Урок 6

Практические рекомендации по линейным моделям

Этот урок будет посвящен сложностям, с которыми можно столкнуться при применении линейных моделей к реальным задачам.

6.1. Масштабирование признаков

6.1.1. Пример: необходимость масштабирования

Понять необходимость масштабирования можно на следующем простом примере. Пусть необходимо найти минимум:

$$w_1^2 + w_2^2 \to \min_w .$$

Ответ в этом случае очевиден — это точка $(0, 0)$. При поиске этого минимума методом градиентного спуска с начальной точкой $w = (1, 1)$, вектор антиградиента будет иметь координаты $(-2, -2)$. Это вектор проходит через точку минимума, а значит при правильно подобранном размере шага, уже на первом шаге градиентного спуска можно попасть строго в точку минимума. Градиентный спуск будет хорошо работать на этой функции.

Изменим функцию:

$$w_1^2 + 100w_2^2 \to \min_w .$$

В этом случае линии уровня представляют собой эллипсы, сильно вытянутые вдоль оси x. Если запустить градиентный спуск из точки $w = (1, 1)$, вектор антиградиента в этой точке будет иметь координаты $(-2, -200)$. Вектор будет смотреть почти строго вниз и проходить мимо точки минимума функции. Более того, существует шанс на следующей итерации уйти еще дальше от минимума.

Итак, градиентный спуск работает хорошо, если линии уровня функции похожи на круги. В этом случае, откуда бы вы не начали, вектор градиента будет всегда смотреть в сторону минимума функции, а итеративный процесс будет сходиться довольно быстро. Если же линии уровня похожи на эллипсы, направление антиградиента будет слабо совпадать с направлением в сторону минимума функции. Градиентный спуск будет делать много лишних шагов. Сходимость будет медленная, и более того, существует риск расхождения итеративного процесса, если размер шага будет подобран неправильно.

6.1.2. Масштабирование выборки

Пусть требуется предсказать, будет ли выдан грант по заявке. Заявка характеризуется двумя признаками — сколько уже успешных заявок было у данного заявителя и год рождения заявителя. Масштабы этих двух признаков существенно отличаются: количество одобренных грантов обычно меряется единицами, а год рождения — тысячами. Из-за такого различия в масштабе линии уровня функции будут выглядеть скорее как эллипсы, а не как круги.

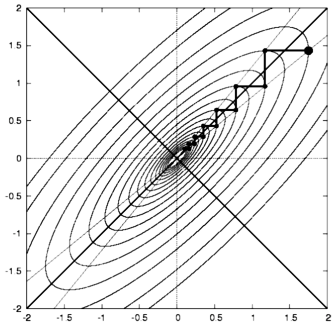

Рис. 6.1: Если масштабирование не было произведено, при использовании метода градиентного спуска будет сделано много лишних шагов, а также существует риск расхождения итеративного процесса.

В таком случае, чтобы без проблем пользоваться градиентным спуском, признаки необходимо привести к одному масштабу, то есть отмасштабировать. Будут рассматриваться два способа масштабирования.

Первый способ называется стандартизацией. Для начала необходимо вычислить вспомогательные величины: средние значения признаков и стандартные отклонения:

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

Чтобы произвести стандартизацию признака, достаточно вычесть из него его среднее значение и разделить на его стандартное отклонение:

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

После того, как это будет выполнено для всех признаков, сдвиги и различия в масштабах будут убраны.

Второй подход называется «масштабирование на отрезок $[0,1]$». В этом случае также необходимо вычислить вспомогательные величины: максимальное и минимальное значение каждого признака на обучающей выборке:

$$m_j = \min(x_1^j, ..., x_\ell^j), \qquad M_j = \max(x_1^j, ..., x_\ell^j).$$

После этого значение каждого признака на конкретном объекте преобразовывается следующим образом:

$$x_i^j := \frac{x_i^j - m_j}{M_j - m_j}.$$

Тогда минимальное значение признака отображается в ноль, а максимальное — в 1, то есть признаки масштабируются на отрезок $[0,1]$.

6.2. Нелинейные зависимости

В этом разделе речь пойдет о спрямляющих пространствах, которые позволяют восстанавливать нелинейные зависимости с помощью линейных моделей. Для начала простой пример.

6.2.1. Нелинейная задача регрессии

Пусть решается задача регрессии с одним признаком, отложенным по оси x. По этому признаку требуется восстановить целевую переменную y.

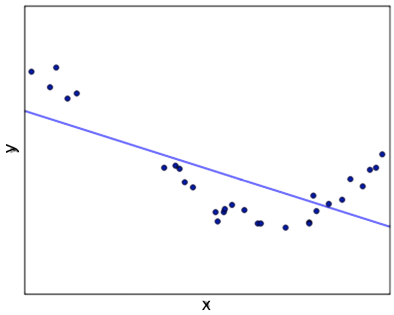

Рис. 6.2: Линейная модель сама по себе не способна восстанавливать нелинейные зависимости.

Как можно в этом убедиться, непосредственное применение линейной модели не дает желаемых результатов и плохо подходит для решения данной задачи. Зависимость y от x явно нелинейная.

Поэтому, кроме признака x, также рассматриваются признаки x^2, x^3 и x^4. Если построить линейную модель на этих признаках, в ней уже будет 5 коэффициентов.

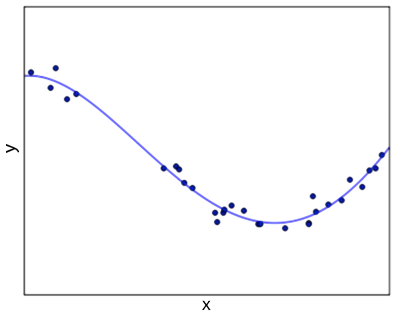

Рис. 6.3: Описать данные удалось с помощью перехода к другому признаковому пространству.

С помощью этой модели уже можно почти идеально описывать данные: она очень хорошо решает задачу, а также не переобучается.

Фактически был совершен переход к новому признаковому пространству из четырех признаков вместо одного, в котором была построена линейная модель. А на исходном пространстве эта модель уже нелинейная и отлично описывает данные.

6.2.2. Задача классификации с нелинейной границей

Другой пример — решается задача классификации с двумя признаками, которые отложены по осям.

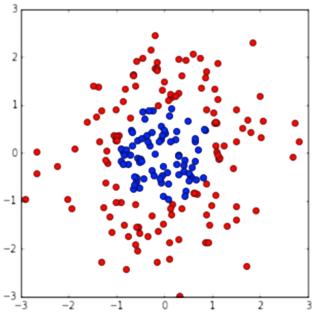

Рис. 6.4: Набор данных для рассматриваемой задачи классификации.

Видно, что разделяющая поверхность здесь не является линейной. Применение линейного классификатора дает следующий результат:

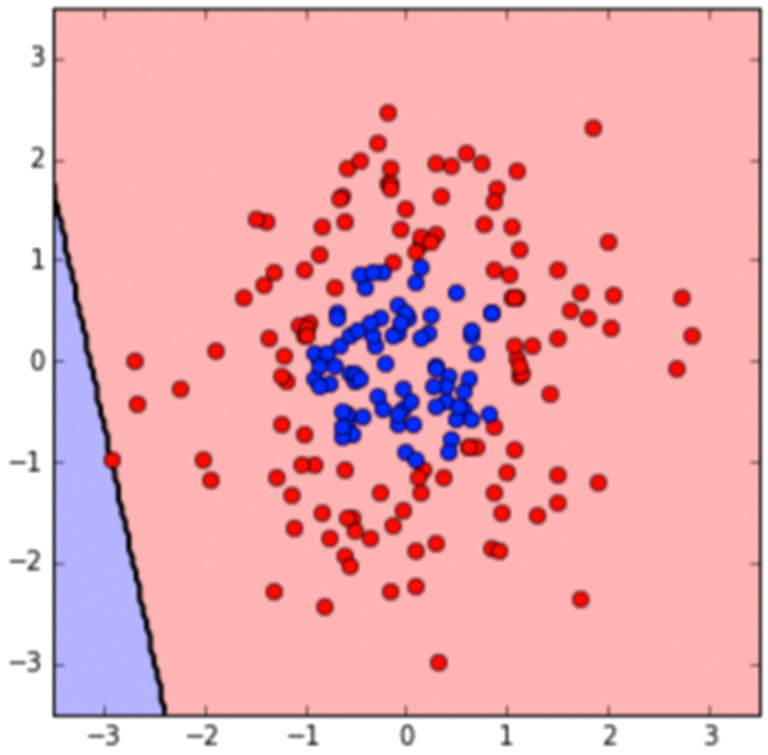

Рис. 6.5: Линейный классификатор не смог решить задачу.

То есть все объекты будут отнесены к красному классу: это лучшее, что он может сделать, но понятно, что это никуда не годится.

Но если кроме признаков x_1 и x_2 рассмотреть признаковое пространство, которое также включает в себя признаки x_1x_2, x_1^2 и x_2^2, результат будет совершенно иной: разделяющая поверхность будет иметь форму окружности и очень хорошо разделять красные и синие точки.

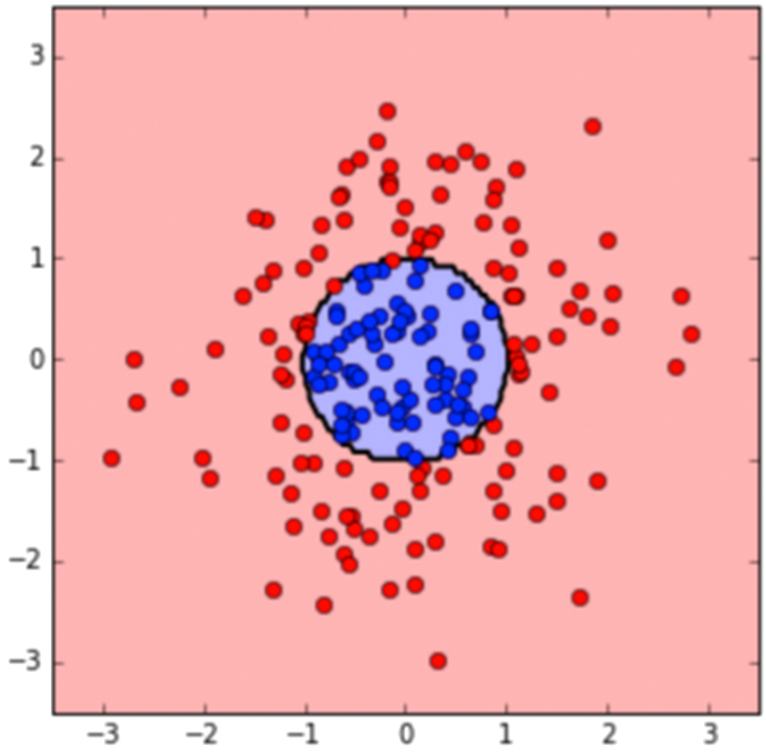

Рис. 6.6: Красные и синие точки хорошо разделились.

Здесь также в новом пространстве была построена линейная модель, которая является нелинейной в исходном признаковом пространстве.

6.2.3. Спрямляющее пространство

Два рассмотренных примера дают представление о спрямляющем пространстве. Спрямляющим пространством признаков называется такое пространство, в котором задача хорошо решается линейной моделью. Спрямляющее пространство может, в том числе, быть построено:

- Через добавление квадратичных признаков, то есть когда к исходным признакам добавляются их квадраты и попарные произведения:

  $$(x_1, ..., x_d) \to (x_1, ..., x_d, x_1^2, ..., x_d^2, x_1x_2, ..., x_{d-1}x_d).$$

  Следует особо отметить, что в таком случае число признаков увеличивается на порядок. Если объектов не слишком много, существует риск переобучения в таком большом признаковом пространстве.

- Через добавление полиномиальных признаков:

  $$(x_1, ..., x_d) \to (x_1, ..., x_d, ..., x_ix_j, ..., x_ix_jx_k, ...).$$

  Этот способ следует использовать только, когда объектов достаточно много, то есть риск переобучения минимален, а также заранее известно, что зависимости очень нелинейные.

- Через логарифмирование (или любые другие нелинейные функции):

  $$x_i \to \ln(x_i + 1), \qquad x_i \to \ln(|x_i| + 1).$$

  Ранее приводился пример про стоимость книг в интернет-магазине. В данном случае признаком будет стоимость книги, значение которого для большинства книг составит несколько сотен рублей. Но существуют и встречаются весьма часто очень дорогие книги, так что распределение этого признака будет иметь «тяжелый правый хвост». Известно, что линейные модели плохо применимы в таком случае и работают гораздо лучше, если распределение признаков близко к нормальному. Чтобы распределение признака «с хвостом» сделать более близким к нормальному, нужно прологарифмировать этот признак.

6.3. Работа с категориальными признаками

В этом разделе пойдет речь о том, как использовать категориальные признаки в линейных или других моделях.

Ранее уже было приведено множество примеров категориальных признаков:

- Город
- Цвет
- Тарифный план
- Марка автомобиля
- и так далее...

Особенность категориальных признаков состоит в том, что это элементы некоторого неупорядоченного множества, и нельзя говорить, что какое-то значение больше или меньше другого. Можно только сравнивать их на равенство. Но в линейных моделях нужно брать значение признака, умножать на вес, а потом складывать с другими числами, и эту операцию нельзя делать со значениями категориальных признаков. Категориальные признаки нужно сначала преобразовать, чтобы их можно было использовать в линейных моделях.

6.3.1. Бинарное кодирование

Один из наиболее популярных подходов к кодированию категориальных признаков — бинарное кодирование. Далее будут использоваться следующие обозначения. Пусть j-ый признак — категориальный и принимает n возможных значений:

$$c_1, ..., c_n.$$

Пусть также $f_j(x)$ — значение этого признака на объекте x. Чтобы закодировать данный признак, вводятся n новых бинарных признаков:

$$b_1(x), ..., b_n(x),$$

причем значение бинарного признака b_i равно единице только в том случае, если на данном объекте x значение категориального признака $f_j(x)$ равно c_i:

$$b_i(x) = [f_j(x) = c_i].$$

В результате один категориальный признак заменяется n бинарными признаками.

6.3.2. Бинарное кодирование (пример)

Пусть в качестве признака рассматривается цвет, причем он может принимать три значения: синий, зеленый или красный. Дано три объекта x_1, x_2, x_3, на которых значения категориального признака:

$$f_j(x_1) = \text{синий}, \qquad f_j(x_2) = \text{красный}, \qquad f_j(x_3) = \text{синий}.$$

Поскольку категориальный признак принимает 3 значения, потребуется 3 бинарных признака, чтобы его закодировать. В результате получается следующая матрица (каждому объекту соответствует своя строка, а каждому столбцу — свое значение признака):

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

6.3.3. Бинарное кодирование: новые значения

Часто возникает следующая проблема. При кодирования тестовой выборки может встретиться объект, на котором категориальный признак принимает новое $(n+1)$-е значение, которое до этого не встречалось в обучающей выборке. В этом случае логичным подходом будет не добавлять новый признак (поскольку это сложно реализуется), а просто приравнять к нулю все существующие признаки. Действительно, по смыслу бинарных признаков (принимает ли категориальный признак значение c_i, $i \in \{1, 2, ..., n\}$), каждый из них в таком случае должен равняться нулю.

6.4. Несбалансированные данные

В этом разделе пойдет речь о том, что такое несбалансированные выборки, к каким проблемам они могут привести, а также как бороться с такими проблемами.

6.4.1. Несбалансированная выборка

Пусть решается задача классификации с несбалансированной выборкой. Задача называется несбалансированной, если объектов одного класса существенно меньше, чем объектов остальных классов. Например, задача бинарной классификации называется несбалансированной, если объектов одного из двух классов менее 10%.

Примеров задач с несбалансированными выборками довольно много:

- Предсказание резких скачков курса доллара. Если определение резкого скачка подразумевает сильные изменения, то примеров таких изменений за всю историю — единицы, но при этом практически каждый день — отрицательный пример, то есть пример, когда такого скачка не было. Выборка в этом случае будет очень несбалансированной.
- Медицинская диагностика (больных, как правило, сильно меньше, чем здоровых)
- Обнаружение мошеннических транзакций (которых существенно меньше, чем обычных транзакций)
- Классификация текстов и так далее.

Основная проблема, связанная с несбалансированными выборками, состоит в том, что классификаторы минимизируют число неправильных ответов и никак не учитывают цены ошибок. Может возникнуть ситуация, когда выгоднее отнести все объекты к большему классу, не пытаясь как-то выделить объекты маленького класса. Другими словами, при работе с несбалансированными выборками классификаторы получаются очень плохие с точки зрения точности или полноты.

6.4.2. Undersampling

Первый подход к работе с несбалансированными выборками — undersampling. Его основная идея состоит в том, что часть объектов из большого класса выбрасываются из выборки. Пусть, например, есть три класса: очень крупный, крупный и небольшой

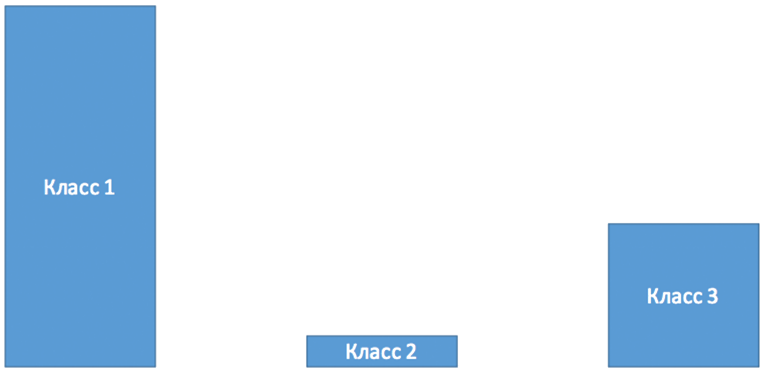

Рис. 6.7: Несбалансированная выборка

В этом случае необходимо выкинуть большую часть 1го класса и половину объектов 3го класса. Размеры классов примерно сравняются.

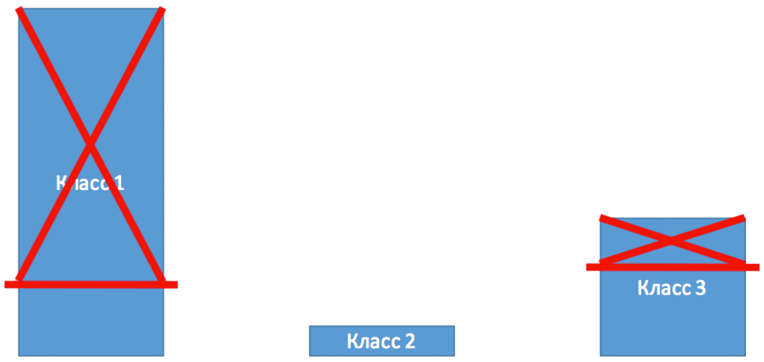

Рис. 6.8: Undersampling

При этом то, сколько именно объектов каждого класса выбрасывается — это гиперпараметр, который имеет смысл настраивать по отложенной выборке или по кросс-валидации.

6.4.3. Oversampling

Второй подход, oversampling, противоположен предыдущему: в данном случае объекты маленьких классов дублируются, чтобы выравнять соотношение классов.

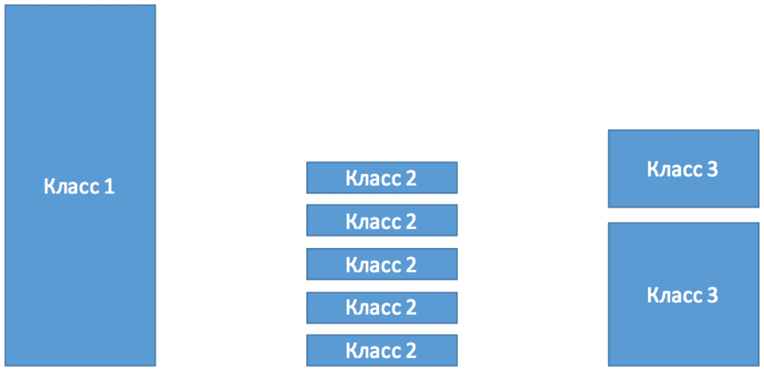

Рис. 6.9: Oversampling

В предыдущем примере 5 раз необходимо продублировать объекты 2го класса, а также продублировать случайную половину 3го класса. В этом случае размеры классов тоже выравняются. То, на сколько будет увеличен каждый класс — тоже гиперпараметр.

Следует обратить внимание на одну особенность. Если задача решается на исходной выборке, то среднеквадратичная ошибка выглядит:

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

Но если делается oversampling, то есть дублируются какие-нибудь объекты, некоторые слагаемые будут входить в эту сумму несколько раз. Таким образом, вместо реального дублирования объектов, можно выставить соответствующие веса ν_i:

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

6.4.4. Стратификация

Еще одна проблема, с которой можно столкнуться при работе с несбалансированными выборками, заключается в том, что при проведении кросс-валидации исходная выборка разбивается на k блоков примерно одинаковой длины. При этом, если выборка несбалансированная, может получиться ситуация, что в некоторые блоки объекты какого-то класса не попадут вообще. Такая ситуация крайне неприятна: при обучении на этом блоке получается классификатор, который никогда не видел один из классов.

Чтобы с этим бороться необходимо использовать стратификацию, то есть делать так, чтобы распределение классов в каждом блоке примерно совпадало с распределением классов в исходной выборке. В этом случае будет гарантироваться, что объекты каждого из классов будут представлены в каждом из блоков разбиения.

6.5. Многоклассовая классификация

В этом разделе рассказывается, как решать задачи многоклассовой классификации с помощью линейных моделей. Как следует из названия, в задачах многоклассовой классификации K возможных классов. Требуется научиться отличать каждый класс от всех остальных.

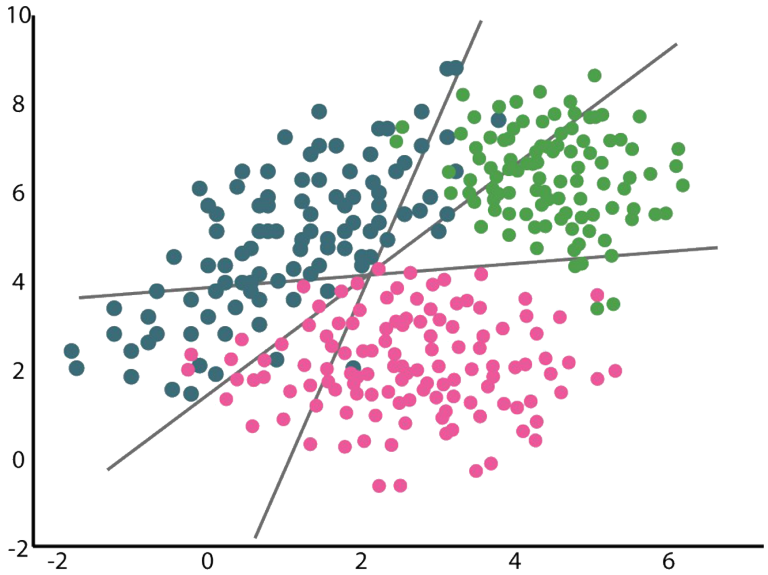

Рис. 6.10: Многоклассовая классификация

В случае бинарной классификации подход был простой — нужно было найти такой вектор весов w, что выражение $a(x) = \text{sign}\langle w, x\rangle$ определяло бы, какому классу этот объект относится.

6.5.1. ONE-VS-ALL

Этот метод можно применить к задаче многоклассовой классификации. Такой подход называется «один против всех». Как следует из названия, для каждого класса будет строиться свой бинарный классификатор. Задачей этого классификатора будет отделение данного класса от всех остальных.

Рис. 6.11: ONE-VS-ALL

Формально говоря, будут решаться K задач бинарной классификации. Для каждой (например k-ой) из этих задач будет весьма специфичная выборка:

$$X = (x_i, [y_i = k])_{i=1}^{\ell},$$

в которой объекты x_i остаются такими же, а ответы становятся бинарными. Будет построен некоторый линейный классификатор, который отделяет k-ый класс от всех остальных:

$$a_k(x) = \text{sign}\langle w_k, x\rangle.$$

Ранее уже было сказано, что уверенность классификатора в своем решении определяется значением скалярного произведения. Если знак скалярного произведения положительный, то чем больше его модуль, тем больше классификатор уверен в том, что данный конкретный объект относится к данному классу. Поэтому для построения многоклассового классификатора можно использовать следующий алгоритм:

$$a(x) = \text{argmax}_{k\in 1,\dots,K}\langle w_k, x\rangle,$$

который будет возвращать тот класс k, для которого уверенность соответствующего классификатора (то есть значение соответствующего скалярного произведения) больше всего.

6.5.2. Матрица ошибок

Для анализа того, насколько хорошо работает многоклассовый классификатор, удобно использовать матрицу ошибок. Каждая строка этой матрицы соответствует тем объектам, которые классификатор отнес к тому или иному классу, а каждый столбец соответствует объектам, которые на самом деле относятся к тому или иному классу.

Например, на пересечении первой строки и второго столбца стоит число q_{12}, которое показывает то, сколько объектов второго класса многоклассовый классификатор отнес к первому. Эта матрица позволяет понять, какие классы перепутываются чаще всего. Также можно измерять уже известные метрики качества, например:

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

Кроме того, можно считать точность и полноту (а также F-меру) для задачи отделения того или иного класса от всех остальных классов. Если точность и полнота будут таким образом вычислены для каждого из классов, они могут быть позднее усреднены, чтобы получить агрегированные оценки.