Лекция 2
Линейная регрессия

Е. А. Соколов
ФКН ВШЭ

13 сентября 2016 г.

1 Линейные модели

На предыдущей лекции мы уже упоминали линейные регрессионные модели. Такие модели сводятся к суммированию значений признаков с некоторыми весами:

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x_j. \tag{1.1}$$

Параметрами модели являются *веса* или *коэффициенты* w_j. Вес w_0 также называется свободным коэффициентом или *сдвигом* (bias). Заметим, что сумма в формуле (1.1) является скалярным произведением вектора признаков на вектор весов. Воспользуемся этим и запишем линейную модель в более компактном виде:

$$a(x) = w_0 + \langle w, x \rangle, \tag{1.2}$$

где $w = (w_1, \ldots, w_d)$ — вектор весов.

Достаточно часто используется следующий приём, позволяющий упростить запись ещё сильнее. Добавим к признаковому описанию каждого объекта $(d+1)$-й признак, равный единице. Вес при этом признаке как раз будет иметь смысл свободного коэффициента, и необходимость в слагаемом w_0 отпадёт:

$$a(x) = \langle w, x \rangle.$$

Тем не менее, при такой форме следует соблюдать осторожность и помнить о наличии в выборке специального признака. Например, мы столкнёмся со сложностями, связанными с этим, когда будем говорить о регуляризации.

За счёт простой формы линейные модели достаточно быстро и легко обучаются, и поэтому популярны при работе с большими объёмами данных. Также у них мало параметров, благодаря чему удаётся контролировать риск переобучения и использовать их для работы с зашумлёнными данными и с небольшими выборками.

2 Измерение ошибки в задачах регрессии

Чтобы обучать регрессионные модели, нужно определиться, как именно измеряется качество предсказаний. Будем обозначать через y значение целевой переменной, через a — прогноз модели. Рассмотрим несколько способов оценить отклонение $L(y, a)$ прогноза от истинного ответа.

MSE и R^2. Основной способ измерить отклонение — посчитать квадрат разности:

$$L(y, a) = (a - y)^2$$

Благодаря своей дифференцируемости эта функция наиболее часто используется в задачах регрессии. Основанный на ней функционал называется среднеквадратичным отклонением (mean squared error, MSE):

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

Отметим, что величина среднеквадратичного отклонения плохо интерпретируется, поскольку не сохраняет единицы измерения — так, если мы предсказываем цену в рублях, то MSE будет измеряться в квадратах рублей. Чтобы избежать этого, используют корень из среднеквадратичной ошибки (root mean squared error, RMSE):

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

Среднеквадратичная ошибка подходит для сравнения двух моделей или для контроля качества во время обучения, но не позволяет сделать выводы том, насколько хорошо данная модель решает задачу. Например, MSE = 10 является очень плохим показателем, если целевая переменная принимает значения от 0 до 1, и очень хорошим, если целевая переменная лежит в интервале $(10000, 100000)$. В таких ситуациях вместо среднеквадратичной ошибки полезно использовать *коэффициент детерминации* (или коэффициент R^2):

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

где $\bar{y} = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$ — среднее значение целевой переменной. Коэффициент детерминации измеряет долю дисперсии, объяснённую моделью, в общей дисперсии целевой переменной. Фактически, данная мера качества — это нормированная среднеквадратичная ошибка. Если она близка к единице, то модель хорошо объясняет данные, если же она близка к нулю, то прогнозы сопоставимы по качеству с константным предсказанием.

MAE. Заменим квадрат отклонения на модуль:

$$L(y, a) = |a - y|$$

Соответствующий функционал называется средним абсолютным отклонением (mean absolute error, MAE):

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

Модуль отклонения не является дифференцируемым, но при этом менее чувствителен к выбросам. Квадрат отклонения, по сути, делает особый акцент на объектах с сильной ошибкой, и метод обучения будет в первую очередь стараться уменьшить отклонения на таких объектах. Если же эти объекты являются выбросами (то есть значение целевой переменной на них либо ошибочно, либо относится к другому распределению и должно быть проигнорировано), то такая расстановка акцентов приведёт к плохому качеству модели. Модуль отклонения в этом смысле гораздо более терпим к сильным ошибкам.

Приведём ещё одно объяснение того, почему модуль отклонения устойчив к выбросам, на простом примере. Допустим, все ℓ объектов выборки имеют одинаковые признаковые описания, но разные значения целевой переменной $y_1, \ldots, y_\ell$. В этом случае модель должна на всех этих объектах выдать один и тот же ответ. Если мы выбрали MSE в качестве функционала ошибки, то получаем следующую задачу:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} (a - y_i)^2 \to \min_a$$

Легко показать, что минимум достигается на среднем значении всех ответов:

$$a^*_{\text{MSE}} = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i.$$

Если один из ответов на порядки отличается от всех остальных (то есть является выбросом), то среднее будет существенно отклоняться в его сторону.

Рассмотрим теперь ту же ситуацию, но с функционалом MAE:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} |a - y_i| \to \min_a$$

Теперь решением будет медиана ответов:

$$a^*_{\text{MAE}} = \text{median}\{y_i\}_{i=1}^{\ell}.$$

Небольшое количество выбросов никак не повлияет на медиану — она существенно более устойчива к величинам, выбивающимся из общего распределения.

MSLE. Перейдём теперь к логарифмам ответов и прогнозов:

$$L(y, a) = (\log(a + 1) - \log(y + 1))^2$$

Соответствующий функционал называется среднеквадратичной логарифмической ошибкой (mean squared logarithmic error, MSLE). Данная метрика подходит для задач с неотрицательной целевой переменной. За счёт логарифмирования ответов и прогнозов мы скорее штрафуем за отклонения в порядке величин, чем за отклонения в их значениях. Также следует помнить, что логарифм не является симметричной функцией, и поэтому данная функция потерь штрафует заниженные прогнозы сильнее, чем завышенные.

MAPE и SMAPE. В задачах прогнозирования обычно измеряется относительная ошибка:

$$L(y, a) = \left| \frac{y - a}{y} \right|$$

Соответствующий функционал называется средней абсолютной процентной ошибкой (mean absolute percentage error, MAPE). Данный функционал часто используется в задачах прогнозирования. Также используется его симметричная модификация (symmetric mean absolute percentage error, SMAPE):

$$L(y, a) = \frac{|y - a|}{(|y| + |a|)/2}$$

3 Обучение линейной регрессии

Чаще всего линейная регрессия обучается с использованием среднеквадратичной ошибки. В этом случае получаем задачу оптимизации (считаем, что среди признаков есть константный, и поэтому свободный коэффициент не нужен):

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

Эту задачу можно переписать в матричном виде. Если X — матрица «объекты-признаки», y — вектор ответов, w — вектор параметров, то приходим к виду

$$\frac{1}{\ell} \|Xw - y\|^2 \to \min_w, \tag{3.1}$$

где используется обычная L_2-норма. Если продифференцировать данный функционал по вектору w, приравнять к нулю и решить уравнение, то получим явную формулу для решения (подробный вывод формулы можно найти в материалах семинаров):

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

Безусловно, наличие явной формулы для оптимального вектора весов — это большое преимущество линейной регрессии с квадратичным функционалом. Но данная формула не всегда применима по ряду причин:

- Обращение матрицы — сложная операция с кубической сложностью от количества признаков. Если в выборке тысячи признаков, то вычисления могут стать слишком трудоёмкими. Решить эту проблему можно путём использования численных методов оптимизации.

- Матрица $X^T X$ может быть вырожденной или плохо обусловленной. В этом случае обращение либо невозможно, либо может привести к неустойчивым результатам. Проблема решается с помощью регуляризации, речь о которой пойдёт ниже.

Следует понимать, что аналитические формулы для решения довольно редки в машинном обучении. Если мы заменим MSE на другой функционал, то найти такую формулу, скорее всего, не получится. Желательно разработать общий подход, в рамках которого можно обучать модель для широкого класса функционалов. Такой подход действительно есть для дифференцируемых функций — обсудим его подробнее.

4 Градиентный спуск и оценивание градиента

Оптимизационные задачи (3.1) можно решать итерационно с помощью градиентных методов (или же методов, использующих как градиент, так и информацию о производных более высокого порядка).

§4.1 Градиент и его свойства

Градиентом функции $f : \mathbb{R}^d \to \mathbb{R}$ называется вектор его частных производных:

$$\nabla f(x_1, \ldots, x_d) = \left(\frac{\partial f}{\partial x_j}\right)_{j=1}^{d}.$$

Градиент является направлением наискорейшего роста функции, а антиградиент (т.е. $-\nabla f$) — направлением наискорейшего убывания. Это ключевое свойство градиента, обосновывающее его использование в методах оптимизации.

Докажем данное утверждение. Пусть $v \in \mathbb{R}^d$ — произвольный вектор, лежащий на единичной сфере: $\|v\| = 1$. Пусть $x_0 \in \mathbb{R}^d$ — фиксированная точка пространства. Скорость роста функции в точке x_0 вдоль вектора v характеризуется производной по направлению $\frac{\partial f}{\partial v}$:

$$\frac{\partial f}{\partial v} = \frac{d}{dt} f(x_{0,1} + tv_1, \ldots, x_{0,d} + tv_d)|_{t=0}.$$

Из курса математического анализа известно, что данную производную сложной функции можно переписать следующим образом:

$$\frac{\partial f}{\partial v} = \sum_{j=1}^{d} \frac{\partial f}{\partial x_j} \frac{d}{dt}(x_{0,j} + tv_j) = \sum_{j=1}^{d} \frac{\partial f}{\partial x_j} v_j = \langle \nabla f, v \rangle.$$

Распишем скалярное произведение:

$$\langle \nabla f, v \rangle = \|\nabla f\| \|v\| \cos\varphi = \|\nabla f\| \cos\varphi,$$

где φ — угол между градиентом и вектором v. Таким образом, производная по направлению будет максимальной, если угол между градиентом и направлением равен нулю, и минимальной, если угол равен 180 градусам. Иными словами, производная по направлению максимальна вдоль градиента и минимальна вдоль антиградиента.

Покажем теперь, что градиент ортогонален линиям уровня. Пусть x_0 — некоторая точка, $S(x_0) = \{x \in \mathbb{R}^d \,|\, f(x) = f(x_0)\}$ — соответствующая линия уровня. Разложим функцию в ряд Тейлора на этой линии в окрестности x_0:

$$f(x_0 + \varepsilon) = f(x_0) + \langle \nabla f, \varepsilon \rangle + o(\|\varepsilon\|),$$

где $x_0 + \varepsilon \in S(x_0)$. Поскольку $f(x_0 + \varepsilon) = f(x_0)$ (как-никак, это линия уровня), получим

$$\langle \nabla f, \varepsilon \rangle = o(\|\varepsilon\|).$$

Поделим обе части на ε:

$$\left\langle \nabla f, \frac{\varepsilon}{\|\varepsilon\|} \right\rangle = o(1).$$

Устремим $\|\varepsilon\|$ к нулю. При этом вектор $\frac{\varepsilon}{\|\varepsilon\|}$ будет стремится к касательной к линии уровня в точке x_0. В пределе получим, что градиент ортогонален этой касательной.

§4.2 Градиентный спуск

Основное свойство антиградиента — он указывает в сторону наискорейшего убывания функции в данной точке. Соответственно, будет логично стартовать из некоторой точки, сдвинуться в сторону антиградиента, пересчитать антиградиент и снова сдвинуться в его сторону и т.д. Запишем это более формально. Пусть $w^{(0)}$ — начальный набор параметров (например, нулевой или сгенерированный из некоторого случайного распределения). Тогда градиентный спуск состоит в повторении следующих шагов до сходимости:

$$w^{(k)} = w^{(k-1)} - \eta_k \nabla Q(w^{(k-1)}). \tag{4.1}$$

Здесь под $Q(w)$ понимается значение функционала ошибки для набора параметров w.

Через η_k обозначается длина шага, которая нужна для контроля скорости движения. Можно делать её константной: $\eta_k = c$. При этом если длина шага слишком большая, то есть риск постоянно «перепрыгивать» через точку минимума, а если шаг слишком маленький, то движение к минимуму может занять слишком много итераций. Иногда длину шага монотонно уменьшают по мере движения — например, по простой формуле

$$\eta_k = \frac{1}{k}.$$

В пакете `vowpal wabbit`, реализующем настройку и применение линейных моделей, используется более сложная формула для шага в градиентном спуске:

$$\eta_k = \lambda \left(\frac{s_0}{s_0 + k} \right)^p,$$

где λ, s_0 и p — параметры (мы опустили в формуле множитель, зависящий от номера прохода по выборке). На практике достаточно настроить параметр λ, а остальным присвоить разумные значения по умолчанию: $s_0 = 1$, $p = 0.5$, $d = 1$.

Останавливать итерационный процесс можно, например, при близости градиента к нулю или при слишком малом изменении вектора весов на последней итерации.

Если функционал $Q(w)$ выпуклый, гладкий и имеет минимум w^*, то имеет место следующая оценка сходимости:

$$Q(w^{(k)}) - Q(w^*) = O(1/k).$$

§4.3 Оценивание градиента

Как правило, в задачах машинного обучения функционал $Q(w)$ представим в виде суммы ℓ функций:

$$Q(w) = \sum_{i=1}^{\ell} q_i(w).$$

В таком виде, например, записан функционал в задаче (3.1), где отдельные функции $q_i(w)$ соответствуют ошибкам на отдельных объектах.

Проблема метода градиентного спуска (4.1) состоит в том, что на каждом шаге необходимо вычислять градиент всей суммы (будем его называть полным градиентом):

$$\nabla_w Q(w) = \sum_{i=1}^{\ell} \nabla_w q_i(w).$$

Это может быть очень трудоёмко при больших размерах выборки. В то же время точное вычисление градиента может быть не так уж необходимо — как правило, мы делаем не очень большие шаги в сторону антиградиента, и наличие в нём неточностей не должно сильно сказаться на общей траектории. Опишем несколько способов оценивания полного градиента.

Оценить градиент суммы функций можно градиентом одного случайно взятого слагаемого. В этом случае мы получим метод *стохастического градиентного спуска* (stochastic gradient descent, SGD) [1]:

$$w^{(k)} = w^{(k-1)} - \eta_k \nabla q_{i_k}(w^{(k-1)}),$$

где i_k — случайно выбранный номер слагаемого из функционала. Для выпуклого и гладкого функционала может быть получена следующая оценка:

$$\mathbb{E}\left[Q(w^{(k)}) - Q(w^*)\right] = O(1/\sqrt{k}).$$

Таким образом, метод стохастического градиента имеет менее трудоемкие итерации по сравнению с полным градиентом, но и скорость сходимости у него существенно меньше.

Отметим одно важное преимущество метода стохастического градиентного спуска. Для выполнения одного шага в данном методе требуется вычислить градиент лишь одного слагаемого — а поскольку одно слагаемое соответствует ошибке на одном объекте, то получается, что на каждом шаге необходимо держать в памяти всего один объект из выборки. Данное наблюдение позволяет обучать линейные модели на очень больших выборках: можно считывать объекты с диска по одному, и по каждому делать один шаг метода SGD.

В 2013 году был предложен метод *среднего стохастического градиента* (stochastic average gradient) [2], который в некотором смысле сочетает низкую сложность итераций стохастического градиентного спуска и высокую скорость сходимости полного градиентного спуска. В начале работы в нём выбирается первое приближение w^0, и инициализируются вспомогательные переменные z_i^0, соответствующие градиентам слагаемых функционала:

$$z_i^{(0)} = \nabla q_i(w^{(0)}), \qquad i = 1, \ldots, \ell.$$

На k-й итерации выбирается случайное слагаемое i_k и обновляются вспомогательные переменные:

$$z_i^{(k)} = \begin{cases} \nabla q_i(w^{(k-1)}), & \text{если } i = i_k; \\ z_i^{(k-1)} & \text{иначе.} \end{cases}$$

Иными словами, пересчитывается один из градиентов слагаемых. Наконец, делается градиентный шаг:

$$w^{(k)} = w^{(k-1)} - \eta_k \sum_{i=1}^{\ell} z_i^{(k)}.$$

Данный метод имеет такой же порядок сходимости для выпуклых и гладких функционалов, как и обычный градиентный спуск:

$$\mathbb{E}\left[Q(w^{(k)}) - Q(w^*)\right] = O(1/k).$$

Существует множество других способов получения оценки градиента. Например, это можно делать без вычисления каких-либо градиентов вообще [3] — достаточно взять случайный вектор u на единичной сфере и домножить его на значение функции в данном направлении:

$$\nabla_w Q(w) = Q(w + \delta u)u.$$

Можно показать, что данная оценка является несмещённой для сглаженной версии функционала Q.

В задаче оценивания градиента можно зайти ещё дальше. Если вычислять градиенты $\nabla_w q_i(w)$ сложно, то можно *обучить модель*, которая будет выдавать оценку градиента на основе текущих значений параметров. Этот подход был предложен для обучения глубинных нейронных сетей [4].

5 Переобучение

Итак, мы выработали достаточно общий метод обучения линейных регрессионных моделей, основанный на градиентных методах оптимизации. При этом модель может оказаться *переобученной* — её качество на новых данных может быть существенно хуже качества на обучающей выборке. Действительно, при обучении мы требуем от модели лишь хорошего качества на обучающей выборки, и совершенно не очевидно, почему она должна при этом хорошо *обобщать* эти результаты на новые объекты.

В следующем разделе мы обсудим подходы к оцениванию обобщающей способности, а пока разберём явление переобучения на простом примере. Рассмотрим некоторую одномерную выборку, значения единственного признака x в которой генерируются равномерно на отрезке $[0, 1]$, а значения целевой переменной выбираются по формуле $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, где $\mathcal{N}(\mu, \sigma^2)$ — нормальное распределение со средним μ и дисперсией σ^2. Попробуем восстановить зависимость с помощью линейных моделей над тремя наборами признаков: $\{x\}$, $\{x, x^2, x^3, x^4\}$ и $\{x, x^2, \ldots, x^{15}\}$. Соответствующие результаты представлены на рис. 1.

Видно, что при использовании признаков высоких степеней модель получает возможность слишком хорошо подогнаться под выборку, из-за чего становится непригодной для дальнейшего использования. Эту проблему можно решать многими способами — например, использовать более узкий класс моделей или штрафовать за излишнюю сложность полученной модели. Так, можно заметить, что у переобученной модели, полученной на третьем наборе признаков, получаются очень большие

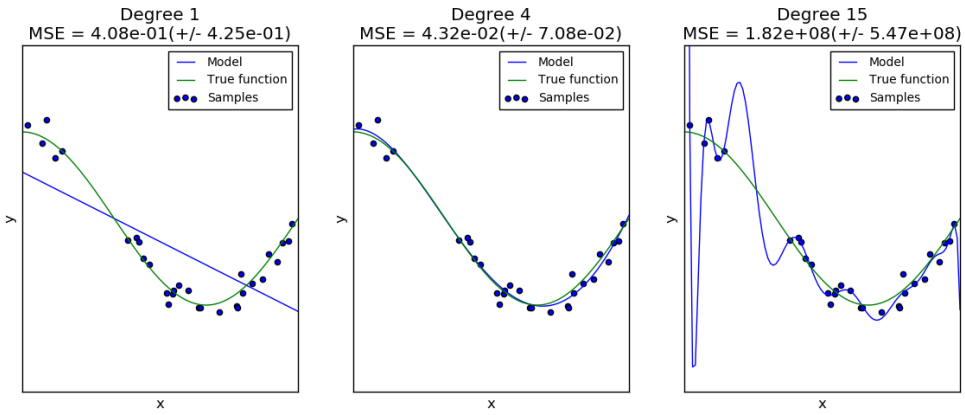

Рис. 1. Регрессионные кривые для признаковых наборов различной сложности.

коэффициенты при признаках. Как правило, именно норма вектора коэффициентов используется как величина, которая штрафуется для контроля сложности модели. Такой подход называется *регуляризацией*, речь о нём пойдёт ниже.

6 Оценивание качества моделей

В примере, о котором только что шла речь, мы не можем обнаружить переобученность модели по обучающей выборке [1]. С другой стороны, если бы у нас были дополнительные объекты с известными ответами, то по ним заметить низкое качество модели было бы довольно легко.

На данной идее основан подход с *отложенной выборкой*. Имеющиеся размеченные данные (т.е. данные с известными ответами) разделяются на две части: обучающую и контрольную. На обучающей выборке, как это следует из названия, модель обучается, а на контрольной выборке проверяется её качество. Если значение функционала на контрольной выборке оказалось удовлетворительным, то можно считать, что модель смогла извлечь закономерности при обучении.

Использование отложенной выборки приводит к одной существенной проблеме: результат существенно зависит от конкретного разбиения данных на обучение и контроль. Мы не знаем, какое качество получилось бы, если бы объекты из данного контроля оказались в обучении. Решить эту проблему можно с помощью *кросс-валидации*. Размеченные данные разбиваются на k блоков $X_1, \ldots, X_k$ примерно одинакового размера. Затем обучается k моделей $a_1(x), \ldots, a_k(x)$, причём i-я модель обучается на объектах из всех блоков, кроме блока i. После этого качество каждой модели оценивается по тому блоку, который не участвовал в её обучении, и результаты усредняются:

$$\mathrm{CV} = \frac{1}{k}\sum_{i=1}^{k} Q\left(a_i(x), X_i\right).$$

[1] Конечно, это можно было бы заметить по большим весам в модели, но связь между нормой весов и обобщающей способностью алгоритма неочевидна.

7 Регуляризация

Выше мы упоминали, что если матрица X^TX не является обратимой, то с оптимизацией среднеквадратичной ошибки могут возникнуть некоторые трудности. Действительно, в ряде случаев (признаков больше чем объектов, коррелирующие признаки) оптимизационная задача $Q(w) \to \min$ может иметь бесконечное число решений, большинство которых являются переобученными и плохо работают на тестовых данных. Покажем это.

Пусть в выборке есть линейно зависимые признаки. Это по определению означает, что существует такой вектор v, что для любого объекта x выполнено $\langle v, x\rangle = 0$. Допустим, мы нашли оптимальный вектор весов w для линейного классификатора. Но тогда классификаторы с векторами $w + \alpha v$ будут давать *точно такие же* ответы на всех объектах, поскольку

$$\langle w + \alpha v, x\rangle = \langle w, x\rangle + \alpha \underbrace{\langle v, x\rangle}_{=0} = \langle w, x\rangle.$$

Это значит, что метод оптимизации может найти решение со сколько угодно большими весами. Такие решения не очень хороши, поскольку классификатор будет чувствителен к крайне маленьким изменениям в признаках объекта, а значит, переобучен.

Мы уже знаем, что переобучение нередко приводит к большим значениям коэффициентов. Чтобы решить проблему, добавим к функционалу *регуляризатор*, который штрафует за слишком большую норму вектора весов:

$$Q_\tau(w) = Q(w) + \tau R(w).$$

Наиболее распространенными являются L_2 и L_1-регуляризаторы:

$$R(w) = \|w\|_2 = \sum_{i=1}^{d} w_i^2,$$

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

Коэффициент τ называется параметром регуляризации и контролирует баланс между подгонкой под обучающую выборку и штрафом за излишнюю сложность. Разумеется, значение данного параметра следует подбирать под каждую задачу.

Отметим, что свободный коэффициент w_0 нет смысла регуляризовывать — если мы будем штрафовать за его величину, то получится, что мы учитываем некие априорные представления о близости целевой переменной к нулю и отсутствии необходимости в учёте её смещения. Такое предположение является достаточно странным. Особенно об этом следует помнить, если в выборке есть константный признак и коэффициент w_0 обучается наряду с остальными весами; в этом случае следует исключить слагаемое, соответствующее константному признаку, из регуляризатора.

Отбор признаков. Особенностью L_1-регуляризатора является то, что он зануляет часть весов, осуществляя тем самым отбор признаков. Попробуем понять, почему это так.

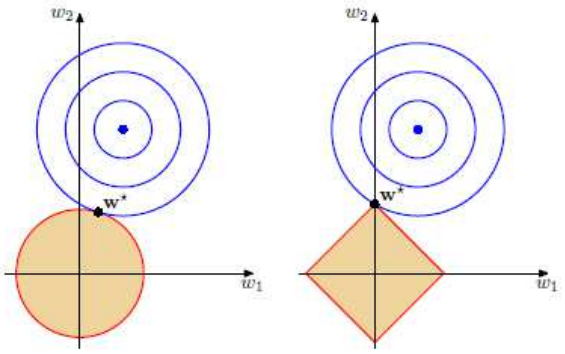

Рис. 2. Линии уровня функционала качества, а также ограничения, задаваемые L_2 и L_1-регуляризаторами.

Можно показать, что если функционал $Q(w)$ является выпуклым, то задача безусловной минимизации функции $Q(w) + \tau\|w\|_1$ эквивалентна задаче условной оптимизации

$$\begin{cases} Q(w) \to \min\limits_{w} \\ \|w\|_1 \leqslant C \end{cases}$$

для некоторого C. На рис. 2 изображены линии уровня функционала $Q(w)$, а также множество, определяемое ограничением $\|w\|_1 \leqslant C$. Решение определяется точкой пересечения допустимого множества с линией уровня. В большинстве случаев эта точка будет лежать на одной из вершин ромба, что соответствует решению с одной зануленной компонентой.

8 Гиперпараметры

В машинном обучении принято разделять подлежащие настройке величины на *параметры* и *гиперпараметры*. Параметрами называют величины, которые настраиваются по обучающей выборки — например, веса в линейной регрессии. К гиперпараметрам относят величины, которые контролируют сам процесс обучения и не могут быть подобраны по обучающей выборке.

Хорошим примером гиперпараметра является коэффициент регуляризации τ. Введение регуляризации мешает модели подгоняться под обучающие данные, и с точки зрения среднеквадратичной ошибки выгодно всегда брать $\tau = 0$. Разумеется, такой выбор не будет оптимальным с точки зрения качества на новых данных, и поэтому коэффициент регуляризации (как и другие гиперпараметры) следует настраивать по отложенной выборке или с помощью кросс-валидации.

При подборе гиперпараметров по кросс-валидации возникает проблема: мы используем отложенные данные, чтобы выбрать лучший набор гиперпараметров. По сути, отложенная выборка тоже становится обучающей, и показатели качества на ней перестают характеризовать обобщающую способность модели. В таких случаях выборку, на которой настраиваются гиперпараметры, называют валидационной, и при этом выделяют третий, тестовый набор данных, на которых оценивается качество итоговой модели.

Список литературы

[1] *Robbins, H., Monro S.* (1951). A stochastic approximation method. // Annals of Mathematical Statistics, 22 (3), p. 400-407.

[2] *Schmidt, M., Le Roux, N., Bach, F.* (2013). Minimizing finite sums with the stochastic average gradient. // Arxiv.org.

[3] *Flaxman, Abraham D. and Kalai, Adam Tauman and McMahan, H. Brendan* (2005). Online Convex Optimization in the Bandit Setting: Gradient Descent Without a Gradient. // Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms.

[4] *Jaderberg, M. et. al* (2016). Decoupled Neural Interfaces using Synthetic Gradients. // Arxiv.org.