Урок 12

Метрические алгоритмы и SVM

12.1. Метод ближайших соседей

В этом разделе будут рассмотрены метрические алгоритмы, которые подразумевают, что в пространстве признаков было введено понятие расстояния, другими словами, определена метрика.

12.1.1. Метод ближайших соседей (kNN)

Самый простой метрический метод в задаче классификации — метод ближайшего соседа, суть которого заключается в том, что новая точка относится к такому же классу, что и ближайшая к ней точка из обучающей выборки.

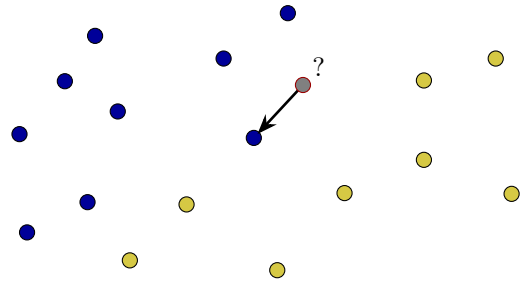

Рис. 12.1: Метод ближайшего соседа

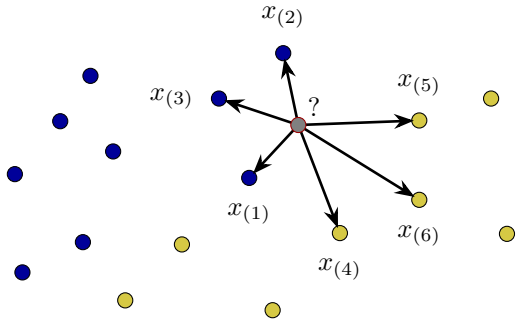

Рис. 12.2: Метод ближайших соседей ($k = 6$)

Для повышения надежности можно принимать решение по более чем одной точке. В этом и состоит метод ближайших соседей: объект относится к тому классу, к которому принадлежит большинство из его k ближайших соседей. Еще больше повысить надежность можно правильным образом определив веса в методе ближайших соседей. Веса могут зависеть как от номера соседа $w(x_{(i)}) = w(i)$, так и от расстояния до него $w(x_{(i)}) = w(d(x, x_{(i)}))$.

Во взвешенном kNN объект x относится к тому классу, взвешенная сумма по объектам из множества k ближайших соседей для которого больше:

$$a(x) = \text{argmax}_{y \in Y} \sum_{i=1}^{k} [x_{(i)} = y] w(x_{(i)}).$$

12.1.2. Центроидный классификатор

Еще один простой метрический классификатор — центроидный классификатор. Сначала по обучающей выборке $\{(x_i, y_i)\}_{i=1}^{m}$ определяются «центры» всех классов (ℓ_y — количество объектов класса y):

$$\mu_y = \frac{1}{\ell_y} \sum_{i: y_i = y} x_i.$$

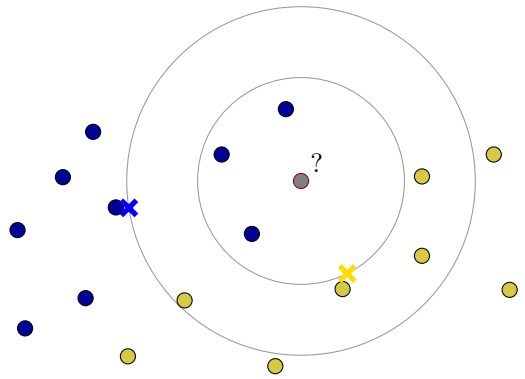

Рис. 12.3: Центроидный классификатор. Крестиками изображены центры классов.

После этого центроидный классификатор относит каждый новый объект x к тому классу, центр которого находится ближе всего в пространстве признаков к признаковому описанию нового объекта:

$$a(x) = \text{argmin}_{y \in Y} d(\mu_y, x).$$

12.1.3. Взвешенный kNN для регрессии

Метод k ближайших соседей можно использовать для решения задачи регрессии. Пусть x — новый объект, который требуется классифицировать, а $x_{(i)}$ — i-ый ближайший сосед из обучающей выборки. Взвешенный kNN для задачи регрессии в таком случае определяется выражением:

$$a(x) = \frac{\sum_{i=1}^{k} w(x_{(i)}) x_{(i)}}{\sum_{i=1}^{k} w(x_{(i)})}.$$

Следует обратить внимание на выбор весовой функции, поскольку это сильно влияет на качество решения.

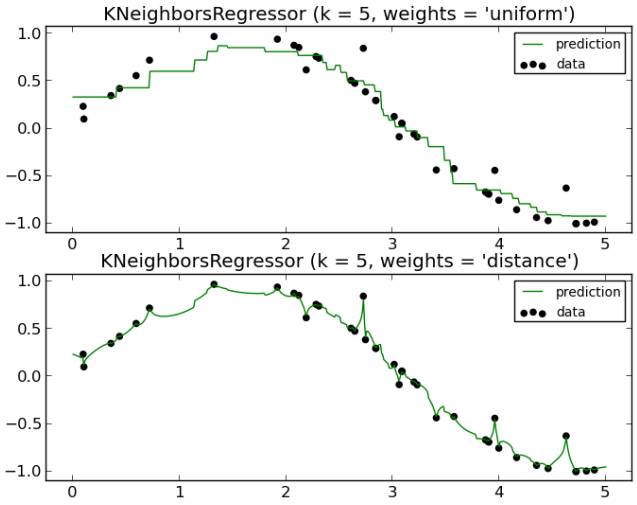

Рис. 12.4: Метод kNN в задаче регрессии в случае различных весовых функций.

Например, если в качестве веса в задаче регрессии использовать величину, обратную к расстоянию, то в результате получится переобученная модель. Это связано с тем, что, если объект уже есть в обучающей выборке, то он будет входить в сумму практически с бесконечным весом и алгоритм вернет для него то же значение, что и было в обучающей выборке.

12.2. Настройка параметров в kNN

Данный раздел посвящен вопросу подбора параметров в методе ближайших соседей так, чтобы качество работы алгоритма было наилучшим. К числу параметров относятся: количество соседей, весовые функции, метрика и так далее.

Проверять качество работы алгоритма с выбранными параметрами лучше не на обучающей выборке, а на отложенной. Также можно использовать кросс-валидацию.

12.2.1. Количество соседей

Основной параметр в kNN — количество соседей.

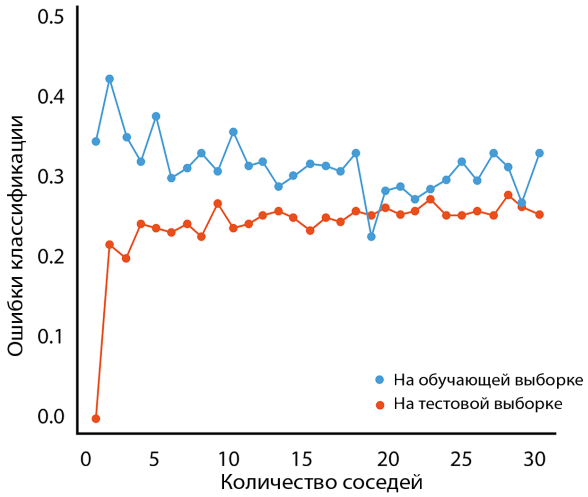

Рис. 12.5: Зависимость ошибки классификации от количества соседей в kNN для объектов из тестовой и обучающей выборок.

Если оценивать качество работы алгоритма по объектам из обучающей выборки, то, как это показывает график, оптимальное значение k будет равно 1. При этом значении k алгоритм совсем не ошибается на объектах обучающей выборки. Действительно, при классификации объекта из обучающей выборки, ближайшим к нему объектом из обучающей выборки будет он сам. Именно поэтому качество алгоритма нужно всегда проверять на тестовой выборке.

В общем зависимость от k следующая: сначала качество на контроле становится все лучше с ростом k, а затем качество начинает ухудшаться.

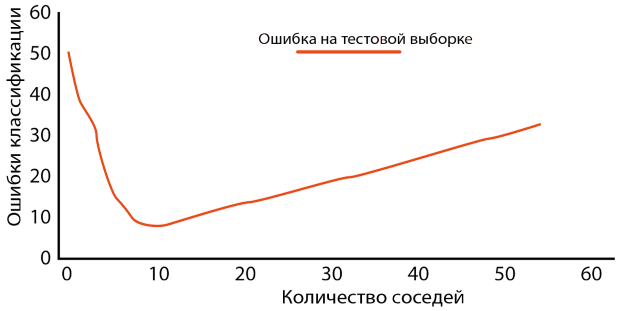

Рис. 12.6: Ошибка классификации для объектов из тестовой выборки.

Выбирать следует такое значение k, при котором достигается наилучшая оценка качества работы алгоритма на контроле.

12.2.2. Веса соседей как функция от номера

Если используется взвешенный kNN, то необходимо задать весовую функцию. Такая функция не должна возрастать с ростом номера объекта. Простейший вариант — это $w(x) = 1$. При выборе весовой функции всегда следует сначала пробовать его перед тем, как рассматривать более сложные варианты.

Если выбор $w(x) = 1$ не дает желаемых результатов, можно попробовать определить веса как функцию от номера соседа:

- $w(i) = q^i, \quad 0 < q < 1$
- $w(i) = \frac{1}{i}$, $w(i) = \frac{1}{i+a}$, $w(i) = \frac{1}{(i+a)^b}$
- $w(i) = 1 - \frac{i-1}{k}$ (не очень удачный вариант).

Функция, которая линейно зависит от номера соседа, не является хорошим выбором. Например, в случае $k = 4$, если 1 и 4 соседи некоторого объекта x принадлежат к первому классу, а 2 и 3 — ко второму, алгоритм не сможет классифицировать этот объект. Это, конечно, не значит, что эта функция вовсе не применима на практике, но эту её особенность следует учитывать.

12.2.3. Веса объектов как функция от расстояния

Другой способ определить весовую функцию — задать ее как функцию от расстояния. Ранее уже было сказано, что в задаче регрессии выбор весовой функции $w(d) = \frac{1}{d}$ приводит к переобученности. Грубо говоря, это было связано с тем, что при $d = 0$ значение весовой функции было бесконечно большим. Поэтому необходимо более аккуратно выбирать весовую функцию. Например, можно попробовать следующие варианты:

- $w(d) = \frac{1}{(d+a)^b}$
- $w(d) = q^d, \quad 0 < q < 1$

Существует более общий подход к придумыванию функции весов, зависящих от расстояний, который основан на использовании так называемых ядер. Но этот подход сейчас не будет обсуждаться.

12.3. Метрики в kNN

12.3.1. Понятие метрики

Метрика является функцией, задающей расстояние в метрическом пространстве, и должна удовлетворять следующим аксиомам:

1. $\rho(x, y) \geq 0$, причем $\rho(x, y) = 0 \iff x = y$.
2. $\rho(x, y) = \rho(y, x)$,
3. $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$.

Эти аксиомы не будут обсуждаться в рамках данного раздела, так как излагаемый далее материал носит исключительно прикладной характер.

Можно привести следующие примеры метрик:

- Евклидова метрика:
$$\rho(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
- Манхэттенская метрика:
$$\rho(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$
- Метрика Минковского (обобщение Евклидовой и Манхэттенской метрик):
$$\rho(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{\frac{1}{q}}.$$

Наглядно демонстрируют структуру метрики изображения единичных окружностей в ней.

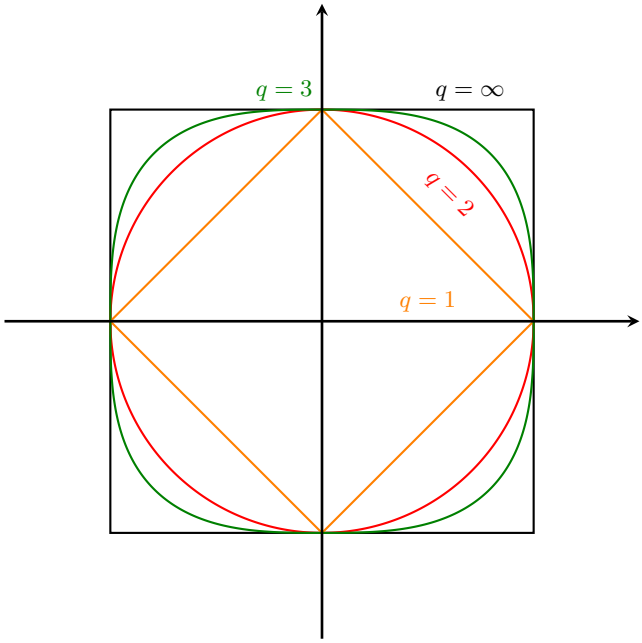

Рис. 12.7: Изображения единичной окружности, то есть множества точек, удаленных на расстояние 1 от начала координат, в различных метриках.

12.3.2. Функции близости

Часто, особенно в задачах анализа текста, используется так называемая косинусная мера, которая представляет собой косинус угла между векторами:

$$similarity = \cos\theta = \frac{x \cdot y}{\|x\| \cdot \|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Причем косинусная мера — не расстояние, а функция близости, то есть такая функция, которая тем больше, чем больше объекты друг на друга похожи.

В рекомендательных системах используется коэффициент корреляции r. Он также может быть использован как функция близости и похож на косинусную меру:

$$r = \frac{\sum_{i=1}^{n} ((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

Конечно, существует и много других функций близости, которые в разной степени учитывают разные различия между векторами:

- Скалярное произведение: $\sum x_i y_i$
- Коэффициент Дайса: $\frac{2\sum x_i y_i}{\sum x_i^2 + \sum y_i^2}$
- Косинусная мера: $\frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$
- Коэффициент Жаккара: $\frac{\sum x_i y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i y_i}$

12.4. Проклятие размерности

В этом разделе будут рассмотрены основные особенности применения понятия расстояния в случае пространств большой размерности. Часто пространства признаков в задачах машинного обучения как раз и относятся к этому случаю.

12.4.1. Небольшие отличия в большом числе координат

Пусть даны три точки, радиус-векторы которых:

$$x_1 = (a_1, a_2, ..., a_N),$$
$$x_2 = (a_1 + \varepsilon, a_2 + \varepsilon, ..., a_N + \varepsilon),$$
$$x_3 = (a_1, a_2 + \Delta, ..., a_N).$$

Вектор x_2 отличается от вектора x_1 незначительно в каждой координате, а вектор x_3 — только в одной, но существенно. При этом расстояние от первой точки до второй и расстояние от первой до третьей могут совпадать.

Другими словами, когда признаков очень много, незначительные различия в каждом признаке могут значить больше, чем одно большое различие в одном. Такое поведение не всегда является желаемым.

12.4.2. Почти одинаковые расстояния

Когда количество объектов сравнимо с количеством признаков, может возникнуть ситуация, что расстояния между двумя любыми объектами будет почти одинаковым.

Действительно, на плоскости три точки, равноудаленные друг от друга, образуют вершины треугольника, в трехмерном пространстве четыре равноудаленные друг от друга, точки являются вершинами тетраэдра, а в N-мерном пространстве можно выбрать $N + 1$ точку так, что расстояние между любыми двумя было одинаковым.

12.4.3. Экспоненциальный рост необходимых данных

Пусть X — вектор в признаковом пространстве из N бинарных признаков, например:

$$X = (0, 0, 1, 0, 1, 1, ..., 1).$$

Всего в этом пространстве 2^N различных векторов, а значит размер обучающей выборки, необходимый, чтобы покрыть все возможные комбинации эти признаков будет также порядка 2^N.

Другими словами, количество необходимых данных с ростом размерности пространства экспоненциально увеличивается.

У данного факта есть красивая геометрическая иллюстрация. Пусть в N-мерном пространстве дан куб с ребром 1 и меньший куб, длина ребер которого равна $\ell < 1$. Меньший куб вложен в больший таким образом, что они имеют общую вершину и их грани попарно параллельны. Доля объема меньшего куба от объема большего выражается формулой:

$$\frac{v}{V} = \ell^N \to 0, \qquad N \to \infty,$$

где v и V — объемы меньшего и большего кубов соответственно.

Пусть $\ell = \frac{1}{2}$. Если ставится задача описать положение какой-либо точки из большого куба с точностью до размеров маленького, то при увеличении размерности пространства и сохранении линейных размеров кубов количество требуемых для этого данных растет экспоненциально.

12.5. Рекомендации фильмов с помощью kNN

12.5.1. Задача

В данном разделе решается задача построения рекомендательной системы с помощью метода ближайших соседей. А именно необходимо построить рекомендации фильмов на основе истории пользовательских оценок.

Можно представить себе таблицу «пользователь-фильм», в которой какие-то значения заполнены, но не все. Таким образом, необходимо научиться прогнозировать значения, отсутствующие в данной таблице.

12.5.2. User-based подход

Один из возможных подходов к задаче заключается в том, что среди пользователей ищутся наиболее похожие на того пользователя, для которого делается прогноз. Этим пользователям в соответствие ставятся веса: тем, кто более похож, вес устанавливается больше, и так далее. В качестве прогноза для исходного пользователя

	Пила	Улица Вязов	Ванильное небо	1 + 1
Маша	5	4	1	2
Юля		5	2	
Вова			3	5
Коля	3		4	5
Петя				4
Ваня		5	3	3

Таб.: Пример возможной таблицы «пользователь-фильм».

указываются усредненные с учетом весовых коэффициентов оценки фильмов этих наиболее похожих на него пользователей.

Описанный выше прогноз называется user-based. Аналогично можно рассмотреть item-based подход, где сначала выбирается интересующий фильм, ищутся похожие на него фильмы и так далее. Далее для определенности речь будет идти только про user-based подход.

12.5.3. Похожесть пользователей

В качестве меры похожести $w_{i,j}$ двух пользователей можно использовать коэффициент корреляции:

$$w_{i,j} = \frac{\sum_a (r_{i,a} - \bar{r}_i)(r_{j,a} - \bar{r}_j)}{\sqrt{\sum_a (r_{i,a} - \bar{r}_i)^2} \sqrt{\sum_a (r_{j,a} - \bar{r}_j)^2}},$$

где $\bar{r}_i = \frac{1}{N_i} \sum_a r_{i,a}$ — средние оценки i-го пользователя, N_i — количество просмотренных им фильмов. Суммирование ведется только по тем фильмам, которые смотрели оба пользователя.

12.5.4. Прогнозирование рейтинга

Теперь можно спрогнозировать рейтинг фильма как средний рейтинг с добавленной к нему взвешенной суммой рейтингов других пользователей:

$$\hat{r}_{i,a} = \bar{r}_i + \frac{\sum_j (r_{j,a} - \bar{r}_j) w_{i,j}}{\sum_j |w_{i,j}|}.$$

Если количество пользователей в системе велико, достаточно производить суммирование только по k ближайших соседей к пользователю, для которого дается оценка:

$$\hat{r}_{i,a} = \bar{r}_i + \frac{\sum_{j \in kNN(i)} (r_{j,a} - \bar{r}_j) w_{i,j}}{\sum_{j \in kNN(i)} |w_{i,j}|}.$$

Таким образом метод kNN может быть адаптирован к задаче рекомендации.

12.6. Метод опорных векторов (SVM)

Давайте познакомимся поближе с еще одним методом, который не вошел в основную часть курса, а именно с методом опорных векторов.

12.6.1. Метод опорных векторов

Конечно, в каком-то виде вы с ним уже знакомы. Это просто линейный классификатор

$$a(x) = \text{sign}(\langle w, x \rangle - w_0),$$

использующий кусочно-линейную функцию потерь

$$L(M_i) = \max\{0, 1 - M_i\} = (1 - M_i)_+$$

и L_2-регуляризатор:

$$\sum_{i=1}^{\ell} \underbrace{L(M_i)}_{\text{Функция потерь}} + \underbrace{\gamma\|w\|^2}_{\text{Квадратичный регуляризатор}} \to \min_w.$$

Но на самом деле метод был придуман не из общего вида линейных классификаторов и не из обобщения с функциями потерь и регуляризаторами. Он был придуман из других довольно простых соображений.

12.6.2. Разделяющая полоса в случае линейно разделимой выборки

Пусть для простоты рассматривается задача бинарной классификации и некоторая линейно разделимая выборка. Выборка называется линейно разделимой, если в пространстве признаков существует такая гиперплоскость, что объекты разных классов будут находиться по разные стороны от этой плоскости.

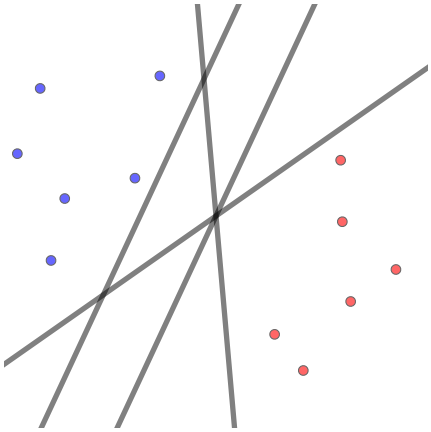

Рис. 12.8: Разделяющая гиперплоскость не единственна.

При этом гиперплоскость может быть проведена не единственным образом и возникает задача отыскания оптимальной разделяющей гиперплоскости.

Пусть разделяющая гиперплоскость существует и задается уравнением $\langle w, x\rangle - w_0 = 0$. Можно выбрать две параллельные ей и расположенные по разные стороны от нее гиперплоскости так, чтобы между ними не было объектов выборки, а расстояние между ними было максимальным. В таком случае каждая из двух получившихся граничных плоскостей будет «приставлена» к соотвествующему классу.

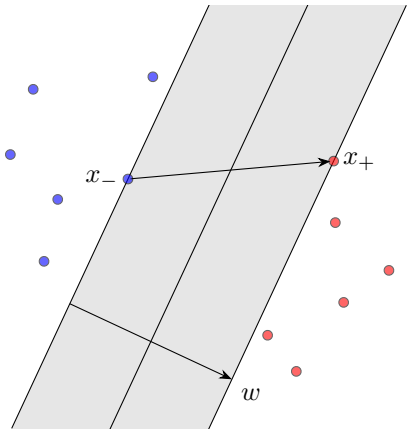

Рис. 12.9: Разделяющая полоса в случае линейно разделимой выборки

Поскольку уравнение плоскости можно умножать на ненулевое число без изменения соответствующей плоскости, всегда можно выбрать (отнормировать) w и w_0 таким образом, чтобы уравнения граничных плоскостей имели вид:

$$\langle w, x\rangle - w_0 = \pm 1.$$

Это условие нормировки можно также сформулировать следующим образом:

$$\min_{i=1,\dots,\ell} y_i(\langle w, x\rangle - w_0) = 1.$$

На каждой из двух граничных плоскостей будет лежать как минимиум один объект из соответствующего ей класса (иначе расстояние между плоскостями можно увеличить). Пусть x_+ и x_- — два таких вектора, лежащие на построенных плоскостях и принадлежащие соответствующим классам.

Тогда для ширины разделяющей полосы будет справедливо выражение (как это следует из аналитической геометрии):

$$\left\langle (x_+ - x_-), \frac{w}{\|w\|} \right\rangle = \frac{2}{\|w\|}$$

Правая часть равенства получена в предположении, что используется описанная выше нормировка.

Теперь можно поставить задачу построения такой разделяющей гиперплоскости, что расстояние между соответствующими ей граничными плоскостями будет максимальным:

$$\begin{cases} \langle w, w\rangle \to \min, \\ y_i(\langle w, x\rangle - w_0) \geq 1, \quad i = 1, \dots, \ell. \end{cases}$$

12.6.3. Случай линейно неразделимой выборки

Поскольку в случае линейно неразделимой выборки по определению любой линейный классификатор будет ошибаться, условие $y_i(\langle w, x\rangle - w_0) \geq 1$ не может быть выполнено для всех i. Естественным обобщением задачи построения оптимальной гиперплоскости на случай линейно неразделимой выборки является введение ошибок $\xi_i \geq 0$ алгоритма и штрафов за эти ошибки в минимизируемую функцию следующим образом:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell} \xi_i \to \min\limits_{w, w_0, \xi}, \\ y_i(\langle w, x\rangle - w_0) \geq 1 - \xi_i, \qquad i = 1, \dots, \ell, \\ \xi_i \geq 0, \qquad i = 1, \dots, \ell. \end{cases}$$

Множитель $1/2$ был введен для удобства, а C задает размер штрафа за ошибки.

12.6.4. Оптимизационная задача для метода опорных векторов

Получившаяся оптимизационная задача:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell} \xi_i \to \min\limits_{w, w_0, \xi}, \\ y_i(\langle w, x\rangle - w_0) \geq 1 - \xi_i, \qquad i = 1, \dots, \ell, \\ \xi_i \geq 0, \qquad i = 1, \dots, \ell \end{cases}$$

является оптимизационной задачей в методе опорных векторов (SVM) и непосредственно связана с задачей линейной классификации. Действительно, поскольку $M_i = y_i(\langle w, x\rangle - w_0)$ — отступ на i-ом объекте выборки:

$$y_i(\langle w, x\rangle - w_0) \geq 1 - \xi_i \implies \xi_i \geq 1 - M_i.$$

Учитывая также условие $\xi_i \geq 0$, можно получить:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell} \xi_i \to \min\limits_{w, w_0, \xi}, \\ \xi_i \geq \max\{0, 1 - M_i\}, \qquad i = 1, \dots, \ell, \end{cases}$$

При фиксированных w и w_0 задача оптимизации по ξ имеет следующий вид:

$$\sum_{i=1}^{\ell} \xi_i \to \min_{\xi}, \quad \text{при условии} \quad \xi_i \geq \max\{0, 1 - M_i\}, \qquad i = 1, \dots, \ell,$$

а ее решением будет $\xi_i = \max\{0, 1 - M_i\} = (1 - M_i)_+$.

Теперь можно вернуться к общей задаче минимизации и переписать ее виде:

$$Q(w, w_0) = \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}.$$

Последнее выражение называется безусловной оптимизационной задачей в SVN. В такой формулировке отчетливо видно и функцию потерь, и L_2 регуляризатор.

12.7. Ядра в методе опорных векторов (Kernel trick)

Пока ансамбли решающих деревьев не набрали своей популярности, SVM очень часто использовали даже в тех задачах, где разделяющая поверхность не похожа на линейную.

12.7.1. Добавление новых признаков и kernel trick

Чтобы применять SVN в нелинейном случае, строилось спрямляющее пространство. В основе этого лежит очень простая и очень красивая идея: если в каком-то исходном пространстве признаков классы не являются линейно разделимыми, то может быть можно отобразить это пространство признаков в какое-то новое, в котором классы уже будут линейно разделимы.

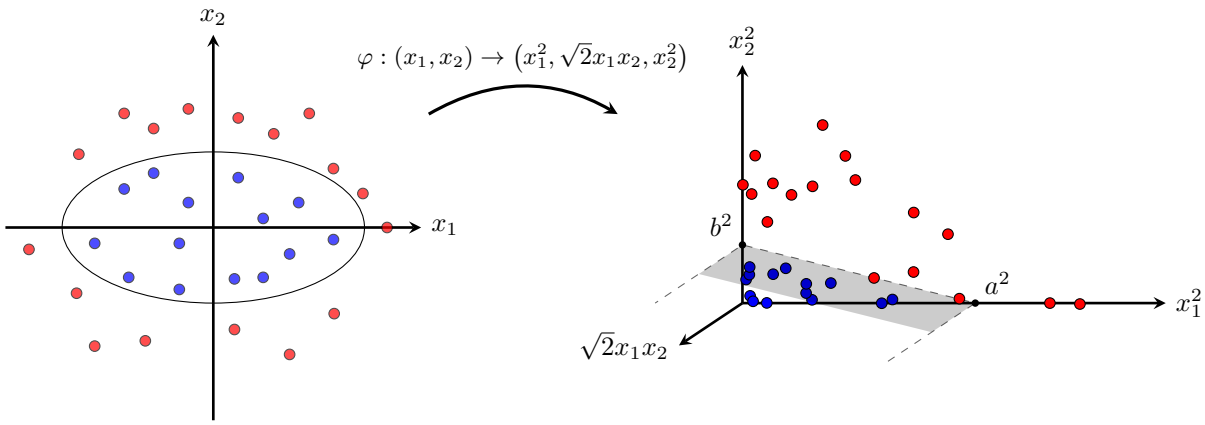

Рис. 12.10: Пример построения спрямляющего пространства.

Не обязательно задавать это отображение явно, так как в SVM везде фигурирует только скалярное произведение вида $\langle w, x\rangle$.

Пусть $\varphi(x)$ — спрямляющее отображение, тогда, чтобы записать SVM в спрямляющем пространстве, необходимо во всех формулах сделать следующие подстановки:

$$x \to \varphi(x), \quad w \to \varphi(w), \quad \langle w, x\rangle \to \langle \varphi(w), \varphi(x)\rangle.$$

Тогда метод SVM может быть сформулирован в исходном пространстве, если в качестве скалярного произведения использовать, возможно, нелинейную симметричную функцию

$$K(w, x) = \langle \varphi(w), \varphi(x)\rangle$$

и таким образом получать нелинейную разделяющую поверхность. Эта идея называется в англоязычной литературе kernel trick.

12.7.2. Линейное ядро

В простейшем случае ядро совпадает со скалярным произведением:

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

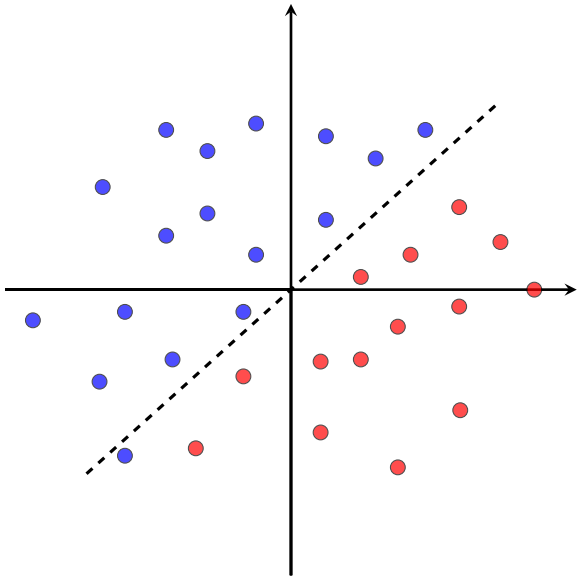

Рис. 12.11: Линейное ядро

Следует отметить, что линейное ядро в некоторых задачах — самый лучший выбор, например в задачах классификации текстов, и не стоит выкидывать его из рассмотрения.

12.7.3. Полиномиальное ядро

Другой пример — это полиномиальное ядро:

$$K(w, x) = (\langle w, x \rangle + r)^d.$$

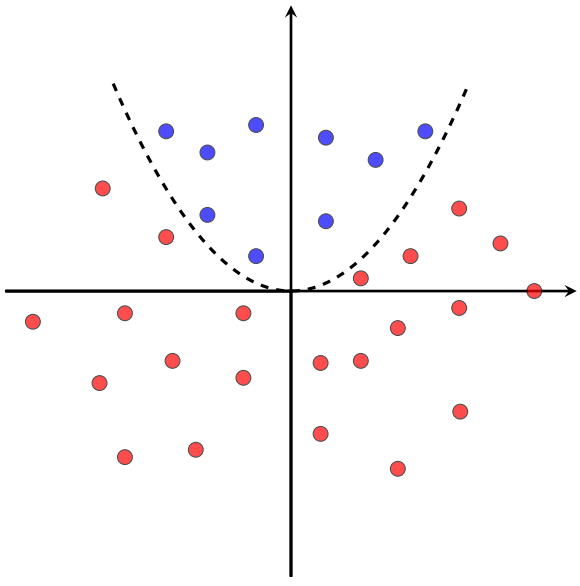

Рис. 12.12: Полиномиальное ядро

Полиномиальное ядро получается, если в качестве спрямляемого пространство выступает пространство многочленов не выше определенной степени.

12.7.4. Радиальное ядро

И другое часто используемое ядро — это радиальное ядро:

$$K(w, x) = \exp\left(-\gamma \|w - x\|^2\right).$$

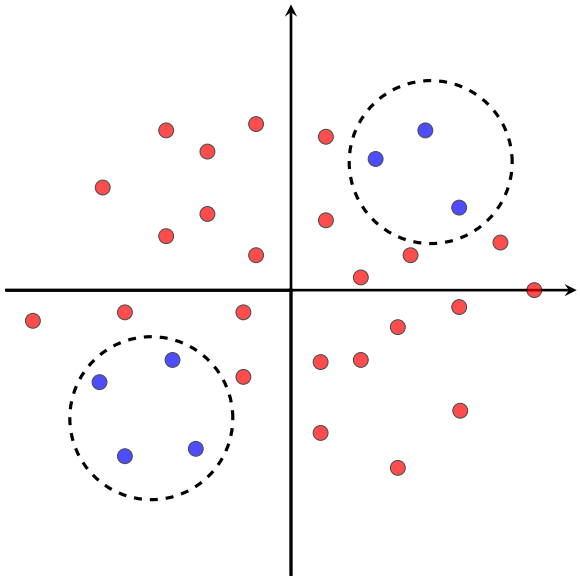

Рис. 12.13: Радиальное ядро

Поскольку радиальное ядро выражается через евклидово расстояние, будут проявляться основные проблемы метрических алгоритмов, в том числе проклятие размерности. Именно поэтому не стоит применять это ядро, если признаков действительно очень-очень много.

Но так или иначе, оно позволяет строить очень сложные границы классов. Спрямляющее пространство, которое соответствует данному ядру, является бесконечномерным.

12.7.5. Ядра и библиотеки

SVM реализован в различных библиотеках: какие-то (например LibSVM) поддерживают различные ядра, но некоторые (например LibLinear) — только линейное ядро. Причем, если на практике потребуется применить SVM с линейным ядром, лучше использовать LibLinear, который лучше оптимизирован для вычислений с линейным ядром, а не LibSVM с указанием линейного ядра.

Scikit-learn же просто предоставляет удобный доступ к LibSVM и LibLinear, поэтому все сказанное выше применимо и здесь: LinearSVC поддерживает только линейное ядро и оптимизирован для этого случая, а SVC поддерживает различные ядра.