Урок 1

«Знакомство с машинным обучением»

1.1. Знакомство с машинным обучением

Приветствуем Вас на курсе «обучение на размеченных данных». Это второй курс специализации «Машинное обучение и анализ данных», в котором начинается знакомство собственно с машинным обучением. Центральной темой этого курса является обучение с учителем. На самом деле эта тема была уже затронута в прошлом курсе, когда речь шла про интерполяцию. Интерполяция — задача восстановления функции по нескольким точкам, в которых известны ее значения.

Обучение с учителем — тоже восстановление общей закономерности по конечному числу примеров. Хотя постановки задач похожи, у них есть много отличий в том, как они решаются и какие требования выдвигаются к решению. Эти различия будут обсуждаться в данном курсе.

1.1.1. Пример: понравится ли фильм пользователю

Для начала будет рассмотрен не очень сложный пример, на котором можно понять, в чем заключается суть обучения с учителем и машинного обучения. Пусть есть некоторый сайт, посвященный кино, на который можно зайти, найти страницу нужного фильма, прочитать информацию про него: когда он снят, кто в нем играет и какой бюджет у этого фильма, а также, возможно, купить его и посмотреть. Пусть есть некоторые пользователи, которые находят страницу нужного фильма, читают и задаются вопросом «смотреть или нет?». Необходимо понять, понравится ли пользователю фильм, если выдать ему рекомендацию о фильме. Есть несколько подходов к решению:

- **Подход первый**, самый глупый — дать пользователю посмотреть этот фильм.
- **Второй подход** — дать случайный ответ и показать случайную рекомендацию. В обоих случаях пользователь может быть разочарован фильмом и он будет недоволен сайтом.
- **Третий подход** — пригласить психолога-киномана, чтобы разрешить ситуацию. Этот человек оценит пользователя, оценит фильм и поймет, понравится ли этот фильм этому пользователю, сопоставив информацию. Этот подход довольно сложный. Скорее всего таких специалистов не очень много, и будет сложно отмасштабировать это решение на миллионы пользователей сайта. Но на самом деле это не нужно.

Существует множество примеров — ситуаций, когда другие пользователи заходили на страницы фильмов, принимали решение посмотреть фильм и далее ставили оценку, по которой можно понять, понравился им фильм или нет. Задача машинного обучения состоит в восстановлении общей закономерности из информации в этих примерах.

1.1.2. Основные обозначения

В рамках данного курса будут использоваться следующие обозначения: x — объект, $\mathbb{X}$ — пространство объектов, $y = y(x)$ — ответ на объекте x, $\mathbb{Y}$ — пространство ответов.

Объектом называется то, для чего нужно сделать предсказание. В данном примере объектом является пара (пользователь, фильм). Пространство объектов — это множество всех возможных объектов, для ко-

торых может потребоваться делать предсказание. В данном примере это множество всех возможных пар (пользователь, фильм).

Ответом будет называться то, что нужно предсказать. В данном случае ответ — понравится пользователю фильм или нет. Пространство ответов, то есть множество всех возможных ответов, состоит из двух возможных элементов: -1 (пользователю фильм не понравился) и +1 (понравился).

Признаковым описанием объекта называется совокупность всех признаков:

$$x = (x^1, x^2, \ldots, x^d).$$

Признак - это число, характеризующее объект. Признаковое описание является d-мерным вектором.

1.1.3. Выборка, алгоритм обучения

Центральным понятием машинного обучения является обучающая выборка $X = (x_i, y_i)_{i=1}^{\ell}$. Это те самые примеры, на основе которых будет строиться общая закономерность. Отдельная задача — получение обучающей выборки. В вышеупомянутом случае y_i - это оценка фильма пользователем.

Предсказание будет делаться на основе некоторой модели (алгоритма) $a(x)$, которая представляет из себя функцию из пространства $\mathbb{X}$ в пространство $\mathbb{Y}$. Эта функция должна быть легко реализуема на компьютере, чтобы ее можно было использовать в системах машинного обучения. Примером такой модели является линейный алгоритм:

$$a(x) = \text{sign}(w_0 + w_1 x^1 + \ldots + w_d x^d).$$

Операция взятия знака sign берется ввиду того, что пространство $\mathbb{Y}$ состоит из двух элементов.

Не все алгоритмы подходят для решения задачи. Например константный алгоритм $a(x) = 1$ не подходит. Это довольно бесполезный алгоритм, который вряд ли принесет пользу сайту.

Поэтому вводится некоторая характеристика качества работы алгоритма — функционал ошибки. $Q(a, X)$ — ошибка алгоритма a на выборке X. Например, функционал ошибки может быть долей неправильных ответов. Следует особо отметить, что Q называется функционалом ошибки, а не функцией. Это связано с тем, что первым его аргументом является функция.

Задача обучения состоит в подборе такого алгоритма a, для которого достигается минимум функционала ошибки. Лучший в этом смысле алгоритм выбирается из некоторого семейства $\mathbb{A}$ алгоритмов.

1.1.4. Решающие пни

Простейшим примером семейства алгоритмов являются решающие пни:

$$\mathbb{A} = \left\{ \left[x^j < t \right] | \forall j, t \right\}.$$

Здесь квадратные скобки соответствуют так называемой нотации Айверсона. Если логическое выражение внутри этих скобок — истина, то значение скобок равно 1, в ином случае — нулю.

Алгоритм работает следующим образом. Если значение определенного признака x^j меньше некоторого порогового значения t, то данный алгоритм возвращает ответ 0 (фильм не понравится), в ином случае — +1 (пользователю фильм понравится).

Решающие пни могут быть использованы для построения сложных композиций алгоритмов.

1.2. Обучение на размеченных данных

1.2.1. Постановка задачи

В этом разделе речь пойдет о том, какие бывают типы задач при обучении на размеченных данных, или обучении с учителем. Общая постановка задачи обучения с учителем следующая. Для обучающей выборки $X = (x_i, y_i)_{i=1}^{\ell}$ нужно найти такой алгоритм $a \in \mathbb{A}$, на котором будет достигаться минимум функционала ошибки:

$$Q(a, X) \to \min_{a \in \mathbb{A}}.$$

В зависимости от множества возможных ответов $\mathbb{Y}$, задачи делятся на несколько типов.

1.2.2. Задача бинарной классификации

В задаче бинарной классификации пространство ответов состоит из двух ответов $\mathbb{Y} = \{0, 1\}$. Множество объектов, которые имеют один ответ, называется классом. Говорят, что нужно относить объекты к одному из двух классов, другими словами, классифицировать эти объекты.

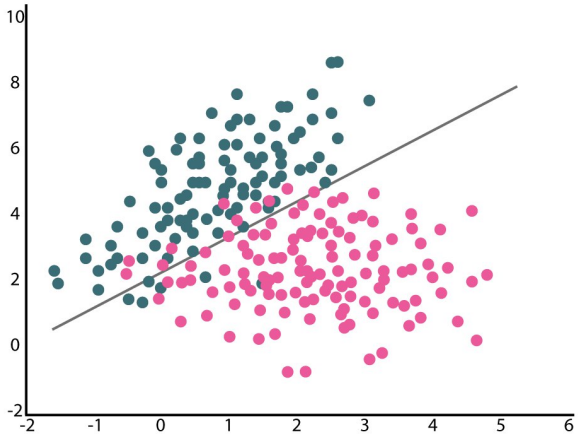

Рис. 1.1: Задача бинарной классификации

Примеры задач бинарной классификации:
- Понравится ли пользователю фильм?
- Вернет ли клиент кредит?

1.2.3. Задача многоклассовой классификации

Классов может быть больше, чем два. В таком случае имеет место задача многоклассовой классификации.

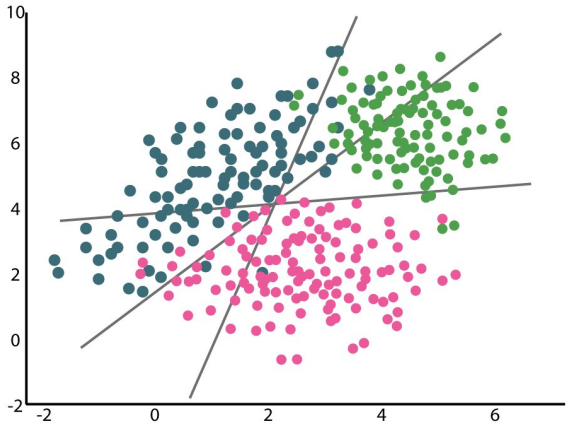

Рис. 1.2: Задача многоклассовой классификации

Примеры задач многоклассовой классификации:
- Из какого сорта винограда сделано вино?
- Какая тема статьи?
- Машина какого типа изображена на фотографии: мотоцикл, легковая или грузовая машина?

1.2.4. Задача регрессии

Когда y является вещественной переменной, говорят о задаче регрессии.

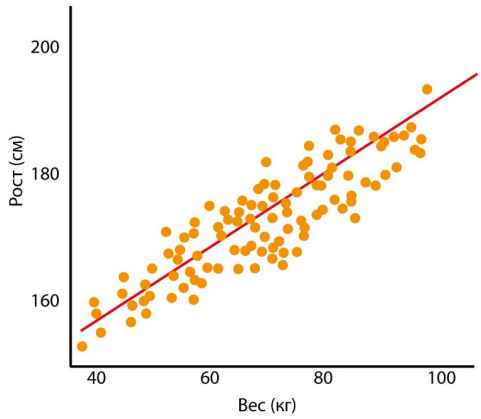

Рис. 1.3: Задача регрессии

Примеры задач регрессии:
- Предсказание температуры на завтра.
- Прогнозирование выручки магазина за год.
- Оценка возраста человека по его фото.

1.2.5. Задача ранжирования

Еще одним примером задачи обучения с учителем является задача ранжирования. Эта задача довольно тяжелая, и речь о ней в данном курсе не пойдет, но знать о ней полезно. Мы сталкиваемся с ней каждый день, когда ищем что-либо в интернете. После того, как мы ввели запрос, происходит ранжирование страниц по релевантности их запросу, то есть для каждой страницы оценивается ее релевантность в виде числа, а затем страницы сортируются по убыванию релевантности. Задача состоит в предсказании релевантности для пары (запрос, страница).

1.3. Обучение без учителя

В этом разделе мы обсудим, какие бывают постановки задач машинного обучения, кроме обучения с учителем.

Обучением с учителем называются такие задачи, в которых есть и объекты, и истинные ответы на них. И нужно по этим парам восстановить общую зависимость. Задача обучения без учителя — это такая задача, в которой есть только объекты, а ответов нет. Также бывают «промежуточные» постановки. В случае частичного обучения есть объекты, некоторые из которых с ответами. В случае активного обучения получение ответа обычно очень дорого, поэтому алгоритм должен сначала решить, для каких объектов нужно узнать ответ, чтобы лучше всего обучиться.

Рассмотрим несколько примеров постановки задач без учителя.

1.3.1. Задача кластеризации

Первый пример — задача кластеризации. Дано множество объектов. Необходимо найти группы похожих объектов. Есть две основные проблемы: не известно количество кластеров и не известны истинные кластеры, которые нужно выделять. Поэтому задача решается очень тяжело — здесь невозможно оценить качество решения. Этим и отличается задача классификации — там тоже нужно делить объекты на группы, но в классификации группы, а точнее классы, фиксированы, и известны примеры объектов из разных групп.

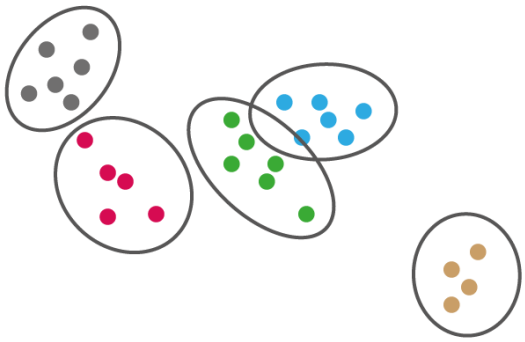

Рис. 1.4: Задача кластеризации

Примеры задач кластеризации:
- Сегментация пользователей (интернет-магазина или оператора связи)
- Поиск схожих пользователей в социальных сетях
- Поиск генов с похожими профилями экспрессии

1.3.2. Задача визуализации

Второй пример — задача визуализации: необходимо нарисовать многомерную (а конкретно, d-мерную) выборку так, чтобы изображение наглядно показывало структуру объектов.

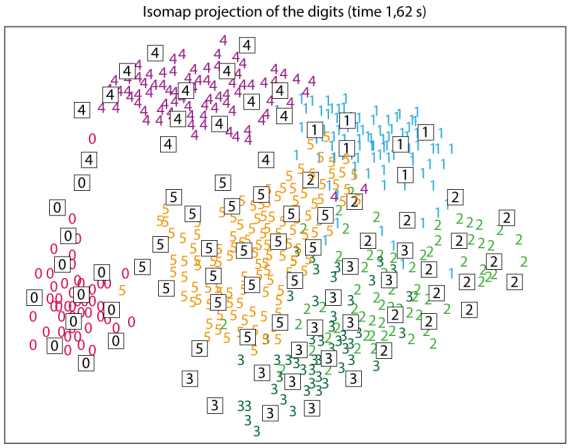

Рис. 1.5: Задача визуализации

Примером задачи визуализации является задача визуализации набора данных MNIST. Этот набор данных был получен в результате оцифровки рукописных начертаний цифр. Каждый скан цифры характеризуется

вектором признаков - яркостей отдельных пикселей. Необходимо таким образом отобразить этот набор данных на плоскость, чтобы разные цифры оказались в разных ее областях.

1.3.3. Поиск аномалий

Третий пример задачи обучения без учителя — поиск аномалий. Необходимо обнаружить, что данный объект не похож на все остальные, то есть является аномальным.

При обучении есть примеры только обычных, не аномальных, объектов. А примеров аномальных объектов либо нет вообще, либо настолько мало, что невозможно воспользоваться классическими методами обучения с учителем (методами бинарной классификации).

При этом задача очень важная. Например, к такому типу задач относится:

- Определение поломки в системах самолета (по показателям сотен датчиков)
- Определение поломки интернет—сайта
- Выявление проблем в модели машинного обучения.

Все упомянутые задачи не будут обсуждаться в рамках данного курса. Им будет посвящен следующий курс — «Поиск структуры в данных».

1.4. Признаки в машинном обучении

В этом разделе речь пойдет о признаках в машинном обучении. Существует несколько классов, или типов признаков. И у всех свои особенности — их нужно по-разному обрабатывать и по-разному учитывать в алгоритмах машинного обучения. В данном разделе будет обсуждаться используемая терминология, о самих же особенностях речь пойдет в следующих уроках.

Признаки описывают объект в доступной и понятной для компьютера форме. Множество значений j-го признака будет обозначаться D_j.

1.4.1. Бинарные признаки

Первый тип признаков — бинарные признаки. Они принимают два значения: $D_j = \{0, 1\}$. К таковым относятся:

- Выше ли доход клиента среднего дохода по городу?
- Цвет фрукта — зеленый?

Если ответ на вопрос да — признак полагается равным 1, если ответ на вопрос нет — то равным 0.

1.4.2. Вещественные признаки

Более сложный класс признаков — вещественные признаки. В этом случае $D_j = \mathbb{R}$. Примерами таких признаков являются:

- Возраст
- Площадь квартиры
- Количество звонков в call-центр

Множество значений последнего указанного признака, строго говоря, является множеством натуральных чисел $\mathbb{N}$, а не $\mathbb{R}$, но такие признаки тоже считают вещественными.

1.4.3. Категориальные признаки

Следующий класс признаков — категориальные признаки. В этом случае D_j — неупорядоченное множество. Отличительная особенность категориальных признаков — невозможность сравнения «больше-меньше» значений признака. К таковым признакам относятся:

- Цвет глаз
- Город
- Образование (В некоторых задачах может быть введен осмысленный порядок)

Категориальные признаки очень трудны в обращении — до сих пор появляются способы учета этих признаков в тех или иных методах машинного обучения.

1.4.4. Порядковые признаки

Частным случаем категориальных признаков являются порядковые признаки. В этом случае D_j — упорядоченное множество. Примеры:

- Роль в фильме (Первый план, второй план, массовка)
- Тип населенного пункта (упорядочены по населенности)
- Образование

Хотя и порядковые, и вещественные признаки упорядочены, они отличаются тем, что в случае порядковых признаков «расстояние» между двумя значениями признака не имеет смысла. Например, отличие значения 3 от значения 2 может быть не таким существенным, как отличие 1 от 0.

1.4.5. Множествозначные признаки

Множествозначный признак — это такой признак, значением которого на объекте является подмножество некоторого множества. Пример:

- Какие фильмы посмотрел пользователь
- Какие слова входят в текст

1.4.6. Распределение признака

Далее речь пойдет о проблемах, с которыми можно столкнуться при работе с признаками. Первая из них — существование выбросов. Выбросом называется такой объект, значение признака на котором отличается от значения признака на большинстве объектов.

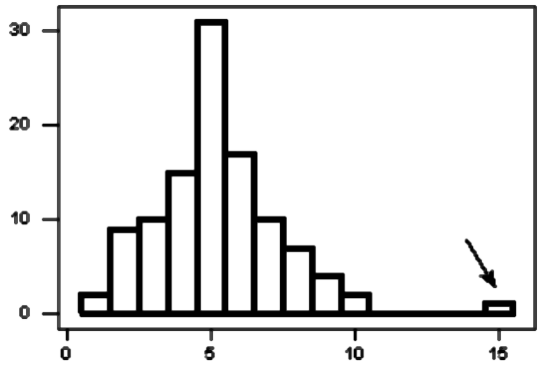

Рис. 1.6: Пример выброса

Наличие выбросов представляет сложность для алгоритмов машинного обучения, которые будут пытаться учесть и их тоже. Поскольку выбросы описываются совершенно другим законом, чем основное множество объектов, выбросы обычно исключают из данных, чтобы не мешать алгоритму машинного обучения искать закономерности в данных.

Проблема может быть и в том, как распределен признак. Не всегда признак имеет такое распределение, которое позволяет ответить на требуемый вопрос. Например, может быть слишком мало данных о клиентах из небольшого города, так как собрать достаточную статистику не представлялось возможным.

Урок 4

Метрики качества

4.1. Метрики качества в задачах регрессии

4.1.1. Применение метрик качества в машинном обучении

Метрики качества могут использоваться:

- Для задания функционала ошибки (используется при обучении).
- Для подбора гиперпараметров (используется при измерении качества на кросс-валидации). В том числе можно использовать другую метрику, которая отличается от метрики, с помощью которой построен функционал ошибки.
- Для оценивания итоговой модели: пригодна ли модель для решения задачи.

Далее мы рассмотрим, какие метрики можно использовать в задачах регрессии.

4.1.2. Среднеквадратичная ошибка

Первая метрика, о которой уже шла речь — среднеквадратичная ошибка:

$$MSE(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 .$$

Такой функционал легко оптимизировать, используя, например, метод градиентного спуска.

Этот функционал сильно штрафует за большие ошибки, так как отклонения возводятся в квадрат. Это приводит к тому, что штраф на выбросе будет очень сильным, и алгоритм будет настраиваться на выбросы. Другими словами, алгоритм будет настраиваться на такие объекты, на которые не имеет смысл настраиваться.

4.1.3. Средняя абсолютная ошибка

Похожий на предыдущий функционал качества — средняя абсолютная ошибка:

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

Этот функционал сложнее минимизировать, так как у модуля производная не существует в нуле. Но у такого функционала больше устойчивость к выбросам, так как штраф за сильное отклонение гораздо меньше.

4.1.4. Коэффициент детерминации

Коэффициент детерминации $R^2(a, X)$:

$$R^2(a, X) = 1 - \frac{\sum_{i=1}^{\ell} \big(a(x_i) - y_i\big)^2}{\sum_{i=1}^{\ell} (y_i - \bar{y})}, \qquad \bar{y} = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i,$$

позволяет интерпретировать значение среднеквадратичной ошибки. Этот коэффициент показывает, какую долю дисперсии (разнообразия ответов) во всем целевом векторе y модель смогла объяснить.

Для разумных моделей коэффициент детерминации лежит в следующих пределах:

$$0 \leq R^2 \leq 1,$$

причем случай $R^2 = 1$ соответствует случаю идеальной модели, $R^2 = 0$ — модели на уровне оптимальной «константной», а $R^2 < 0$ — модели хуже «константной» (такие алгоритмы никогда не нужно рассматривать). Оптимальным константым алгоритмом называется такой алгоритм, который возвращает всегда среднее значение ответов $\bar{y}$ для объектов обучающей выборки.

4.1.5. Несимметричные потери

До этого рассматривались симметричные модели, то есть такие, которые штрафуют как за недопрогноз, так и за перепрогноз. Но существуют такие задачи, в которых эти ошибки имеют разную цену.

Пусть, например, требуется оценить спрос на ноутбуки. В этом случае заниженный прогноз приведет к потере лояльности покупателей и потенциальной прибыли (будет закуплено недостаточное количество ноутбуков), а завышенный — только к не очень большим дополнительным расходам на хранение непроданных ноутбуков. Чтобы учесть это, функция потерь должна быть несимметричной и сильнее штрафовать за недопрогноз, чем за перепрогноз.

4.1.6. Квантильная ошибка

В таких случаях хорошо подходит квантильная ошибка или квантильная функция потерь:

$$\rho_\tau(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \Big((\tau - 1)\big[y_i < a(x_i)\big] + \tau\big[y_i \geq a(x_i)\big] \Big) \left(y_i - a(x_i)\right).$$

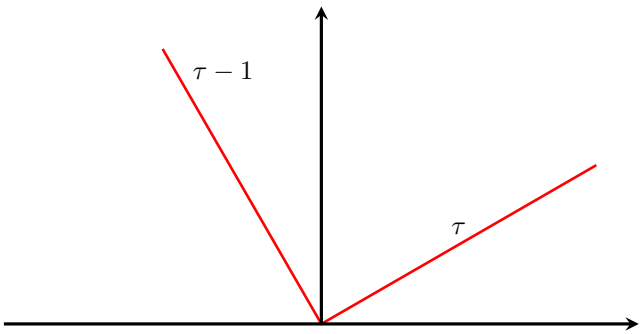

Рис. 4.1: График квантильной функции потерь

Параметр $\tau \in [0, 1]$ определяет то, за что нужно штрафовать сильнее — за недопрогноз или перепрогноз. Если τ ближе к 1, штраф будет больше за недопрогноз, а если, наоборот, ближе к 0 — за перепрогноз.

4.1.7. Вероятностный смысл квантильной ошибки

Чтобы разобраться, почему такая функция потерь называется квантильной, нужно разобраться с ее вероятностным смыслом. Пусть один и тот же объект x с одним и тем же признаковым описанием повторяется в выборке n раз, но на каждом из повторов — свой ответ $y_1, ..., y_n$.

Такое может возникнуть при измерении роста человека. Измерения роста одного и того же человека могут отличаться ввиду ошибки прибора, а также зависеть от самого человека (может сгорбиться или выпрямиться).

При этом алгоритм должен для одного и того же признакового описания возвращать одинаковый прогноз. Другими словами, необходимо решить, какой прогноз оптимален для x с точки зрения различных функционалов ошибки.

Оказывается, что если используется квадратичный функционал ошибки, то наиболее оптимальным прогнозом будет средний ответ на объектах, если абсолютный, то медиана ответов. Если же будет использоваться квантильная функция потерь, наиболее оптимальным прогнозом, будет τ-квантиль. В этом и состоит вероятностный смысл квантильной ошибки.

4.2. Метрика качества классификации

В этом блоке речь пойдет о том, как измерять качество в задачах классификации.

4.2.1. Доля правильных ответов

Как меру качества в задачах классификации естественно использовать долю неправильных ответов:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}[a(x_i) \neq y_i]$$

Однако в задачах классификации принято выбирать метрики таким образом, чтобы их нужно было максимизировать, тогда как в задачах регрессии — так, чтобы их нужно было минимизировать. Поэтому определяют:

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

Эта метрика качества проста и широко используется, однако имеет несколько существенных недостатков.

4.2.2. Несбалансированные выборки

Первая проблема связана с несбалансированными выборками. Показателен следующий пример. Пусть в выборке 1000 объектов, из которых 950 относятся к классу -1 и 50 — к классу $+1$. Рассматривается бесполезный (поскольку не восстанавливает никаких закономерностей в данных) константный классификатор, который на всех объектах возвращает ответ -1. Но доля правильных ответов на этих данных будет равна 0.95, что несколько много для бесполезного классификатора.

Чтобы «бороться» с этой проблемой, используется следующий факт. Пусть q_0 — доля объектов самого крупного класса, тогда доля правильных ответов для разумных алгоритмов $accuracy \in [q_0, 1]$, а не $[1/2, 1]$, как это можно было бы ожидать. Поэтому, если получается высокий процент правильных ответов, это может быть связано не с тем, что построен хороший классификатор, а с тем, что какого-то класса сильно больше, чем остальных.

4.2.3. Цены ошибок

Вторая проблема с долей верных ответов состоит в том, что она никак не учитывает разные цены разных типов ошибок. Тогда как цены действительно могут быть разными.

Например, в задаче кредитного скоринга, то есть в задаче принятия решения относительно выдачи кредита, сравниваются две модели. При использовании первой модели кредит будет выдан 100 клиентам, 80 из которых его вернут. Во второй модели, более консервативной, кредит был выдан только 50 клиентам, причем вернули его в 48 случаях. То, какая из двух моделей лучше, зависит от того, цена какой из ошибок выше: не дать кредит клиенту, который мог бы его вернуть, или выдать кредит клиенту, который его не вернет. Таким образом, нужны дополнительные метрики качества, которые учитывают цены той или иной ошибки.

4.3. Точность и полнота

4.3.1. Цены ошибок

В этом разделе пойдет речь о метриках качества классификации, которые позволяют учитывать разные цены ошибок. В конце предыдущего разделе уже было сказано, что цены ошибок действительно могут быть разными. Так, в задаче банковского скоринга необходимо принять решение, что хуже: выдать кредит «плохому» клиенту или не выдать кредит «хорошему» клиенту. Доля верных ответов не способна учитывать цены разных ошибок и поэтому не может дать ответа на этот вопрос.

4.3.2. Матрица ошибок

Удобно классифицировать различные случаи, как соотносятся между собой результат работы алгоритма и истинный ответ, с помощью так называемой матрицы ошибок.

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

Когда алгоритм относит объект к классу $+1$, говорят, что алгоритм срабатывает. Если алгоритм сработал и объект действительно относится к классу $+1$, имеет место верное срабатывание (true positive), а если объект на самом деле относится к классу -1, имеет место ложное срабатывание (false positive).

Если алгоритм дает ответ -1, говорят, что он пропускает объект. Если имеет место пропуск объекта класса $+1$, то это ложный пропуск (false negative). Если же алгоритм пропускает объект класса -1, имеет место истинный пропуск (true negative).

Таким образом, существуют два вида ошибок: ложные срабатывания и ложные пропуски. Для каждого из них нужна своя метрика качества, чтобы измерить, какое количество ошибок какого типа совершается.

4.3.3. Точность и полнота

Пусть для примера рассматриваются две модели $a_1(x)$ и $a_2(x)$. Выборка состоит из 200 объектов, из которых 100 относятся к классу 1 и 100 — к классу -1. Матрицы ошибок имеют вид:

	$y = 1$	$y = -1$
$a_1(x) = 1$	80	20
$a_1(x) = -1$	20	80

	$y = 1$	$y = -1$
$a_2(x) = 1$	48	2
$a_2(x) = -1$	52	98

Введем две метрики. Первая метрика, точность (precision), показывает, насколько можно доверять классификатору в случае срабатывания:

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

Вторая метрика, полнота (recall), показывает, на какой доле истинных объектов первого класса алгоритм срабатывает:

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

В примере выше точность и полнота первого алгоритма оказываются равными:

$$\begin{aligned} precision(a_1, X) &= 0.8, & precision(a_2, X) &= 0.96 \\ recall(a_1, X) &= 0.8, & recall(a_2, X) &= 0.48 \end{aligned}$$

Вторая модель является очень точной, но в ущерб полноте.

4.3.4. Примеры использования точности и полноты

Первый пример — использование в задаче кредитного скоринга. Пусть в задаче кредитного скоринга ставится условие, что неудачных кредитов должно быть не больше 5%. В таком случае задача является задачей максимизации полноты при условии $\text{precision}(a, X) \geq 0.95$.

Второй пример — использование в медицинской диагностике. Необходимо построить модель, которая определяет, есть или нет определенное заболевание у пациента. При этом требуется, чтобы были выявлены как минимум 80% пациентов, которые действительно имеют данное заболевание. Тогда ставят задачу максимизации точности при условии $\text{recall}(a, X) \geq 0.8$.

Следует особо обратить внимание на то, как точность и полнота работают в случае несбалансированных выборок. Пусть рассматривается выборка со следующей матрицей ошибок:

	$y = 1$	$y = -1$
$a(x) = 1$	10	20
$a(x) = -1$	90	10000

Доля верных ответов (accuracy), точность(precision) и полнота(recall) для данного случая:

$$accuracy(a, X) = 0.99, \qquad precision(a, X) = 0.33, \qquad recall(a, X) = 0.1.$$

То, что доля верных ответов равняется 0.99, ни о чем не говорит: алгоритм все равно делает 66% ложных срабатываний и выявляет только 10% положительных случаев. Благодаря введению точности и полноты становится понятно, что алгоритм нужно улучшать.

4.4. Объединение точности и полноты

В этом разделе пойдет речь о том, как соединить точность и полноту в одну метрику качества классификации.

4.4.1. Точность и полнота (напоминание)

Точность показывает, насколько можно доверять классификатору в случае срабатывания:

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

Полнота показывает, на какой доле истинных объектов первого класса алгоритм срабатывает:

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

В некоторых задачах есть ограничения на одну из этих метрик, тогда как по второй метрике будет производиться оптимизация. Но в некоторых случаях хочется максимизировать и точность, и полноту одновременно. Встает вопрос об объединении этих двух метрик.

4.4.2. Арифметическое среднее

Единая метрика может быть получена как арифметическое среднее точности и полноты:

$$A = \frac{1}{2}(precision + recall)$$

Пусть есть алгоритм, точность которого равна 10%, а полнота — 100%:

$$precision = 0.1 \qquad recall = 1.$$

Это может быть случай, когда в выборке всего 10% объектов класса +1, а алгоритм является константным и всегда возвращает +1. Очевидно, что этот алгоритм плохой, но введенная выше метрика для него равна $A = 0.55$. В свою очередь другой, гораздо более лучший алгоритм, с $precision = 0.55$ и $recall = 55$ также характеризуется $A = 0.55$.

Ситуация, когда константный и разумный алгоритмы могут лежать на одной линии, является недопустимой, поэтому следует искать другой способ построения единой метрики.

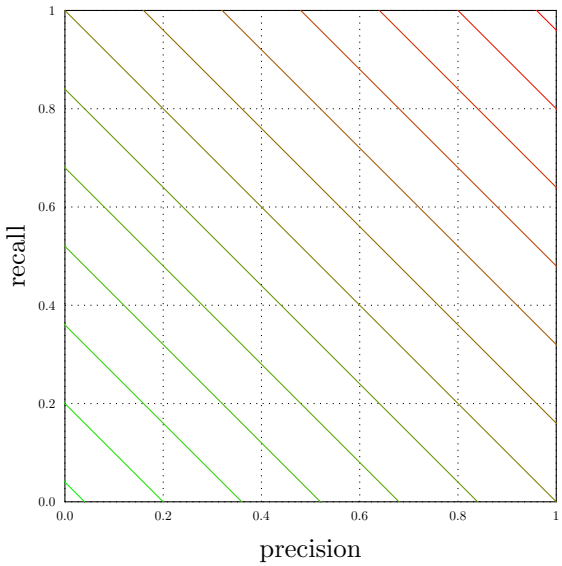

Рис. 4.2: Линии $A = \frac{1}{2}(precision + recall) = const$ в координатах precision–recall

4.4.3. Минимум

Чтобы констрантный и разумный алгоритмы не лежали на одной линии уровня, можно рассматривать:

$$M = \min(precision, recall).$$

Данный подход решает вышеупомянутую проблему, например:

$$precision = 0.05, \quad recall = 1 \qquad \Longrightarrow \qquad M = 0.05.$$

Но есть другой нюанс: два алгоритма, для которых точности одинаковы, но отличаются значения полноты, будут лежать на одной линии уровня M:

$$\begin{aligned} precision = 0.4, \quad recall = 0.5 \qquad &\Longrightarrow \qquad M = 0.4, \\ precision = 0.4, \quad recall = 0.9 \qquad &\Longrightarrow \qquad M = 0.4. \end{aligned}$$

Такое тоже недопустимо, так как второй алгоритм существенно лучше первого.

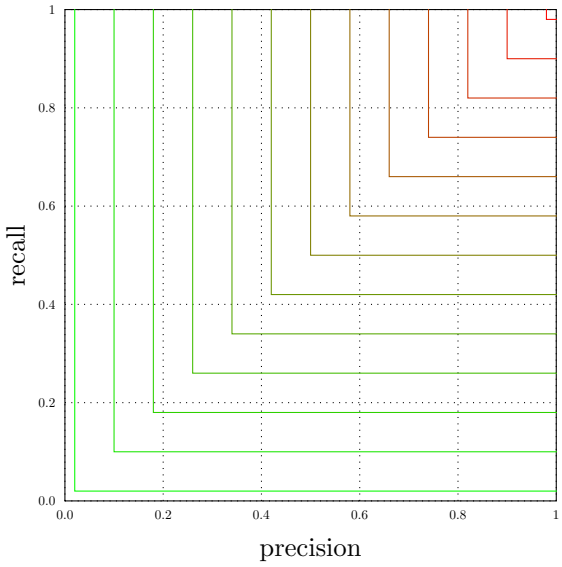

Рис. 4.3: Линии $M = \min(precision, recall) = const$ в координатах precision–recall

4.4.4. F-мера

«Сгладить» минимум можно с помощью гармонического среднего, или F-меры:

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

Для двух упомянутых выше алгоритма значения F-меры, в отличие от M, будут отличаться:

$$precision = 0.4, \quad recall = 0.5 \quad \Longrightarrow \quad F = 0.44,$$
$$precision = 0.4, \quad recall = 0.9 \quad \Longrightarrow \quad F = 0.55.$$

Если необходимо отдать предпочтение точности или полноте, следует использовать расширенную F-меру, в которой есть параметр β:

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

Например, при $\beta = 0.5$ важнее оказывается полнота, а в случае $\beta = 2$, наоборот, важнее оказывается точность.

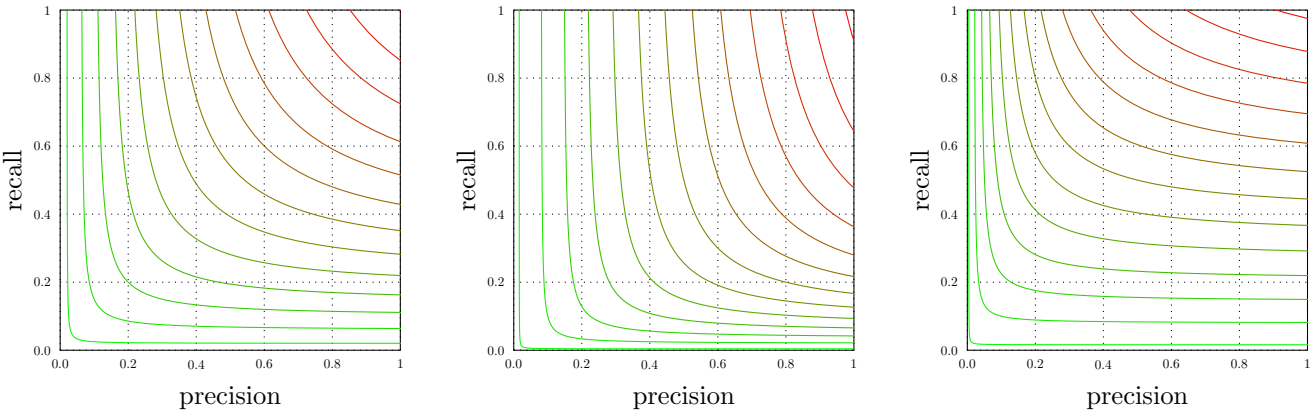

Рис. 4.4: Линии $F = const$ в координатах precision–recall при значениях $\beta = 1$, $\beta = 0.5$ и $\beta = 2$ соответственно

4.5. Качество оценок принадлежности классу

4.5.1. Оценка принадлежности

Многие алгоритмы бинарной классификации устроены следующим образом: сначала вычисляется некоторое вещественное число $b(x)$, которое сравнивается с порогом t.

$$a(x) = [b(x) > t],$$

где $b(x)$ — оценка принадлежности классу $+1$. Другими словами, $b(x)$ выступает в роли некоторой оценки уверенности, что x принадлежит классу $+1$.

В случае линейного классификатора $a(x) = [\langle w, x\rangle > t]$ оценка принадлежности классу $+1$ имеет вид $b(x) = \langle w, x\rangle$.

Часто бывает необходимо оценить качество именно оценки принадлежности, а порог выбирается позже из соображений на точность или полноту.

4.5.2. Оценка принадлежности в задаче кредитного скоринга

Пусть рассматривается задачного кредитного скоринга и была построена некоторая функция $b(x)$, которая оценивает вероятность возврата кредита клиентом x. Далее классификатор строится следующим образом:

$$a(x) = [b(x) > 0.5]$$

При этом получилось, что точность (precision) равна 10%, а полнота (recall) — 70%. Это очень плохой алгоритм, так как 90% клиентов, которым будет выдан кредит, не вернут его.

При этом не понятно, в чем дело: был плохо выбран порог или алгоритм не подходит для решения данной задачи. Именно для этого необходимо измерять качество самих оценок $b(x)$.

4.5.3. PR-кривая

Первый способ оценки принадлежности классу основан на использовании кривой точности-полноты. По оси X откладывается полнота, а по оси Y — точность. Каждой точке на этой кривой будет соответствовать классификатор с некоторым значением порога.

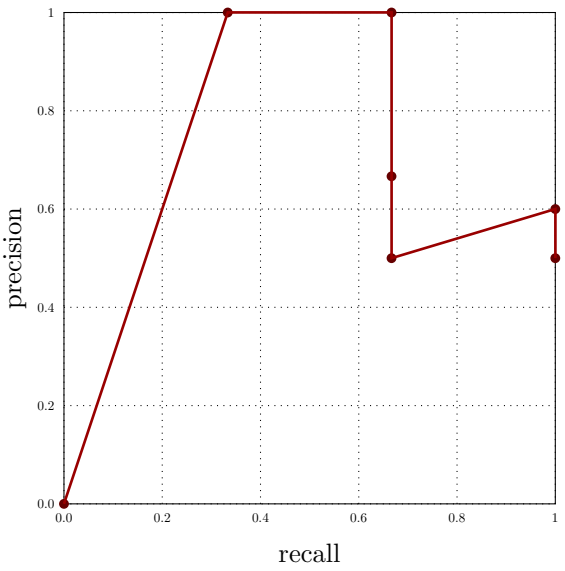

Рис. 4.5: Кривая полноты–точности

Для примера будет приведено построение PR-кривой для выборки из 6 объектов, три из которых относятся к классу 1 и 3 — к классу 0. Соответствующий ей график изображен выше.

b(x)	0.14	0.23	0.39	0.54	0.73	0.90
y	0	1	0	0	1	1

1. При достаточно большом пороге ни один объект не будет отнесен к классу 1. В этом случае и точность и полнота равны 0.
2. При таком пороге, что ровно один объект отнесен к классу 1, точность будет 100% (поскольку этот объект действительно из 1 класса), а полнота — 1/3 (поскольку всего 3 объекта 1 класса).
3. При дальнейшем уменьшении порога уже два объекта отнесены к классу 1, точность также остается 100%, а полнота становится равной 2/3.
4. При таком пороге, что уже три объекта будут отнесены к классу 1, точность становится равной 2/3, а полнота остается такой же.
5. При таком пороге, что четыре объекта отнесены к классу 1, точность уменьшится до 0.5, а полнота опять не изменится.
6. При дальнейшем уменьшении порога уже 5 объектов будут отнесены к 1 классу, полнота станет равной 100%, а точность — 3/5.

В реальных задачах с числом объектов порядка нескольких тысяч или десятков тысяч, кривая точности-полноты выглядит примерно следующим образом.

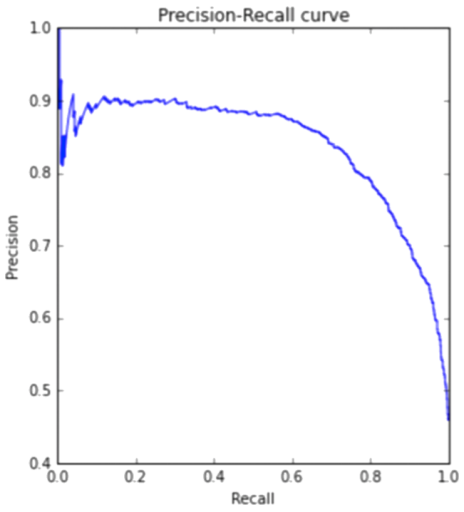

Рис. 4.6: Кривая точности-полноты в реальных задачах с десятками тысяч объектов

Следует отметить, что начинается PR-кривая всегда из точки $(0, 0)$, а заканчивается точной $(1, r)$, где r — доля объектов класса 1.

В случае идеального классификатора, то есть если существует такой порог, что и точность, и полнота равны 100%, кривая будет проходить через точку $(1, 1)$. Таким образом, чем ближе кривая пройдет к этой точке, тем лучше оценки. Площадь под этой кривой может быть хорошей мерой качества оценок принадлежности к классу 1. Такая метрика называется AUC–PRC, или площадь под PR-кривой.

4.5.4. ROC-кривая

Второй способ измерить качество оценок принадлежности к классу 1 — ROC-кривая, которая строится в осях False Positive Rate (ось X) и True Positive Rate (ось Y):

$$FPR = \frac{FP}{FP + TN}, \qquad TPR = \frac{TP}{TP + FN}.$$

ROC-кривая строится аналогично PR-кривой: постепенно рассматриваются случаи различных значений порогов и отмечаются точки на графике. Для упомянутой выше выборки ROC-кривая имеет следующий вид:

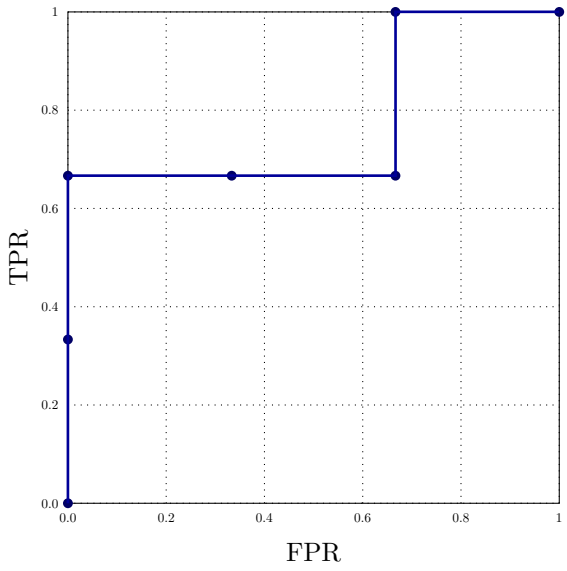

Рис. 4.7: ROC–кривая

В случае с большой выборкой ROC-кривая выглядит следующим образом:

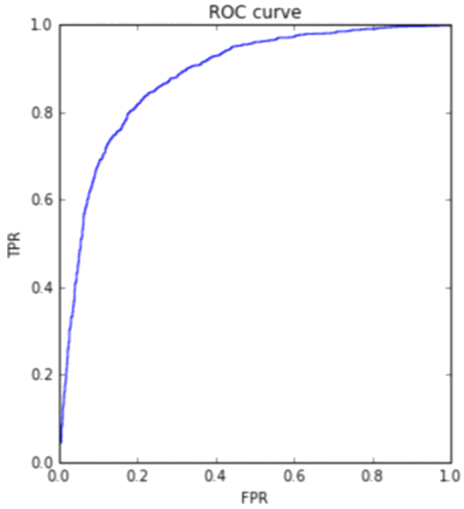

Рис. 4.8: Кривая ROC в реальных задачах с десятками тысяч объектов

Кривая стартует с точки $(0, 0)$ и приходит в точку $(1, 1)$. При этом, если существует идеальный классификатор, кривая должна пройти через точку $(0, 1)$. Чем ближе кривая к этой точке, тем лучше будут оценки, а площадь под кривой будет характеризовать качество оценок принадлежности к первому классу. Такая метрика называется AUC–ROC,или площадь под ROC-кривой.

4.5.5. Особенности AUC-ROC

Как было написано выше, ROC-кривая строится в осях FPR и TPR, которые нормируются на размеры классов:

$$FPR = \frac{FP}{FP + TN}, \qquad TPR = \frac{TP}{TP + FN}.$$

Следовательно, при изменении баланса классов величина AUC-ROC и неизменных свойствах объектов выборки площадь под ROC-кривой не изменится. В случае идеального алгоритма $AUC - ROC = 1$, а в случае худшего $AUC - ROC = \frac{1}{2}$.

Значение $AUC-ROC$ имеет смысл вероятности того, что если были выбраны случайный положительный и случайный отрицаельный объекты выборки, положительный объект получит оценку принадлежности выше, чем отрицательный объект.

4.5.6. Особенности AUC-PRC

PR-кривая строится в осях precision и recall:

$$precision = \frac{TP}{TP + FP}, \qquad recall = \frac{TP}{TP + FN},$$

а следовательно изменяется при изменении баланса классов.

Урок 3

Проблема переобучения и борьба с ней

3.1. Проблема переобучения

3.1.1. Пример: проблема переобучения в задачах классификации

Допустим при решении задачи классификации был построен некоторый алгоритм, например линейный классификатор, причем доля ошибок на объектах из обучающей выборки была равна 0.2, и такая доля ошибок является допустимой.

Но поскольку алгоритм не обладает обобщающей способностью, нет никаких гарантий, что такая же доля ошибок будет для новой выборки. Вполне может возникнуть ситуация, что для новой выборки ошибка станет равной 0.9. Это значит, что алгоритм не смог обобщить обучающую выборку, не смог извлечь из нее закономерности и применить их для классификации новых объектов. При этом алгоритм как-то смог подогнаться под обучающую выборку и показал хорошие результаты при обучении без извлечения истинной закономерности. В этом и состоит проблема переобучения.

3.1.2. Пример: проблема переобучения в задачах линейной регрессии

Глубже понять проблему переобучения можно на данном примере. На следующем графике изображена истинная зависимость и объекты обучающей выборки:

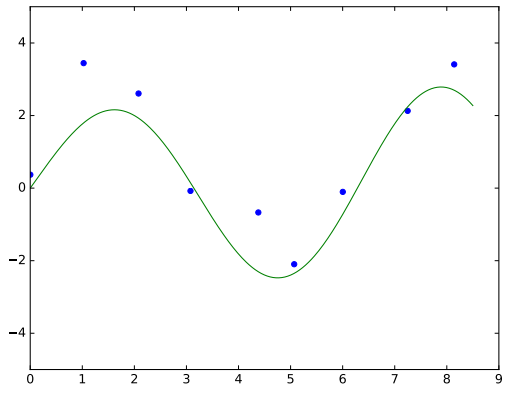

Рис. 3.1: Истинная зависимость (зеленая линия) и элементы обучающей выборки (изображены синими точками).

Видно, что истинная зависимость является нелинейной и имеет два экстремума.

В модели $a(x) = w_0$, после того, как она будет настроена под данные, на графике получается некоторая горизотальная кривая, которая довольно плохо обобщает информацию об объектах из выборки.

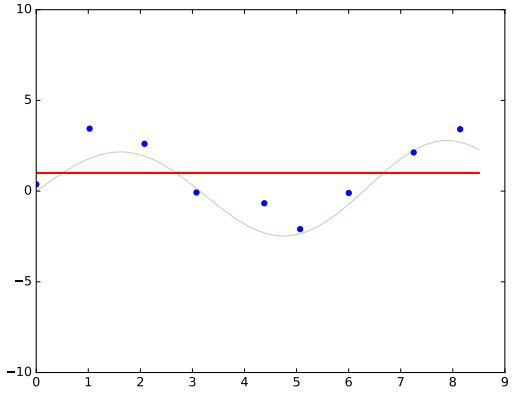

Рис. 3.2: Модель $a(x) = w_0$.

Имеет место недообучение. Хороший алгоритм не был построен, поскольку семейство алгоритмов слишком мало и с его помощью невозможно уловить закономерность.

В линейной регрессии используется семейство алгоритмов $a(x) = w_0 + w_1 x$.

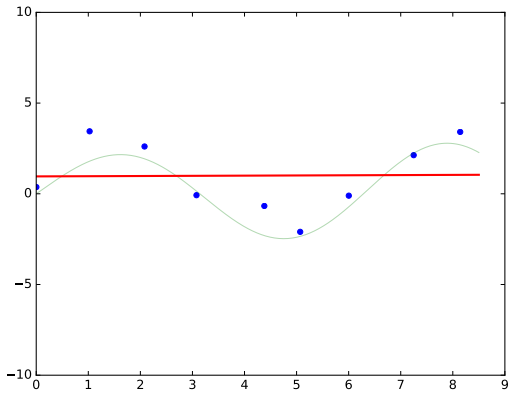

Рис. 3.3: Модель $a(x) = w_0 + w_1 x$.

В этом случае также будет иметь место недообучение. Получилось лучше, но прямая тоже плохо описывает данные.

Если семейство алгоритмов — множество многочленов 4-ей степени:

$$a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_4 x^4,$$

то после обучения получившаяся кривая будет достаточно хорошо описывать и обучающую выборку, и истинную зависимость.

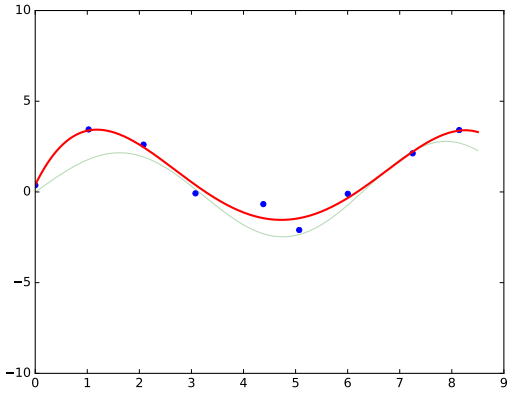

Рис. 3.4: Модель $a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_4 x^4$.

В таком случае качество алгоритма хорошее, но нет идеального совпадения. Встает вопрос, а можно ли добиться совпадения увеличением сложности алгоритма.

При использовании многочленов 9-ой степени уже имеет место переобучение.

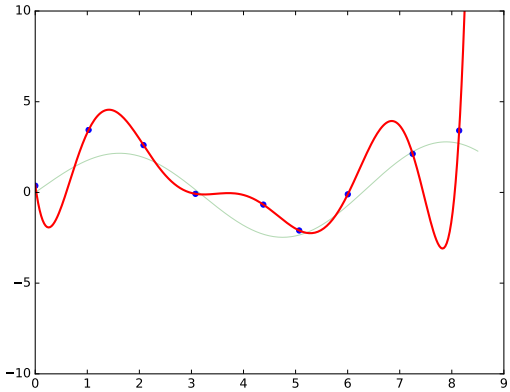

Рис. 3.5: Модель $a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_9 x^9$.

Восстановленная зависимость дает идеальные ответы на всех объектах обучающей выборки, но при этом в любой другой точке сильно отличается от истинной зависимости. Такая ситуация называется переобучением. Алгоритм слишком сильно подогнался под обучающую выборку ценой того, что он будет давать плохие ответы на новых точках.

3.1.3. Недообучение и переобучение

Таким образом, недообучение — ситуация, когда алгоритм плохо описывает и обучающую выборку, и новые данные. В этом случае алгоритм необходимо усложнять.

В случае переобучения, данные из обучающей выборки будут описываться хорошо, а новые данные плохо. Выявить переобучение, используя только обучающую выборку, невозможно, поскольку и хорошо обученный, и переобученный алгоритмы будут хорошо ее описывать. Необходимо использовать дополнительные данные.

Существуют несколько подходов к выявлению переобучения:

- Отложенная выборка. Часть данных из обучающей выборки не участвуют в обучении, чтобы позже проверять на ней обученный алгоритм.
- Кросс-валидация, несколько усложненный метод отложенной выборки. (Об этом способе речь пойдет позже.)

- Использовать меры сложности модели. Об этом пойдет речь далее.

3.2. Регуляризация

В этом разделе речь пойдет о регуляризации — способе борьбы с переобучением в линейных моделях.

3.2.1. «Симптомы» переобучения. Мультиколлинеарность.

Мерой сложности, то есть «симптомом» переобученности модели, являются большие веса при признаках. Например, в предыдущем разделе при обучении модели

$$a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_9 x^9$$

веса оказывались огромными:

$$a(x) = 0.5 + 12458922x + 43983740x^2 + ... + 2740x^9.$$

Другая ситуация, в которой можно встретиться с переобучением — мультиколлинеарность. Так называется проблема, при которой признаки в выборке являются линейно зависимыми. Другими словами, существуют коэффициенты $\alpha_1, ..., \alpha_d$ такие, что для любого объекта x_i из выборки выполняется:

$$\alpha_1 x_i^1 + ... + \alpha_d x_i^d = 0.$$

Более компактно последнее выражение можно переписать в виде:

$$\langle \alpha, x_i \rangle = 0.$$

Допустим, было найдено решение задачи оптимизации:

$$w_* = \text{argmin}_w \frac{1}{\ell} \sum_{i=1}^{\ell} \big(\langle w, x_i \rangle - y_i \big)^2$$

Другой вектор весов, полученный сдвигом в направлении вектора α:

$$w_1 = w_* + t\alpha,$$

так как для элементов x выборки выполняется:

$$\langle w_* + t\alpha, x \rangle = \langle w_*, x \rangle + t\langle \alpha, x \rangle = \langle w_*, x \rangle,$$

также будет являться решением задачи оптимизации. Другими словами, он будет также хорошо описывать данные в выборке, как и исходный алгоритм. Фактически, решениями задачи оптимизации являются бесконечное множество алгоритмов, но многие из них имеют большие веса, и далеко не все обладают хорошей обобщающей способностью. Поэтому здесь тоже легко столкнуться с переобучением.

3.2.2. Регуляризация

Выше было продемонстрировано, что если веса в линейной модели большие, существует высокий риск переобучения. Чтобы бороться с этим, минимизируется уже не выражение для функционала ошибки $Q(a, X)$, а новый функционал, получаемый прибавлением регуляризатора. Самый простой регуляризатор — квадратичный регуляризатор:

$$\|w\|^2 = \sum_{j=1}^{d} w_j^2.$$

В этом случае имеет место следующая задача оптимизации:

$$Q(w, X) + \lambda \|w\|^2 \to \min_w .$$

Таким образом, при обучении будет учитываться также то, что не следует слишком сильно увеличивать веса признаков.

3.2.3. Коэффициент регуляризации

Введенный выше коэффициент λ, который стоит перед регуляризатором, называется коэффициентом регуляризации. Чем больше λ, тем ниже сложность модели. Например, при очень больших его значениях оптимально просто занулить все веса. В то же время при слишком низких значениях λ высок риск переобучения, то есть модель становится слишком сложной.

Поэтому нужно найти некоторое оптимальное значение λ, достаточно большое, чтобы не допустить переобучения, и не очень большое, чтобы уловить закономерности в данных. Обычно λ подбирается на кросс-валидации, о которой пойдет речь в следующем разделе.

3.2.4. Смысл регуляризации

Чтобы понять смысл регуляризации, вместо задачи оптимизации с квадратичным оптимизатором нагляднее рассмотреть задачу условной оптимизации:

$$\begin{cases} Q(w, X) \to \min\limits_{w} \\ \|w\|^2 \leq C \end{cases}$$

Добавление регуляризатора вводит требование, чтобы решение задачи минимизации искалось в некоторой круглой области с центром в нуле.

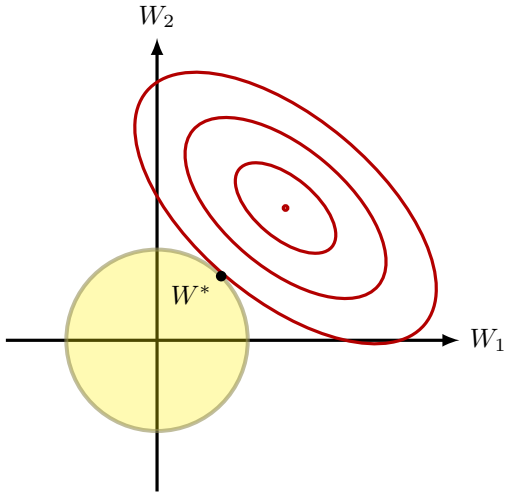

Рис. 3.6: Геометрический смысл условной регуляризации. Красная точка — настоящий оптимум функции, красные линии — линии уровня функции, черная точка — оптимум функции при введенном ограничении.

Таким образом, решение задачи с регуляризатором не будет характеризоваться слишком большими значениями весовых коэффициентов.

3.2.5. Виды регуляризаторов

Рассмотренный выше квадратичный регуляризатор (L_2-регуляризатор) является гладким и выпуклым, что позволяет использовать градиентный спуск.

Также существует L_1-регуляризатор:

$$\|w\|_1 = \sum_{j=1}^{d} |w_j|,$$

который представляет собой L_1-норму вектора весов. Он уже не является гладким, а также обладает интересным свойством. Если применять такой регуляризатор, некоторые веса оказываются равными нулю. Другими словами, такой регуляризатор производит отбор признаков и позволяет использовать в модели не все признаки, а только самые важные из них.

3.3. Оценка качества алгоритмов. Кросс-валидация

В этом разделе речь пойдет об оценке качества алгоритмов и о том, как понять, как поведет себя алгоритм на новых данных.

3.3.1. Выявление переобучения

Уже было сказано, что переобучение сложно выявить, используя только обучающую выборку: и хороший, и переобученный алгоритмы будут показывать хорошее качество на объектах обучающей выборки. Рассмотренные в предыдущем разделе меры переобученности (значения регуляризаторов), безусловно, можно применять, но они не дают ответа на вопрос, насколько хорошо алгоритм поведет себя на новых данных, то есть какая у него будет доля ошибок на новых данных.

3.3.2. Отложенная выборка

Самый простой способ оценить качество алгоритма — использование отложенной выборки. В этом случае следует разбить выборку на две части: первая из двух частей будет использоваться для обучения алгоритма, а вторая, тестовая выборка, — для оценки его качества, в том числе для нахождения доли ошибок в задаче классификации, MSE (среднеквадратичной ошибки) в задаче регрессии и других мер качества в зависимости от специфики задачи.

Естественный вопрос — о том, в какой пропорции производить разбиение. Если взять тестовую выборку слишком маленькой, оценка качества будет ненадежной, хотя обучающая выборка будет почти совпадать с полной выборкой. В противоположенном случае, если отложенная часть будет большой, оценка качества будет надежной, но низкое качество алгоритма может свидетельствовать о недостаточном объёме первой, обучающей, части выборки. Обычно выборку разбивают в соотношениях $70/30$, $80/20$ или $0.632/0.368$.

Преимуществом отложенной выборки является то, что обучать алгоритм приходится всего лишь один раз, но при этом результат сильно зависит от того, как было произведено разбиение.

Например, оценивается стоимость жилья по некоторым признакам. И есть особая категория жилья, например двухэтажные квартиры. И если окажется, что все двухэтажные квартиры, которых немного, попали в отложенную выборку, то после обучения алгоритм будет давать на них очень плохое качество, поскольку в обучающей выборке таких объектов не было.

Чтобы решить эту проблему, можно использовать следующий подход: построить n различных разбиений выборки на 2 части, для каждого разбиения найти оценку качества, а в качестве итоговой оценки качества работы алгоритма использовать усредненное по всем разбиениям значение. Но и в данном случае, поскольку разбиения строятся случайно, нет никаких гарантий, что особый объект хотя бы раз попадет на обучение.

3.3.3. Кросс-валидация

Более системный подход — кросс валидация. В этом случае выборка делится на k блоков примерно одинакового размера. Далее по очереди каждый из этих блоков используется в качестве тестового, а все остальные — в качестве обучающей выборки.

После того, как каждый блок побывает в качестве тестового, будут получены k показателей качества. В результате усреднения получается оценка качества по кросс-валидации.

При этом встает вопрос, какое число блоков использовать. Если блоков мало, получаются надежные, но смещенные оценки. В случае большого числа блоков оценки, наоборот, получаются ненадежными (большой разброс оценок), но несмещенными.

Нет конкретных рекомендаций относительно выбора k. Обычно выбирают $k = 3, 5, 10$. Чем больше k, тем больше раз приходится обучать алгоритм. Поэтому на больших выборках следует выбирать небольшие значения k, так как даже при удалении $1/3$ выборки (а она большая) оставшихся данных будет достаточно для обучения.

3.3.4. Совет: перемешивайте данные в выборке

Часто данные в файле записаны в отсортированном виде по какого-нибудь признаку. Поэтому всегда следует перемешивать выборку прежде, чем производить кросс-валидацию. В ином случае алгоритм будет показывать плохое качество и причина этого будет не так очевидна.

При этом есть задачи, в которых выборку нельзя перемешивать. Это задачи предсказания будущего, например предсказание погоды на следующий день. В этом случае нужно особо следить за тем, как происходит деление выборки.

3.4. Выбор гиперпараметров и сравнение алгоритмов

В этом разделе речь пойдет о выборе гиперпараметров и сравнении различных алгоритмов с помощью изученных ранее схем.

3.4.1. Гиперпараметры

Гиперпараметрами называются такие параметры алгоритмов, которые не могут быть получены из обучающей выборки при обучении, поэтому их надо подбирать путем многократного обучения алгоритма. Примерами гиперпараметров являются:

- Параметр регуляризации λ (при использовании регуляризатора)
- Степень полинома в задаче регрессии с семейством алгоритмов, заданным множеством полиномов определенной степени.

3.4.2. Сравнение разных алгоритмов

Более общая задача — сравнение разных алгоритмов:

- обученных с разными значениями гиперпараметров;
- использующих различный способ регуляризации;
- настроенных с использованием разного функционала ошибки, например среднеквадратичной ошибки и средней абсолютной ошибки;
- которые принадлежат разным классам алгоритмов.

При сравнении алгоритмов можно использовать как отложенную выборку, так и кросс-валидацию, но при этом следует соблюдать осторожность.

Действительно, пусть 1000 алгоритмов сравниваются по качеству на отложенной выборке. Каждый из 1000 алгоритмов, обученных на обучающей выборке, тестируется на отложенной, и в результате выбирается лучший. Фактически на этом шаге отложенная выборка также становится своего рода обучающей, и возникает проблема переобучения: из большого числа алгоритмов выбирается тот, который лучше всего ведет себя на отложенной выборке, лучше подогнан под нее.

3.4.3. Улучшенная схема сравнения алгоритмов

Чтобы бороться с этим, следует использовать несколько усовершенствованную схему оценивания качества алгоритмов, а именно все данные нужно будет делить на 3 части (в случае использования отложенной выборки): обучение, валидация и контроль. Каждый из тысячи алгоритмов будет обучен на обучающей выборке, а его качество будет измерено на валидационной. Алгоритм с наилучшим качеством будет проверен на тестовой выборке, чтобы исключить переобучение и проверить алгоритм на адекватность. По сути именно тестовая выборка будет играть роль новых данных.

Если предпочтительно использовать кросс-валидацию, то данные следует разбить на 2 части. Первая из них будет использоваться для обучения алгоритмов и оценки качества с помощью кросс-валидации, после чего лучший алгоритм будет проверен на адекватность на контрольной выборке.

Урок 2

Линейные модели

2.1. Линейные модели в задачах регрессии

Данный урок будет посвящен линейным моделям. Речь пойдет о задачах классификации и регрессии, как их обучать, и с какими проблемами можно столкнуться при использовании этих моделей. В этом блоке мы обсудим, как выглядят линейные модели в задачах регрессии.

2.1.1. Повторение обозначений из прошлого урока

Для начала необходимо напомнить некоторые обозначения, которые были введены на прошлом уроке.

- $\mathbb{X}$ — пространство объектов
- $\mathbb{Y}$ — пространство ответов
- $x = (x^1, ..., x^d)$ — признаковое описание объекта
- $X = (x_i, y_i)_{i=1}^{\ell}$ — обучающая выборка
- $a(x)$ — алгоритм, модель
- $Q(a, X)$ — функционал ошибки алгоритма a на выборке X
- Обучение: $a(x) = \text{argmin}_{a \in \mathbb{A}} Q(a, X)$

Напомним, что в задаче регрессии пространство ответов $\mathbb{Y} = \mathbb{R}$. Чтобы научиться решать задачу регрессии, необходимо задать:

- **Функционал ошибки** Q: способ измерения того, хорошо или плохо работает алгоритм на конкретной выборке.
- **Семейство алгоритмов** $\mathbb{A}$: как выглядит множество алгоритмов, из которых выбирается лучший.
- **Метод обучения**: как именно выбирается лучший алгоритм из семейства алгоритмов.

2.1.2. Пример задачи регрессии: предсказание прибыли магазина

Пусть известен один признак — прибыль магазина в прошлом месяце, а предсказать необходимо прибыль магазина в следующем. Поскольку прибыль — вещественная переменная, здесь идет речь о задаче регрессии.

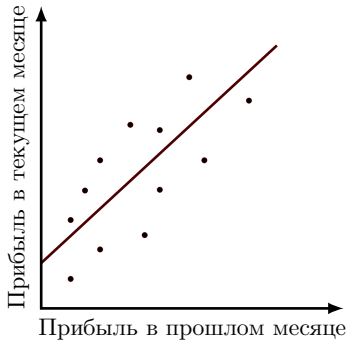

Рис. 2.1: Точки обучающей выборки.

По этому графику можно сделать вывод о существовании зависимости между прибылью в следующем и прошлом месяцах. Если предположить, что зависимость приблизительно линейная, ее можно представить в виде прямой на этом графике. По этой прямой и можно будет предсказывать прибыль в следующем месяце, если известна прибыль в прошлом.

В целом такая модель угадывает тенденцию, то есть описывает зависимость между ответом и признаком. При этом, разумеется, она делает это не идеально, с некоторой ошибкой. Истинный ответ на каждом объекте несколько отклоняется от прогноза.

Один признак — это не очень серьезно. Гораздо сложнее и интереснее работать с многомерными выборками, которые описываются большим количеством признаков. В этом случае нарисовать выборку и понять, подходит или нет линейная модель, нельзя. Можно лишь оценить ее качество и по нему уже понять, подходит ли эта модель.

Следует отметить, что вообще нельзя придумать модель, которая идеально описывает ваши данные, то есть идеально описывает, как порождается ответ по признакам.

2.1.3. Описание линейной модели

Далее обсудим, как выглядит семейство алгоритмов в случае с линейными моделями. Линейный алгоритм в задачах регрессии выглядит следующим образом:

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

где w_0 — свободный коэффициент, x^j — признаки, а w_j — их веса.

Если добавить $(d+1)$-й признак, который на каждом объекте принимает значение 1, линейный алгоритм можно будет записать в более компактной форме

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

где используется обозначение $\langle w, x \rangle$ для скалярного произведения двух векторов.

В качестве меры ошибки не может быть выбрано отклонение от прогноза $Q(a, y) = a(x) - y$, так как в этом случае минимум функционала не будет достигаться при правильном ответе $a(x) = y$. Самый простой способ — считать модуль отклонения:

$$|a(x) - y|.$$

Но функция модуля не является гладкой функцией, и для оптимизации такого функционала неудобно использовать градиентные методы. Поэтому в качестве меры ошибки часто выбирается квадрат отклонения:

$$(a(x) - y)^2.$$

Функционал ошибки, именуемый среднеквадратичной ошибкой алгоритма, задается следующим образом:

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

В случае линейной модели его можно переписать в виде функции (поскольку теперь Q зависит от вектора, а не от функции) ошибок:

$$Q(w, x) = \frac{1}{\ell} \sum_{i=1}^{\ell} (\langle w, x_i \rangle - y_i)^2 .$$

2.2. Обучение модели линейной регрессии

В этом блоке речь пойдет о том, как обучать модель линейной регрессии, то есть как настраивать ее параметры. В прошлый раз было введено следующее выражение для качества линейной модели на обучающей выборке:

$$Q(w, x) = \frac{1}{\ell} \sum_{i=1}^{\ell} (\langle w, x_i \rangle - y_i)^2 \to \min_w .$$

Следует напомнить, что в число признаков входит также постоянный признак, равный 1 для всех объектов, что позволяет исключить постоянную составляющую в последнем соотношении.

2.2.1. Переход к матричной форме записи

Прежде, чем будет рассмотрена задача оптимизации этой функции, имеет смысл переписать используемые соотношения в матричной форме. Матрица «объекты–признаки» X составлена из признаковых описаний всех объектов из обучающей выборки:

$$X = \begin{pmatrix} x_{11} & \dots & x_{1d} \\ \dots & \dots & \dots \\ x_{\ell 1} & \dots & x_{\ell d} \end{pmatrix}$$

Таким образом, в ij элементе матрицы X записано значение j-го признака на i объекте обучающей выборки. Также понадобится вектор ответов y, который составлен из истинных ответов для всех объектов:

$$y = \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}.$$

В этом случае среднеквадратичная ошибка может быть переписана в матричном виде:

$$Q(w, X) = \frac{1}{\ell}\|Xw - y\|^2 \to \min_w.$$

Эта формула пригодится, в частности, при реализации линейной регрессии на компьютере.

2.2.2. Аналитический метод решения

Можно найти аналитическое решение задачи минимизации:

$$w_* = (X^T X)^{-1} X^T y.$$

Основные сложности при нахождении решения таким способом:

- Для нахождения решения необходимо вычислять обратную матрицу. Операция обращения матрицы требует, в случае d признаков, выполнение порядка d^3 операции, и является вычислительно сложной уже в задачах с десятком признаков.
- Численный способ нахождения обратной матрицы не может быть применен в некоторых случаях (когда матрица плохо обусловлена).

2.2.3. Оптимизационный подход к решению

Другой, несколько более удобный, способ найти решение — использовать численные методы оптимизации.

Несложно показать, что среднеквадратическая ошибка — это выпуклая и гладкая функция. Выпуклость гарантирует существование лишь одного минимума, а гладкость — существование вектора градиента в каждой точке. Это позволяет использовать метод градиентного спуска.

При использовании метода градиентного спуска необходимо указать начальное приближение. Есть много подходов к тому, как это сделать, в том числе инициализировать случайными числами (не очень большими). Самый простой способ это сделать — инициализировать значения всех весов равными нулю:

$$w^0 = 0.$$

На каждой следующей итерации, $t = 1, 2, 3, ...$, из приближения, полученного в предыдущей итерации w^{t-1}, вычитается вектор градиента в соответствующей точке w^{t-1}, умноженный на некоторый коэффициент η_t, называемый шагом:

$$w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, X).$$

Остановить итерации следует, когда наступает сходимость. Сходимость можно определять по-разному. В данном случае разумно определить сходимость следующим образом: итерации следует завершить, если разница двух последовательных приближений не слишком велика:

$$\|w^t - w^{t-1}\| < \varepsilon.$$

Подробно метод градиентного спуска обсуждался в прошлом курсе этой специализации.

2.3. Градиентный спуск для линейной регрессии

2.3.1. Случай парной регрессии

В случае парной регрессии признак всего один, а линейная модель выглядит следующим образом:

$$a(x) = w_1 x + w_0,$$

где w_1 и w_0 — два параметра.

Среднеквадратичная ошибка принимает вид:

$$Q(w_0, w_1, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i)^2.$$

Для нахождения оптимальных параметров будет применяться метод градиентного спуска, про который уже было сказано ранее. Чтобы это сделать, необходимо сначала вычислить частные производные функции ошибки:

$$\frac{\partial Q}{\partial w_1} = \frac{2}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i) x_i, \qquad \frac{\partial Q}{\partial w_0} = \frac{2}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i).$$

2.3.2. Демонстрация градиентного спуска в случае парной регрессии

Следующие два графика демонстрируют применение метода градиентного спуска в случае парной регрессии. Справа изображены точки выборки, а слева — пространство параметров. Точка в этом пространстве обозначает конретную модель.

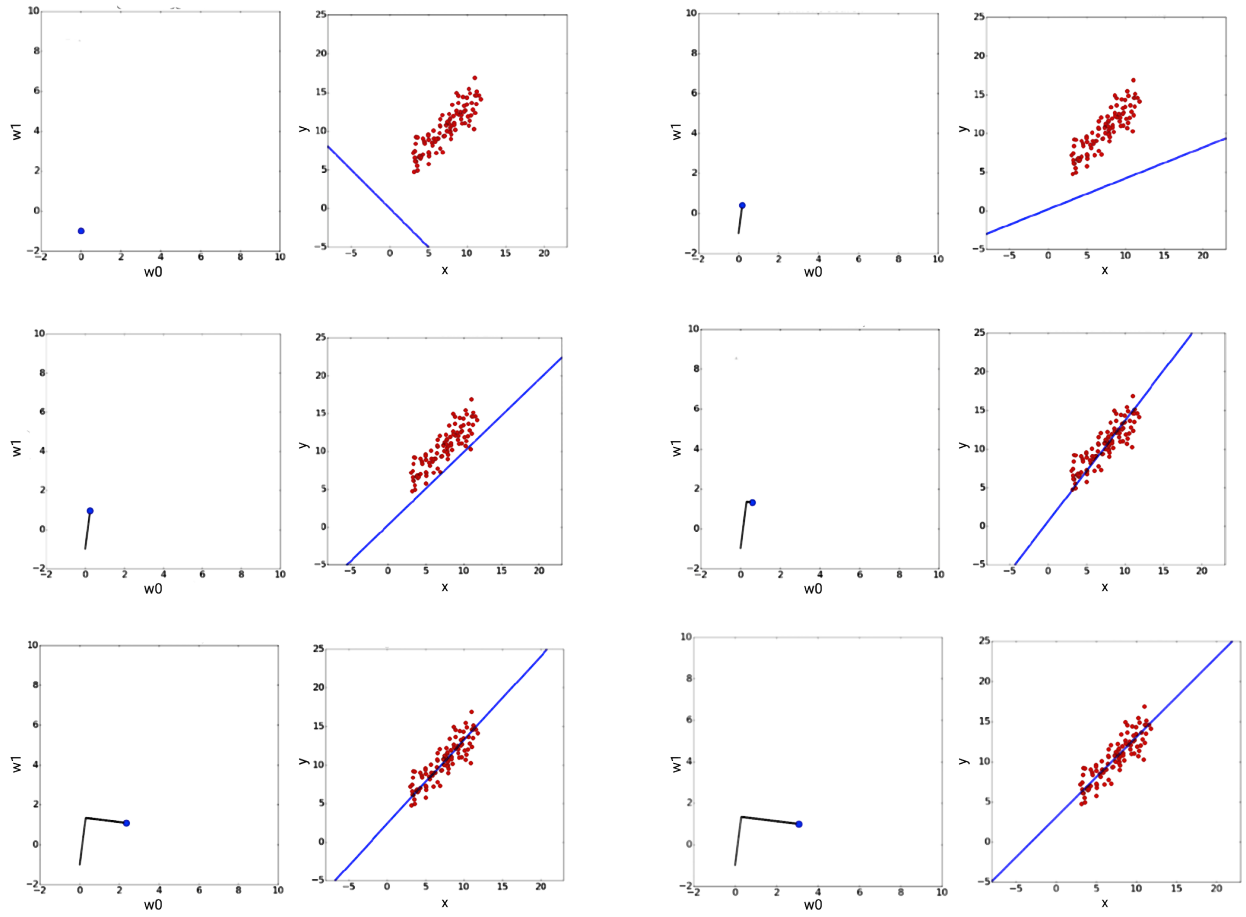

Рис. 2.2: Демонстрация метода градиентного спуска.

График зависимости функции ошибки от числа произведенных операции выглядит следующим образом:

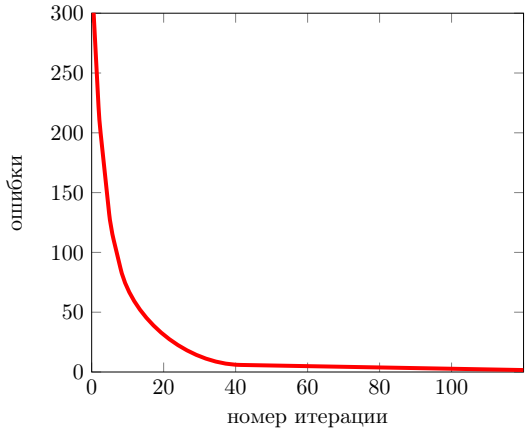

Рис. 2.3: Ошибка в зависимости от номера итерации

2.3.3. Выбор размера шага в методе градиентного спуска

Очень важно при использовании метода градиентного спуска правильно подбирать шаг. Каких-либо конкретных правил подбора шага не существует, выбор шага — это искусство, но существует несколько полезных закономерностей.

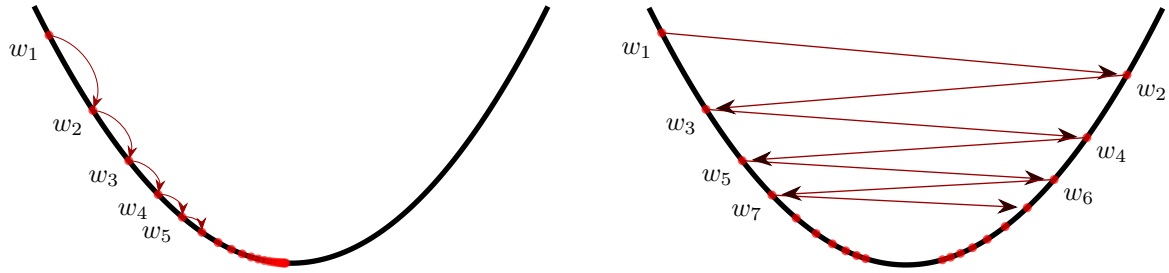

Рис. 2.4: Случаи маленького и большого шага

Если длина шага слишком мала, то метод будет неспеша, но верно шагать в сторону минимума. Если же взять размер шага очень большим, появляется риск, что метод будет перепрыгивать через минимум. Более того, есть риск того, что градиентный спуск не сойдется.

Имеет смысл использовать переменный размер шага: сначала, когда точка минимума находится еще далеко, двигаться быстро, а позже, спустя некоторое количество итерации — делать более аккуратные шаги. Один из способов задать размер шага следующий:

$$\eta_t = \frac{k}{t},$$

где k — константа, которую необходимо подобрать, а t — номер шага.

2.3.4. Случай многомерной линейной регрессии

В случае многомерной линейной регрессии используется тот же самый подход — необходимо решать задачу минимизации:

$$Q(w, X) = \frac{1}{\ell}\|Xw - y\|^2 \to \min_w,$$

где $\|x\|$ — норма вектора x. Формула для вычисления градиента принимает следующий вид:

$$\nabla_w Q(w, X) = \frac{2}{\ell} X^T (Xw - y)$$

Стоит отметить, что вектор $Xw - y$, который присутствует в данном выражении, представляет собой вектор ошибок.

2.4. Стохастический градиентный спуск

В этом блоке речь пойдет о стохастическом градиентном спуске, который особенно хорошо подходит для обучения линейных моделей.

2.4.1. Недостатки обычного метода градиентного спуска

В обычном методе градиентного спуска на каждом шаге итерации следующее приближение получается из предыдущего вычитанием вектора градиента, умноженного на шаг η_t:

$$w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, X).$$

При этом выражение для градиента в матричной форме имеет вид:

$$\nabla_w Q(w, X) = \frac{2}{\ell} X^T (Xw - y)$$

Выражение для j-ой компоненты градиента, таким образом, содержит суммирование по всем объектам обучающей выборки:

$$\frac{\partial Q}{\partial w_j} = \frac{2}{\ell} \sum_{i=1}^{\ell} x_i^j (\langle w, x_i \rangle - y_i).$$

В этом и состоит основной недостаток метода градиентного спуска — в случае большой выборки даже одна итерация метода градиентного спуска будет производиться долго.

2.4.2. Стохастический градиентный спуск

Идея стохастического градиентного спуска основана на том, что в сумме в выражении для j-компоненты градиента i-ое слагаемое указывает то, как нужно поменять вес w_j, чтобы качество увеличилось для i-го объекта выборки. Вся сумма при этом задает, как нужно изменить этот вес, чтобы повысить качество для всех объектов выборки. В стохастическом методе градиентного спуска градиент функции качества вычисляется только на одном случайно выбранном объекте обучающей выборки. Это позволяет обойти вышеупомянутый недостаток обычного градиентного спуска.

Таким образом, алгоритм стохастического градиентного спуска следующий. Сначала выбирается начальное приближение:

$$w^0 = 0$$

Далее последовательно вычисляются итерации w^t: сначала случайным образом выбирается объект x_i из обучающей выборки X и вычисляется вектор градиента функции качества на этом объекте, а следующее приближение получается из предыдущего вычитанием умноженного на шаг η_t полученного вектора:

$$w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, \{x_i\}).$$

Итерации прекращаются при достижении определенного условия, например:

$$\|w^t - w^{t-1}\| < \varepsilon.$$

2.4.3. Сходимость стохастического градиентного спуска

Показательно посмотреть на графики сходимости градиентного спуска и стохастического градиентного спуска. В обычном градиентном спуске на каждом шаге уменьшается суммарная ошибка на всех элементах обучающей выборки. График в таком случае обычно получается монотонным.

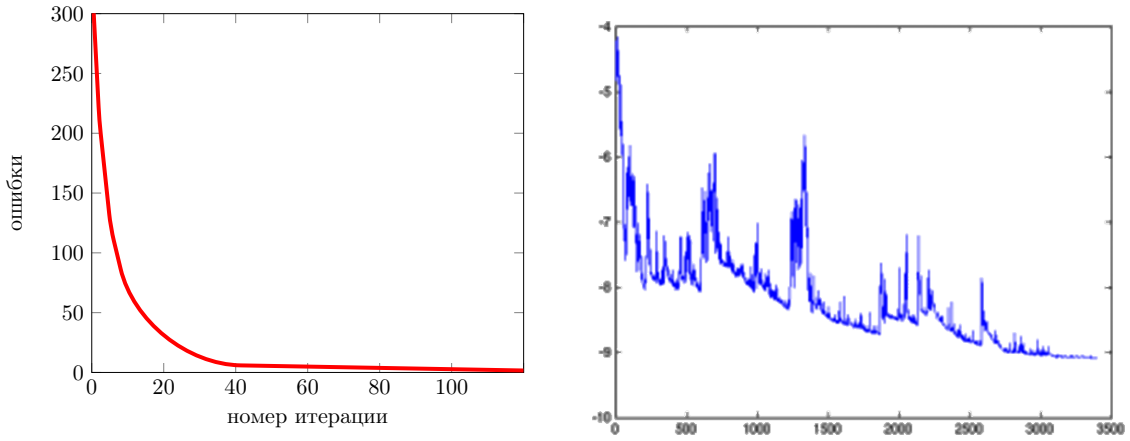

Рис. 2.5: Ошибка в зависимости от номера итерации

Напротив, в стохастическом методе весовые коэффициенты меняются таким образом, чтобы максимально уменьшить ошибку для одного случайно выбранного объекта. Это приводит к тому, что график выглядит пилообразным, то есть на каждой конкретной итерации полная ошибка может как увеличиваться, так и уменьшаться. Но в итоге с ростом номера итерации значение функции уменьшается.

2.4.4. Особенности стохастического градиентного спуска

Стохастический градиентный спуск (SGD) обладает целым рядом преймуществ. Во-первых, каждый шаг выполняется существенно быстрее шага обычного градиентного метода, а также не требуется постоянно хранить всю обучающую выборку в памяти. Это позволяет использовать для обучения выборки настолько большие, что они не помещаются в память компьютера. Стохастический градиентный спуск также можно использовать для онлайн-обучения, то есть в ситуации, когда на каждом шаге алгоритм получает только один объект и должен учесть его для коррекции модели.

2.5. Линейная классификация

2.5.1. Задача бинарной классификации

В этом разделе рассматривается задача линейной классификации, то есть применение линейных моделей к задаче классификации. Речь пойдет о самом простом виде классификации — бинарной классификации. В случае бинарной классификации множество возможных значений ответов состоит из двух элементов:

$$\mathbb{Y} = \{-1, +1\}.$$

Как уже было сказано, чтобы работать с той или иной моделью нужно:

- Выбрать функционал (функцию) ошибки, то есть задать способ определения качества работы того или иного алгоритма на обучающей выборке.
- Построить семейство алгоритмов, то есть множество алгоритмов, из которого потом будет выбираться наилучший с точки зрения определенного функционала ошибки.
- Ввести метод обучения, то есть определить способ выбора лучшего алгоритма из семейства.

2.5.2. Линейный классификатор

Ранее была рассмотрена задача линейной регрессии. В этом случае алгоритм представлял собой линейную комбинацию признаков с некоторыми весами и свободным коэффициентом.

Линейные классификаторы устроены похожим образом, но они должны возвращать бинарные значения, а следовательно требуется также брать знак от получившегося выражения:

$$a(x) = \text{sign}\left(w_0 + \sum_{j=1}^{d} w_j x^j\right).$$

Как и раньше, добавлением еще одного постоянного для всех объектов признака можно привести формулу к более однородному виду:

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

2.5.3. Геометрический смысл линейного классификатора

Выражение $\langle w, x\rangle = 0$ является уравнением некоторой плоскости в пространстве признаков.

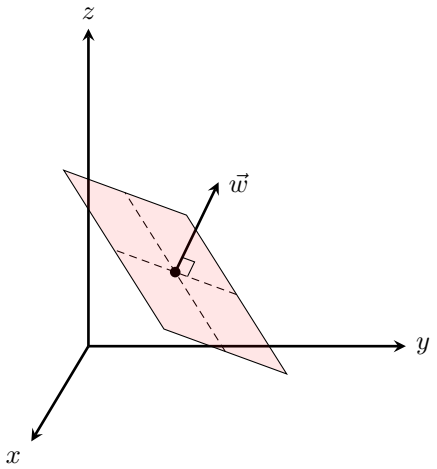

Рис. 2.6: Геометрический смысл линейного классификатора

При этом для точек по одну сторону от этой плоскости скалярное произведение $\langle w, x\rangle$ будет положительным, а с другой — отрицательным.

Таким образом, линейный классификатор проводит плоскость в пространстве признаков и относит объекты по разные стороны от нее к разным классам.

Согласно геометрическому смыслу скалярного произведения, расстояние от конкретного объекта, который имеет признаковое описание x, до гиперплоскости $\langle w, x\rangle = 0$ равно $\frac{|\langle w,x\rangle|}{\|w\|}$. С этим связано такое важное понятие в задачах линейной классификации как понятие отступа:

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

Отступ является величиной, определяющей корректность ответа. Если оступ больше нуля $M_i > 0$, то классификатор дает верный ответ для i-го объекта, в ином случае — ошибается.

Причем чем дальше отступ от нуля, тем больше уверенность как в правильном ответе, так и в том, что алгоритм ошибается. Если отступ для некоторого объекта отрицательный и большой по модулю, это значит, что алгоритм неправильно описывает данные: либо этот объект является выбросом, либо алгоритм не пригоден для решения данной задачи.

2.6. Функции потерь в задачах классификации

2.6.1. Пороговая функция потерь

В случае линейной классификации естественный способ определить качество того или иного алгоритма — вычислить для объектов обучающей выборки долю неправильных ответов:

$$Q(a, x) = \frac{1}{\ell}\sum_{i=1}^{\ell} [a(x_i) \neq y_i]$$

С помощью введенного ранее понятия отступа можно переписать это выражение для случая линейной классификации в следующем виде:

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

Функция, стоящая под знаком суммы, называется функцией потерь. В данном случае это пороговая функция потерь, график которой в зависимости от отступа выглядит следующим образом:

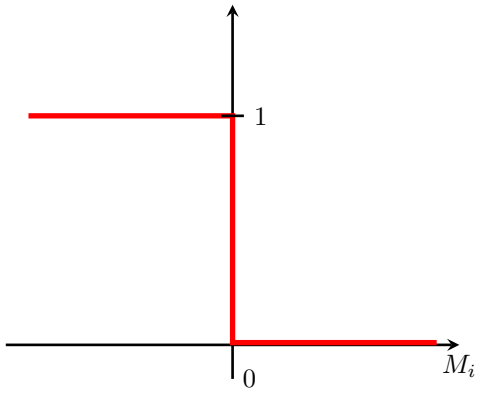

Рис. 2.7: График пороговой функции потерь

Такая функция является разрывной в точке 0, что делает невозможным применение метода градиентного спуска. Можно, конечно, использовать методы негладкой оптимизации, о которых шла речь в прошлом курсе, но они сложны в реализации.

2.6.2. Оценка функции потерь

Используя любую гладкую оценку пороговой функции:

$$\big[M_i < 0\big] \leq \tilde{L}(M_i)$$

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

$$Q(a,X) \leq \tilde{Q}(a,X) = \frac{1}{\ell}\sum_{i=1}^{\ell}\tilde{L}(M_i).$$

В этом случае минимизировать нужно будет не долю неправильных ответов, а некоторую другую функцию, которая является оценкой сверху:

$$Q(a,X) \leq \tilde{Q}(a,X) = \frac{1}{\ell}\sum_{i=1}^{\ell}\tilde{L}(M_i) \to \min_a.$$

Здесь используется предположение, что в точке минимума этой верхней оценки число ошибок также будет минимально. Строго говоря, это не всегда так.

2.6.3. Примеры оценок функции потерь

Примерами таких оценок функции потерь являются:

- Логистическая функция потерь (используется в логистической регрессии, о которой пойдет речь позже в данном курсе):
$$\tilde{L}(M) = \log_2\big(\exp(-M)\big)$$
- Экспоненциальная функция потерь:
$$\tilde{L}(M) = \exp(-M)$$

- Кусочно-линейная функция потерь (используется в методе опорных векторов):

$$\tilde{L}(M) = \max(0, 1 - M)$$

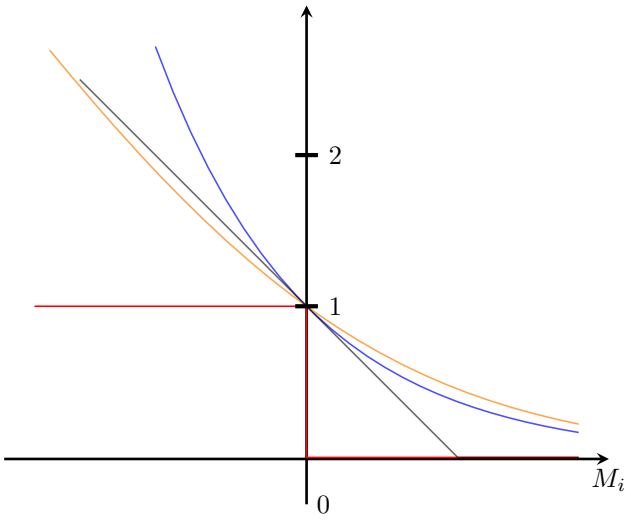

Рис. 2.8: Графики различных функций потерь: пороговая (красная линия), экспоненциальная (синяя), логистическая (оранжевая) и кусочно-линейная (серая).

2.6.4. Логистическая функция потерь

В случае логистической функции потерь функционал ошибки имеет вид:

$$\tilde{Q}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \ln\big(\exp(-M_i)\big) = \frac{1}{\ell} \sum_{i=1}^{\ell} \ln\big(\exp(-y_i \langle w, x_i \rangle)\big).$$

Получившееся выражение является гладким, а, следовательно, можно использовать, например, метод градиентного спуска.

Следует обратить внимание, что в случае, если число ошибок стало равно нулю, все равно в ходе обучения алгоритма линейной классификации будут увеличиваться отступы, то есть будет увеличиваться уверенность в полученных результатах.