагается, что диагноз ОРВИ. Если есть ломота в теле, то предполагается диагноз грипп. Если ломоты в теле нет, то проверяется боль в горле, если горло не болит, то рекомендуется вызвать скорую помощь (диагноз не ясен). Если болит горло, то предполагается диагноз ангина. Если температура меньше 37 °С

и есть кашель, то проверяется, есть ли слабость долгое время. Если есть слабость долгое время, то рекомендуется обратиться к пульмонологу, если слабости нет, то предполагается диагноз ОРЗ. Также диагноз ОРЗ предполагается, если нет высокой температуры и нет кашля.

Рис. 15. Пример решающего дерева

В принципе правила, ведущие к каждому заключению, можно сформировать как логическую формулу. Например, если температура меньше 37 °C, есть кашель и слабость долгое время, то нужно обратиться к пульмонологу.

Рассмотрим, как применяется решающее дерево к выборке. Допустим, у нас есть 50 пациентов, по которым заполнены данные о температуре тела, скорости ее роста, наличие кашля, ломоты и боли в горле. Тогда на первом шаге применения решающего дерева (рис. 15) выборка разделится на 2 части: в одной будут пациенты с температурой выше 37 °C, в другой — ниже 37 °C. Далее часть выборки с пациентами, имеющими температуру выше 37 °C, будет снова разделена на две части. В одной — будут пациенты с быстро растущей температурой,

в другой — со стабильной температурой. Затем часть выборки с быстро растущей температурой снова будет разделена по признаку наличия ломоты в теле. Та часть выборки, для которой признак ломоты в теле отсутствует, будет снова поделена по признаку боли в горле. Аналогично будет разделена и часть выборки с температурой тела ниже 37 °C. В конечном итоге будут получены 7 групп (по числу листов дерева) из первоначальной выборки.

Решающие деревья чаще всего применяются для задач классификации. Как известно, геометрический смысл классификации сводится к нахождению линии или плоскости, разделяющей объекты выборки на классы. На рис. 16 приведен пример классификации с помощью решающего дерева.

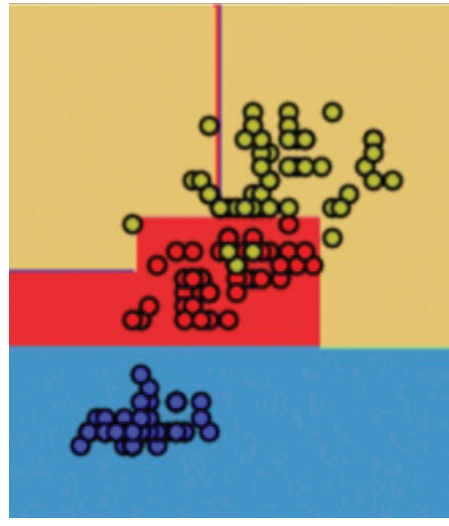

Рис. 16. Классификация решающим деревом

Как видно из рис. 16, выборка поделена на 3 класса — синий, желтый и красный. Для этого потребовалось дерево с 2 уровнями иерархии. На первом (самом верхнем уровне) провели

разделение на синий и желтый классы, на втором уровне желтый класс разделили на красный и желтый классы. Некоторые объекты желтого класса попали в красный класс (см. рис. 16). Что же будет, если продолжить разделение выборки по классам, применяя новые признаки? Скорее всего, модель переобучится. Решающие деревья очень легко переобучаются, то есть решающее дерево можно очень хорошо «заточить» под конкретную выборку, но потом его нельзя будет применять для другой выборки. Пример результатов работы переобученного дерева приведен на рис. 17.

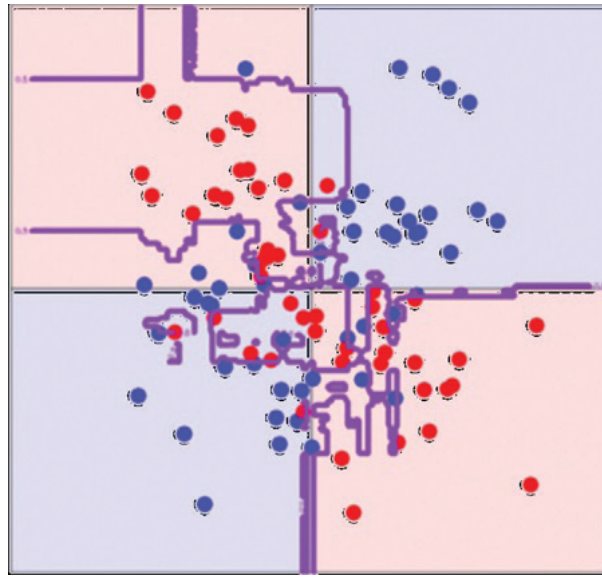

Рис. 17. Пример переобученного дерева

Как видно из рис. 17, поверхность, разделяющая классы, имеет очень сложную форму и наверняка при использовании ее на другой выборке даст большую ошибку.

Обучение решающих деревьев

Построение решающих деревьев идет путем разделения выборки на части по вводимым признакам. Признаки и порог их значения, по которым делится выборка, нужно подбирать так, чтобы в листьях дерева оставались объекты одного класса.

Пусть в вершине X_m объектов. Выбираем порог t по критерию ошибки Q для признака j, минимизируя критерий ошибки:

$$Q(X_m, j, t) \to \min,$$

причем признаки j и значения порога перебираем так, чтобы выполнить данное условие. В итоге получаем две части выборки:

$$X_l = \left\{ x \in X_m \middle| \left[x^j \le t \right] \right\},$$

$$X_r = \left\{ x \in X_m \middle| \left[x^j > t \right] \right\}.$$

Однако дерево нельзя строить до бесконечности: нужно знать, когда остановиться. Остановка в разбиении происходит в 3 случаях. В первом случае, если в вершину попал только один объект обучающей выборки или все объекты принадлежат одному классу (в задачах классификации). Во втором случае, если глубина дерева достигла определенного значения. В третьем случае задают количество объектов в листе дерева. Причем чаще на практике задают глубину разделения.

Прогноз для листа дерева

При использовании в качестве функционала ошибки в задачах регрессии значения среднеквадратичной ошибки, прогнозом будет среднее значение в листе a_m:

$$a_m = \frac{1}{|X_m|} \sum_{i \in X_m} y_i .$$

В задачах классификации прогнозом будет наиболее популярный класс в листе a_m:

$$a_m = \arg\max_{y \in Y} \sum_{i \in X_m} [y_i = y].$$

Критерий информативности

Выбор оптимального разбиения определяется критерием ошибки $Q(X_m, j, t)$:

$$Q(X_m, j, t) = \frac{|X_l|}{|X_m|} H(X_l) + \frac{|X_r|}{|X_m|} H(X_r),$$

где $H(X)$ — критерий информативности.

Критерий информативности показывает, насколько хорошо решающее дерево компонует объекты в своих листьях, то есть насколько они близки друг к другу. В случае задачи регрессии близость объектов в листе характеризуется степенью разброса значений в листе:

$$H(X) = \frac{1}{|X|} \sum_{i \in X} (y_i - \bar{y}(X))^2.$$

В случае задач классификации используется критерий информативности Джини. Он не отрицателен и имеет максимум, если все объекты в листе принадлежат классу 1, а точнее, если в листе вообще 1 объект. Вычисляется критерий Джини по формуле

$$H(X) = \sum_{k=1}^{K} p_k (1 - p_k),$$

где p_k — доля объектов класса k в выборке X:

$$p_k = \frac{1}{X} \sum_{i \in X} [y_i = k].$$

Также может быть использован энтропийный критерий информативности

$$H(X) = -\sum_{k=1}^{K} p_k \ln p_k .$$

Он оптимален при условии, что все объекты X принадлежат классу 1.

Стрижка деревьев

Суть данного приема заключается в том, что строится дерево максимальной глубины, то есть до того момента, когда в листе останется по 1 объекту. А затем убираются лишние листья по заданному критерию.

Решающие деревья и категориальные признаки

Все вышеописанные примеры относились к моделям с бинарными признаками. Но есть множество моделей, у которых есть категориальные признаки со множеством значений. Как к ним применяются решающие деревья?

В этом случае есть 2 подхода.

Первый подход — использование n-арных деревьев. Принцип очень простой: у каждого узла, в котором содержится категориальный признак, имеется не два, а n потомков по числу категорий признака. Пусть признак X имеет $\{c_1, c_2, ..., c_n\}$ значений. Тогда вершина разбивается на n значений так, чтобы $X = c_i$.

Разбиение проводится по принципу минимизации функционала ошибки разбиения $Q(X_m, j)$:

$$Q(X_m, j) = \sum_{i=1}^{n} \frac{|X_i|}{|X_m|} H(X_i) \to \min_j .$$

Второй подход — использование бинарных деревьев с разбиением на множество значений. В этом подходе категориальные признаки заменяют на натуральные и сортируют значения. Далее назначают порог, по которому идет деление вершины на две части.

Композиция алгоритмов

Одно дерево — это хорошо, а много — еще лучше. А когда деревьев много — это уже лес, причем это вовсе не шутка. Объединение множества решающих деревьев дает композицию алгоритмов, одной из разновидностей которой является «Случайный лес».

Зачем нужно объединять алгоритмы? Опыт показывает, что объединение множества слабых алгоритмов с невысокой точностью дает один сильный алгоритм с хорошей точностью.

Случайный лес

В методе «Случайный лес» обучают каждый алгоритм из композиции, а ответом является усредненный результат по всем алгоритмам, входящим в композицию.

В случае регрессии ответ $a(x)$ находится по формуле

$$a(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x).$$

В случае классификации ответ $a(x)$ находится по формуле

$$a(x) = \operatorname{sign}\frac{1}{N}\sum_{n=1}^{N} b_n(x).$$

Например, результаты композиции из 10 бинарных классификаторов показали следующие ответы: 1, 1, −1, −1, −1, 1, 1, −1, −1. Тогда итоговый ответ имеет вид

$$a(x) = \operatorname{sign}\left(-\frac{1}{10}\right) = -1.$$

Казалось бы хороший метод. Но есть проблема. Обучать каждый алгоритм надо на своей части выборки. Почему нельзя

на всей выборке обучить 10 раз разные алгоритмы? Дело в том, что есть следующая закономерность, выведенная экспериментально и не имеющая объяснения. Если 10 раз использовать одни и те же данные для разных алгоритмов обучения, то с каждым алгоритмом результат будет все лучше, но это искусственный эффект. На самом деле, если немного изменить выборку, результаты не улучшаются. При использовании одних и тех же данных идет подстройка алгоритмов под данные, и она дает кажущиеся улучшения качества алгоритмов. Поэтому выборку надо разбивать на части и обучать алгоритмы на разных частях выборки.

Для разбиения выборки на подвыборки можно использовать случайный выбор подмножества из начальной выборки, то есть случайным образом задается номер объекта из выборки и он помещается в подвыборку. Это работает, если выборка большая и есть из чего выбрать. Но если выборка небольшая, то приходится прибегать к ухищрениям. Имя ему бутстрап («тянуть себя за шнурки», если точно перевести с английского). В чем смысл фокуса, при котором из 4 объектов можно сделать 10 выборок по 4 объекта?

Допустим, у нас есть 4 шарика — синий, желтый, зеленый и красный (рис. 18).

Рис. 18. Начальная выборка

Берем из начальной выборки синий шар и помещаем его копию в выборку, а сам шар возвращаем в начальную выборку. Повторяем процедуру, потом берем желтый шар и помещаем его копию в выборку, а его возвращаем в начальную выборку, повторяем процедуру с зеленым шаром. В результате получи-

ли выборку из 4 шаров. Аналогично набираем еще 3 выборки (рис. 19).

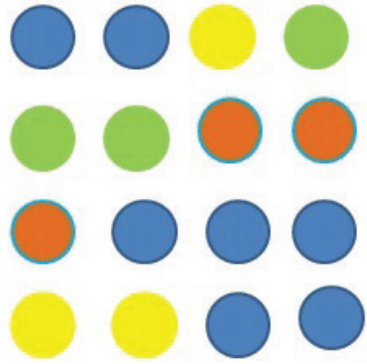

Рис. 19. Сгенерированные бутстрапом подвыборки

Вот таким незамысловатым способом можно сделать сколько угодно выборок из 4 шаров.

Bagging

Теперь рассмотрим методы построения и обучения композиций алгоритмов. Начнем с bagging. В этом методе на сгенерированных бутстрапом выборках обучаются алгоритмы, входящие в композицию. Каждый алгоритм получает свою выборку для обучения. Причем при обучении каждое решающее дерево строится до конца (как правило, до 1 элемента в листе или $n_{\min}$ элементов), а оптимальный признак, по которому идет разделение в выборки на каждом шаге построения дерева, выбирается случайно из множества признаков, и каждый раз используется новое множество признаков. Результаты, полученные на алгоритмах, усредняются.

Boosting

В данном методе алгоритмы обучаются последовательно, и каждый последующий алгоритм учитывает ошибки предыдущего. Кроме того, при вычислении общего ответа композиции, вклад каждого алгоритма учитывается в зависимости от его точности, то есть для каждого алгоритма в композиции есть свой вес. За счет такого подхода можно получить хороший результат на меньшем количестве алгоритмов.

Рассмотрим работу boosting на примере регрессии.

На первом шаге обучается первый алгоритм, минимизируя среднеквадратичную ошибку:

$$b_1 = \arg\min\left(\frac{1}{n}\sum_{i=1}^{n}\left(b(x_i) - y_i\right)^2\right),$$

где $b(x_i)$ — ответ алгоритма композиции на объекте x_i; y_i — ответ объекта x_i в выборке.

На вход первому алгоритму подаются объекты выборки X и ответы на них Y.

Далее второй алгоритм обучается, минимизируя ошибки, полученные первым алгоритмом:

$$b_2(x) = \arg\min\left(\frac{1}{n}\sum_{i=1}^{n}\left(b(x_i) - (y_i - b_1(x_i))\right)^2\right).$$

На вход ко второму алгоритму подаются объекты выборки X и ошибки первого алгоритма на всей выборке.

Продолжаем набирать алгоритмы, так чтобы n алгоритм уменьшал ошибку всей композиции:

$$b_N(x) = \arg\min\left(\frac{1}{n}\sum\left(b(x_i) - \left(y_i - \sum_{i=1}^{N-1} b_n(x_i)\right)\right)\right)^2.$$

Алгоритм бустинга

Приведем алгоритм бустинга:
1. Создаем первую модель (дерево решений или регрессор).
2. Обучаем на всей выборке.
3. Делаем предсказание.
4. Считаем среднеквадратичную ошибку.
5. Создаем следующую модель и обучаем ее на полученных в п. 4 ошибках и том же наборе параметров.
6. Суммируем предсказания полученных моделей.
7. Повторяем с п. 4 по п. 6, пока сумма предсказаний не перестанет меняться или разница не станет меньше заданной точности.

Градиентный бустинг

Несмотря на то что бустинг является довольно эффективным методом обучения композиций алгоритмов, его можно ускорить за счет учета направления уменьшения ошибки алгоритма. Это и происходит в градиентном бустинге. Здесь каждый новый алгоритм обучается уже на антиградиенте функционала ошибки предыдущего алгоритма.

Алгоритм градиентного бустинга

Приведем алгоритм градиентного бустинга:
1. Создаем первую модель (дерево решений или регрессор).
2. Обучаем на всей выборке.
3. Делаем предсказание.
4. Считаем антиградиент от функционала ошибки (например, от среднеквадратичной ошибки).

5. Создаем следующую модель и обучаем ее на полученных в п. 4 антиградиентах и том же наборе параметров.
6. Суммируем предсказания полученных моделей.
7. Повторяем с п. 4 по п. 6, пока сумма предсказаний не перестанет меняться или разница не станет меньше заданной точности.

Градиентный бустинг легко переобучается, что и показано на рис. 20.

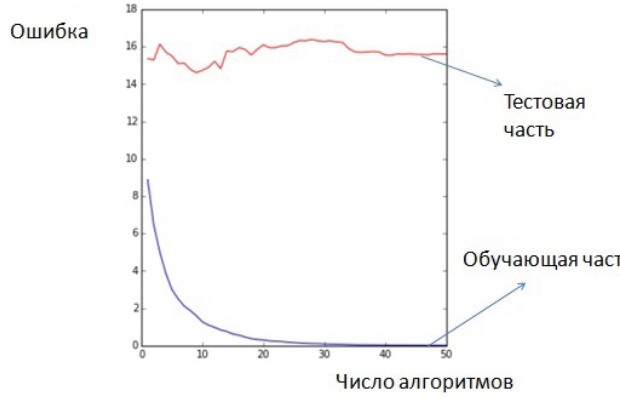

Рис. 20. Переобучение градиентного спуска

Причина переобучения банальна — траектория спуска не является оптимальной.

Для исправления ситуации есть два пути.

Первый — уменьшение шага на каждой итерации. Для этого достаточно добавлять каждый новый алгоритм с меньшим весом, чем предыдущий.

Второй путь — изменение объема выборки. Второй путь осуществляется в стохастическом градиентном бустинге.

Стохастический градиентный бустинг

Суть его та же, что и градиентного бустинга с той разницей, что на каждой итерации берется для обучения алгоритма не вся выборка, а ее случайная часть. На следующей итерации берется уже другая случайная часть выборки. После каждой итерации части выборки возвращаются обратно в выборку.

На рис. 21 представлены результаты по улучшению градиентного бустинга.

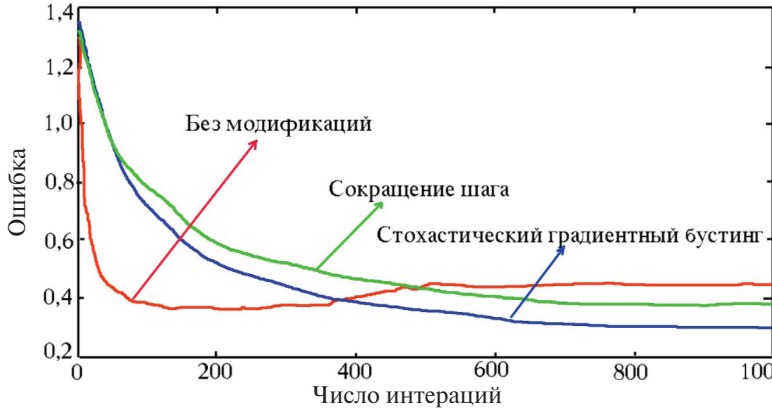

Рис. 21. Сравнение вариантов улучшения градиентного бустинга

Как видно из рис. 21, градиентный бустинг без модификаций имеет неравномерное уменьшение ошибки на итерациях. Сокращение шага дает более плавное уменьшение ошибки по мере роста итераций, но при этом требует большего числа итераций для достижения той же ошибки, что и градиентный бустинг. Использование стохастического градиентного бустинга дает плавное уменьшение ошибки по мере роста итераций и в то же время требует практически столько же итераций, что и градиентный бустинг без модификаций.

Контрольные вопросы

1. Что является ответом в методе случайного леса?
2. Как обучается каждый последующий алгоритм в бустинге?
3. Что используется для обучения каждой последующей модели в алгоритме градиентного бустинга?

Обучение без учителя

Основные понятия и области применения

Все случаи, рассмотренные ранее, имели ответы на обучающей выборке. Но бывают такие ситуации, когда ответов нет даже на обучающей выборке. Например, есть данные о посетителях сайта, и нужно понять, к каким социальным группам они относятся. В этом случае ответов, то есть данных о социальной группе посетителя, в обучающей выборке не будет. Другой случай: есть набор точек, и нужно понять, на что этот набор больше похож, то есть какой предмет он может описывать.

Все случаи, когда ответов на выборке нет, относятся к обучению без учителя. Как правило, обучение без учителя является частью подготовки данных для анализа методами обучения с учителем, поскольку предсказательных моделей этот метод дать не может. Задачами обучения без учителя являются кластеризация, визуализация и понижение размерности.

Задачи кластеризации

В задачах кластеризации нужны группы объектов, схожих между собой. Схожесть объектов определяется расстоянием между ними: чем объекты ближе друг к другу, тем они более схожи.

На рис. 22 близкие друг к другу объекты выделили в 5 кластеров.

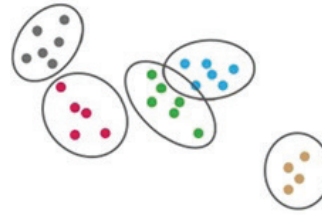

Рис. 22. Пример кластеризации

В задачах кластеризации невозможно изначально понять, сколько будет кластеров и какой будет размер кластеров. Кластеризацию часто проводят перед задачей классификации, чтобы понять, насколько разделимы данные. Например, по данным социологического опроса американцев была проведена кластеризация (рис. 23). И как можно видеть на рис. 23, четко выделились две непересекающиеся группы граждан, которые, как оказалось, коррелируют с политическими убеждениями. Одна группа отображает республиканцев, другая — демократов.

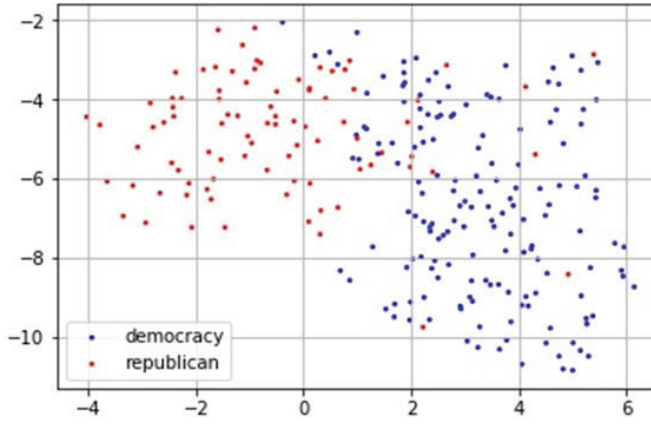

Рис. 23. Пример кластеризации

Подробнее кластеризацию изучим в разделе «Кластеризация».

Задача визуализации многомерных данных на плоскости

Цель данной процедуры — снизить размерность данных до 2 или 3, чтобы представить их наглядно. После этого данные на плоскости могут образовать группы, а значит, данные разделимы на классы. Такой подход очень полезен перед построением классификации.

На рис. 24 приведена визуализация на плоскости данных о рукописных цифрах.

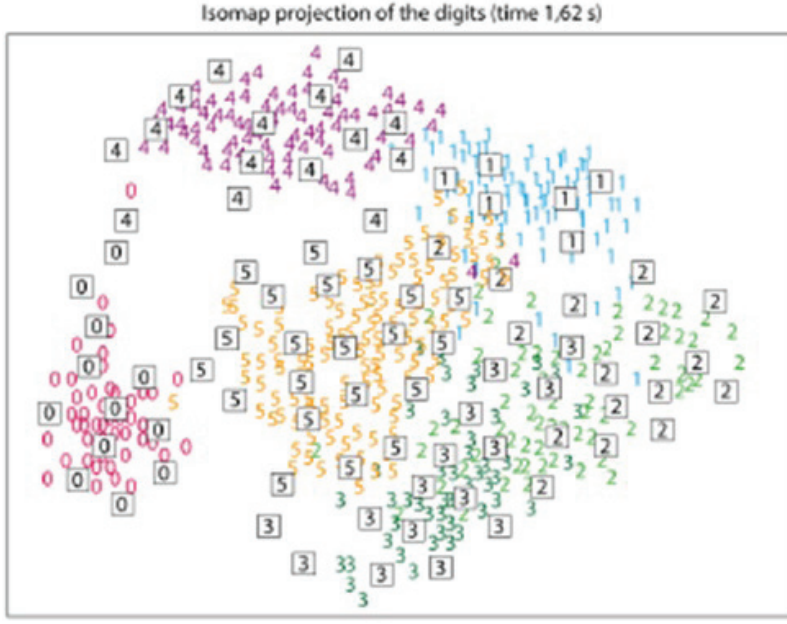

Рис. 24. Визуализация данных о рукописных цифрах

Как можно видеть из рис. 24, данные образуют группы по типу цифры, которую они отображают.

Подробнее о методах понижения размерности будет в разделе «Понижение размерности».

Задача поиска аномалий

Задача поиска аномалий — найти те объекты выборки, которые по своим свойствам резко выделяются из выборки. Это позволяет определить нестандартные действия пользователя на сайте и заподозрить, что сайт взломан. Определить несвойственные держателю карты траты и заподозрить, что карта украдена и пр. Объекты, обладающие аномальными свойствами, в дальнейшем исследуются другими методами.

Контрольные вопросы

1. Есть ли ответы на выборке в методе обучения без учителя?
2. По какому принципу объединяются объекты выборки в кластеры?
3. Зачем нужна визуализация данных?

Понижение размерности

Понижение размерности сводится к уменьшению числа параметров модели, причем важные параметры модели должны быть оставлены. Есть 2 подхода к решению задачи понижения размерности: отбор признаков и проекция признаков. Цель — отобрать значащие признаки, а незначащие признаки убрать из модели. Значащими признаками считаются те, которые имеют влияние на функционал ошибки модели, а именно уменьшают его.

Алгоритмы отбора признаков

Существует 3 основных метода отбора признаков: методы обертывания, методы вложения и методы фильтров:

- методы обертывания — используют априорную оценку качества модели, модель обучается на наборе признаков и оценивается ее ошибка, выбирается набор с минимальной ошибкой;
- методы фильтров — используют косвенные оценки качества модели (например, расстояние между классами), выбирается набор с минимумом по выбранному критерию;
- методы вложения — осуществляют отбор признаков в процессе построения модели (например, Lasso).

Рассмотрим подробнее каждый из них.

Методы обертывания

Методы обертывания представлены 3 алгоритмами: полный перебор, «жадный» алгоритм и ADD-DEL.

При полном переборе считается ошибка на всех возможных комбинациях параметров и выбирается комбинация, дающая минимальную ошибку. Очевидно, что этот метод долгий и затратный по ресурсам.

В «жадном» алгоритме (рис. 25) берется признак, дающий минимальную ошибку модели, и затем к нему добавляется по 1 признаку. И если наблюдается уменьшение функционала ошибки, то продолжают добавлять, если уменьшения не наблюдается, то добавление останавливают.

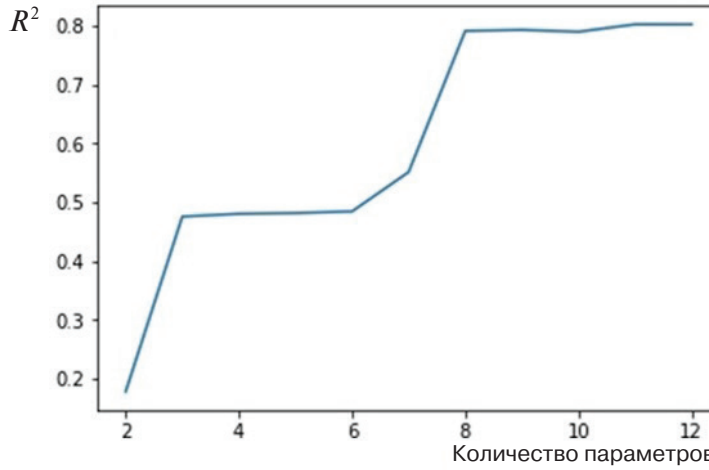

Рис. 25. Пример работы жадного алгоритма

Как видно из рис. 24, точность модели, оцениваемая через коэффициент детерминации R^2, растет по мере добавления параметров, но после 10 параметров точность модели уже почти не меняется.

Алгоритм ADD-DEL является модернизацией «жадного» алгоритма. Если в «жадном» алгоритме новые параметры только добавляются, то в алгоритме ADD-DEL параметр, который увеличивает функционал ошибки после его добавления в модель, удаляется из набора параметров.

На рис. 26 показан результат работы алгоритма ADD-DEL на тех же данных, что и у «жадного» алгоритма.

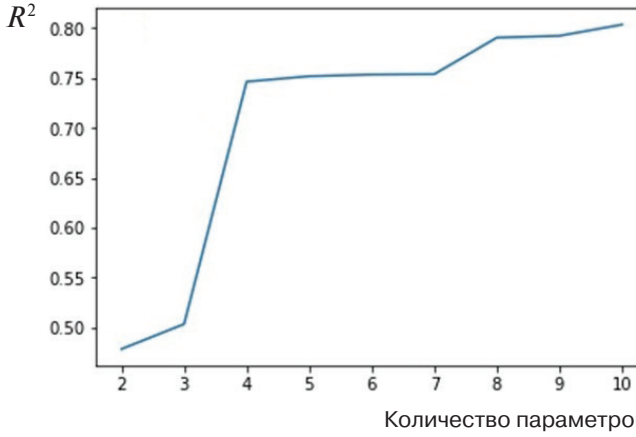

Рис. 26. Пример работы алгоритма ADD-DEL

Как видно из рисунка, алгоритм ADD-DEL быстрее, чем «жадный» алгоритм, достигает стабильной точности — ему хватило 10 параметров.

Методы фильтров

Метод фильтров заключается в отборе параметров по заданному условию. Например, можно убрать из модели все сильно коррелирующие между собой признаки.

Методы вложения

Методы вложения осуществляют отбор признаков сразу во время построения модели за счет, например, введения регуляризаторов. Метод Lasso является одним из методов вложения. Мы рассмотрели его работу подробно ранее и сейчас не будем повторяться.

Проекция признаков

Второй способ понижение размерности — это проекция признаков. Целью его является перевод данных из пространства высокой размерности в пространство малой размерности. Проекция признаков бывает линейной и нелинейной. При линейной проекции признаков последние проецируются на прямую линию. Линейную проекцию признаков реализует метод главных компонент (PCA). В методах нелинейной проекции признаков последние проецируются на различные кривые. Методов, реализующих нелинейную проекцию признаков, много. Ниже мы рассмотрим два наиболее популярных сейчас метода — SNE и t-SNE.

Метод главных компонент (principal component analysis — PCA)

Метод был изобретен Карлом Пирсоном в 1901 г. Постановка задачи в методе сформулирована так: выполнить аппроксимацию данных линейными многообразиями, то есть найти направления максимальной дисперсии данных. Суть метода заключается в поиске направлений максимальной дисперсии данных. Эти направления и будут служить основами для проекций данных. Действительно, если аппроксимировать точки на прямую,

которая идет вдоль точек, то суть начальных данных исказится несильно и этими искажениями можно будет пренебречь. Зато вместо множества точек можно будет записать уравнение прямой, что существенно снизит размерность данных.

Рассмотрим математическое описание постановки задачи.

Дано — множество векторов

$$x_1, x_2, ..., x_m.$$

Требуется для каждого из них в k-мерном пространстве, где $k = 0, 1, 2, ..., n-1$, найти такое линейное многообразие L, что выполняется условие

$$\sum_{i=1}^{m} \text{dist}^2\left(x_i, L_k\right) \to \min,$$

где dist(x, L) — евклидово расстояние от точки x до линейного многообразия L.

Линейное многообразие L представляет собой взвешенную сумму ортогональных векторов a_0, a_1, ..., a_k:

$$L_k = \left\{a_0 + \beta_1 a_1 + \beta_2 a_2 + ... + \beta_k a_k\right\},$$

то есть необходимо найти семейство прямых, вдоль которых идет наибольший разброс данных и которые проходят как можно ближе к данным.

Приведем алгоритм метода:

1. Централизуются данные (вычитанием среднего).
2. Отыскивается первая главная компонента

$$a_1 = \arg\min\left(\sum\left\|x_i - a_1\left(a_1, x_i\right)\right\|^2\right).$$

3. Из данных вычитается проекция на первую главную компоненту

$$x_i = x_i - a_1(a_1, x_i).$$

4. Повторяются п. 2 и п. 3 до тех пор, пока a_n не станет единичным вектором.

Разберем пример понижения размерности с 2 до 1*.

Дан набор точек (рис. 27).

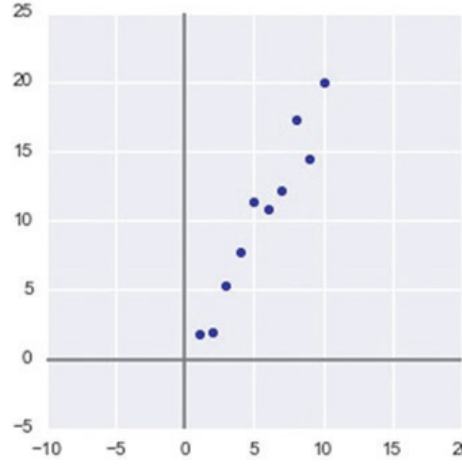

Рис. 27. Начальная выборка

Отцентрируем выборку, вычтя средние значения из X и из Y (рис. 28).

Теперь нужно понять форму распределения точек в начальной выборке, для чего используется ковариационная матрица. Ковариационная матрица — это матрица, у которой (i, j)-элемент является корреляцией признаков (X_i, X_j). Формула ковариации имеет вид

$$\mathrm{Cov}\left(X_i, X_j\right) = E\left[\left(X_i - E\left(X_i\right)\right)\cdot\left(X_j - E(X_j)\right)\right] =$$
$$= E\left(X_i X_j\right) - E(X_i)\cdot E(X_j).$$

* Полный текст статьи с примером есть на сайте habr https://habr.com/ru/post/304214/.

Диагональные элементы рассчитываются по формуле

$$\mathrm{Cov}(X_i, X_j) = \mathrm{Var}(X_i).$$

Диагональные ее элементы показывают отклонение величины от математического ожидания. Величина диагонального элемента покажет, в какую сторону вытянута форма данных выборки.

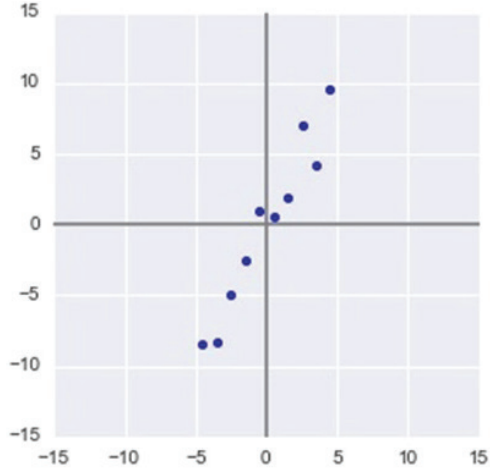

Рис. 28. Отцентрированная выборка

Вернемся к примеру. Таким образом, в нашей матрице по диагонали будут дисперсии признаков (так как $i = j$), а в остальных ячейках — ковариации соответствующих пар признаков. А в силу симметричности ковариации матрица тоже будет симметрична.

Проанализируем данные матрицы ковариации. Дисперсия по X составляет 9,17, дисперсия по Y составляет 37,26, коэффициент корреляции X и Y составляет 17,93. Поскольку дисперсия по Y больше, чем по X, следует ожидать, что данные будут вытянуты вдоль Y.

Теперь нужно найти векторы, по направлению которых наблюдается максимальная дисперсия данных. Направление максимальной дисперсии у проекции всегда совпадает с собственным вектором, имеющим максимальное собственное значение, равное величине этой дисперсии. Размерность нашей выборки равна двум и количество собственных векторов (айгенвекторов) у нее соответственно 2 (*X* и *Y*). То есть мы берем вектор оси *X* и умножаем его на дисперсию по *X*, берем вектор оси *Y* и умножаем его на дисперсию по *Y*. Поскольку дисперсия по *Y* больше, то он более значительно отклонится от оси *Y*. Построим эти векторы (рис. 29).

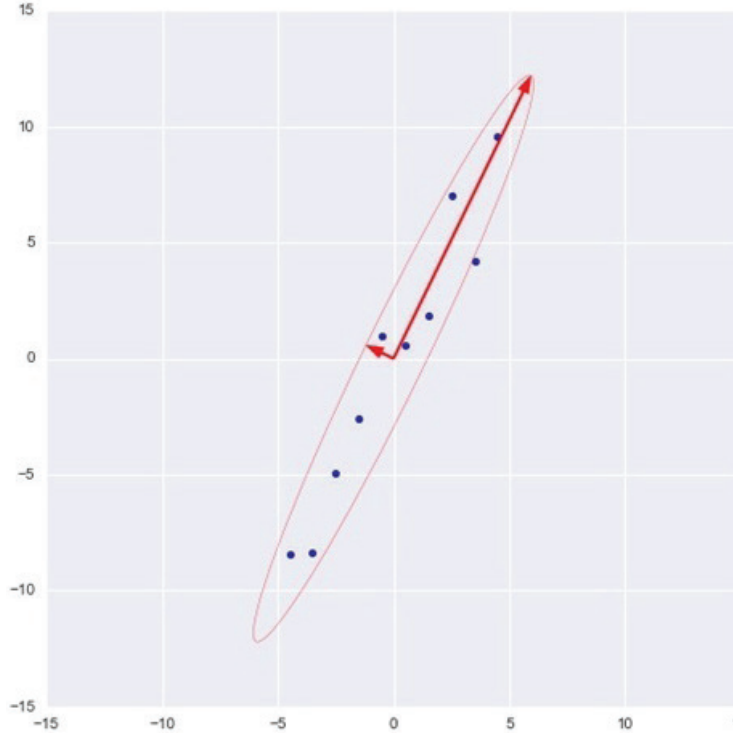

Рис. 29. Собственные вектора

Выберем наибольший вектор из них (в нашем случае *Y*). Его направление даст направление линии регрессии, на которую мы будем проецировать данные выборки. Спроецировав на него нашу выборку, мы потеряем информацию, сравнимую с суммой остаточных членов регрессии (только расстояние теперь евклидово, а не дельта по *Y*). В итоге мы получили одномерный вектор данных.

Метод SNE (Stochastic Neighbor Embedding)

Метод основан на вычислении вероятностей нахождения объектов рядом друг с другом как в многомерном пространстве, так и в двухмерном.

На первом этапе в методе вычисляется вероятность нахождения двух объектов рядом в многомерном пространстве. Чем она выше, тем объекты ближе. Далее вычисляется вероятность нахождения этих же объектов рядом в двухмерном пространстве. Затем оценивается разница вероятностей при переходе из многомерного пространства в двухмерное. На последнем этапе вычисляются координаты прообразов многомерных объектов в двухмерном пространстве путем минимизации разницы вероятностей.

Приведем алгоритм метода:

1. Вычисляем вероятность нахождения объекта X_i рядом с объектом X_j в многомерном пространстве $p(x_i|x_j)$, используя евклидову метрику и распределение Гаусса:

$$p\left(x_j \mid x_i\right) = \frac{\exp\left(\left\|x_i - x_j\right\|^2 / 2\sigma^2\right)}{\sum \exp\left(\left\|x_i - x_j\right\|^2 / 2\sigma^2\right)}. \qquad (18)$$

2. Вычисляем вероятность нахождения объекта X_i рядом с объектом X_j в двухмерном пространстве $q(\tilde{x}_i|\tilde{x}_j)$, используя евклидову метрику и распределение Гаусса:

$$q(\tilde{x}_j \mid \tilde{x}_i) = \frac{\exp\left(\left\|\tilde{x}_i - \tilde{x}_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(\left\|\tilde{x}_i - \tilde{x}_k\right\|^2 / 2\sigma^2\right)}.$$

3. Находим новые координаты, минимизируя дивергенцию Кульбака-Лейблера:

$$p\left(x_j \mid x_i\right) \log \frac{p(x_j \mid x_i)}{q(\tilde{x}_j \mid \tilde{x}_i)} \to \min. \qquad (19)$$

Отметим недостатки метода.

При переходе из многомерного пространства в маломерное расстояние между объектами уменьшается и пропорции искажаются, а евклидова метрика слишком сильно штрафует за искажение пропорций.

Метод t-SNE

В этом методе распределение объектов в новом пространстве берется как распределение Стьюдента с 1 степенью свободы. При таком подходе нет сильного штрафа за искажение пропорций.

Приведем алгоритм метода:

1. Вычисляем вероятность нахождения объекта X_i рядом с объектом X_j в многомерном пространстве $p\ (x_i|x_j)$, используя евклидову метрику и распределение Гаусса (18).
2. Вычисляем вероятность нахождения объекта X_i рядом с объектом X_j в двухмерном пространстве $q(\tilde{x}_i|\tilde{x}_j)$, используя евклидову метрику и распределение Стьюдента:

$$q(\tilde{x}_j \mid \tilde{x}_i) = \frac{\left(1 + \left\|\tilde{x}_i - \tilde{x}_j\right\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \left\|\tilde{x}_i - \tilde{x}_j\right\|^2\right)^{-1}}.$$

3. Находим новые координаты, минимизируя дивергенцию Кульбака-Лейблера (19).

Контрольные вопросы

1. По какому критерию выбирается первая компонента, на которую проецируются объекты из многомерного пространства?
2. Какая метрика используется для оценки расстояния между объектом и компонентой в методе главных компонент?
3. Одинаковые ли метрики используются для оценки близости объектов в многомерном пространстве и в маломерном в методе SNE?
4. Одинаковые ли метрики используются для оценки близости объектов в многомерном пространстве и в маломерном в методе t-SNE?

Кластеризация

Задача, алгоритм и этапы кластеризации

Целью кластеризации является выделение близких по параметрам объектов и объединение их в кластеры. Строго говоря, задача кластеризации формируется следующим образом. Имеется обучающая выборка $X_l = \{x_1, ..., x_l\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X_l$ приписывается метка (номер) кластера y_i.

Алгоритм кластеризации — это функция $a: X \to Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров с точки зрения того или иного критерия качества кластеризации.

Перечислим этапы кластеризации:

- Выделение параметров объекта, по которым можно проводить кластеризацию.
- Определение меры, по которой будет определяться расстояние между объектами.
- Разбиение объектов на кластеры.
- Визуализация кластеров.

Меры расстояний между объектами ρ(x, x')

Существует несколько мер расстояний между объектами.

- евклидово расстояние;
- квадрат евклидова расстояния;
- расстояние городских кварталов (манхэттенское расстояние);
- расстояние Чебышева;
- степенное расстояние.

Евклидово расстояние

Геометрическое расстояние между 2 точками в многомерном пространстве:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2},$$

где x, x' — координаты точек.

Квадрат евклидова расстояния

Применяется для придания большего веса более отдаленным друг от друга объектам. Определяется по формуле

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2.$$

Расстояние городских кварталов (манхэттенское расстояние)

Такое расстояние является средним разностей по координатам:

$$\rho(x, x') = \sum_{i}^{n} |x_i - x_i'|.$$

Для данной меры влияние отдельных больших разностей (выбросов) уменьшается, так как они не возводятся в квадрат.

Расстояние Чебышева

Такое расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Определяется по формуле

$$\rho(x, x') = \sum_{i}^{n} |x_i - x_i'|.$$

Степенное расстояние

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Вычисляется по формуле

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p},$$

где r и p — параметры, определяемые пользователем. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра — r и p — равны двум, то это расстояние совпадает с расстоянием Евклида.

Метод ближайших соседей (kNN)

Берем точку и ищем ее ближайших соседей. Определяем, к какому кластеру относятся большинство ближайших соседей. Считаем, что точка относится к тому кластеру, что и большинство ближайших соседей.

Строго говоря, этот метод — не совсем метод кластеризации: он является универсальным методом и применим как в кластеризации, так и в классификации и регрессии.

В случае кластеризации здесь формально не строятся кластеры, а определяются ближайшие соседи и их параметры. Тогда объект, для которого проводится кластеризация и для которого нужно определить недостающие параметры, принимает параметры ближайших соседей.

В случае классификации нужна обучающая выборка с указанными классами, а затем новый объект приписывают тому классу, к которому он ближе.

В случае регрессии объекту присваивается среднее значение от его соседей.

Рассмотрим пример использования kNN. Допустим, у нас есть данные о рейтингах фильмов, выставленные разными пользователями (табл. 4).

Таблица 4

Рейтинги фильмов

Имя пользователя	Титаник	Унесенные ветром	Дикая роза	Star track	Человек-паук
Маша	5	5	5	–	1
Дима	3	1	1	–	5
Саша	3	3	–	5	4
Катя	4	4	–	–	1
Петя	3	1	–	–	5

Необходимо вычислить на ее основе рейтинг фильмов для тех пользователей, которые его не проставили. Для этого используем user-based подход.

Вычисляем коэффициент корреляции между пользователями i и j $w_{i,j}$:

$$w_{i,j} = \frac{\sum_a (r_{i,a} - \overline{r}_i)(r_{j,a} - \overline{r}_j)}{\sqrt{\sum_a (r_{i,a} - \overline{r}_i)^2} \sqrt{\sum_a (r_{j,a} - \overline{r}_j)^2}},$$

где $r_{i,a}$ — рейтинг фильма a, выставленный пользователем i; $\overline{r}_i$ — средний рейтинг, выставленный пользователем i; $r_{j,a}$ — рейтинг фильма a, выставленный пользователем j.

Далее вычисляем рейтинг фильма *a* для пользователя *i*, который он не выставил:

$$\hat{r}_{i,a} = \overline{r}_i + \frac{\sum_j \left(r_{j,a} - \overline{r}_j\right) w_{i,j}}{\sum_j \left|w_{i,j}\right|}.$$

Метод k средних (k-means)

Итерационный метод, в котором на первой стадии наносят случайным образом центры кластеров и определяют точки, которые к ним относятся. Затем пересчитываются координаты центров, и алгоритм повторяется. Остановка алгоритма происходит тогда, когда центры кластеров перестают сдвигаться.

Приведем алгоритм метода:

- выбираем количество кластеров;
- наносим центры кластеров;
- определяем для каждой точки центр, к которому она ближе по выбранной мере (обычно евклидово расстояние, реже манхэттенское);
- пересчитываем координаты центров кластеров как среднее значение координат всех точек кластера;
- повторяем, пока расстояние до центра кластера не достигнет значения меньше заданной величины.

Недостаток метода состоит в том, что нужно задавать начальное количество кластеров и их центры.

Плотностной алгоритм пространственной кластеризации с присутствием шума (Density-based spatial clustering of applications with noise — DBSCAN)

Плотностной алгоритм пространственной кластеризации с присутствием шума основан на определении плотности точек и объединении соседних точек в кластеры.

Берем точку и смотрим, есть ли у нее соседи ближе, чем на заданное расстояние. Если есть, то причисляем этих соседей к кластеру. В конце алгоритма точки, у которых более 3 соседей, считаем коренными точками, менее 3, но все же есть соседи, — краевыми точками, нет соседей — выбросы.

Математическое описание:

- введем симметричную функцию расстояния $\rho(x, y)$ и константы ε и m;
- ε-окрестность объекта — это область $E(x)$, для которой $\forall y$: $\rho(x, y) \le \varepsilon$;
- корневой (ядерный) объект степени m — это объект, ε-окрестность которого содержит не менее m объектов: $|E(x)| \ge m$;
- объект p непосредственно плотно достижим из объекта q, если $p \in E(q)$ и q-корневой объект;
- объект p плотно достижим из объекта q, если $\exists p_1, p_2, ..., p_n$, $p_1 = q$, $p_n = p$, такие, что $\forall i \in 1 \ldots n-1$: p_{i+1} непосредственно плотно достижим из p_i.

Приведем алгоритм метода:

- выбирается корневой объект p;
- его непосредственно плотно достижимые соседи помещаются в список обхода;
- рассматриваем список обхода;
- если объект из списка обхода корневой, то добавляются в список обхода все его соседи.

Иерархические методы кластеризации

Иерархические алгоритмы кластеризации, называемые также алгоритмами таксономии, строят не одно разбиение выборки на непересекающиеся классы, а систему вложенных разбиений. Результат таксономии обычно представляется в виде таксономического дерева — дендрограммы.

Типы иерархических методов:

- дивизимные или нисходящие алгоритмы разбивают выборку на все более и более мелкие кластеры;
- агломеративные или восходящие алгоритмы, в которых объекты объединяются во все более и более крупные кластеры.

Агломеративная иерархическая кластеризация

В этом методе на первом этапе каждый объект считается кластером. Далее рассчитывается расстояние между объектами по выбранной метрике (описанных ранее). Два ближайших объекта объединяются в кластер. Затем объединяются более дальние объекты. В конечном итоге получится один кластер.

На рис. 30 представлены этапы кластеризации. На каждом этапе объекты объединяются в кластеры, а по вертикали откладывается расстояние между объединяемыми объектами или кластерами.

По дендрограмме можно сделать срез на любом этапе кластеризации и получить нужную глубину кластеризации. Преимуществом этого метода является то, что сразу строится полная кластеризация, а затем можно получить срезы любого этапа кластеризации.

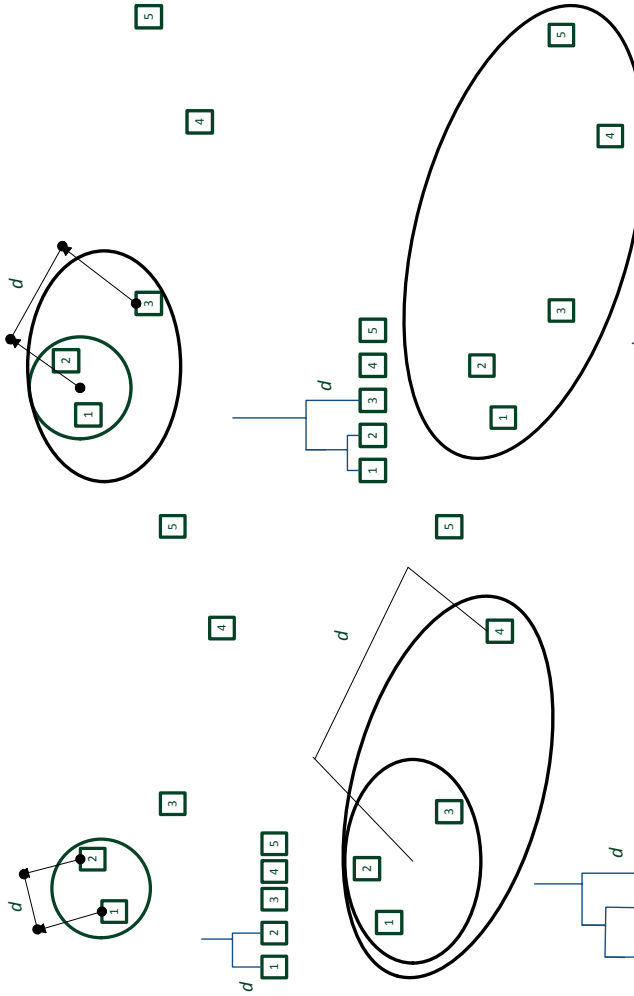

Рис. 30. Этапы кластеризации

Контрольные вопросы

1. Нужно ли знать количество кластеров в методе *k*-means?
2. Задается ли количество кластеров в методе DBSCAN?
3. Какой из описанных методов кластеризации более подходит для выявления кластеров сложной формы, такой как спираль?

Оглавление

Для заметок

Учебное издание

Лимановская Оксана Викторовна,
Алферьева Татьяна Игоревна

**ОСНОВЫ
МАШИННОГО ОБУЧЕНИЯ**

Редактор И. В. Коршунова
Верстка О. П. Игнатьевой

Подписано в печать 02.07.2020. Формат 60×84/16.
Бумага офсетная. Цифровая печать. Усл. печ. л. 5,1.
Уч.-изд. л. 4,1. Тираж 100 экз. Заказ 100.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: +7 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620083, Екатеринбург, ул. Тургенева, 4
Тел.: +7 (343) 358-93-06, 350-58-20, 350-90-13
Факс: +7 (343) 358-93-06
http://print.urfu.ru


УНИВЕРСИТЕТСКАЯ КНИГА
2018


9 785799 630157