Введение в машинное обучение для Java-разработчиков

Лекция 1

Ермилов Сергей

education

Мета

О курсе

- 6 лекций, 6 домашек
- На каждое ДЗ дается 2 недели с момента «выкладывания» ДЗ
- В каждом ДЗ будут написаны баллы для получения определенной оценки
- Все ДЗ должны быть сданы до 3 декабря включительно. Время сдачи ДЗ == времени коммита в репозиторий по московскому времени.
- Если вы не успели вовремя сдать ДЗ, но очень хотите получить высокую оценку, то это обсуждается в индивидуальном порядке, но лучше сдавать все вовремя!

Коммуникация

- Вопросы вне занятий можно задавать в личку или в общем чате курса в vk в любое время.
- Отвечаем в рабочее время с 10:00 до 18:00, SLA на ответ ~8 часов. Если мы не ответили, то можно смело писать еще раз.

Как задать вопрос на лекции

- Текстом в чатике или голосом
- Хорошим тоном будет «поднять руку» перед тем как задать вопрос

Если что-то идет не так

- Пропал голос, видео, презентация. Ставим «минусы» в чатик
- Если при этом у вас все хорошо, ставим «плюсики»
- Если совсем все поломалось, пишем в чатик ВК (он на резервном канале интернета)

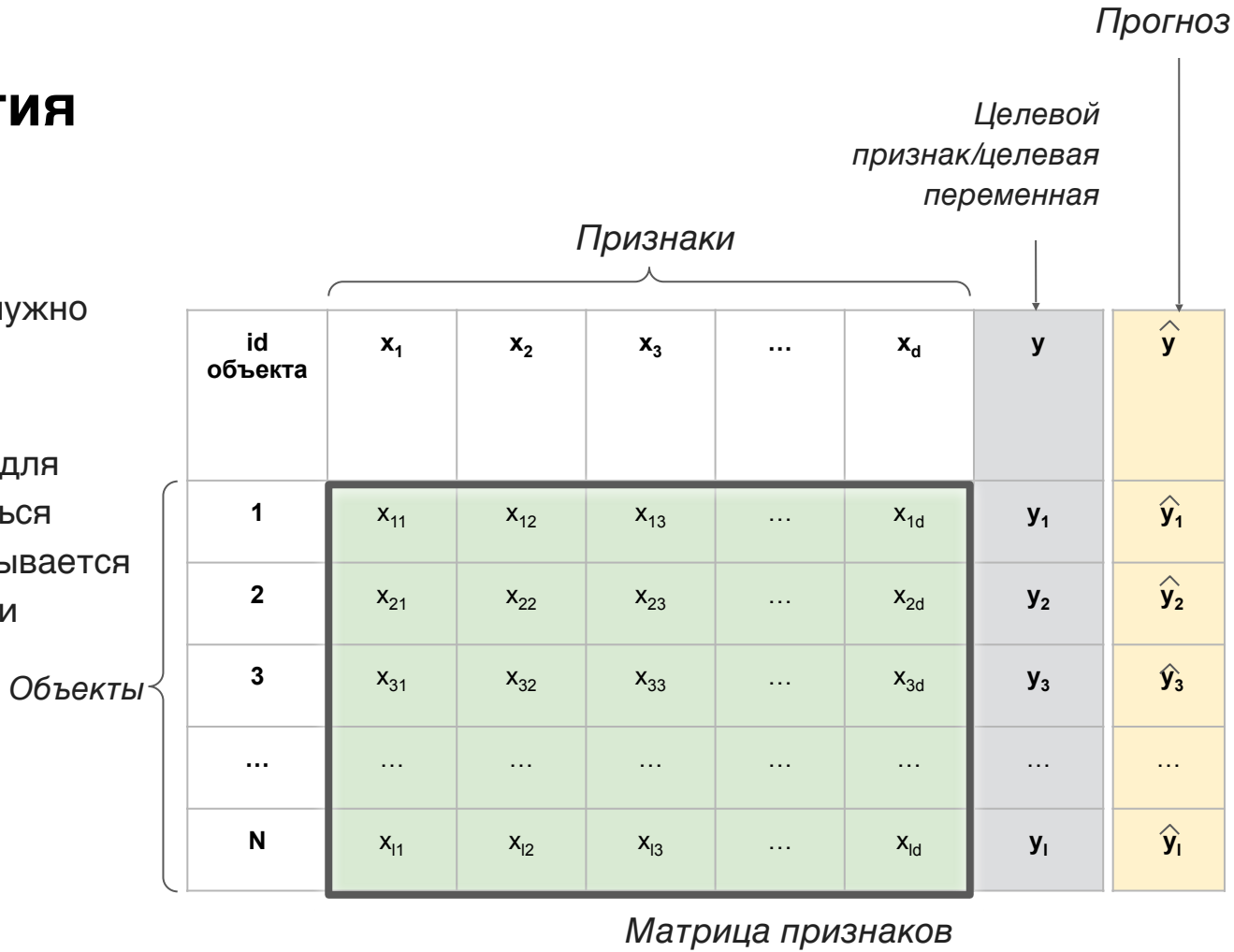
Основные понятия

Основные понятия

Машинное обучение — дисциплина, заключающаяся в извлечении знаний из известных данных. Является разделом математики, поэтому формул будет много.

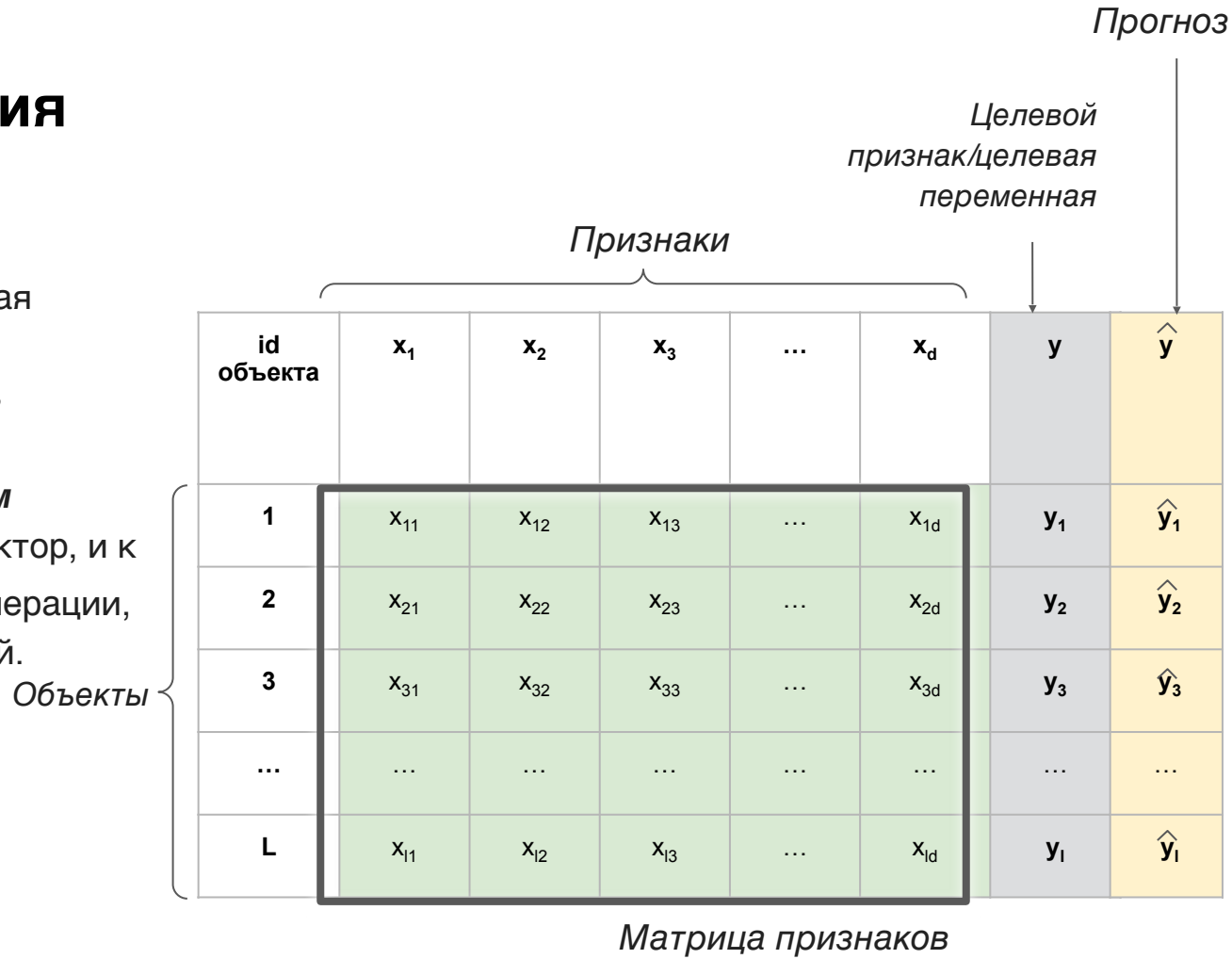
Основные понятия

- **Объект** - то, для чего нам нужно сделать **прогноз**. Объекты обозначаются буквой x.
- Множество всех объектов, для которых может потребоваться сделать предсказания, называется ***пространством объектов*** и обозначается $\mathbb{X}$.



Основные понятия

- **Признак** — это некая числовая характеристика объекта
- Совокупность всех признаков объекта $x = (x_1, x_2, \ldots, x_d)$ называется его ***признаковым описанием.*** Это d-мерный вектор, и к нему можно применять все операции, описанные линейной алгеброй.



Виды признаков

Множество значений i-го признака будем обозначать D_i. Признаки можно разделить на следующие виды:

- Бинарные
- Вещественные
- Категориальные
- Порядковые
- Множественные

Виды признаков

Бинарные признаки принимают два значения: $D_i = \{0,1\}$

Примером в задаче кредитного скоринга может служить ответ, выше ли доход клиента определенной установленной суммы.

При положительном ответе признак полагается равным 1, при отрицательном – 0.

Виды признаков

Вещественные признаки могут принимать в качестве значений все вещественные числа: $D_i = \mathbb{R}$.

Примерами могут выступать:

возраст человека, заработная плата, количество звонков в колл-центр в месяц и т.д.

Виды признаков

Категориальные признаки — это такие признаки, значения которых можно сравнивать только на равенство, и нельзя на "больше-меньше". В этом случае D_i — неупорядоченное множество. Примерами таких признаков могут выступать город, в котором родился клиент банка, или его образование.

Виды признаков

Порядковые признаки — частный случай категориальных признаков. В этом случае D_i – упорядоченное множество. Признаки можно сравнивать между собой, но нельзя определить расстояние между ними. Например, то же образование, но с введенным осмысленным порядком (высшее образование больше среднего профессионального, которое в свою очередь больше среднего и т.д.)

Виды признаков

Множественные признаки — признаки, значения которых на объекте являются подмножеством некоторого множества. Например, в задачах анализа текстов таким признаком является множество слов, которые входят в текст. Оно является подмножеством большого словаря.

Общие понятия

Центральным понятием машинного обучения является ***обучающая выборка (train)***. Это примеры, на основе которых мы планируем строить общую закономерность. Она обозначается X и состоит из N пар объектов x_i и известных ответов y_i:

$$X = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

Валидационная выборка (val) - те объекты, на которым мы проверяем метрики, чтобы слишком сильно не подстраиваться под обучающую выборку

Тестовая выборка (test) - те объекты, для которых мы будем делать прогнозы. Иногда она же валидационная

Общие понятия

Функция, отображающая пространство объектов $\mathbb{X}$ в пространство ответов $\mathbb{Y}$, помогающая нам делать предсказания, называется ***алгоритмом* или *моделью*** и обозначается $a(x)$. Она принимает на вход объект и выдает ответ .

Общие понятия

Для решения определенной задачи не все алгоритмы одинаково хорошо подходят. Для определения наиболее подходящего алгоритма введена характеристика, называемая ***функционалом ошибки*** $Q(a,X)$. Он принимает на вход алгоритм и выборку и сообщает, насколько хорошо данный алгоритм работает на данной выборке. В примере распознавания спам-писем в качестве такого функционала может выступать доля неправильных ответов (предсказаний). Задача обучения заключается в подборе такого алгоритма, при котором достигается минимум функционала ошибки $Q(a,X) \to min$.

Наиболее подходящий алгоритм при этом выбирается из множества, называемого *семейством алгоритмов* $\mathbb{A}$.

Линейная регрессия

Линейная регрессия

Линейные модели — это такие модели, которые сводятся к суммированию значений признаков с некоторыми **весами.** Вектор весов одинаковый для всех объектов. Само название модели говорит о том, что зависимость предсказываемой переменной от признаков будет линейной:

$$a(x) = w_0 + \sum_{i=1}^{N} w_i x_i$$

Линейная регрессия

В этом случае параметрами моделей являются веса w_i. Вес w_0 называется *свободным коэффициентом* или ***сдвигом***. Для упрощения свободный коэффициент можно добавить, как признак, равный 1 на всех объектах.

Оптимизация модели в таком случае заключается в подборе оптимальных значений весов. Сумму в формуле также можно описать как скалярное произведение вектора признаков $x = (x_1, x_2, \ldots, x_n)$ на вектор весов $w = (w_1, w_2, \ldots, w_n)$

$$a(x) = < w, x >$$

Линейная регрессия

Для обучения модели необходимо иметь возможность измерять точность линейного алгоритма на выборке (обучающей или тестовой).

В качестве меры ошибки можно взять абсолютное отклонение истинного значения от прогноза $Q(a,y)=a(x)-y$, но тогда минимизация функционала ошибки (в которой и состоит задача обучения) будет достигаться при принятии им отрицательного значения. Например, если истинное значение ответа равно 10, а алгоритм $a(x)$ выдает ответ 11, отклонение будет равно 1, а при значении предсказания равном 1, отклонение будет равно 1−10=−9. Значение ошибки во втором случае ниже, однако разница истинного значения и предсказания нашей модели больше. Таким образом, такой функционал ошибки не поддается минимизации.

Логичным кажется решение использовать в качестве функционала ошибки модуль отклонения $Q(a,y)=|a(x)-y|$. Соответствующий функционал ошибки называется **средним абсолютным отклонением (mean absolute error, MAE)**:

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

Линейная регрессия

Зачастую методы оптимизации включают в себя дифференцирование, а функция модуля не является гладкой — она не имеет производной в нуле, поэтому ее оптимизация бывает затруднительной.

Поэтому сейчас основной способ измерить отклонение — посчитать квадрат разности $Q(a,y)=(a(x)-y)^2$. Такая функция является гладкой и имеет производную в каждой точке, а ее минимум достигается при равенстве истинного ответа y и прогноза $a(x)$.

Основанный на этой функции функционал ошибки называется **суммой квадратов отклонений (residual sum of squares, RSS)**:

$$RSS = e_1^2 + e_2^2 + \ldots + e_N^2 -> min$$

RSS можно нормировать на количество элементов в выборке и получить функционал **MSE**

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

Метод наименьших квадратов

Метод наименьших квадратов

Обучение модели регрессии заключается в минимизации функционала ошибки.

В случае использования среднеквадратичной ошибки получаем задачу оптимизации

$$Q(w, x) = \frac{1}{N} \sum_{i=1}^{N} (\langle w, x \rangle - y_i)^2 \to min$$

Способ вычисления весов путем минимизации среднеквадратичного отклонения называется ***методом наименьших квадратов*.**

Метод наименьших квадратов

Имеет смысл переписать имеющиеся соотношения в матричном виде. В матрицу "объекты-признаки" впишем по строкам d признаков для всех l объектов из обучающей выборки:

$$X = \begin{pmatrix} & x_{1d} \\ & \dots \\ & \dots \\ & \dots \\ x_{ld} & \end{pmatrix},$$

и составим вектор ответов y из истинных ответов для данной выборки:

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

Метод наименьших квадратов

Помня, что w — вектор параметров, переписанная в матричном виде задача будет выглядеть следующим образом:

$$Q(w, x) = \frac{1}{N} \quad Xw - y \quad {}^{2}_{2} \rightarrow min_{w}$$

где используется евклидова (L_2) норма.

$$x \quad {}_{2} = \sqrt{\sum_{i} \; x_i \; {}^{2}}$$

Метод наименьших квадратов

Продифференцировав данную функцию по вектору w и приравняв к нулю, можно получить явную аналитическую формулу для решения задачи минимизации:

$$w = (X^T X)^{-1} X^T y$$

Это решение называется *нормальным уравнением* линейной регрессии. Наличие аналитического решения кажется положительным фактором, однако, у него есть некоторые минусы, среди которых вычислительная сложность операции (обращение матрицы $X^T X$ будет иметь кубическую сложность от количества признаков d^3), а также тот факт, что матрица $X^T X$ может быть вырожденной и поэтому необратимой. Тогда найти решение будет невозможно.

Градиентный спуск

Градиентный спуск

Среднеквадратичная ошибка имеет один минимум и непрерывна на всей области значений (то есть является выпуклой и гладкой), а значит в каждой ее точке можно рассчитать частные производные.

Вспомним, что *градиентом* функции f называется n-мерный вектор из частных производных.

$$\nabla f(x_1, \ldots, x_d) = (\partial f / \partial x_i)^d_{i=1}.$$

При этом известно, что **градиент задает направление наискорейшего роста функции**. Значит, **антиградиент будет показывать направление ее скорейшего убывания**, что будет полезно нам в нашей задаче минимизации функционала ошибки.

Градиентный спуск

Для решения задачи нам требуется определить некоторую стартовую точку и итерационно сдвигаться от нее в сторону антиградиента с определенным *шагом* η_k, на каждом шагу пересчитывая градиент в точке, в которой мы находимся. Таким образом, имея начальный вектор весов w^0, k-й шаг градиентного спуска будет иметь вид

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

Градиентный спуск

Итерации следует продолжать, пока не наступает сходимость. Она определяется разными способами, но в данном случае удобно определять как ситуацию, когда векторы весов от шага к шагу изменяются незначительно, то есть норма отклонения вектора весов на текущем шаге от предыдущего не превышает заданное значение ε:

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

Градиентный спуск

Начальный вектор весов w_0 также можно определять различными способами, обычно его берут нулевым или состоящим из случайных небольших чисел.

Градиентный спуск

В случае многомерной регрессии (при количестве признаков больше 1) при оптимизации функционала ошибки MSE

$$Q(w, X) = \frac{\|Xw - y\|_2^2}{N}$$

формула вычисления градиента принимает вид

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

Масштабирование признаков

При работе с линейными моделями полезной является практика ***масштабирования признаков*.** Многие методы машинного обучения наиболее эффективны в том случае, когда признаки имеют одинаковый масштаб. По сути масштабирование означает приведение признаков к некоей единой шкале.

Существует большое количество методов масштабирования, наиболее популярными из которых являются *нормализация* и *стандартизация*.

Масштабирование признаков

Метод **нормализации заключается в приведении признаков к масштабу в диапазоне [0-1].**

Для его реализации необходимо найти минимальное $min_j(x^j_i)$ и максимальное $max_j(x^j_i)$ значение признака на обучающей выборке. При этом масштабированное значение признака будет находиться по формуле

$$x^j_i = (x^j_i - min_j(x^j_i)) / (max_j(x^j_i) - min_j(x^j_i)).$$

После преобразования значений признаков минимальное значение превратится в 0, а максимальное в 1.

Масштабирование признаков

Стандартизация заключается в приведении к виду стандартного нормального распределения. Для ее реализации необходимо вычислить среднее значение признака

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_i^j$$

и стандартное отклонение, которое находится путем суммирования квадратов отклонения значений признака на объектах выборки от среднего μ_j и делением на число объектов выборки с последующим извлечением корня:

$$\sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i^j - \mu_j)^2}$$

Чтобы масштабировать признак, каждое его значение преобразуется по формуле

$$x_i^j = \frac{x_i^j - \mu_j}{\sigma_j}$$

Масштабирование признаков

Важным свойством масштабирования является факт, что после масштабирования признаков в линейных моделях веса при них могут **интерпретироваться** как мера значимости этих признаков.

Существуют различные ситуации, когда целесообразно применять тот или иной метод масштабирования. Нормализовать полезно признаки, опирающиеся на величину значений — такие как расстояние. Стандартизировать полезно признаки для модели, которая опирается на распределение.

Регуляризация

Регуляризация

При работе с линейными моделями очень важно, чтобы признаки не были мультиколлинеарными. В случае коррелированности признаков веса при связанных признаках могут неконтролируемо расти и, в целом, модель становится неустойчивой.

Как один из методов борьбы с мультиколлинеарностью и в целом переобучением, служит **регуляризация** - накладывание дополнительных ограничений через добавление в функцию потерь.

L1 и L2 регуляризации

L1 Regularization

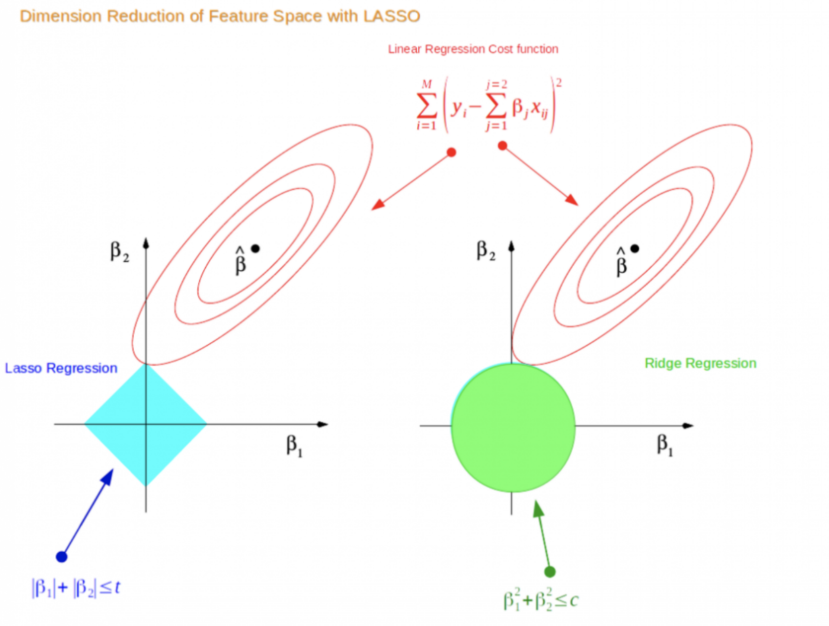
$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

Loss function

Regularization Term



Перерыв
10 минут

education

Линейная классификация

Линейная классификация

В отличии от задач регрессии, в которых восстанавливается непрерывная зависимость по имеющимся данным, существуют задачи *классификации*, где объекты делятся на конечное количество классов, и целью обучения является получение модели, способной соотносить объекты к тому или иному классу.

Простейшим случаем является *бинарная классификация*, когда у нас имеется два класса. Единственное отличие от линейной регрессии здесь в том, что пространство ответов состоит из двух элементов, в нашем случае возьмем $\mathbb{Y}=\{-1,1\}$, где -1 и 1 означают принадлежность к первому или второму классу, соответственно. Примером такой задачи может являться задача распознавания спам-писем. В этом случае, -1 означает, что письмо не является спамом, а 1 — что является.

Линейная классификация

Как и в случае регрессии, в классификации можно использовать линейные модели. Это называется *линейной классификацией.* Линейные классификаторы устроены похожим на линейную регрессию образом, за одним лишь различием — для получения бинарных значений берется только знак от значения $a(x)$:

$$a(x) = \text{sign}(w_0 + \sum^{d}_{i=1} w_i x^i).$$

Аналогично линейной регрессии, после добавления константного признака формула имеет вид

$$\text{sign}(\sum^{d+1}_{i=1} w_i x^i) = \text{sign}(\langle w, x \rangle).$$

Множество точек $\langle w, x \rangle$=0 образует *гиперплоскость* в пространстве признаков и делит его на две части. Объекты, расположенные по разные стороны от нее, относятся к разным классам.

Линейная классификация

Стоит отметить, что для некоторого объекта x расстояние до этой гиперплоскости будет равняться $|\langle w,x\rangle| / \|w\|$, соответственно, при классификации нам важен не только знак скалярного произведения $\langle w,x\rangle$, но и его значение: чем выше оно, тем больше будет расстояние от объекта до разделяющей гиперплоскости, что будет означать, что алгоритм более уверен в отнесении объекта к данному классу. Это приводит нас к значению *отступа*, который равен скалярному произведению вектора весов w на вектор признаков x, умноженному на истинное значение ответа y, которое, как мы помним, принимает значения -1 и 1:

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

Таким образом, если скалярное произведение отрицательно, и истинный ответ равен -1, отступ будет больше нуля. Если скалярное произведение положительно, и истинный ответ равен 1, отступ также будет положителен. То есть $M_i > 0$, когда классификатор дает верный ответ, и $M_i < 0$, когда классификатор ошибается. Отступ характеризует корректность ответа, а его абсолютное значение свидетельствует о расстоянии от разделяющей гиперплоскости, то есть о мере уверенности в ответе.

Ошибка линейной классификации

Как и в случае линейной регрессии, для обучения алгоритма линейной классификации требуется измерять ошибку. По аналогии с средней абсолютной ошибкой и среднеквадратичной ошибкой в случае линейной классификации можно использовать естественный подход: так как возможных ответов конечное число, можно требовать полного совпадения предсказанного класса $a(x_i)$ и истинного y_i. Тогда в качестве функционала ошибки можно использовать долю неправильных ответов:

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

или, используя понятие отступа,

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

Функция, стоящая под знаком суммы, называется *функцией потерь*. График ее в зависимости от отступа будет иметь пороговый вид.

Логистическая регрессия

Логистическая регрессия — частный случай линейного классификатора, обладающий одной полезной особенностью: помимо отнесения объекта к определенному классу она умеет прогнозировать вероятность P того, что объект относится к этому классу.

Во многих задачах такая особенность является очень важной. Например, в задачах кредитного скоринга (предсказание, вернет клиент кредит или нет) прогнозируют вероятность невозврата кредита и на основании нее принимают решение о выдаче или невыдаче.

Логистическая регрессия

Пусть в каждой точке пространства объектов $\mathbb{X}$ задана вероятность того, что объект x будет принадлежать к классу "+1" $P\,(y{=}1|x)$ (условная вероятность y=1 при условии x). Она будет принимать значения от 0 до 1, и нам нужно каким-то образом ее предсказывать, но пока мы умеем только строить прогноз методами линейной регрессии с помощью некоего алгоритма $b(x) = \langle w,\, x_i \rangle$. У него есть проблема, связанная с тем, что скалярное произведение $\langle w,\, x_i \rangle$ не всегда возвращает значения в отрезке [0, 1]. Чтобы достичь такого условия, можно использовать некую функцию $\sigma : \mathbb{R} \to [0,1]$, которая будет переводить полученное в скалярном произведении значение в вероятность, пределы которой будут лежать в промежутке от 0 до 1. В модели логистической регрессии в качестве такой функции берется сигмоида, которая имеет вид:

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$

Таким образом, чем больше будет скалярное произведение $z = \langle w,\, x_i \rangle$, тем выше будет предсказанная вероятность.

Оценка качества классификации

Accuracy

Как и в случае линейной регрессии, в задачах классификации требуется оценивать качество обученной модели. Для этого существует большое количество подходов.

Наиболее очевидным и простым способом является расчет *доли правильных ответов*:

$$accuracy(a, x) = \frac{1}{N} \sum_{i=1}^{N} [a(x_i) = y_i]$$

Accuracy

Эта метрика имеет определенные недостатки:

- Она может неадекватно работать на несбалансированных выборках, в которых объектов одного класса намного больше остальных: например, если у нас имеется выборка с 950 объектами класса +1 и 50 класса -1, обыкновенная константная модель классификатора, которая на всех объектах отдает ответ +1, будет иметь долю правильных ответов 0.95, при этом сам классификатор является абсолютно бесполезным. Методом борьбы с этим заключается в введении коэффициента q_0, равного доле объектов самого большого класса.

  Доля правильных ответов для корректных алгоритмов должна лежать в промежутке [q_0, 1]

- Она не учитывает "цены ошибок". В некоторых прикладных задачах ошибки разного рода могут иметь разную важность. Например, если говорить о кредитном скоринге, при постановке задачи необходимо определить, какая ошибка будет хуже: выдать кредит "плохому" клиенту или не выдать "хорошему". При этом используемая метрика качества должна учитывать цены разных ошибок.

Матрица ошибок

Удобно представлять ответы в виде комбинации истинного ответа и ответа алгоритма. При этом получается так называемая *матрица ошибок*.

	y = +1	y = -1
$a(x)$ = +1	TP	FP
$a(x)$ = -1	FN	TN

В матрице сверху отложены истинные ответы, а слева – ответы алгоритма. Когда алгоритм относит объект к классу "+1", говорят, что он *срабатывает*, а когда к "-1" — *пропускает*. Если алгоритм дал положительный ответ и объект действительно относится к классу "+1", говорят, что имеет место верное срабатывание (True Positive, TP), а если объект не относится к классу "+1", это ложное срабатывание (False Positive, FP). Если алгоритм пропускает объект, а его истинный класс "+1", это ложный пропуск (False Negative, FN), а если истинный класс объекта "-1", имеет место истинный пропуск (True Negative, TN). При такой классификации уже есть два вида ошибок: ложные срабатывания и ложные пропуски. По главной диагонали в матрице ошибок располагаются верные ответы, по побочной — неверные.

Точность и полнота

Часто используются две метрики - *точность* и *полнота*.

Точность (precision) представляет из себя долю истинных срабатываний от общего количества срабатываний. Она показывает, насколько можно доверять алгоритму классификации в случае срабатывания

$$precision(a,X) = TP / (TP+FP).$$

Полнота (recall) считается как доля объектов, истинно относящихся к классу "+1", которые алгоритм отнес к этому классу

$$recall(a,X) = TP / (TP+FN),$$

здесь $TP+FN$ как раз будут вместе составлять весь список объектов класса "+1".

Точность и полнота

Пусть у нас есть выборка из 100 объектов, из которых 50 относится к классу "+1" и 50 к классу "-1" и для этой работы с этой выборкой мы рассматриваем две модели:

$a_1(x)$ с матрицей ошибок

	y = +1	y = -1
$a_1(x)$ = +1	40	10
$a_1(x)$ = -1	10	40

и $a_2(x)$ с матрицей ошибок:

	y = +1	y = -1
$a_2(x)$ = +1	22	2
$a_2(x)$ = -1	29	48

Тогда:

$$precision(a_1, X) = 0.8 \quad recall(a_1, X) = 0.8$$

$$precision(a_2, X) = 0.92 \quad recall(a_2, X) = 0.44$$

Точность и полнота

Видно, что точность второй модели очень высока, но при этом сильно снижена полнота. Поэтому нужно правильно формировать бизнес-требования к модели, какой именно показатель должен быть определяющим. Например, если в задаче кредитного скоринга банк ставит цель возврата 90% кредитов, задачей ставится максимизация полноты при условии точности не ниже 0.9. А если при распознавании спама стоит требование, например, распознавать 95% спам-писем, задача состоит в максимизации точности при условии полноты не ниже 0.95.

Точность и полнота

Однако, такое ограничение есть не всегда, и в остальных случаях требуется максимизировать и полноту и точность. Есть различные варианты объединения их в одну метрику, одним из наиболее удобных из них является *F-мера*, которая представляет собой среднее гармоническое между точностью и полнотой

$$F = (2 \cdot precision \cdot recall) / (presision + recall).$$

В отличие от, например, среднего арифметического, если хотя бы один из аргументов близок к нулю, то и среднее гармоническое будет близко к нулю. По сути, F-мера является сглаженной версией минимума из точности и полноты.

ROC-кривая

ROC-кривая позволяет оценить модель в целом, не привязываясь к конкретному порогу. Такая кривая строится в следующих координатах:

по оси x откладывается доля ложных срабатываний (False Positive Rate) — отношение числа ложных срабатываний к общему размеру отрицательного класса:

$$FPR = FP / (FP+TN),$$

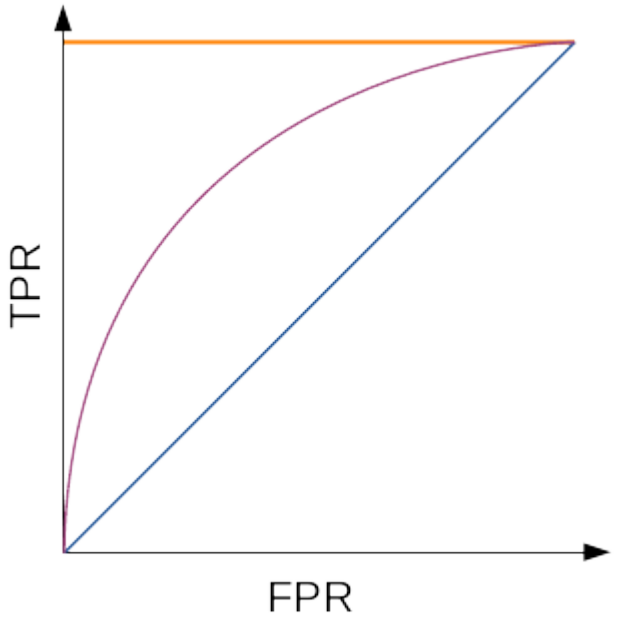
по оси y откладывается доля верных срабатываний (True Positive Rate) — отношение числа верных срабатываний к размеру положительного класса:

$$TPR = TP / (TP+FN),$$

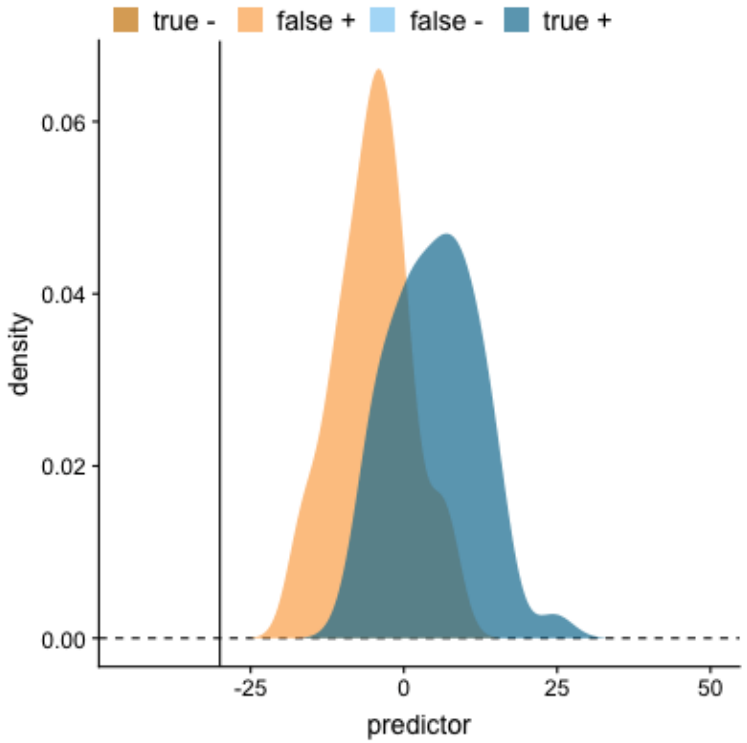
то есть TPR по сути представляет из себя полноту, о которой мы говорили ранее.

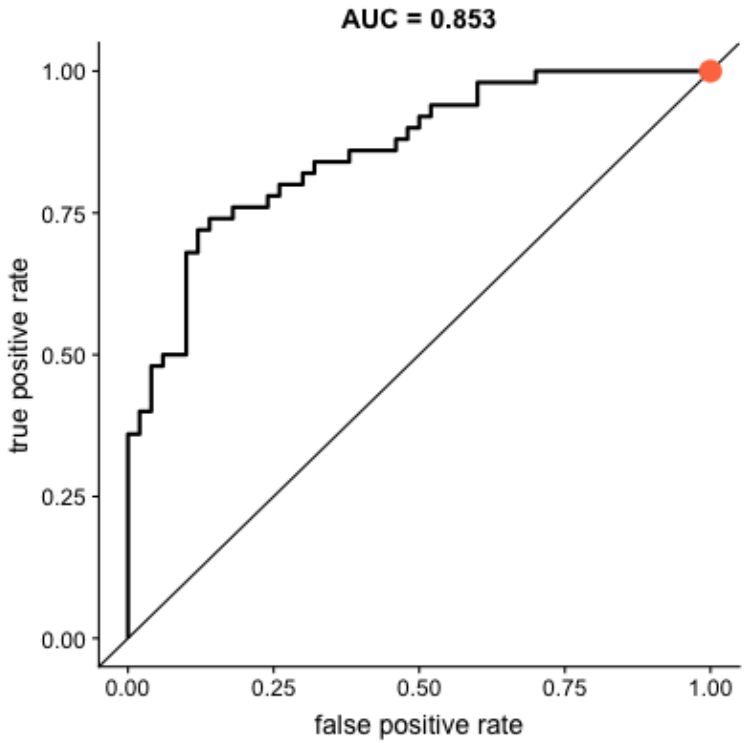
ROC-кривая

ROC кривая всегда идет из точки (0,0) в точку (1,1). При этом в случае наличия идеального классификатора с определенным порогом доля его верных ответов будет равна 1, а доля ложных срабатываний 0, то есть график будет проходить через точку (0,1). Таким образом, чем ближе к этой точке проходит ROC-кривая, тем лучше наши оценки и лучше используемое семейство алгоритмов.



AUC-ROC


true -
false +
false -
true +
density
0.06
0.04
0.02
0.00
-25
0
25
50
predictor


AUC = 0.853
true positive rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
false positive rate

AUC-ROC

Мерой качества оценок принадлежности к классу 1 может служить площадь под ROC-кривой. Такая метрика называется AUC-ROC (Area Under Curve - площадь под кривой ROC). В случае идеального алгоритма $AUC–ROC$=1, а в случае худшего приближается к 0.5.

Критерий AUC-ROC можно интерпретировать как вероятность того, что если выбрать случайные положительный и отрицательный объект выборки, положительный объект получит оценку принадлежности выше, чем отрицательный.

AUC-ROC

Как и accuracy мера AUC-ROC не устойчива к несбалансированным выборкам. Допустим, нам нужно выбрать 100 релевантных документов из выборки в 1000000 документов. И у нас есть алгоритм, который дает выборку из 5000 документов, 90 из которых релевантны. В этом случае

$$TPR = TP / (TP+FN) = 90 / (90 + 10) = 0.9$$

$$FPR = FP / (FP+TN) = 4910 / (4910+994990) = 0.00491,$$

Что является показателями очень хорошего алгоритма: AUC-ROC близка к 1, хотя на самом деле 1900 из 2000 выданных документов являются нерелевантными.

Чтобы посмотреть реальное положение дел, рассчитаем точность и полноту:

$$precision = TP / (TP+FP) = 90 / (90+4910) = 0.018$$

$$recall = TPR = 0.9.$$

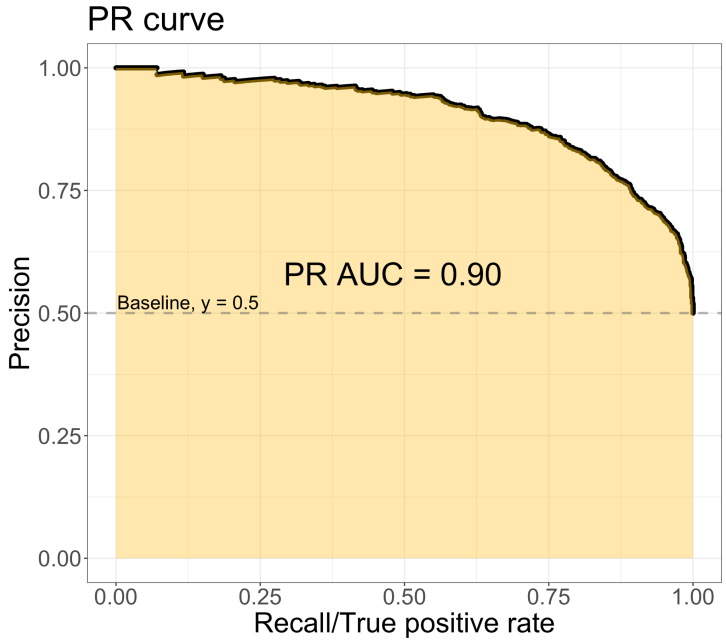
Здесь уже видно, что алгоритм является недостаточно точным.

PR-кривая

Оценить несбалансированные выборки можно посредством *кривой точности-полноты (PR-кривой)*. По оси x откладывается полнота, по оси y — точность.

Точка на графике, аналогично ROC-кривой, будет соответствовать конкретному классификатору с некоторым значением порога.

В случае наличия идеального классификатора, у которого точность и полнота 100%, кривая пройдет через точку (1,1). Таким образом, чем ближе к этой точке кривая проходит, тем лучше оценки. Так что, как и в случае ROC-кривой, можно ввести метрику качества в виде площади под PR-кривой AUC-PR.



Практика

education