



328,33

Рейтинг

Open Data Science

Крупнейшее русскоязычное Data Science сообщество



yorko 20 марта 2017 в 19:38

Открытый курс машинного обучения. Тема 4. Линейные модели классификации и регрессии

Блог компании Open Data Science, Python, Алгоритмы, Математика, Машинное обучение

Всем привет!

Сегодня мы детально обсудим очень важный класс моделей машинного обучения – линейных. Ключевое отличие нашей подачи материала от аналогичной в курсах эконометрики и статистики – это акцент на практическом применении линейных моделей в реальных задачах (хотя и математики тоже будет немало).

Пример такой задачи – это соревнование Kaggle Inclass по идентификации пользователя в Интернете по его последовательности переходов по сайтам.

UPD: теперь курс — на английском языке под брендом [mlcourse.ai](#) со статьями на Medium, а материалами — на Kaggle (Dataset) и на GitHub.

Все материалы доступны на GitHub.

А вот [видеозапись](#) лекции по мотивам этой статьи в рамках второго запуска открытого курса (сентябрь–ноябрь 2017). В ней, в частности рассмотрены два бенчмарка соревнования, полученные с помощью логистической регрессии.

Список статей серии

План этой статьи:

1. Линейная регрессия

- Метод наименьших квадратов
- Метод максимального правдоподобия
- Разложение ошибки на смещение и разброс (Bias-variance decomposition)
- Регуляризация линейной регрессии

2. Логистическая регрессия

- Линейный классификатор
- Логистическая регрессия как линейный классификатор
- Принцип максимального правдоподобия и логистическая регрессия
- L2-регуляризация логистической функции потерь

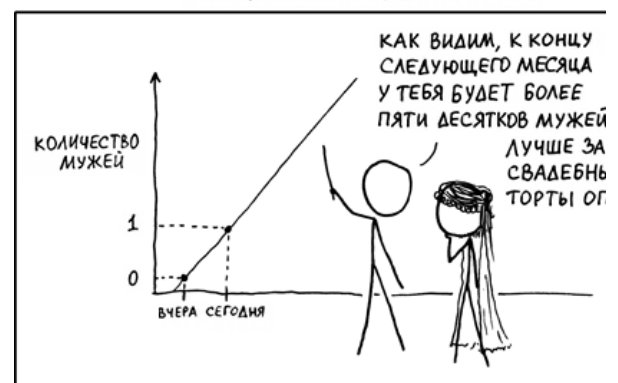
3. Наглядный пример регуляризации логистической регрессии

4. Где логистическая регрессия хороша и где не очень

- Анализ отзывов IMDB к фильмам
- XOR-проблема

5. Кривые валидации и обучения

МОЁ ХОББИ: ЭКСТРАПОЛИРОВАТЬ



- 6. Плюсы и минусы линейных моделей в задачах машинного обучения
- 7. Домашнее задание №4
- 8. Полезные ресурсы

1. Линейная регрессия

Метод наименьших квадратов

Рассказ про линейные модели мы начнем с линейной регрессии. В первую очередь, необходимо задать модель зависимости объясняемой переменной y от объясняющих ее факторов, функция зависимости будет линейной: $y = w_0 + \sum_{i=1}^m w_i x_i$. Если мы добавим фиктивную размерность $x_0 = 1$ для каждого наблюдения, тогда линейную форму можно переписать чуть более компактно, записав свободный член под сумму: $y = \sum_{i=0}^m w_i x_i = \vec{w}^T \vec{x}$. Если рассматривать матрицу наблюдения-признаки, у которой в строках находятся примеры из n данных, то нам необходимо добавить единичную колонку слева. Зададим модель следующим образом:

$$\vec{y} = X\vec{w} + \epsilon,$$

где

- $\vec{y} \in \mathbb{R}^n$ – объясняемая (или целевая) переменная;
- \vec{w} – вектор параметров модели (в машинном обучении эти параметры часто называют весами);
- X – матрица наблюдений и признаков размерности n строк на $m + 1$ столбцов (включая фиктивную единичную колонку слева) с рангом по столбцам: $\text{rank}(X) = m + 1$;
- ϵ – случайная переменная, соответствующая случайной, непрогнозируемой ошибке модели.

Можем выписать выражение для каждого конкретного наблюдения

$$y_i = \sum_{j=0}^m w_j X_{ij} + \epsilon_i$$

Также на модель накладываются следующие ограничения (иначе это будет какая то другая регрессия, но точно не линейная):

- матожидание случайных ошибок равно нулю: $\forall i : \mathbb{E}[\epsilon_i] = 0$;
- дисперсия случайных ошибок одинакова и конечна, это свойство называется гомоскедастичностью: $\forall i : \text{Var}(\epsilon_i) = \sigma^2 < \infty$;
- случайные ошибки не скоррелированы: $\forall i \neq j : \text{Cov}(\epsilon_i, \epsilon_j) = 0$.

Оценка \hat{w}_i весов w_i называется линейной, если

$$\hat{w}_i = w_{1i}y_1 + w_{2i}y_2 + \dots + w_{ni}y_n,$$

где $\forall i$ w_{ki} зависит только от наблюдаемых данных X и почти наверняка нелинейно. Так как решением задачи поиска оптимальных весов будет именно линейная оценка, то и модель называется *линейной регрессией*. Введем еще одно определение. Оценка \hat{w}_i называется несмещенной тогда, когда матожидание оценки равно реальному, но неизвестному значению оцениваемого параметра:

$$\mathbb{E}[\hat{w}_i] = w_i$$

Один из способов вычислить значения параметров модели является **метод наименьших квадратов** (МНК), который минимизирует среднеквадратичную ошибку между реальным значением зависимой переменной и прогнозом, выданным моделью:

$$\begin{aligned}
 \mathcal{L}(X, \vec{y}, \vec{w}) &= \frac{1}{2n} \sum_{i=1}^n \left(y_i - \vec{w}^T \vec{x}_i \right)^2 \\
 &= \frac{1}{2n} \|\vec{y} - X\vec{w}\|_2^2 \\
 &= \frac{1}{2n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w})
 \end{aligned}$$

Для решения данной оптимизационной задачи необходимо вычислить производные по параметрам модели, приравнять их к нулю и решить полученные уравнения относительно \vec{w} (матричное дифференцирование неподготовленному читателю может показаться затруднительным, попробуйте расписать все через суммы, чтобы убедиться в ответе):

Шпаргалка по матричным производным

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial \vec{w}} &= \frac{\partial}{\partial \vec{w}} \frac{1}{2n} \left(\vec{y}^T \vec{y} - 2\vec{y}^T X\vec{w} + \vec{w}^T X^T X\vec{w} \right) \\
 &= \frac{1}{2n} (-2X^T \vec{y} + 2X^T X\vec{w})
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 &\Leftrightarrow \frac{1}{2n} (-2X^T \vec{y} + 2X^T X\vec{w}) = 0 \\
 &\Leftrightarrow -X^T \vec{y} + X^T X\vec{w} = 0 \\
 &\Leftrightarrow X^T X\vec{w} = X^T \vec{y} \\
 &\Leftrightarrow \vec{w} = (X^T X)^{-1} X^T \vec{y}
 \end{aligned}$$

Итак, имея в виду все определения и условия описанные выше, мы можем утверждать, опираясь на теорему Маркова-Гаусса, что оценка является лучшей оценкой параметров модели, среди всех *линейных* и *несмещенных* оценок, то есть обладающей наименьшей дисперсией.

Метод максимального правдоподобия

У читателя вполне резонно могли возникнуть вопросы: например, почему мы минимизируем среднеквадратичную ошибку, а не что-то другое? Ведь можно минимизировать среднее абсолютное значение невязки или еще что-то. Единственное, что произойдет в случае изменения минимизируемого значения, так это то, что мы выйдем из условий теоремы Маркова-Гаусса, и наши оценки перестанут быть лучшими среди линейных и несмещенных.

Давайте перед тем как продолжить, сделаем лирическое отступление, чтобы проиллюстрировать метод максимального правдоподобия простом примере.

Как-то после школы я заметил, что все помнят формулу этилового спирта. Тогда я решил провести эксперимент: помнят ли люди более простую формулу метилового спирта: CH_3OH . Мы опросили 400 человек и оказалось, что формулу помнят всего 117 человек. Разумно предположить, что вероятность того, что следующий опрошенный знает формулу метилового спирта – $\frac{117}{400} \approx 0.29$. Покажем, что такая интуитивно понятная оценка не просто хороша, а еще и является оценкой максимального правдоподобия.

Разберемся, откуда берется эта оценка, а для этого вспомним определение распределения Бернулли: случайная величина X имеет распределение Бернулли, если она принимает всего два значения (1 и 0 с вероятностями θ и $1 - \theta$ соответственно) и имеет следующую функцию распределения вероятности:

$$p(\theta, x) = \theta^x (1 - \theta)^{(1-x)}, x \in \{0, 1\}$$

Похоже, это распределение – то, что нам нужно, а параметр распределения θ и есть та оценка вероятности того, что человек знает формулу метилового спирта. Мы проделали **400 независимых** экспериментов, обозначим их исходы как $\vec{x} = (x_1, x_2, \dots, x_{400})$. Запишем *правдоподобие* наших данных (наблюдений), то есть вероятность наблюдать 117 реализаций случайной величины $X = 1$ и 283 реализации $X = 0$:

$$p(\vec{x} | \theta) = \prod_{i=1}^{400} \theta^{x_i} (1 - \theta)^{(1-x_i)} = \theta^{117} (1 - \theta)^{283}$$

Далее будем максимизировать это выражение по θ , и чаще всего это делают не с правдоподобием $p(\vec{x} | \theta)$, а с его логарифмом (применение монотонного преобразования не изменит решение, но упростит вычисления):

$$\begin{aligned}\log p(\vec{x} \mid \theta) &= \log \prod_{i=1}^{400} \theta^{x_i} (1 - \theta)^{(1-x_i)} = \\ &= \log \theta^{117} (1 - \theta)^{283} = 117 \log \theta + 283 \log(1 - \theta)\end{aligned}$$

Теперь мы хотим найти такое значение θ , которое максимизирует правдоподобие, для этого мы возьмем производную по θ , приравняем нулю и решим полученное уравнение:

$$\frac{\partial p(\vec{x} \mid \theta)}{\partial \theta} = \frac{\partial}{\partial \theta} (117 \log \theta + 283 \log(1 - \theta)) = \frac{117}{\theta} - \frac{283}{1 - \theta};$$

$$\frac{117}{\theta} - \frac{283}{1 - \theta} = 0 \Rightarrow \theta = \frac{117}{400}.$$

Получается, что наша интуитивная оценка – это и есть оценка максимального правдоподобия. Применим теперь те же рассуждения для задачи линейной регрессии и попробуем выяснить, что лежит за среднеквадратичной ошибкой. Для этого нам придется посмотреть на линейную регрессию с вероятностной точки зрения. Модель, естественно, остается такой же:

$$\vec{y} = X\vec{w} + \epsilon,$$

но будем теперь считать, что случайные ошибки берутся из централизованного нормального распределения:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Перепишем модель в новом свете:

$$\begin{aligned}p(y_i \mid X, \vec{w}) &= \sum_{j=1}^m w_j X_{ij} + \mathcal{N}(0, \sigma^2) \\ &= \mathcal{N}\left(\sum_{j=1}^m w_j X_{ij}, \sigma^2\right)\end{aligned}$$

Так как примеры берутся независимо (ошибки не скоррелированы – одно из условий теоремы Маркова-Гаусса), то полное правдоподобие данных будет выглядеть как произведение функций плотности $p(y_i)$. Рассмотрим логарифм правдоподобия, что позволит нам перейти произведения к сумме:

$$\begin{aligned}\log p(\vec{y} \mid X, \vec{w}) &= \log \prod_{i=1}^n \mathcal{N}\left(\sum_{j=1}^m w_j X_{ij}, \sigma^2\right) \\ &= \sum_{i=1}^n \log \mathcal{N}\left(\sum_{j=1}^m w_j X_{ij}, \sigma^2\right) \\ &= -\frac{n}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^n \left(y_i - \vec{w}^T \vec{x}_i\right)^2\end{aligned}$$

Мы хотим найти гипотезу максимального правдоподобия, т.е. нам нужно максимизировать выражение $p(\vec{y} \mid X, \vec{w})$, а это то же самое, что максимизация его логарифма. Обратите внимание, что при максимизации функции по какому-то параметру можно выкинуть все члены зависящие от этого параметра:

$$\begin{aligned}
\hat{w} &= \arg \max_w p(\vec{y} | X, \vec{w}) \\
&= \arg \max_w -\frac{n}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^n \left(y_i - \vec{w}^T \vec{x}_i\right)^2 \\
&= \arg \max_w -\frac{1}{2\sigma^2} \sum_{i=1}^n \left(y_i - \vec{w}^T \vec{x}_i\right)^2 \\
&= \arg \max_w -\mathcal{L}(X, \vec{y}, \vec{w})
\end{aligned}$$

Таким образом, мы увидели, что максимизация правдоподобия данных – это то же самое, что и минимизация среднеквадратичной ощи (при справедливости указанных выше предположений). Получается, что именно такая функция стоимости является следствием того, что ошибка распределена нормально, а не как-то по-другому.

Разложение ошибки на смещение и разброс (Bias-variance decomposition)

Поговорим немного о свойствах ошибки прогноза линейной регрессии (в принципе эти рассуждения верны для всех алгоритмов машинного обучения). В свете предыдущего пункта мы выяснили, что:

- истинное значение целевой переменной складывается из некоторой детерминированной функции $f(\vec{x})$ и случайной ошибки ϵ : $y = f(\vec{x}) + \epsilon$;
- ошибка распределена нормально с центром в нуле и некоторым разбросом: $\epsilon \sim \mathcal{N}(0, \sigma^2)$;
- истинное значение целевой переменной тоже распределено нормально: $y \sim \mathcal{N}(f(\vec{x}), \sigma^2)$
- мы пытаемся приблизить детерминированную, но неизвестную функцию $f(\vec{x})$ линейной функцией от регрессоров $\hat{f}(\vec{x})$, которая, в очередь, является точечной оценкой функции f в пространстве функций (точнее, мы ограничили пространство функций параметрическим семейством линейных функций), т.е. случайной переменной, у которой есть среднее значение и дисперсия.

Тогда ошибка в точке \vec{x} раскладывается следующим образом:

$$\begin{aligned}
\text{Err}(\vec{x}) &= \mathbb{E} \left[\left(y - \hat{f}(\vec{x}) \right)^2 \right] \\
&= \mathbb{E} [y^2] + \mathbb{E} \left[\left(\hat{f}(\vec{x}) \right)^2 \right] - 2\mathbb{E} [y \hat{f}(\vec{x})] \\
&= \mathbb{E} [y^2] + \mathbb{E} [\hat{f}^2] - 2\mathbb{E} [y \hat{f}]
\end{aligned}$$

Для наглядности опустим обозначение аргумента функций. Рассмотрим каждый член в отдельности, первые два расписываются легко в формуле $\text{Var}(z) = \mathbb{E}[z^2] - \mathbb{E}[z]^2$:

$$\begin{aligned}
\mathbb{E}[y^2] &= \text{Var}(y) + \mathbb{E}[y]^2 = \sigma^2 + f^2 \\
\mathbb{E}[\hat{f}^2] &= \text{Var}(\hat{f}) + \mathbb{E}[\hat{f}]^2
\end{aligned}$$

Пояснения:

$$\begin{aligned}
\text{Var}(y) &= \mathbb{E} \left[(y - \mathbb{E}[y])^2 \right] \\
&= \mathbb{E} \left[(y - f)^2 \right] \\
&= \mathbb{E} \left[(f + \epsilon - f)^2 \right] \\
&= \mathbb{E} [\epsilon^2] = \sigma^2
\end{aligned}$$

$$\mathbb{E}[y] = \mathbb{E}[f + \epsilon] = \mathbb{E}[f] + \mathbb{E}[\epsilon] = f$$

И теперь последний член суммы. Мы помним, что ошибка и целевая переменная независимы друг от друга:

$$\begin{aligned}\mathbb{E}[y\hat{f}] &= \mathbb{E}[(f + \epsilon)\hat{f}] \\ &= \mathbb{E}[f\hat{f}] + \mathbb{E}[\epsilon\hat{f}] \\ &= f\mathbb{E}[\hat{f}] + \mathbb{E}[\epsilon]\mathbb{E}[\hat{f}] = f\mathbb{E}[\hat{f}]\end{aligned}$$

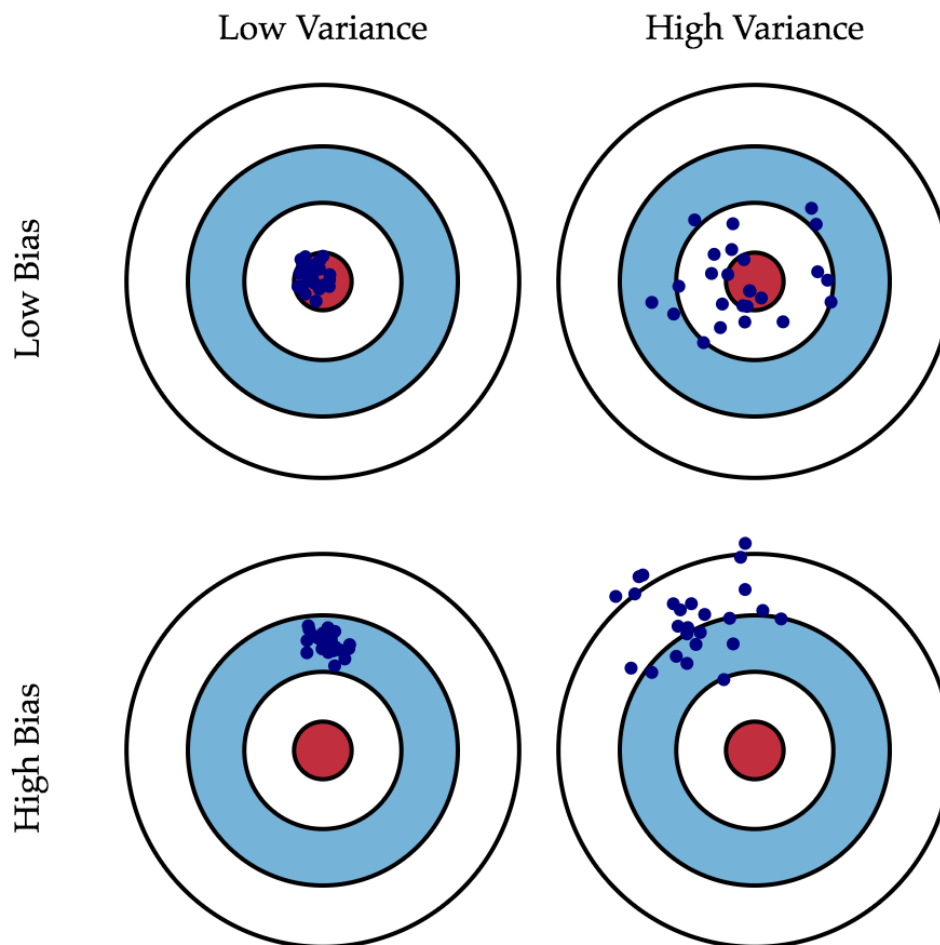
Наконец, собираем все вместе:

$$\begin{aligned}\text{Err}(\vec{x}) &= \mathbb{E}\left[\left(y - \hat{f}(\vec{x})\right)^2\right] \\ &= \sigma^2 + f^2 + \text{Var}(\hat{f}) + \mathbb{E}[\hat{f}]^2 - 2f\mathbb{E}[\hat{f}] \\ &= \left(f - \mathbb{E}[\hat{f}]\right)^2 + \text{Var}(\hat{f}) + \sigma^2 \\ &= \text{Bias}(\hat{f})^2 + \text{Var}(\hat{f}) + \sigma^2\end{aligned}$$

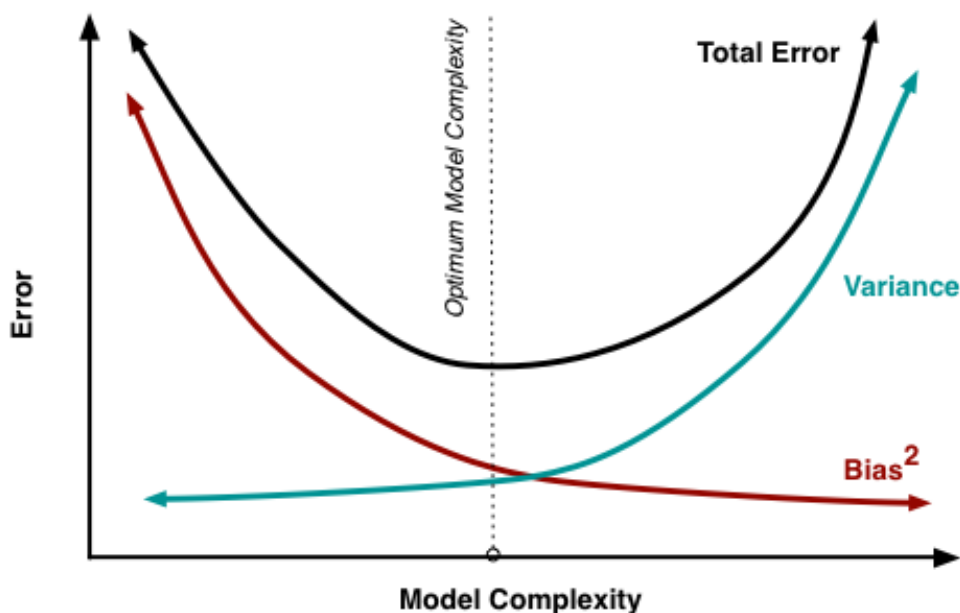
Итак, мы достигли цели всех вычислений, описанных выше, последняя формула говорит нам, что ошибка прогноза любой модели вида $y = f(\vec{x}) + \epsilon$ складывается из:

- квадрата смещения: $\text{Bias}(\hat{f})$ – средняя ошибка по всевозможным наборам данных;
- дисперсии: $\text{Var}(\hat{f})$ – вариативность ошибки, то, на сколько ошибка будет отличаться, если обучать модель на разных наборах дан
- неустранимой ошибки: σ^2 .

Если с последней мы ничего сделать не можем, то на первые два слагаемых мы можем как-то влиять. В идеале, конечно же, хотелось бы свести на нет оба этих слагаемых (левый верхний квадрат рисунка), но на практике часто приходится балансировать между смещенным нестабильными оценками (высокая дисперсия).



Как правило, при увеличении сложности модели (например, при увеличении количества свободных параметров) увеличивается дисперсия (разброс) оценки, но уменьшается смещение. Из-за того что тренировочный набор данных полностью запоминается вместо обобщения небольшие изменения приводят к неожиданным результатам (переобучение). Если же модель слабая, то она не в состоянии выучить закономерность, в результате выучивается что-то другое, смещенное относительно правильного решения.



Теорема Маркова-Гаусса как раз утверждает, что МНК-оценка параметров линейной модели является самой лучшей в классе несмещенных линейных оценок, то есть с наименьшей дисперсией. Это значит, что если существует какая-либо другая несмещенная модель g тоже и класса линейных моделей, то мы можем быть уверены, что $\text{Var}(\hat{f}) \leq \text{Var}(g)$.

Регуляризация линейной регрессии

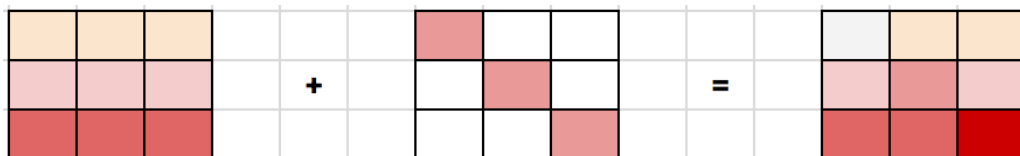
Иногда бывают ситуации, когда мы намеренно увеличиваем смещенность модели ради ее стабильности, т.е. ради уменьшения дисперсии модели $\text{Var}(\hat{f})$. Одним из условий теоремы Маркова-Гаусса является полный столбцовый ранг матрицы X . В противном случае решение МНК $\vec{w} = (X^T X)^{-1} X^T \vec{y}$ не существует, т.к. не будет существовать обратная матрица $(X^T X)^{-1}$. Другими словами, матрица $X^T X$ будет сингулярна, или вырождена. Такая задача называется некорректно поставленной. Задачу нужно скорректировать, а именно, сделать матрицу $X^T X$ невырожденной, или регулярной (именно поэтому этот процесс называется регуляризацией). Чаще в данных мы можем наблюдать так называемую *мультиколлинеарность* — когда два или несколько признаков сильно коррелированы, в матрице X это проявляется в виде "почти" линейной зависимости столбцов. Например, в задаче прогнозирования цены квартиры по ее параметрам "площадь с учетом балкона" и "площадь без учета балкона". Формально для таких данных матрица $X^T X$ будет обратима, но из-за мультиколлинеарности у матрицы $X^T X$ некоторые собственные значения будут близки к нулю, а в обратной матрице $(X^T X)^{-1}$ появятся экстремально большие собственные значения, т.к. собственные значения обратной матрицы — это $\frac{1}{\lambda_i}$. Из-за такого шатания собственных значений станет нестабильная оценка параметров модели, т.е. добавление нового наблюдения в набор тренировочных данных приведет к совершенно другому решению. Иллюстрации роста коэффициентов вы найдете в одном из наших постов. Одним из способов регуляризации является регуляризация Тихонова, которая в общем виде выглядит как добавление нового члена среднеквадратичной ошибки:

$$\mathcal{L}(X, \vec{y}, \vec{w}) = \frac{1}{2n} \|\vec{y} - X\vec{w}\|_2^2 + \|\Gamma\vec{w}\|_2^2$$

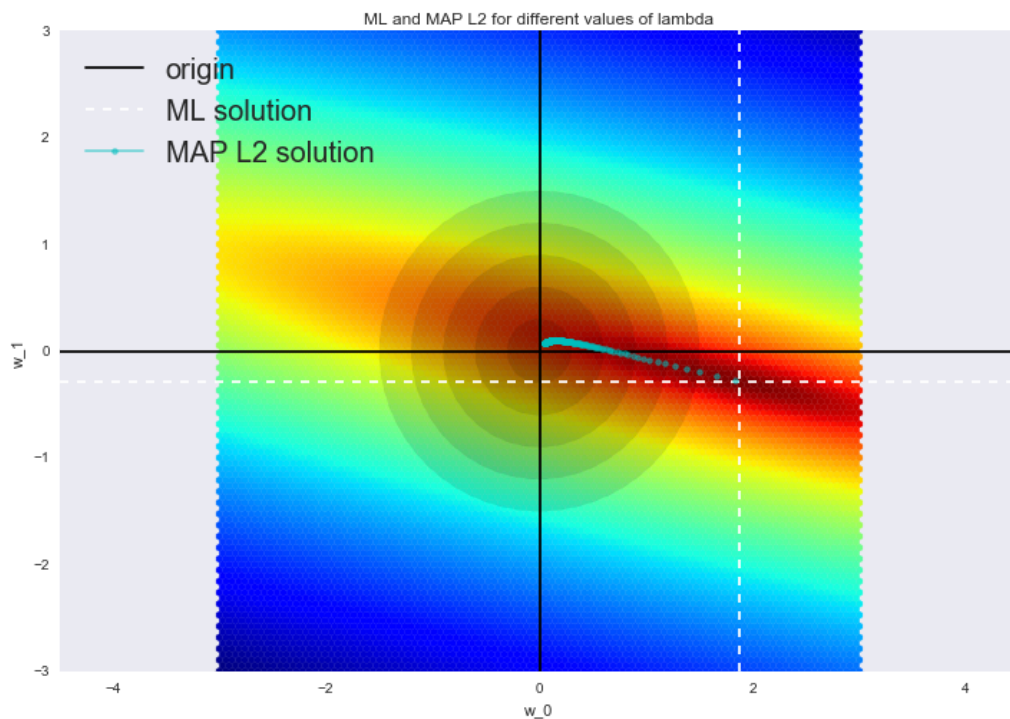
Часто матрица Тихонова выражается как произведение некоторого числа на единичную матрицу: $\Gamma = \frac{\lambda}{2} E$. В этом случае задача минимизации среднеквадратичной ошибки становится задачей с ограничением на L_2 норму. Если продифференцировать новую функцию стоимости по параметрам модели, приравнять полученную функцию к нулю и выразить \vec{w} , то мы получим точное решение задачи.

$$\vec{w} = (X^T X + \lambda E)^{-1} X^T \vec{y}$$

Такая регрессия называется гребневой регрессией (ridge regression). А гребнем является как раз диагональная матрица, которую мы прибавляем к матрице $X^T X$, в результате получается гарантированно регулярная матрица.



Такое решение уменьшает дисперсию, но становится смещенным, т.к. минимизируется также и норма вектора параметров, что заставляет решение сдвигаться в сторону нуля. На рисунке ниже на пересечении белых пунктирных линий находится МНК-решение. Голубыми точками обозначены различные решения гребневой регрессии. Видно, что при увеличении параметра регуляризации λ решение сдвигается в сторону нуля.

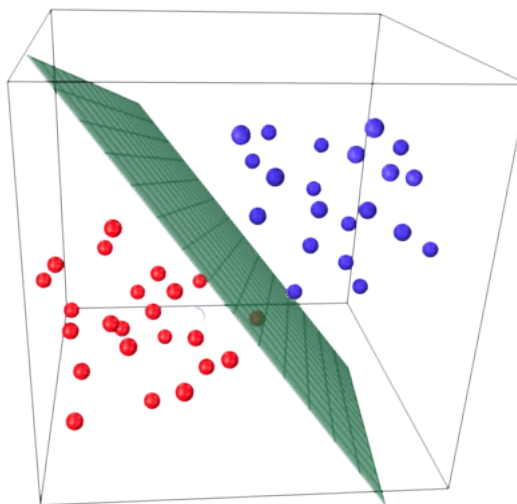


Советуем обратиться в наш [прошлый пост](#) за примером того, как L_2 регуляризация справляется с проблемой мультиколлинеарности, а чтобы освежить в памяти еще несколько интерпретаций регуляризации.

2. Логистическая регрессия

Линейный классификатор

Основная идея линейного классификатора заключается в том, что признаковое пространство может быть разделено гиперплоскостью на полупространства, в каждом из которых прогнозируется одно из двух значений целевого класса. Если это можно сделать без ошибок, то обучающая выборка называется *линейно разделимой*.



Мы уже знакомы с линейной регрессией и методом наименьших квадратов. Рассмотрим задачу бинарной классификации, причем метк целевого класса обозначим "+1" (положительные примеры) и "-1" (отрицательные примеры).

Один из самых простых линейных классификаторов получается на основе регрессии вот таким образом:

$$a(\vec{x}) = \text{sign}(\vec{w}^T x),$$

где

- \vec{x} – вектор признаков примера (вместе с единицей);
- \vec{w} – веса в линейной модели (вместе со смещением w_0);

- $\text{sign}(\bullet)$ – функция "сигнум", возвращающая знак своего аргумента;
- $a(\vec{x})$ – ответ классификатора на примере \vec{x} .

Логистическая регрессия как линейный классификатор

Логистическая регрессия является частным случаем линейного классификатора, но она обладает хорошим "умением" – прогнозировать вероятность p_+ отнесения примера \vec{x}_i к классу "+":

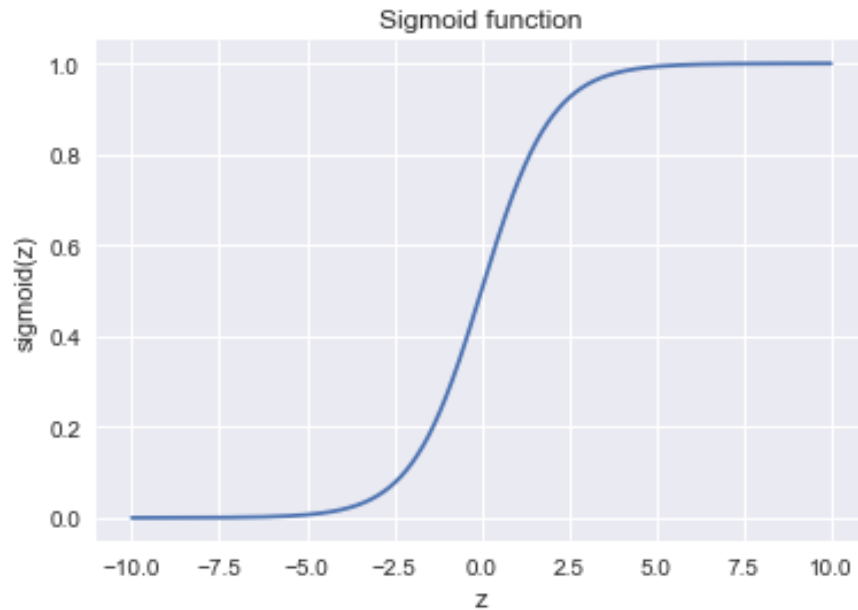
$$p_+ = P(y_i = 1 \mid \vec{x}_i, \vec{w})$$

Прогнозирование не просто ответа ("+" или "-"), а именно *вероятности* отнесения к классу "+" во многих задачах является очень важным бизнес-требованием. Например, в задаче кредитного скоринга, где традиционно применяется логистическая регрессия, часто прогноз вероятности невозврата кредита (p_+). Клиентов, обратившихся за кредитом, сортируют по этой предсказанной вероятности (по убыванию получается скоркарта — по сути, рейтинг клиентов от плохих к хорошим. Ниже приведен игрушечный пример такой скоркарты.

Клиент	Вероятность невозврата
Mike	0.78
Jack	0.45
Larry	0.13
Kate	0.06
William	0.03
Jessica	0.02

Банк выбирает для себя порог p_* предсказанной вероятности невозврата кредита (на картинке – **0.15**) и начиная с этого значения уже не выдает кредит. Более того, можно умножить предсказанную вероятность на выданную сумму и получить матожидание потерь с клиента тоже будет хорошей бизнес-метрикой (Далее в комментариях специалисты по скорингу могут поправить, но главная суть примерно так)

Итак, мы хотим прогнозировать вероятность $p_+ \in [0, 1]$, а пока умеем строить линейный прогноз с помощью МНК: $b(\vec{x}) = \vec{w}^T \vec{x} \in \mathbb{R}$. К образом преобразовать полученное значение в вероятность, пределы которой – $[0, 1]$? Очевидно, для этого нужна некоторая функция $f: \mathbb{R} \rightarrow [0, 1]$. В модели логистической регрессии для этого берется конкретная функция: $\sigma(z) = \frac{1}{1 + \exp(-z)}$. И сейчас разберемся, каков этого предпосылки.



Обозначим $P(X)$ вероятностью происходящего события X . Тогда отношение вероятностей $OR(X)$ определяется из $\frac{P(X)}{1-P(X)}$, а это — отношение вероятностей того, произойдет ли событие или не произойдет. Очевидно, что вероятность и отношение шансов содержат одинаковую информацию. Но в то время как $P(X)$ находится в пределах от 0 до 1, $OR(X)$ находится в пределах от 0 до ∞ .

Если вычислить логарифм $OR(X)$ (то есть называется логарифм шансов, или логарифм отношения вероятностей), то легко заметить, что $\log OR(X) \in \mathbb{R}$. Его-то мы и будем прогнозировать с помощью МНК.

Посмотрим, как логистическая регрессия будет делать прогноз $p_+ = P(y_i = 1 | \vec{x}_i, \vec{w})$ (пока считаем, что веса \vec{w} мы как-то получили обучили модель), далее разберемся, как именно).

- **Шаг 1.** Вычислить значение $w_0 + w_1 x_1 + w_2 x_2 + \dots = \vec{w}^T \vec{x}$. (уравнение $\vec{w}^T \vec{x} = 0$ задает гиперплоскость, разделяющую примеры класса);
- **Шаг 2.** Вычислить логарифм отношения шансов: $\log(OR_+) = \vec{w}^T \vec{x}$.
- **Шаг 3.** Имея прогноз шансов на отнесение к классу "+" — OR_+ , вычислить p_+ с помощью простой зависимости:

$$p_+ = \frac{OR_+}{1 + OR_+} = \frac{\exp^{\vec{w}^T \vec{x}}}{1 + \exp^{\vec{w}^T \vec{x}}} = \frac{1}{1 + \exp^{-\vec{w}^T \vec{x}}} = \sigma(\vec{w}^T \vec{x})$$

В правой части мы получили как раз сигмоид-функцию.

Итак, логистическая регрессия прогнозирует вероятность отнесения примера к классу "+" (при условии, что мы знаем его признаки и в модели) как сигмоид-преобразование линейной комбинации вектора весов модели и вектора признаков примера:

$$p_+(x_i) = P(y_i = 1 | \vec{x}_i, \vec{w}) = \sigma(\vec{w}^T \vec{x}_i).$$

Следующий вопрос: как модель обучается? Тут мы опять обращаемся к принципу максимального правдоподобия.

Принцип максимального правдоподобия и логистическая регрессия

Теперь посмотрим, как из принципа максимального правдоподобия получается оптимизационная задача, которую решает логистическая регрессия, а именно, — минимизация *логистической* функции потерь.

Только что мы увидели, что логистическая регрессия моделирует вероятность отнесения примера к классу "+" как

$$p_+(\vec{x}_i) = P(y_i = 1 | \vec{x}_i, \vec{w}) = \sigma(\vec{w}^T \vec{x}_i)$$

Тогда для класса "-" аналогичная вероятность:

$$p_-(\vec{x}_i) = P(y_i = -1 | \vec{x}_i, \vec{w}) = 1 - \sigma(\vec{w}^T \vec{x}_i) = \sigma(-\vec{w}^T \vec{x}_i)$$

Оба этих выражения можно ловко объединить в одно (следите за моими руками – не обманывают ли вас):

$$P(y = y_i | \vec{x}_i, \vec{w}) = \sigma(y_i \vec{w}^T \vec{x}_i)$$

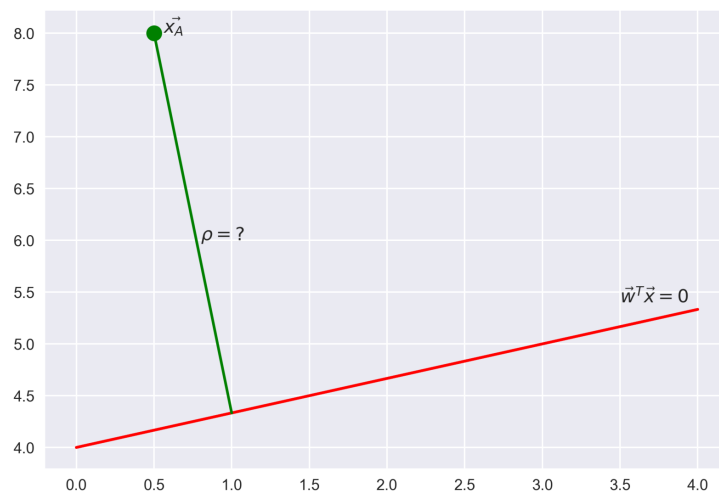
Выражение $M(\vec{x}_i) = y_i \vec{w}^T \vec{x}_i$ называется *отступом (margin)* классификации на объекте \vec{x}_i (не путать с зазором (тоже margin), про который чаще всего говорят в контексте SVM). Если он неотрицателен, модель не ошибается на объекте \vec{x}_i , если же отрицателен – значит, класс \vec{x}_i спрогнозирован неправильно.

Заметим, что отступ определен для объектов именно обучающей выборки, для которых известны реальные метки целевого класса y_i .

Чтобы понять, почему это мы сделали такие выводы, обратимся к геометрической интерпретации линейного классификатора. Подробнее это можно почитать в материалах Евгения Соколова.

Рекомендую решить почти классическую задачу из начального курса линейной алгебры: найти расстояние от точки с радиус-вектором плоскости, которая задается уравнением $\vec{w}^T \vec{x} = 0$.

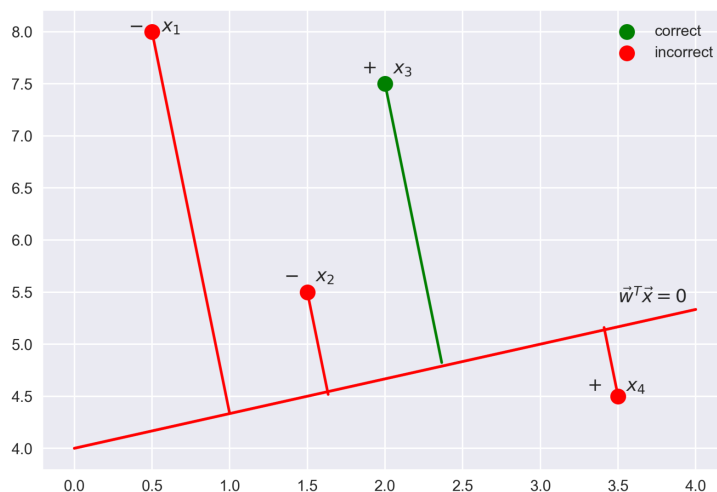
Ответ



Когда получим (или посмотрим) ответ, то поймем, что чем больше по модулю выражение $\vec{w}^T \vec{x}_i$, тем дальше точка \vec{x}_i находится от плоскости $\vec{w}^T \vec{x} = 0$.

Значит, выражение $M(\vec{x}_i) = y_i \vec{w}^T \vec{x}_i$ – это своего рода "уверенность" модели в классификации объекта \vec{x}_i :

- если отступ большой (по модулю) и положительный, это значит, что метка класса поставлена правильно, а объект находится далеко от разделяющей гиперплоскости (такой объект классифицируется уверенно). На рисунке – \mathbf{x}_3 .
- если отступ большой (по модулю) и отрицательный, значит метка класса поставлена неправильно, а объект находится далеко от разделяющей гиперплоскости (скорее всего такой объект – аномалия, например, его метка в обучающей выборке поставлена неправильно). На рисунке – \mathbf{x}_1 .
- если отступ малый (по модулю), то объект находится близко к разделяющей гиперплоскости, а знак отступа определяет, правильно объект классифицирован. На рисунке – \mathbf{x}_2 и \mathbf{x}_4 .



Теперь распишем правдоподобие выборки, а именно, вероятность наблюдать данный вектор \vec{y} у выборки \mathbf{X} . Делаем сильное предположение: объекты приходят независимо, из одного распределения (*i.i.d.*). Тогда

$$P(\vec{y} | \mathbf{X}, \vec{w}) = \prod_{i=1}^{\ell} P(y = y_i | \vec{x}_i, \vec{w}),$$

где ℓ – длина выборки \mathbf{X} (число строк).

Как водится, возьмем логарифм данного выражения (сумму оптимизировать намного проще, чем произведение):

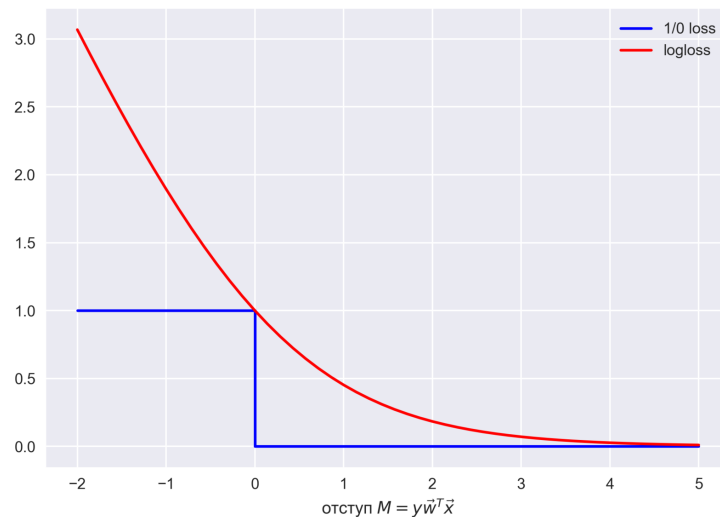
$$\begin{aligned} \log P(\vec{y} | \mathbf{X}, \vec{w}) &= \log \prod_{i=1}^{\ell} P(y = y_i | \vec{x}_i, \vec{w}) \\ &= \log \prod_{i=1}^{\ell} \sigma(y_i \vec{w}^T \vec{x}_i) \\ &= \sum_{i=1}^{\ell} \log \sigma(y_i \vec{w}^T \vec{x}_i) \\ &= \sum_{i=1}^{\ell} \log \frac{1}{1 + \exp^{-y_i \vec{w}^T \vec{x}_i}} \\ &= - \sum_{i=1}^{\ell} \log(1 + \exp^{-y_i \vec{w}^T \vec{x}_i}) \end{aligned}$$

То есть в данном случае принцип максимизации правдоподобия приводит к минимизации выражения

$$\mathcal{L}_{\log}(\mathbf{X}, \vec{y}, \vec{w}) = \sum_{i=1}^{\ell} \log(1 + \exp^{-y_i \vec{w}^T \vec{x}_i}).$$

Это *логистическая* функция потерь, просуммированная по всем объектам обучающей выборки.

Посмотрим на новую функцию как на функцию от отступа: $L(\mathbf{M}) = \log(1 + \exp^{-\mathbf{M}})$. Нарисуем ее график, а также график 1/0 функций потерь (*zero-one loss*), которая просто штрафует модель на 1 за ошибку на каждом объекте (отступ отрицательный): $L_{1/0}(\mathbf{M}) = [\mathbf{M} < 0]$



Картинка отражает общую идею, что в задаче классификации, не умея напрямую минимизировать число ошибок (по крайней мере, градиентными методами это не сделать — производная $1/0$ функций потерь в нуле обращается в бесконечность), мы минимизируем некоторую ее верхнюю оценку. В данном случае это логистическая функция потерь (где логарифм двоичный, но это не принципиально), справедливо

$$\begin{aligned}\mathcal{L}_{1/0}(X, \vec{y}, \vec{w}) &= \sum_{i=1}^{\ell} [M(\vec{x}_i) < 0] \\ &\leq \sum_{i=1}^{\ell} \log(1 + \exp^{-y_i \vec{w}^T \vec{x}_i}) \\ &= \mathcal{L}_{\log}(X, \vec{y}, \vec{w})\end{aligned}$$

где $\mathcal{L}_{1/0}(X, \vec{y}, \vec{w})$ — попросту число ошибок логистической регрессии с весами \vec{w} на выборке (X, \vec{y}) .

То есть уменьшая верхнюю оценку \mathcal{L}_{\log} на число ошибок классификации, мы таким образом надеемся уменьшить и само число ошибок

L_2 -регуляризация логистических потерь

L_2 -регуляризация логистической регрессии устроена почти так же, как и в случае с гребневой (Ridge регрессией). Вместо функционала $\mathcal{L}_{\log}(X, \vec{y}, \vec{w})$ минимизируется следующий:

$$J(X, \vec{y}, \vec{w}) = \mathcal{L}_{\log}(X, \vec{y}, \vec{w}) + \lambda |\vec{w}|^2$$

В случае логистической регрессии принято введение обратного коэффициента регуляризации $C = \frac{1}{\lambda}$. И тогда решением задачи будет

$$\hat{w} = \arg \min_{\vec{w}} J(X, \vec{y}, \vec{w}) = \arg \min_{\vec{w}} (C \sum_{i=1}^{\ell} \log(1 + \exp^{-y_i \vec{w}^T \vec{x}_i}) + |\vec{w}|^2)$$

Далее рассмотрим пример, позволяющий интуитивно понять один из смыслов регуляризации.

3. Наглядный пример регуляризации логистической регрессии

В 1 статье уже приводился пример того, как полиномиальные признаки позволяют линейным моделям строить нелинейные разделяющие поверхности. Покажем это в картинках.

Посмотрим, как регуляризация влияет на качество классификации на наборе данных по тестированию микрочипов из курса Andrew Ng машинному обучению.

Будем использовать логистическую регрессию с полиномиальными признаками и варьировать параметр регуляризации C .

Сначала посмотрим, как регуляризация влияет на разделяющую границу классификатора, интуитивно распознаем переобучение и недообучение.

Потом численно установим близкий к оптимальному параметр регуляризации с помощью кросс-валидации (cross-validation) и перебора сетке (GridSearch).

Подключение библиотек

Загружаем данные с помощью метода `read_csv` библиотеки `pandas`. В этом наборе данных для 118 микрочипов (объекты) указаны результаты двух тестов по контролю качества (два числовых признака) и сказано, пустили ли микрочип в производство. Признаки уже центрированы, из всех значений вычтены средние по столбцам. Таким образом, "среднему" микрочипу соответствуют нулевые значения результатов тестов.

Загрузка данных

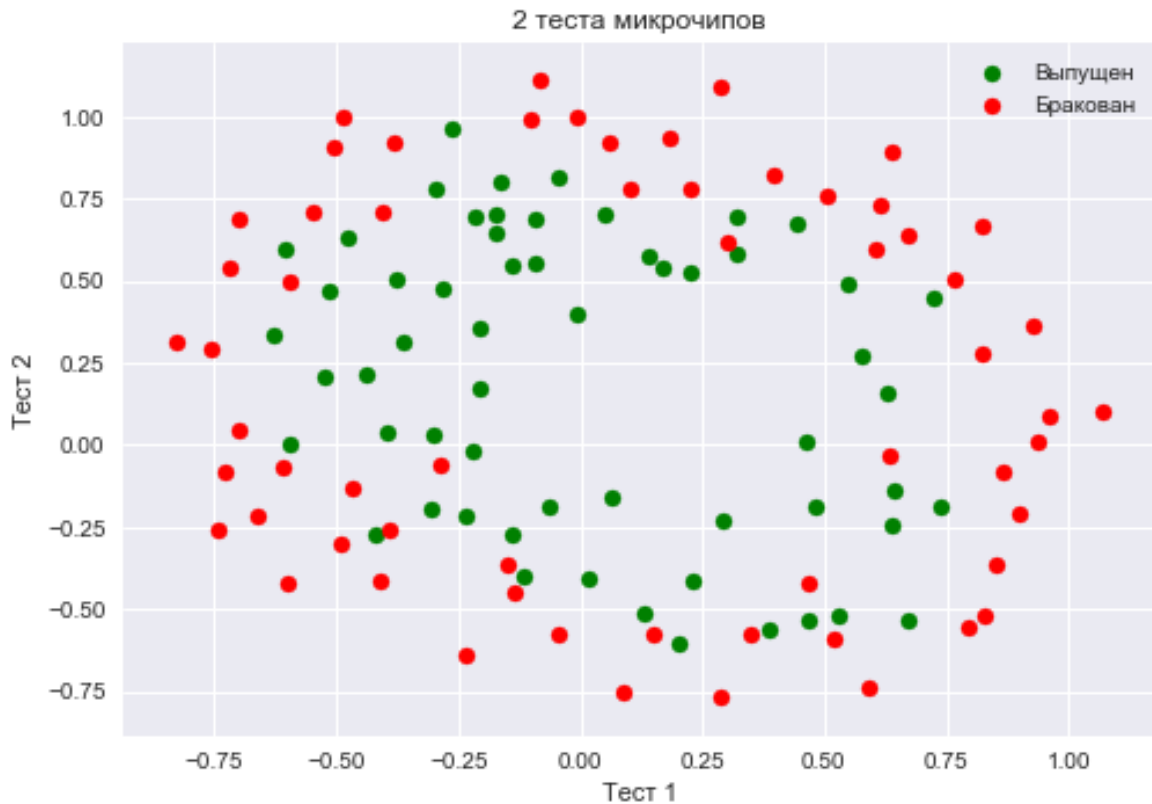
Посмотрим на первые и последние 5 строк.

	test1	test2	released
0	0.051267	0.69956	1
1	-0.092742	0.68494	1
2	-0.213710	0.69225	1
3	-0.375000	0.50219	1
4	-0.513250	0.46564	1

	test1	test2	released
113	-0.720620	0.538740	0
114	-0.593890	0.494880	0
115	-0.484450	0.999270	0
116	-0.006336	0.999270	0
117	0.632650	-0.030612	0

Сохраним обучающую выборку и метки целевого класса в отдельных массивах NumPy. Отобразим данные. Красный цвет соответствует бракованным чипам, зеленый – нормальным.

Код



Определяем функцию для отображения разделяющей кривой классификатора

[Код](#)

Полиномиальными признаками до степени d для двух переменных x_1 и x_2 мы называем следующие:

$$\{x_1^d, x_1^{d-1}x_2, \dots, x_2^d\} = \{x_1^i x_2^j\}_{i+j \leq d, i, j \in \mathbb{N}}$$

Например, для $d = 3$ это будут следующие признаки:

$$1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3$$

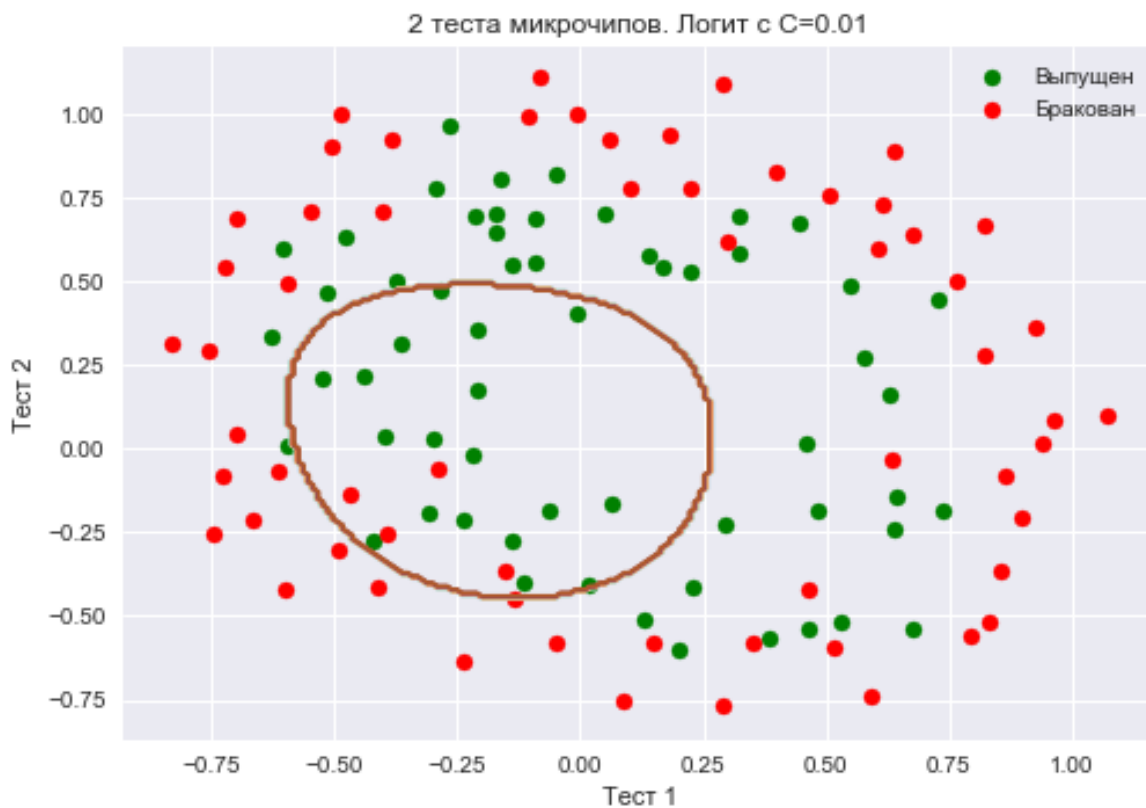
Нарисовав треугольник Пифагора, Вы сообразите, сколько таких признаков будет для $d = 4, 5, \dots$ и вообще для любого d .

Попросту говоря, таких признаков экспоненциально много, и строить, скажем, для 100 признаков полиномиальные степени 10 может оказаться затратно (а более того, и не нужно).

Создадим объект `sklearn`, который добавит в матрицу X полиномиальные признаки вплоть до степени 7 и обучим логистическую регрессию параметром регуляризации $C = 10^{-2}$. Изобразим разделяющую границу.

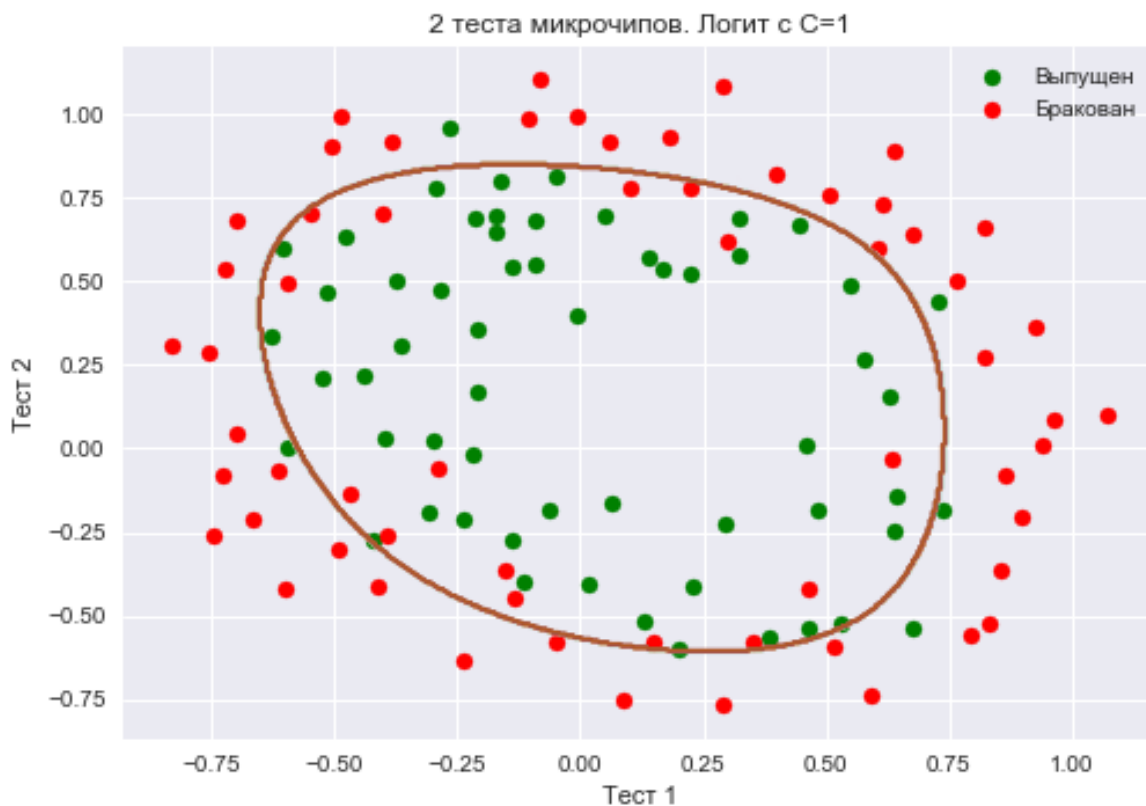
Также проверим долю правильных ответов классификатора на обучающей выборке. Видим, что регуляризация оказалась слишком сильной модель "недообучилась". Доля правильных ответов классификатора на обучающей выборке оказалась равной 0.627.

[Код](#)



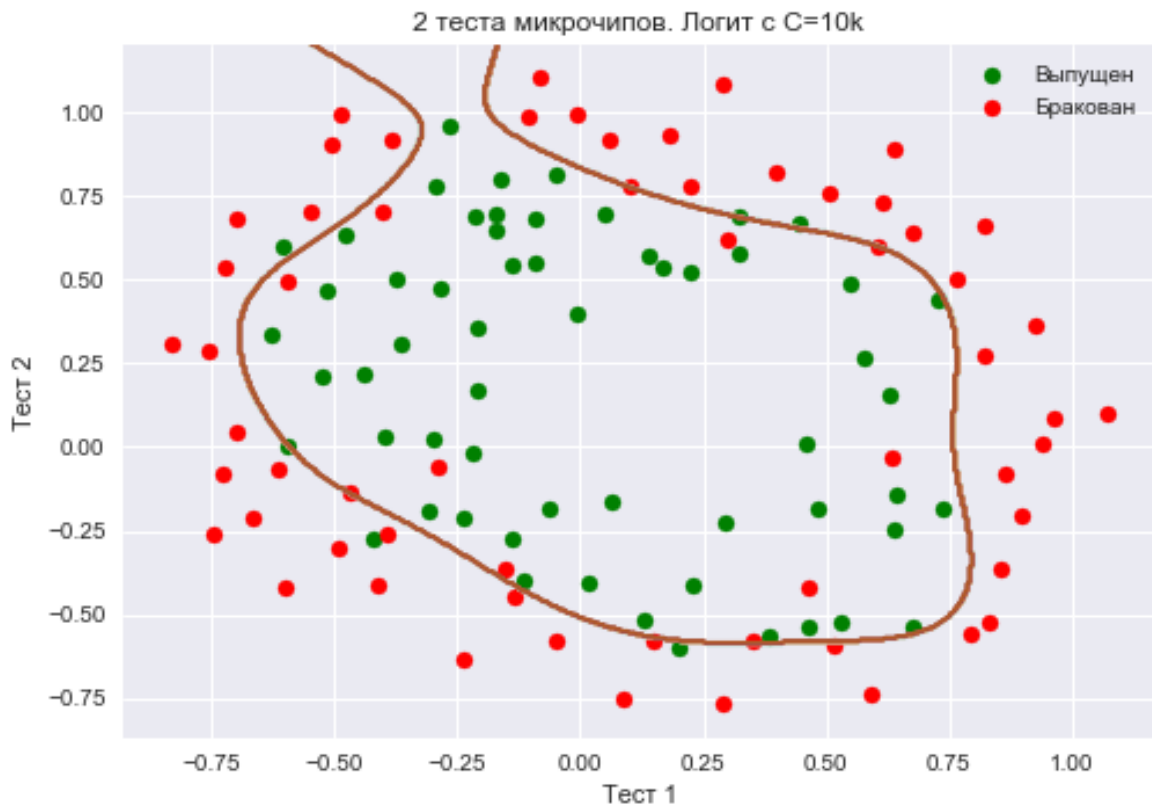
Увеличим C до 1. Тем самым мы *ослабляем* регуляризацию, теперь в решении значения весов логистической регрессии могут оказаться больше (по модулю), чем в прошлом случае. Теперь доля правильных ответов классификатора на обучающей выборке – 0.831.

[Код](#)



Еще увеличим C – до 10 тысяч. Теперь регуляризации явно недостаточно, и мы наблюдаем переобучение. Можно заметить, что в прошл случае (при $C=1$ и "гладкой" границе) доля правильных ответов модели на обучающей выборке не намного ниже, чем в 3 случае, зато на новой выборке, можно себе представить, 2 модель сработает намного лучше.

Доля правильных ответов классификатора на обучающей выборке – 0.873.

[Код](#)

Чтоб обсудить результаты, перепишем формулу для функционала, который оптимизируется в логистической регрессии, в таком виде:

$$J(X, y, w) = \mathcal{L} + \frac{1}{C} \|w\|^2,$$

где

- \mathcal{L} – логистическая функция потерь, просуммированная по всей выборке
- C – обратный коэффициент регуляризации (тот самый C в sklearn-реализации LogisticRegression)

Промежуточные выводы:

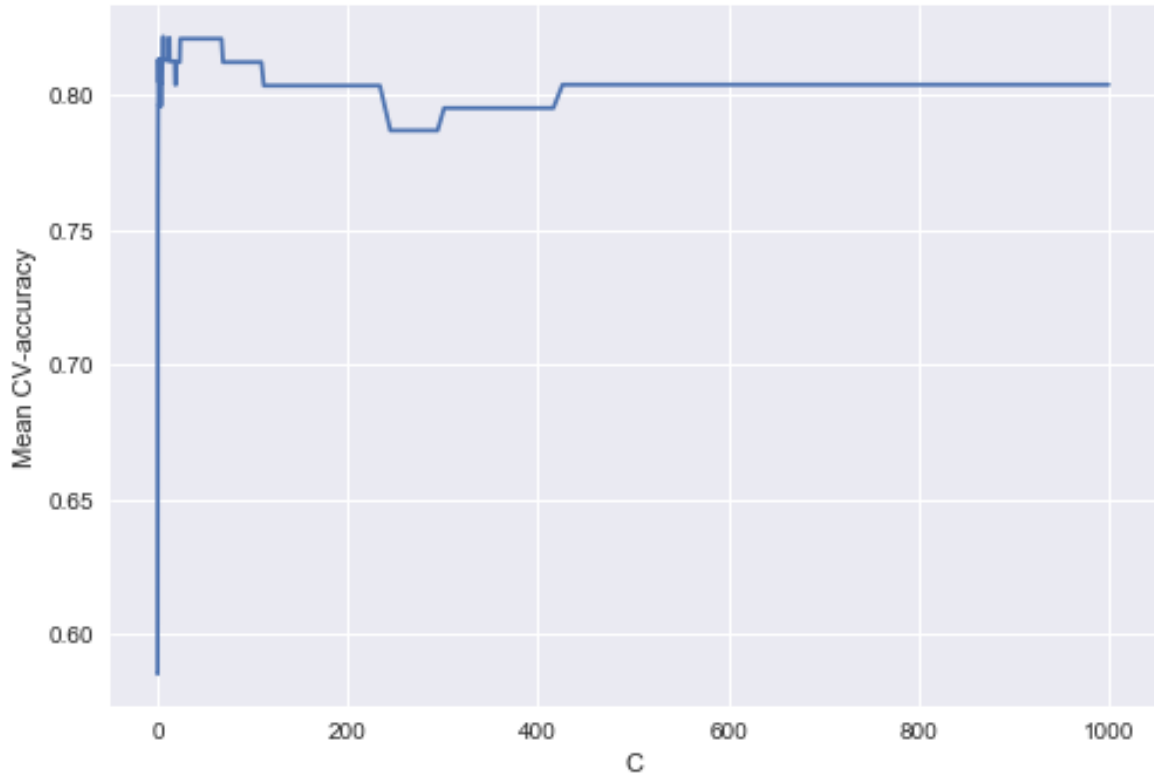
- чем больше параметр C , тем более сложные зависимости в данных может восстанавливать модель (интуитивно C соответствует "сложности" модели (model capacity))
- если регуляризация слишком сильная (малые значения C), то решением задачи минимизации логистической функции потерь может оказаться то, когда многие веса занулились или стали слишком малыми. Еще говорят, что модель недостаточно "штрафуется" за ош (то есть в функционале J "перевешивает" сумма квадратов весов, а ошибка \mathcal{L} может быть относительно большой). В таком случае модель окажется *недообученной* (1 случай)
- наоборот, если регуляризация слишком слабая (большие значения C), то решением задачи оптимизации может стать вектор w с большими по модулю компонентами. В таком случае большой вклад в оптимизируемый функционал J имеет \mathcal{L} и, вольно выражаясь, модель слишком "боится" ошибиться на объектах обучающей выборки, поэтому окажется *переобученной* (3 случай)
- то, какое значение C выбрать, сама логистическая регрессия "не поймет" (или еще говорят "не выучит"), то есть это не может быть определено решением оптимизационной задачи, которой является логистическая регрессия (в отличие от весов w). Так же точно, д решений не может "само понять", какое ограничение на глубину выбрать (за один процесс обучения). Поэтому C – это *гиперпараметр* модели, который настраивается на кросс-валидации, как и *max_depth* для дерева.

Настройка параметра регуляризации

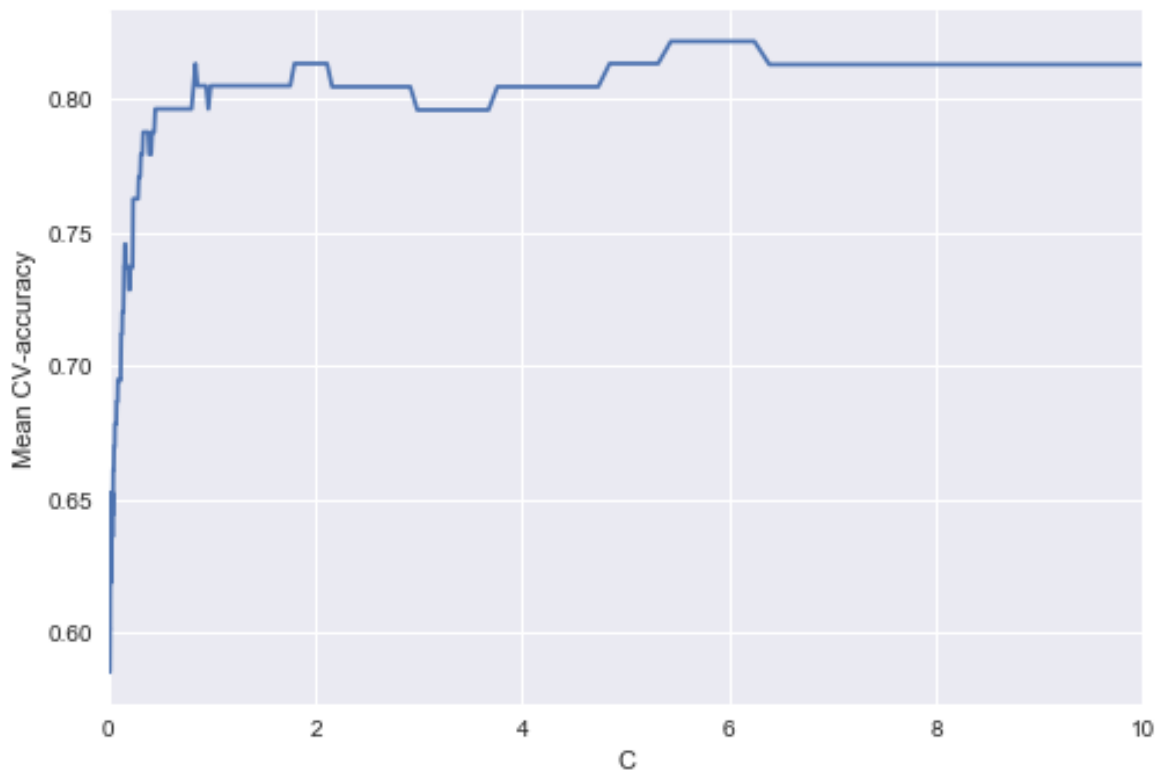
Теперь найдем оптимальное (в данном примере) значение параметра регуляризации C . Сделать это можно с помощью `LogisticRegression` перебора параметров по сетке с последующей кросс-валидацией. Этот класс создан специально для логистической регрессии (для него известны эффективные алгоритмы перебора параметров), для произвольной модели мы бы использовали `GridSearchCV`, `RandomizedSearchCV` или, например, специальные алгоритмы оптимизации гиперпараметров, реализованные в `hyperopt`.

[Код](#)

Посмотрим, как качество модели (доля правильных ответов на обучающей и валидационной выборках) меняется при изменении гиперпараметра C .



Выделим участок с "лучшими" значениями C .

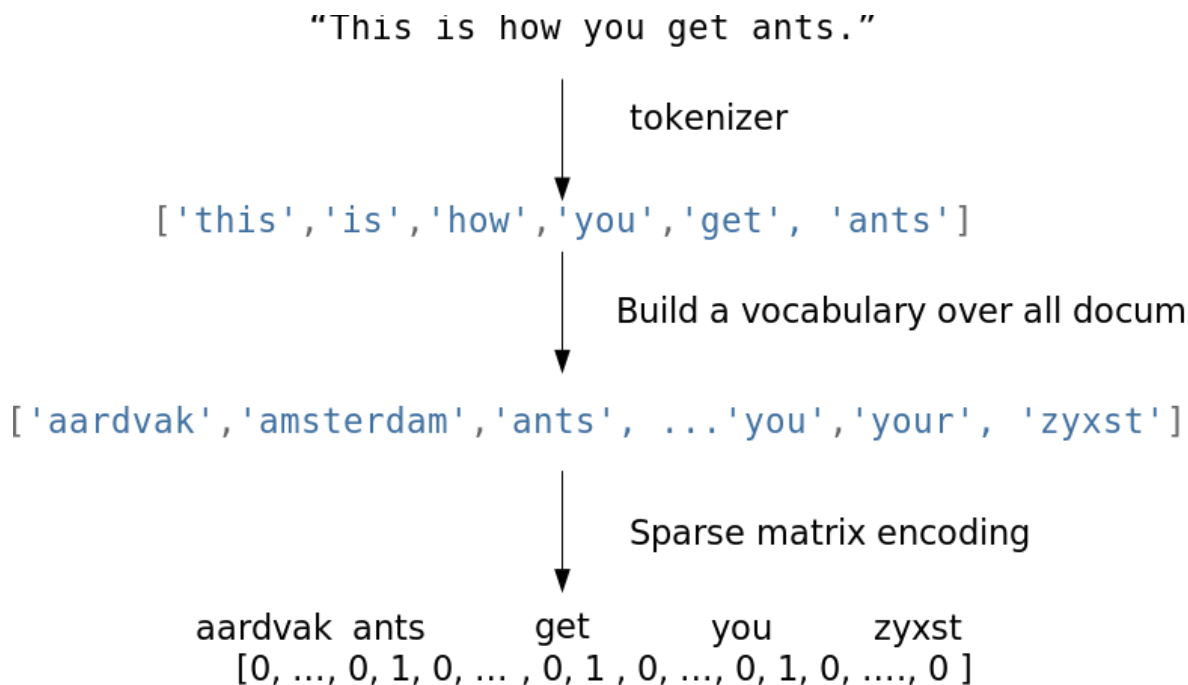


Как мы помним, такие кривые называются *валидационными*, раньше мы их строили вручную, но в sklearn для них их построения есть специальные методы, которые мы тоже сейчас будем использовать.

4. Где логистическая регрессия хороша и где не очень

Анализ отзывов IMDB к фильмам

Будем решать задачу бинарной классификации отзывов IMDB к фильмам. Имеется обучающая выборка с размеченными отзывами, по 12 отзывам известно, что они хорошие, еще про 12500 – что они плохие. Здесь уже не так просто сразу приступить к машинному обучению потому что готовой матрицы X нет – ее надо приготовить. Будем использовать самый простой подход – мешок слов ("Bag of words"). При таком подходе признаками отзыва будут индикаторы наличия в нем каждого слова из всего корпуса, где корпус – это множество всех отзывов. Идея иллюстрируется картинкой



Импорт библиотек и загрузка данных

Пример плохого отзыва:

'Words can't describe how bad this movie is. I can't explain it by writing only. You have too see it for yourself to get at grip of how horrible a m really can be. Not that I recommend you to do that. There are so many clich\xc3\xa9s, mistakes (and all other negative things you can imagine) here that will just make you cry. To start with the technical first, there are a LOT of mistakes regarding the airplane. I won't list them here, but j mention the coloring of the plane. They didn't even manage to show an airliner in the colors of a fictional airline, but instead used a 747 painter the original Boeing livery. Very bad. The plot is stupid and has been done many times before, only much, much better. There are so many ridiculous moments here that i lost count of it really early. Also, I was on the bad guys' side all the time in the movie, because the good guys w so stupid. "Executive Decision" should without a doubt be you're choice over this one, even the "Turbulence"-movies are better. In fact, every other movie in the world is better than this one.'

Пример хорошего отзыва:

'Everyone plays their part pretty well in this "little nice movie". Belushi gets the chance to live part of his life differently, but ends up realizing th what he had was going to be just as good or maybe even better. The movie shows us that we ought to take advantage of the opportunities we have, not the ones we do not or cannot have. If U can get this movie on video for around \$10, it\xc2\xbd be an investment!'

Простой подсчет слов

Составим словарь всех слов с помощью CountVectorizer. Всего в выборке 74849 уникальных слов. Если посмотреть на примеры получен "слов" (лучше их называть токенами), то можно увидеть, что многие важные этапы обработки текста мы тут пропустили (автоматическая обработка текстов – это могло бы быть темой отдельной серии статей).

Код

Закодируем предложения из текстов обучающей выборки индексами входящих слов. Используем разреженный формат. Преобразуем тестовую выборку.

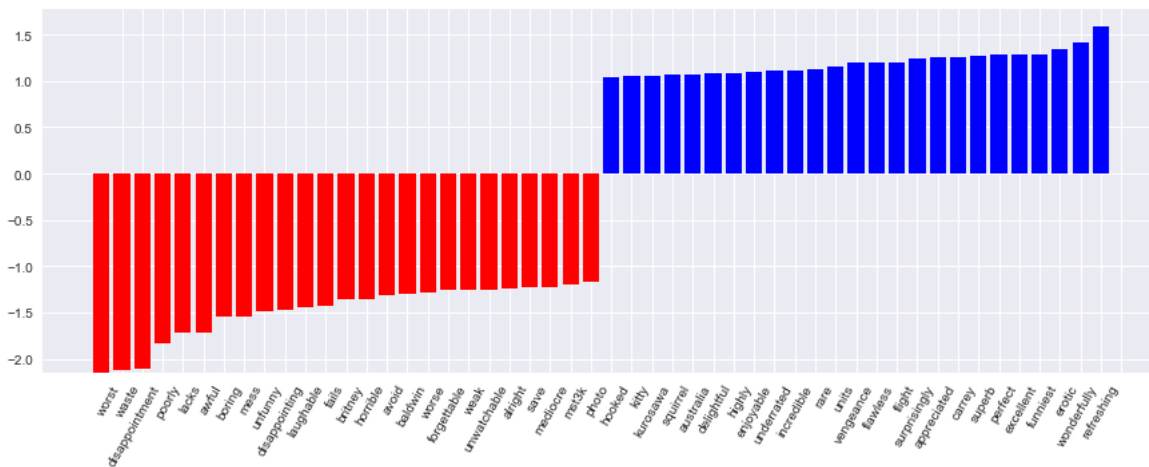
```
X_train = cv.transform(text_train)
X_test = cv.transform(text_test)
```

Обучим логистическую регрессию и посмотрим на доли правильных ответов на обучающей и тестовой выборках. Получается, на тестовой выборке мы правильно угадываем тональность примерно 86.7% отзывов.

Код

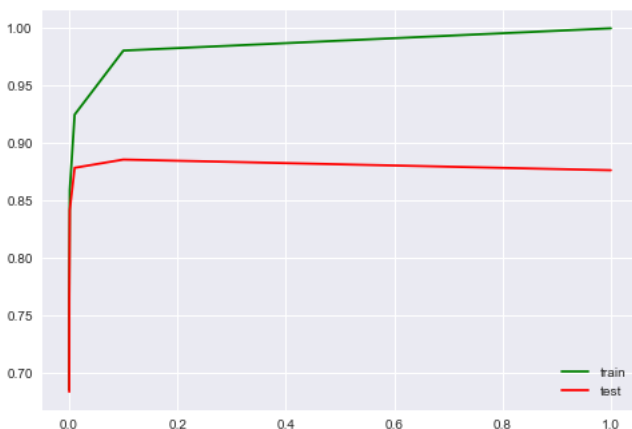
Коэффициенты модели можно красиво отобразить.

Код визуализации коэффициентов модели



Подберем коэффициент регуляризации для логистической регрессии. Используем sklearn.pipeline, поскольку CountVectorizer правильно применять только на тех данных, на которых в текущий момент обучается модель (чтоб не "подсматривать" в тестовую выборку и не считать по ней частоты вхождения слов). В данном случае pipeline задает последовательность действий: применить CountVectorizer, затем обучить логистическую регрессию. Так мы поднимаем долю правильных ответов до 88.5% на кросс-валидации и 87.9% – на отложенной выборке.

Код



Теперь то же самое, но со случайным лесом. Видим, что с логистической регрессией мы достигаем большей доли правильных ответов меньшими усилиями. Лес работает дольше, на отложенной выборке 85.5% правильных ответов.

Код для обучения случайного леса

XOR-проблема

Теперь рассмотрим пример, где линейные модели справляются хуже.

Линейные методы классификации строят все же очень простую разделяющую поверхность – гиперплоскость. Самый известный игрушечный пример, в котором классы нельзя без ошибок поделить гиперплоскостью (то есть прямой, если это 2D), получил имя "the XOR problem".

XOR – это "исключающее ИЛИ", булева функция со следующей таблицей истинности:

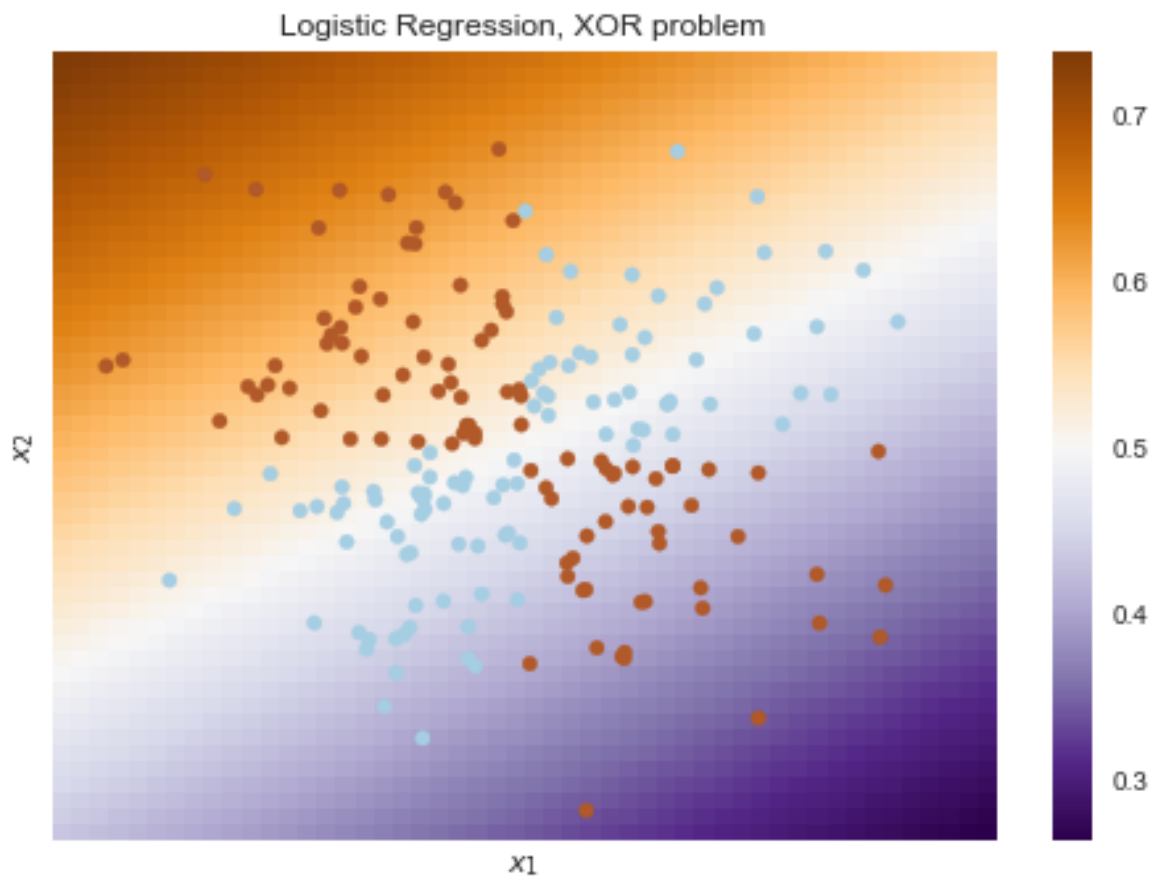
XOR	Input 1	
	0	1
Input 2	0	1
	0	1
	1	0

XOR дал имя простой задаче бинарной классификации, в которой классы представлены вытянутыми по диагоналям и пересекающимися облаками точек.

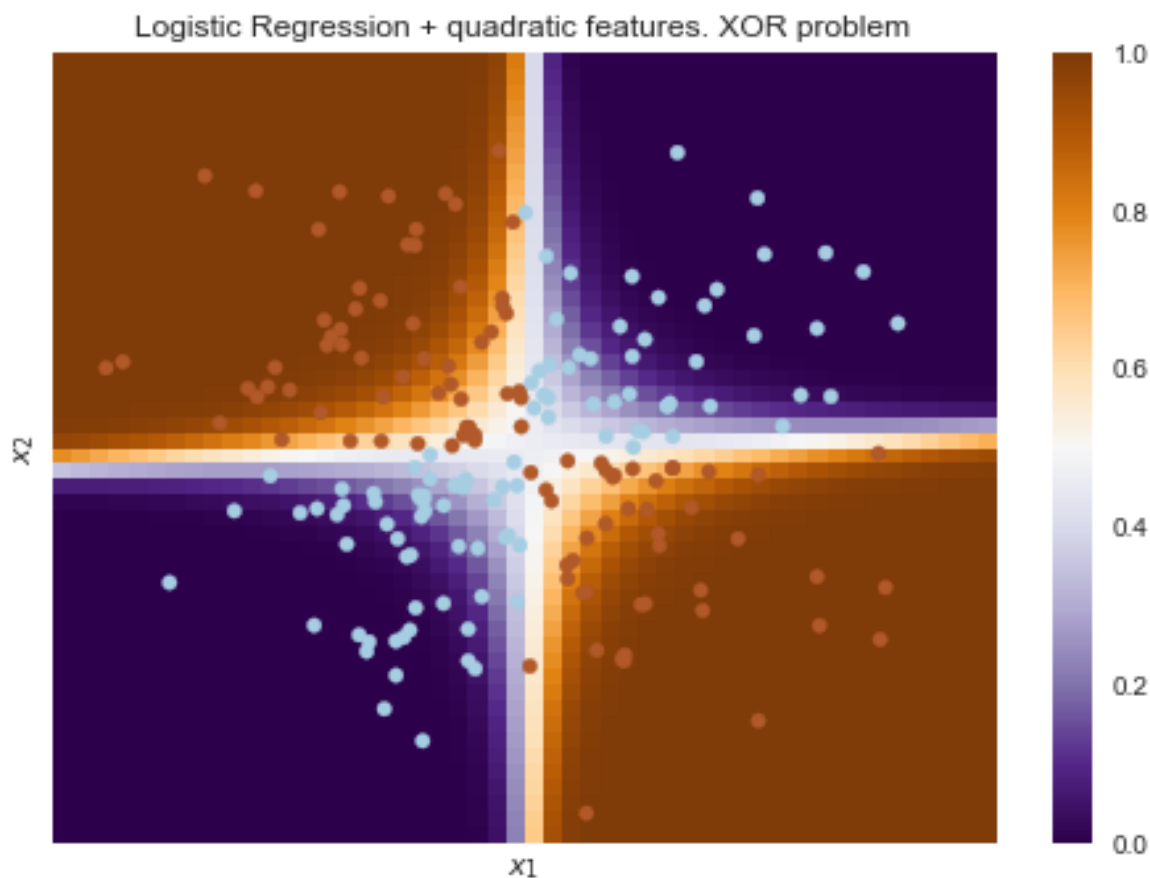
[Код, рисующий следующие 3 картинки](#)



Очевидно, нельзя провести прямую так, чтобы без ошибок отделить один класс от другого. Поэтому логистическая регрессия плохо справляется с такой задачей.



А вот если на вход подать полиномиальные признаки, в данном случае до 2 степени, то проблема решается.



Здесь логистическая регрессия все равно строила гиперплоскость, но в 6-мерном пространстве признаков $1, x_1, x_2, x_1^2, x_1x_2$ и x_2^2 . В проекции на исходное пространство признаков x_1, x_2 граница получилась нелинейной.

На практике полиномиальные признаки действительно помогают, но строить их явно – вычислительно неэффективно. Гораздо быстрее работает SVM с ядровым трюком. При таком подходе в пространстве высокой размерности считается только расстояние между объектами (задаваемое функцией-ядром), а явно плодить комбинаторно большое число признаков не приходится. Про это подробно можно почитать на курсе Евгения Соколова (математика уже серьезная).

5. Кривые валидации и обучения

Мы уже получили представление о проверке модели, кросс-валидации и регуляризации. Теперь рассмотрим главный вопрос:

Если качество модели нас не устраивает, что делать?

- Сделать модель сложнее или упростить?
- Добавить больше признаков?
- Или нам просто нужно больше данных для обучения?

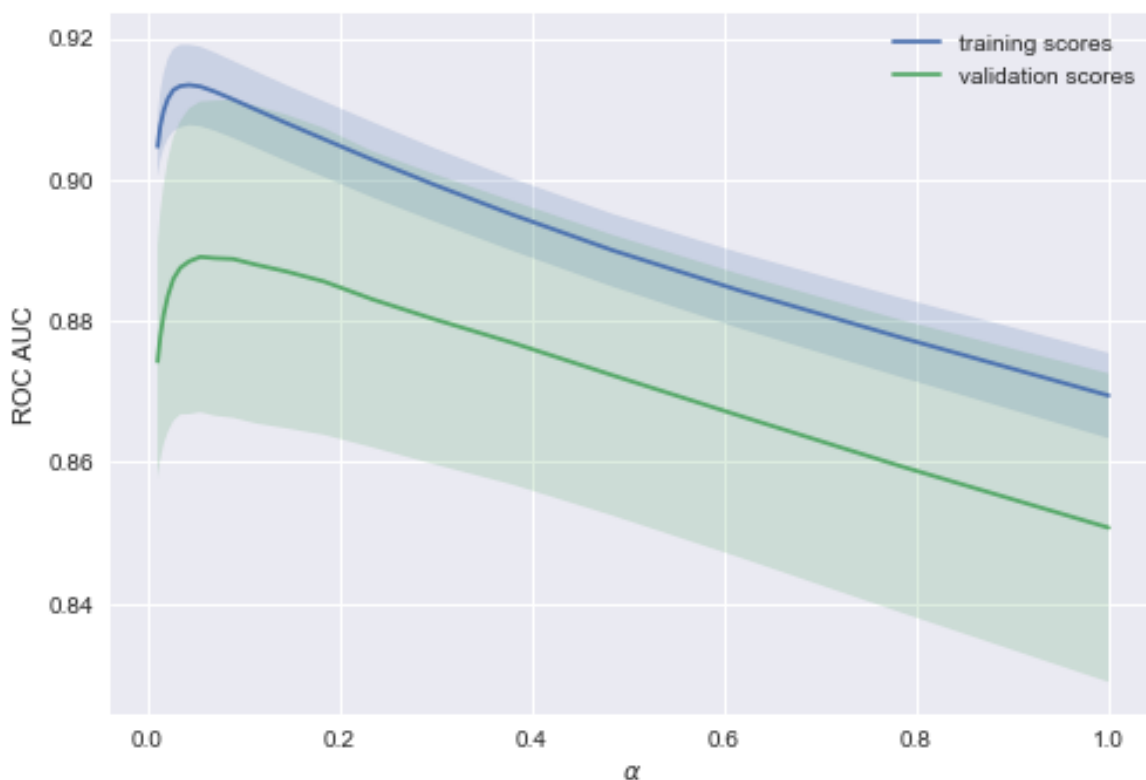
Ответы на данные вопросы не всегда лежат на поверхности. В частности, иногда использование более сложной модели приведет к ухудшению показателей. Либо добавление наблюдений не приведет к ощутимым изменениям. Способность принять правильное решение выбрать правильный способ улучшения модели, собственно говоря, и отличает хорошего специалиста от плохого.

Будем работать со знакомыми данными по оттоку клиентов телеком-оператора.

Импорт библиотек и чтение данных

Логистическую регрессию будем обучать стохастическим градиентным спуском. Пока объясним это тем, что так быстрее, но далее в программе у нас отдельная статья про это дело. Построим валидационные кривые, показывающие, как качество (ROC AUC) на обучающей и проверочной выборке меняется с изменением параметра регуляризации.

Код



Тенденция видна сразу, и она очень часто встречается.

- Для простых моделей тренировочная и валидационная ошибка находятся где-то рядом, и они велики. Это говорит о том, что модель **недообучилась**: то есть она не имеет достаточное кол-во параметров.

- Для сильно усложненных моделей тренировочная и валидационная ошибки значительно отличаются. Это можно объяснить **переобучением**: когда параметров слишком много либо не хватает регуляризации, алгоритм может "отвлекаться" на шум в данных и упускать основной тренд.

Сколько нужно данных?

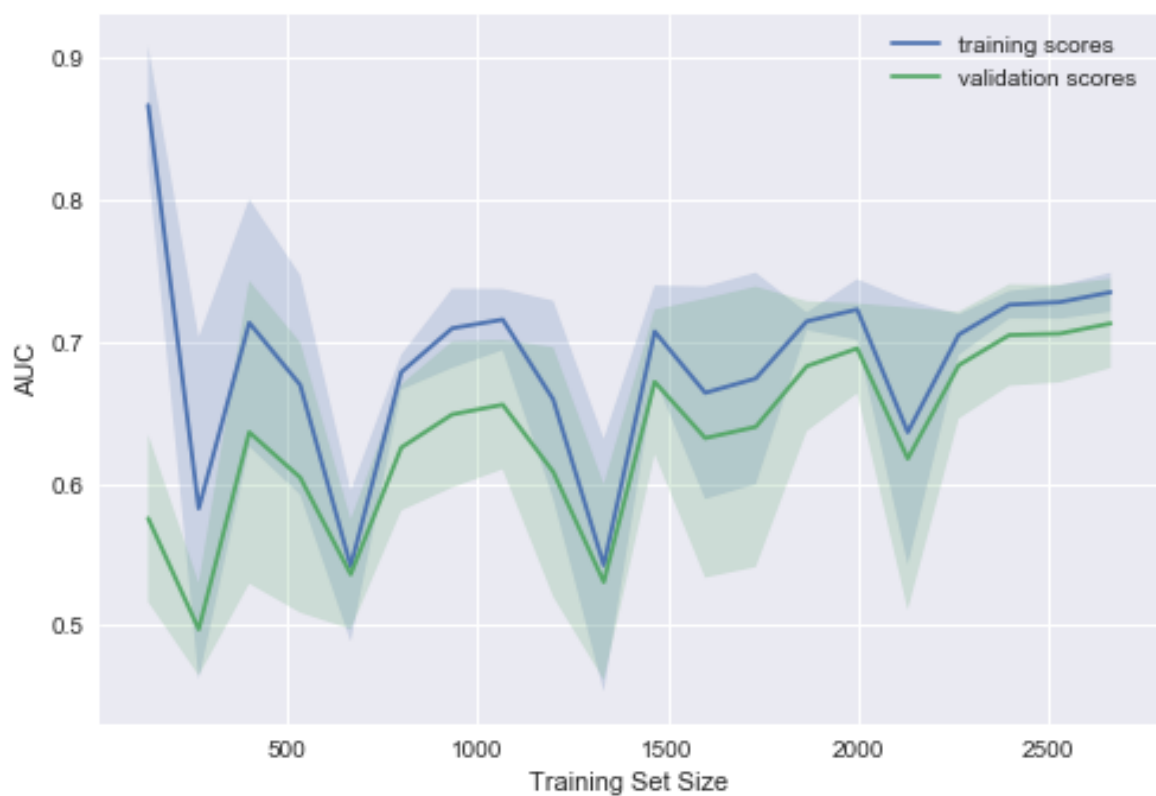
Известно, что чем больше данных использует модель, тем лучше. Но как нам понять в конкретной ситуации, помогут ли новые данные? Скажем, целесообразно ли нам потратить N\$ на труд ассессоров, чтобы увеличить выборку вдвое?

Поскольку новых данных пока может и не быть, разумно поварьировать размер имеющейся обучающей выборки и посмотреть, как качество решения задачи зависит от объема данных, на котором мы обучали модель. Так получаются **кривые обучения (learning curves)**.

Идея простая: мы отображаем ошибку как функцию от количества примеров, используемых для обучения. При этом параметры модели фиксируются заранее.

Давайте посмотрим, что мы получим для линейной модели. Коэффициент регуляризации выставим большим.

[Код](#)



Типичная ситуация: для небольшого объема данных ошибки на обучающей выборке и в процессе кросс-валидации довольно сильно отличаются, что указывает на переобучение. Для той же модели, но с большим объемом данных ошибки "сходятся", что указывает на недообучение.

Если добавить еще данные, ошибка на обучающей выборке не будет расти, но с другой стороны, ошибка на тестовых данных не будет уменьшаться.

Получается, ошибки "сошлись", и добавление новых данных не поможет. Собственно, это случай – самый интересный для бизнеса. Возьмем ситуацию, когда мы увеличиваем выборку в 10 раз. Но если не менять сложность модели, это может и не помочь. То есть стратегия "на один раз – дальше использую 10 раз" может и не работать.

Что будет, если изменить коэффициент регуляризации (уменьшить до 0.05)?

Видим хорошую тенденцию – кривые постепенно сходятся, и если дальше двигаться направо (добавлять в модель данные), можно еще повысить качество на валидации.



А если усложнить модель ещё больше ($\alpha = 10^{-4}$)?

Проявляется переобучение – AUC падает как на обучении, так и на валидации.



Строя подобные кривые, можно понять, в какую сторону двигаться, и как правильно настроить сложность модели на новых данных.

Выводы по кривым валидации и обучения

- Ошибка на обучающей выборке сама по себе ничего не говорит о качестве модели

- Кросс-валидационная ошибка показывает, насколько хорошо модель подстраивается под данные (имеющийся тренд в данных), сохранив при этом способность обобщения на новые данные
- **Валидационная кривая** представляет собой график, показывающий результат на тренировочной и валидационной выборке в зависимости от **сложности модели**:
- если две кривые располагаются близко, и обе ошибки велики, — это признак *недообучения*
- если две кривые далеко друг от друга, — это показатель *переобучения*
- **Кривая обучения** — это график, показывающий результаты на валидации и тренировочной подвыборке в зависимости от количества наблюдений:
- если кривые сошлись друг к другу, добавление новых данных не поможет — надо менять сложность модели
- если кривые еще не сошлись, добавление новых данных может улучшить результат.

6. Плюсы и минусы линейных моделей в задачах машинного обучения

Плюсы:

- Хорошо изучены
- Очень быстрые, могут работать на очень больших выборках
- Практически вне конкуренции, когда признаков очень много (от сотен тысяч и более), и они разреженные (хотя есть еще факторизационные машины)
- Коэффициенты перед признаками могут интерпретироваться (при условии что признаки масштабированы) — в линейной регрессии частные производные зависимой переменной от признаков, в логистической — как изменение шансов на отнесение к одному из классов \exp^{β_i} раз при изменении признака x_i на 1 ед., подробнее тут
- Логистическая регрессия выдает вероятности отнесения к разным классам (это очень ценится, например, в кредитном скоринге)
- Модель может строить и нелинейную границу, если на вход подать полиномиальные признаки

Минусы:

- Плохо работают в задачах, в которых зависимость ответов от признаков сложная, нелинейная
- На практике предположения теоремы Маркова-Гаусса почти никогда не выполняются, поэтому чаще линейные методы работают хуже, например, SVM и ансамбли (по качеству решения задачи классификации/регрессии)

7. Домашнее задание № 4

Актуальные домашние задания объявляются во время очередной сессии курса, следить можно в группе ВК и в репозитории курса.

В качестве закрепления изученного материала предлагаем следующее задание: разобраться с тем, как работает `TfidfVectorizer` и `DictVectorizer`, обучить и настроить модель линейной регрессии `Ridge` на данных о публикациях на Хабрахабре и воспроизвести бенчмарк соревнования. Проверить себя можно отправив ответы в веб-форме (там же найдете и решение).

8. Полезные ресурсы

- Перевод материала этой статьи на английский — Jupyter notebooks в репозитории курса
- Видеозаписи лекций по мотивам этой статьи: классификация, регрессия
- Основательный обзор классики машинного обучения и, конечно же, линейных моделей сделан в книге "Deep Learning" (I. Goodfellow, A. Courville, 2016);
- Реализация многих алгоритмов машинного обучения с нуля — репозиторий @rushter. Рекомендуем изучить реализацию логистической регрессии;
- Курс Евгения Соколова по машинному обучению (материалы на GitHub). Хорошая теория, нужна неплохая математическая подготовка
- Курс Дмитрия Ефимова на GitHub (англ.). Тоже очень качественные материалы.

Статья написана в соавторстве с @mephistopheies (Павлом Нестеровым). Он же — автор домашнего задания. Авторы домашнего задания первой сессии курса (февраль-май 2017)— @aiho (Ольга Дайховская) и @das19 (Юрий Исаков). Благодарю @bauchgefuehl (Анастасию Манохину) за редактирование.

Теги: machine learning, logistic regression, linear regression, ods, mlcourse.ai

↑ +50 ↓ 360 157k 41



Open Data Science 328,33

Крупнейшее русскоязычное Data Science сообщество



99,5

Карма

2,6

Рейтинг

210

Подписчики

16

Подписки

Yury Kashnitsky @yorko

Data Scientist, NLP practitioner

Сайт

Поддержать автора

Отправить деньги

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

13 марта 2017 в 14:03

Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей

↑ +61 181k 505 46

6 марта 2017 в 15:58

Открытый курс машинного обучения. Тема 2: Визуализация данных с Python

↑ +52 167k 474 45

20 февраля 2017 в 16:01

Базовые принципы машинного обучения на примере линейной регрессии

↑ +75 82,3k 475 22

Комментарии 41



mephistopheies 20 марта 2017 в 19:44

приветтики



Roman_Kh 20 марта 2017 в 22:47

После слов "Обратите внимание, что при максимизации функции по какому-то параметру можно выкинуть все члены, не зависящие от этого параметра:" в формуле ошибка в последней строке — должно быть **arg min**.



mephistopheies 20 марта 2017 в 22:53

fixed



Roman_Kh 20 марта 2017 в 22:54

...Поговорим немного о свойствах оценки, полученной линейной регрессией. В свете предыдущего пункта мы выяснили, что:

- ошибка распределена нормально с центром в нуле и некоторым разбросом: $\epsilon \sim N(0, \sigma^2)$;
- истинное значение целевой переменной тоже распределено нормально: $y \sim N(f(x), \sigma^2)$

Оба этих утверждения неверны. Линейная регрессия не требует нормальности ошибок, а только соблюдения условий Гаусса-Маркова. Нормальность ввели **вы** для того чтобы построить функционал максимального правдоподобия. А его вы строите, чтобы показать, что при нормальных ошибках МП равнозначно МНК.

Но сам МНК отлично работает и при ненормальных ошибках.

 **mephistopheies** 20 марта 2017 в 23:00    

так а разве говорится где то, что нормальность это требование теоремы Гаусса-Маркова? предыдущий пункт как раз про МЛ оценку, а пред-предыдущие это про МНК оценку

 **Roman_Kh** 20 марта 2017 в 23:12    

Цитирую дословно: "Поговорим немного о свойствах оценки, полученной линейной регрессией"
Но все что идет дальше это не свойства оценки, полученной линейной регрессией

>> В свете предыдущего пункта мы выяснили, что... ошибка распределена нормально с центром в нуле и некоторым разбросом: $\epsilon \sim N(0, \sigma^2)$;
Вы это не выяснили, а сами ввели. А если чуть строже, то сделали такое параметрическое предположение, что если ошибка распределена нормально, то тогда...

>> предыдущий пункт как раз про МЛ оценку, а пред-предыдущие это про МНК оценку
А в этом навалено и про МНК, и не про МНК, так что разобраться что о чем невозможно.

 **mephistopheies** 20 марта 2017 в 23:24    



формально, вы правы

 **antklen** 21 марта 2017 в 14:37  

ребят, а вам не кажется, что столько материала и домашки строго на одну неделю — это многовато для начинающих? Учитывая еще, что домашк далеко не полностью коррелирует с теорией и в ней нужно тоже много в чем разбираться?

 **Dsvolkov** 21 марта 2017 в 17:27    

Мне кажется нормально, как раз хорошо, динамично. Математика и базовый питон были в пререквизитах, а зная их тут не так уж и много, в домашках всё довольно подробно разжёвано. Каггловские же соревнования за доп. баллы на весь курс, не на неделю.

 **yorko** 21 марта 2017 в 17:30    

Ну там задания то — только признаки подготовить и логит/ридж запустить. Статья да, обширная, но на мой взгляд, за 3 присеста по 2 часа уж точно можно осилить. Все зависит от Вашей мотивации.

 **Dsvolkov** 21 марта 2017 в 17:49    

Хабр в рассылке мне говорит "26 минут на прочтение" и крутись как хочешь))

 **yorko** 21 марта 2017 в 18:18    

за 26 минут можешь понять, надо ли тебе это читать)

 **makkos** 21 марта 2017 в 18:17    

тяжело в учении — легко в бою

 **antklen** 22 марта 2017 в 15:11    

я-то сам не начинающий и курс не прохожу) просто взгляд со стороны.
в заданиях помимо самих вопросов нужно же еще понять и разобрать весь код, который там написан, для начинающего может быть не так просто/быстро

 **yorko**  22 марта 2017 в 15:19    

Согласен. Но никто не говорил, что будет просто. Это Data Science.
Ко мне в ВШЭ люди ходят учиться с 19 до 22 после работы в будни. И еще в субботу.

 **yorko**  22 марта 2017 в 15:24    

К тому же, начинающим не обязательно гнаться сразу за всем. Я согласен, что в начале многое сложно. В этой статье некоторые темы — у) для продолжающих. Мы учитываем пожелания широкой публики (перед стартом курса проводили обширный опрос), в том числе и тех, кто применяет разные модели, но не особо понимает, что за этим стоит.

**justm57** 21 марта 2017 в 17:27

Очередной раз спасибо за замечательную статью. В том моменте, когда мы сказали, что исходных признаков недостаточно и пытались сделать полиномиальные — это разве не ведёт к мультиколлинеарности? Или я чего-то не понимаю

**mephistopheies** 21 марта 2017 в 17:40

действительно это так, для демонстрации можете глянуть сюда <https://habrahabr.ru/company/ods/blog/322076/>

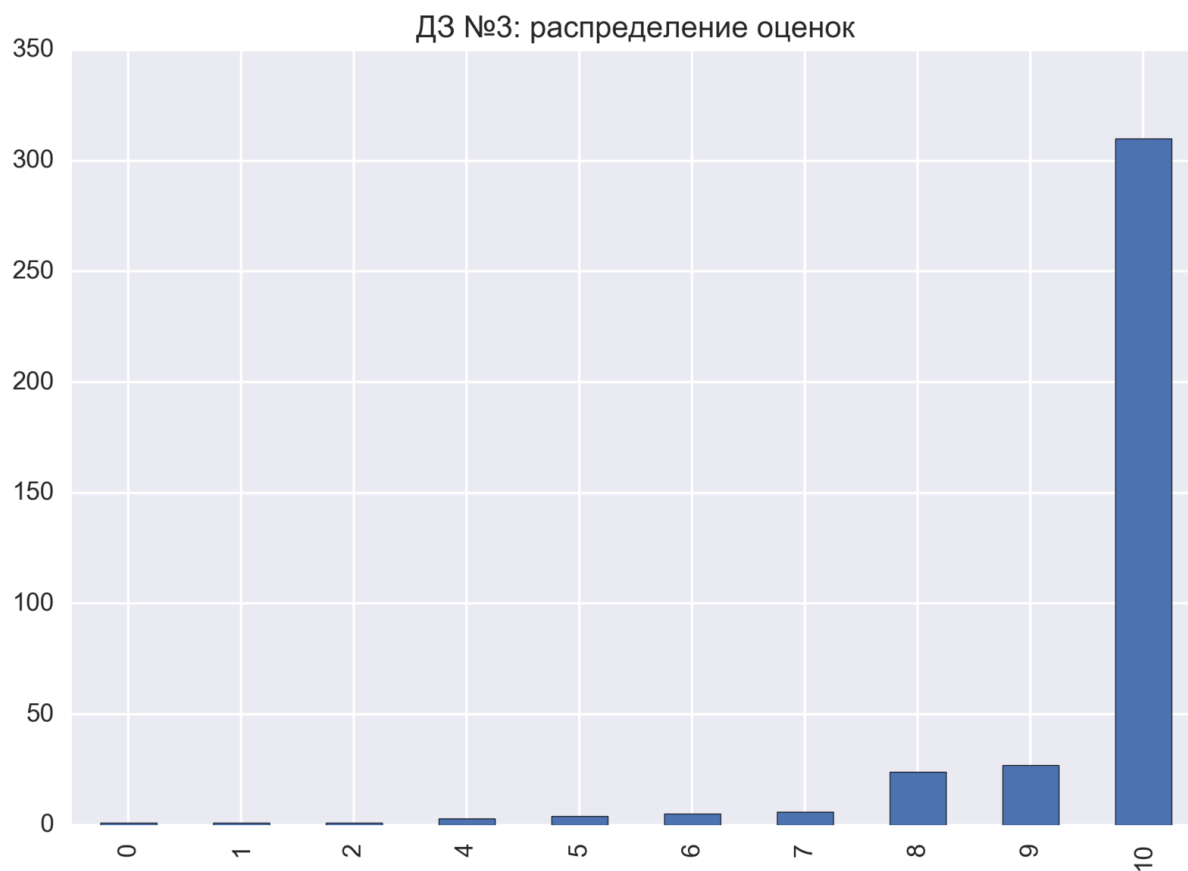
добавление полиномиальных действительно ведет к мультиколлинеарности, в линейных моделях это увеличивает сложность модели, что в итоге приводит к переобучению, но если остановить раньше, то все ок

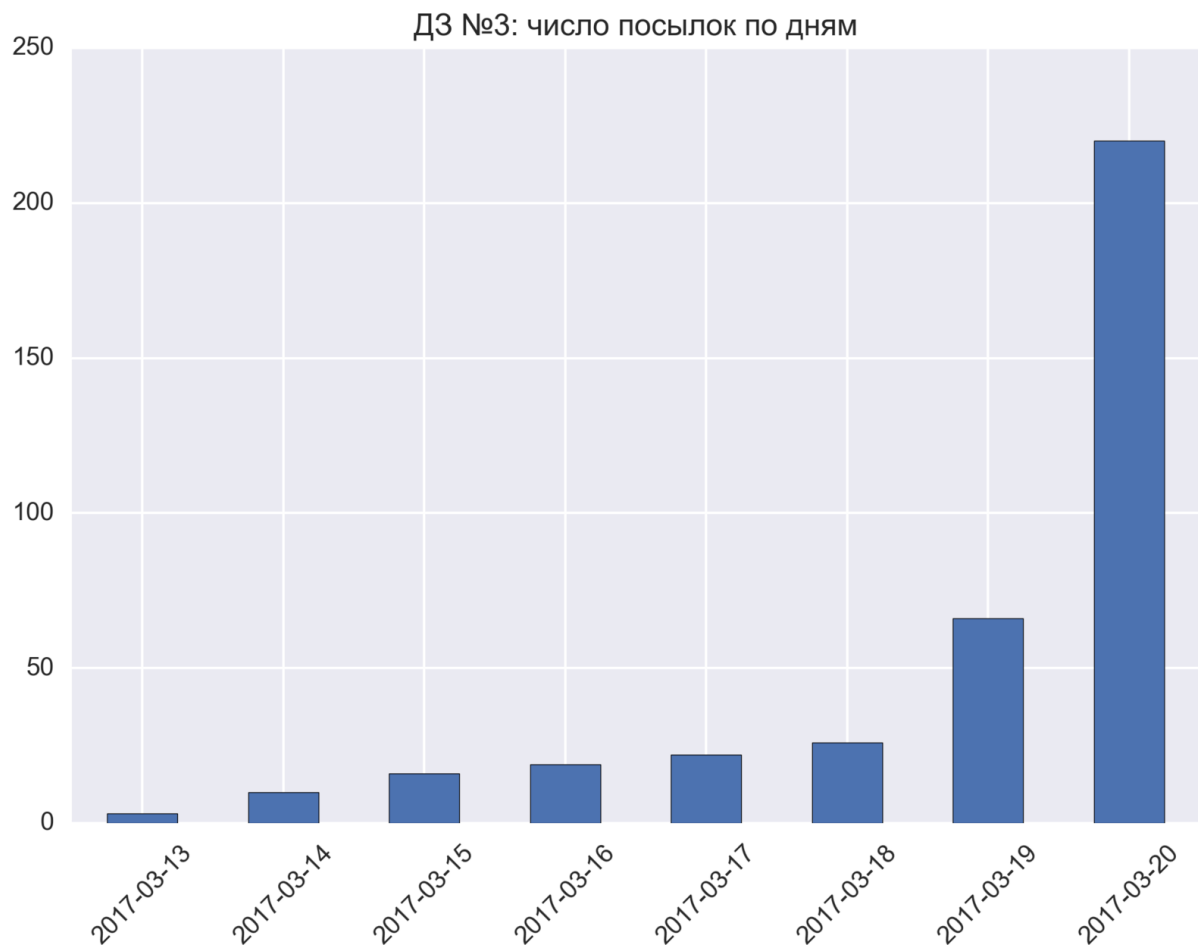
но вообще не всякое добавление полиномиальных фичей приведет к мультиколлинеарности, представьте такой случай

```
x = np.random.uniform(-1, 1, size=10000)
print np.corrcoef(x**2, x**6)[0, 1]
print np.corrcoef(x**2, x**7)[0, 1]
```

**yorko** 21 марта 2017 в 18:23



Минимальная статистика по 3 заданию. Оно оказалось уж больно простым, но ничего, есть 4-ое.





 **quiethorror** 23 марта 2017 в 00:07  

Можете пояснить такой вопрос: предположим в задаче бинарной классификации среди признаков есть такой индикатор, что если он равен 1, то гарантированно целевая переменная =1, но при этом срабатывает крайне редко (например, =1 на <1% от всех объектов). Это хороший признак для регрессии?

 **yorko** 23 марта 2017 в 00:09    

Это еще от баланса классов зависит. Вообще на кросс-валидации надо смотреть – он может как переобучать, так и, наоборот, улучшать модель.

 **ivanovserg990** 23 марта 2017 в 08:36  

А какие подходы существуют предсказывать значения с помощью логистической регрессии? То есть использовать логистическую регрессию для оценки стоимости квартиры, например. Понятно, что лог-регрессия используется для оценки вероятности принадлежности к классу. Как перейти от этого значения вероятности к интересующему значению y ?

 **yorko** 23 марта 2017 в 08:41    

Возможно, тут сказалась историческая путаница: логистическая регрессия — это модель классификации, а не регрессии. Так что если вы по «значениями» подразумеваете количественный признак, то логит — не для этого.

Если вопрос в том, как по предсказанной вероятности получить отнесение к одному из классов, то тут выбирается порог, начиная с которого прогнозируется класс 1. Это проиллюстрировано в статье картинкой с невозвратом кредита.

 **HKA** 24 марта 2017 в 15:45  

Иногда бывают ситуации, когда мы намеренно увеличиваем смещенность модели ради ее стабильности, т.е. ради уменьшения дисперсии модели $\text{Var}(f^*)$.

Можете привести *практический* пример такой ситуации?

 **yorko** 24 марта 2017 в 15:51    

Наиболее подробное изложение этого вопроса я видел в «Elements of Statistical Learning», если побыстрее – то в «Deep Learning» тоже есть с классического ML.



Roman_Kh 24 марта 2017 в 16:15



Допустим вам нужно предсказать стоимость куска адамантия. У вас есть данные о стоимости предыдущих сделок, а также размер, вес, качество и т.п.

Из этого можно было бы построить регрессию, однако между различными параметрами (в частности между размером и весом) есть связь. А значит, что регрессионная модель будет очень неустойчивой, т.е. небольшие изменения в исходных данных могут дать огромные изменения значениях рассчитанных коэффициентов.

Например, если у нас есть данные о 15 сделках, то при расчете модели на сделках с 1 по 10-ю, со 2 по 11-ю, с 3 по 12-ю и т.д., мы будем получать очень разные результаты. А они должны быть близкими или даже вообще одинаковыми.

Ведь правильные коэффициенты все равно существуют, просто мы их не знаем.

В нормальных условиях (без взаимозависимости в исходных данных) обычная регрессия дала бы лучшую оценку искомых коэффициентов — несмещенную и с наименьшей дисперсией.

Но в данном случае обычное не работает. Поэтому мы сознательно идем на добавление некоторого смещения (при идеальном раскладе — незначительного), но с резким уменьшением дисперсии.



yorko 24 марта 2017 в 16:46



Кстати, вам попадалось "простое" доказательство того, что разброс оценок у GLM ниже? Я какое-то видел, но четырехэтажное. А в классических книгах типа ESL этот аспект как-то игнорируют.



yorko 24 марта 2017 в 15:50



Речь как раз о том, что идет далее. На практике применяют Lasso и Ridge, которые добавляют смещения (bias) — оценки коэффициентов в модель будут дальше от истинных, нежели оценки МНК. Но зато и разброс (variance) этих оценок уменьшается по сравнению с МНК-оценками. А поскольку ошибка модели складывается из шума, разброса и квадрата смещения, сильно снизив разброс, можно уменьшить и ошибку, даже несмотря на смещение. Именно это и делают регуляризованные линейные модели Lasso и Ridge. Скажу даже больше: *по моему опыту* чистый МНК редко применяется, почти всегда нужна регуляризация.



iamtodor 15 мая 2017 в 15:44



ошибка: Подключение библиотек



yorko 15 мая 2017 в 16:53



спасибо!



enabokov 24 сентября 2017 в 06:52



Офигенная статья. Очень редкий экземпляр для Хабра.



yorko 27 сентября 2017 в 23:28



UPD: Видеозапись лекции по мотивам этой статьи в рамках нового запуска открытого курса (сентябрь-ноябрь 2017). Немного по теме статьи + разбор бенчмарков соревнования.



3draven 4 октября 2017 в 21:58



Я с ваших статей и лекций просто тащусь. Правда. Очень редко когда человек может в трех словах объяснить сложные вещи. Пишите учебник, срочно!



yorko 5 октября 2017 в 19:49



Спасибо! И за совет тоже спасибо! :trollface:

Впрочем, можно и без тролфейса задаться вопросом, зачем нужен учебник, когда есть статьи на хабре и видео, и как будет выглядеть учебник будущего.

Пададжи... учебник писать — это все мемасики выкинуть? No way



3draven 5 октября 2017 в 19:53



Тут все просто. Когда человек пишет учебник, он систематически излагает то, что позволит построить стабильную базу для развития. Помогая этому присылая это личным опытом.

Статьи на хабре это «нахватал».




О учебниках будущего я тоже когда-то думал, когда в образовании работал :) Так вот, на практике — пиши учебник. В каждой области есть всего пара книг «маст хэв». Их пишут люди, способные на пальцах пятикласснику объяснить как работает сложнейшая штука и чем же они занимаются на работе.

Такие книжки знает любой спец в своей области и никакие статьи такие книжки не заменят.




3draven 5 октября 2017 в 19:53






 **yorko** 18 декабря 2017 в 09:37  




Новый запуск курса – 5 февраля 2018 г. Регистрация не требуется, но чтобы мы о вас знали, заполните [форму](#). Главные новости будут в группе ВКонтакте, а жизнь во время курса будет теплиться в Slack OpenDataScience ([вступить](#)) в канале #mlcourse_open.

 **yorko** 13 мая 2018 в 22:46  






Новый запуск – 1 октября 2018 г., на английском. Подробности – [тут](#).

 **yorko** 13 июня 2018 в 17:25  

Теперь курс можно проходить и самостоятельно – появились демо-версии заданий с решениями. Они описываются в конце каждой статьи, но е общий список. Решения доступны после отправки соотв. веб-формы.

 **quiethorror** 12 февраля 2019 в 14:04  

Не могу понять, почему в пункте «Метод максимального правдоподобия» мы в $P(y_i | x, w)$ подставляем плотность вероятности. Смущает, что распределение — не дискретное и вероятность конкретного значения $y_i = 0$

 **yorko** 15 февраля 2019 в 00:43    

Функция правдоподобия как раз через плотность вероятности определяется.

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

- Сутки
- Неделя
- Месяц

Зубы мудрости: удалить нельзя оставить

 +109

 36,3k

 129

 174

Осторожно доктор

 +123

 17,4k

 68

 71

Ответ психиатра на статью «Болен-здоров»

 +101

 28k

 160

 157

«Яндекс» заявил о решении проблемы с ключами шифрования для ФСБ, а Павел Дуров зовет разработчиков компании к себе

 +22

 24,2k

 6

 68



Автоматика для дома своими руками

 +39

 10,8k

 150

 26

Аккаунт	Разделы	Информация	Услуги	Приложения
Войти	Публикации	Правила	Реклама	 
Регистрация	Новости	Помощь	Тарифы	
	Хабы	Документация	Контент	
	Компании	Соглашение	Семинары	

[Пользователи](#)[Конфиденциальность](#)[Песочница](#)

Если нашли опечатку в посте, выделите ее и нажмите Ctrl+Enter, чтобы сообщить автору.



© 2006 – 2019 «ТМ»

[Настройка языка](#)[О сайте](#)[Служба поддержки](#)[Мобильная версия](#)