

Математические методы машинного обучения

Кафедра Машинного Обучения и
Цифровой Гуманистике

Оглавление

Глава 1

Общие термины и обозначения

1.1. Модели зависимости и обучения

Предполагается, что имеется пространство X признаковых описаний исследуемых объектов, и пространство Y ответов. Также существует неизвестная целевая зависимость $y: X \rightarrow Y$, значения которой (в случае обучения с учителем) известны только на некоторых элементах $x = (x_1, \dots, x_m) \in X^m$. Требуется восстановить эту неизвестную зависимость (а точнее, приблизить её как можно точнее).

Чтобы восстановить зависимость, вводят некоторое семейство W параметров (весов в линейном случае) и функцию $f: X \times W \rightarrow Y$, которая по параметрам восстанавливает зависимость y . Для всякого $w \in W$ функция $a_w: X \rightarrow Y$, определённая как $a_w(x) = f(w, x)$, называется моделью зависимости.

Нашей задачей теперь является найти такую $w^* \in W$, что функция a_{w^*} хорошо приближает зависимость y .

1.2. Функция потерь и функционал качества

Чтобы формализовать понятие «хорошо приближает», вводят т.н. функцию потерь $L: Y^2 \rightarrow \mathbb{R}$, которая является неотрицательно определённой симметричной формой. Типичные примеры:

- $L(y^*, y) = [y^* \neq y]$ для задач классификации,
- $L(y^*, y) = (y^* - y)^2$ для задач регрессии.

Функционал качества $Q: Y^X \times X^m \times Y^m$ характеризует средние потери и определяется как

$$Q(a, x, y) = \frac{1}{m} \sum_{i=1}^m L(a(x_i), y_i).$$

В терминах теории вероятностей это значение называют эмпирическим риском.

Принцип минимизации эмпирического риска утверждает, что для решения поставленной задачи нужно найти

$$w^* = \arg \min_{w \in W} Q(a_w, x, y).$$

Задача 1

Сформулируйте задачу минимизации эмпирического риска в терминах метода максимума правдоподобия.

Задача 2

Сведите задачу с функцией потерь MAPE к задаче с функцией МАЕ, но с относительными весами.

Задача 3

Допустим, мы решаем задачу проведения разделяющей гиперплоскости, но в аффинном пространстве. Как будет выглядеть функция потерь в таком случае?

Переобучение. Борьба с переобучением

Переобучение (overfitting) - это явление, при котором модель машинного обучения показывает существенно лучшие результаты на обучающих данных по сравнению с тестовыми. Формально переобученность определяется как разность между ошибками на контрольной и обучающей выборках:

$$\delta(\mu, X^l, X^k) = \nu(\mu(X^l), X^k) - \nu(\mu(X^l), X^l)$$

где:

- μ - метод обучения
- X^l - обучающая выборка
- X^k - контрольная выборка
- ν - частота ошибок

О наличии переобучения говорят, когда $\delta > \varepsilon$ для некоторого порогового значения ε .

Признаки переобучения

1. Большая разница между качеством работы на обучающей и тестовой/валидационных выборках.
2. Продолжение улучшения качества на обучении при ухудшении на валидации в процессе обучения.
3. Слишком много параметров модели
4. Слишком большие значения весов
5. Избыточное количество признаков

Основные методы борьбы с переобучением

1. **Регуляризация** - добавление штрафа за сложность модели:

L1-регуляризация:

$$J(w) = L(w) + \lambda \sum_{i=1}^n |w_i|$$

L2-регуляризация:

$$J(w) = L(w) + \lambda \sum_{i=1}^n w_i^2$$

где $L(w)$ - исходная функция потерь, λ - коэффициент регуляризации.

2. **Использование кросс-валидации** Кросс-валидация - оценка качества и подбор гиперпараметров на разных разбиениях выборки:

- K-fold: Выборка разбивается на k равных частей. На каждой итерации одна часть используется для валидации, остальные k-1 частей - для обучения. Итоговая оценка усредняется по всем разбиениям.
- Leave-one-out (LOO): Частный случай k-fold CV при k = n, где n - размер выборки. На каждой итерации только один объект используется для валидации. Этот метод дает несмещенную оценку ошибки, но требует большого числа итераций обучения.

3. **Уменьшение сложности модели:**

- Сокращение числа параметров
- Отбор значимых признаков
- Упрощение архитектуры

4. **Увеличение объема обучающей выборки**

Оценка эффективности борьбы с переобучением

1. Мониторинг динамики ошибок на обучающей и валидационной выборках.
2. Расчет величины переобученности δ .
3. Оценка значимости признаков и параметров модели.
4. Анализ сложности модели относительно размера выборки.

Задача 1

В процессе обучения нейронной сети для задачи бинарной классификации получены следующие значения ошибок по эпохам обучения:

Эпоха	Ошибка на обучении	Ошибка на валидации
1	0.40	0.42
5	0.25	0.30
10	0.15	0.25
15	0.08	0.28
20	0.03	0.35
25	0.01	0.45

Определите оптимальное количество эпох обучения для минимизации эффекта переобучения и обоснуйте свой выбор. Рассчитайте величину переобученности δ для последней эпохи.

Решение: Для определения оптимального количества эпох проанализируем динамику ошибок:

1. До 10-й эпохи обе ошибки монотонно убывают, что говорит о нормальном процессе обучения.
2. После 10-й эпохи ошибка на обучении продолжает уменьшаться, но ошибка на валидации начинает расти, что является явным признаком переобучения.
3. Оптимальное количество эпох - 10, так как при дальнейшем обучении начинается переобучение

Величина переобученности на последней эпохе:

$$\delta = \nu(X^k) - \nu(X^l) = 0.45 - 0.01 = 0.44$$

Ответ: 10 эпох; $\delta = 0.44$

Задача 2

Дана выборка из 8 объектов с целевой переменной $y \in \{0, 1\}$ и одним признаком x :

$$(x_1 = 1, y_1 = 0), (x_2 = 2, y_2 = 0), (x_3 = 3, y_3 = 1), (x_4 = 4, y_4 = 1), \\ (x_5 = 5, y_5 = 1), (x_6 = 6, y_6 = 1), (x_7 = 7, y_7 = 0), (x_8 = 8, y_8 = 0)$$

Модель представляет собой пороговое правило вида:

$$a(x) = [x \geq t]$$

где t - порог, [...] - индикатор.

Используя метод полного скользящего контроля (leave-one-out cross-validation), найдите оптимальное значение порога t , минимизирующее среднюю ошибку на контроле.

Решение: 1) При leave-one-out каждый объект по очереди становится контрольным, а остальные - обучающими.

2) Возможные значения порога t следует искать между соседними значениями признака x , принадлежащими разным классам. Таким образом, получаем множество потенциальных порогов: $t \in \{1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5\}$

3) Рассмотрим подробно процесс для порога $t = 2.5$. Запишем пороговое правило:

$$a(x) = [x \geq 2.5]$$

Для первого объекта ($x_1 = 1, y_1 = 0$):

$$a(1) = [1 \geq 2.5] = 0$$

Правильный ответ $y_1 = 0$, следовательно ошибка на первом объекте:

$$e_1 = |y_1 - a(x_1)| = |0 - 0| = 0$$

Для второго объекта ($x_2 = 2, y_2 = 0$):

$$a(2) = [2 \geq 2.5] = 0$$

$$e_2 = |y_2 - a(x_2)| = |0 - 0| = 0$$

Для третьего объекта ($x_3 = 3, y_3 = 1$):

$$a(3) = [3 \geq 2.5] = 1$$

$$e_3 = |y_3 - a(x_3)| = |1 - 1| = 0$$

Аналогично для остальных объектов получаем вектор ошибок:

$$e = (0, 0, 0, 0, 0, 0, 1, 1)$$

Средняя ошибка для порога $t = 3.5$ равна:

$$\text{err}(2.5) = \frac{1}{8} \sum_{i=1}^8 e_i = \frac{0 + 0 + 0 + 0 + 0 + 0 + 1 + 1}{8} = \frac{2}{8} = 0.25$$

Проведем аналогичные вычисления для остальных порогов и получим:

Значение порога t	Средняя ошибка $\text{err}(t)$	Количество ошибок
1.5	0.375	3
2.5	0.250	2
3.5	0.375	3
4.5	0.500	4
5.5	0.625	5
6.5	0.750	6
7.5	0.625	5

Ответ: Оптимальное значение порога $t = 2.5$, что дает среднюю ошибку 0.25.

Задача 3

При обучении модели классификации изображений исследователь записывал значения accuracy каждые 3 эпохи для двух различных архитектур нейронных сетей (A и B). Размер обучающей выборки: 500 изображений.

Эпоха	Модель А		Модель В	
	Обучение	Валидация	Обучение	Валидация
3	0.65	0.62	0.70	0.68
6	0.75	0.70	0.82	0.75
9	0.85	0.72	0.95	0.69
12	0.95	0.68	0.98	0.63

Требуется:

Определить, какая из моделей показывает более сильное переобучение. Обосновать ответ. Рассчитать величину переобученности δ для обеих моделей на последней эпохе. Для модели с более сильным переобучением указать оптимальное количество эпох обучения.

Решение: Давайте проанализируем поведение каждой модели. Для модели A: На первых эпохах наблюдается нормальное обучение - точность растет как на обучающей, так и на валидационной выборке. После 6-й эпохи начинается расхождение: точность на обучении продолжает расти, а на валидации начинает падать. К 12-й эпохе разрыв становится существенным. Для модели B: Начальное обучение идет быстрее, чем у модели A. Однако уже после 6-й эпохи наблюдается резкое расхождение между обучающей и валидационной точностью. К 9-й эпохе точность на обучении достигает 0.95, но на валидации падает до 0.69. Рассчитаем величину переобученности δ на последней эпохе:

$$\text{Модель A: } \delta_A = 0.68 - 0.95 = -0.27 \quad \text{Модель B: } \delta_B = 0.63 - 0.98 = -0.35$$

Модель B показывает более сильное переобучение, потому что:

Больше абсолютная величина δ на последней эпохе (-0.35 против -0.27) Более быстрая скорость расхождения метрик после 6-й эпохи Более высокая точность на обучении при более низкой точности на валидации

Для модели В оптимальным является остановка на 6-й эпохе, так как:

Достигнута хорошая точность на валидации (0.75) Разрыв между обучением и валидацией еще не критический После этой эпохи начинается явное переобучение

Ответ: Модель В показывает более сильное переобучение ($\delta_B = -0.35$), в то время как ($\delta_A = -0.27$). Оптимальное количество эпох для модели В - 6 эпох, что обеспечивает наилучшее качество на валидации (0.75) без явного переобучения.

1.3. Линейные модели классификации и регрессии

Обе линейные модели описываются следующим образом. На пространстве X объектов заданы числовые признаки $f_1, \dots, f_n: X \rightarrow \mathbb{R}$. По заданному вектору $w \in \mathbb{R}^n$ весов определяются предсказания объектов

$$a(x, w) := \sum_{i=1}^n w_i f_i(x) = w^T f(x)$$

в задаче регрессии; в задаче классификации от этого выражения берётся знак. При фиксированной (но неизвестной) таргет-функции $y: X \rightarrow Y$ определяется функция потерь $L(x, w)$, которая в задаче классификации равна индикатору того, что $a(x, w) \neq y(x)$, а в задаче регрессии равна

$$L(x, w) = |a(x, w) - y(x)| \quad \text{или} \quad L(x, w) = |a(x, w) - y(x)|^2,$$

или любому другому симметричному неотрицательному функционалу. Далее в обеих моделях нужно решить задачу оптимизации

$$\sum_{i=1}^N L(x_i, w) \rightarrow \min_w,$$

где x_i суть элементы выборки размера N . То есть, неформально, нужно придумать линейную модель, которая по $x \in X$ восстанавливает $y(x) \in Y$.

1.4. Метод наименьших квадратов

В методе наименьших квадратов в задаче линейной регрессии необходимо решить задачу минимизации функционала

$$\|w^T f(X) - Y\|_2^2 \rightarrow \min_w,$$

где Y это вектор из таргетов $y_i = y(X_i)$, X это вектор наблюдений, а $f(X)$ это матрица признаков размера $n \times N$. Из линейной алгебры известно, что решением задачи минимизации является вектор

$$\hat{w} = (f(X)f(X)^T)^{-1}f(X)Y.$$

Известно, что он является несмешённой оценкой параметра w , и, кроме того, наилучшей оценкой этого параметра в классе всех линейных оценок вида $A(X)Y$ с квадратичной функцией потерь.

В гауссовской линейной модели дополнительно известно, что

$$Y \sim N(w^T f(X), \sigma^2 I_{N \times N}).$$

Тогда оценка \hat{w} является оптимальной оценкой вектора w в среднеквадратичном подходе в классе несмешённых оценок.

Задача 1

Дан набор точек в \mathbb{R}^3 , отмеченных ± 1 . Необходимо описать задачу проведения разделяющей плоскости, проходящей через начало координат, как задачу линейной регрессии.

Задача 2

В задаче линейной регрессии получить несмешённую оценку ошибки σ^2 .

Задача 3

В гауссовской линейной модели найдите оптимальную оценку параметра $w_1 + \dots + w_n$ и её распределение.

1.5. Скользящий контроль

Скользящий контроль (или кросс-проверка, кросс-валидация, англ. cross-validation, CV) — это метод эмпирической оценки обобщающей способности алгоритмов, когда они обучаются на примерах из данных.

Метод основан на использовании некоторого числа разбиений исходной выборки на два подмножества: обучающей и контрольной подвыборок. Для каждого разбиения выполняется обучение алгоритма на обучающей подвыборке, а затем рассчитывается средняя ошибка на контрольной подвыборке. Итоговой оценкой скользящего контроля является среднее значение ошибки по всем контрольным подвыборкам.

При условии независимости выборки средняя ошибка кросс-валидации даёт несмещённую оценку вероятности ошибки. Это преимущество выделяет её по сравнению со средней ошибкой на обучающей выборке, которая может быть смещена (занижена) из-за эффекта переобучения.

Скользящий контроль является стандартным методом для тестирования и сравнения алгоритмов классификации, регрессии и прогнозирования.

1.6. Определения и обозначения

Рассмотрим задачу обучения с учителем.

Пусть X — множество, содержащее описания объектов, а Y — множество возможных ответов.

Предположим, что имеется конечная выборка прецедентов $X^L = \{(x_i, y_i)\}_{i=1}^L \subset X \times Y$.

Задан алгоритм обучения — отображение μ , которое сопоставляет произвольной конечной выборке прецедентов X^m некоторую функцию (алгоритм) $a : X \rightarrow Y$.

Качество алгоритма a оценивается по произвольной выборке прецедентов X^m с использованием функционала качества $Q(a, X^m)$. Для процедуры скользящего контроля способ вычисления данного функционала может варьироваться, однако обычно он аддитивен по элементам выборки:

$$Q(a, X^m) = \frac{1}{m} \sum_{x_i \in X^m} \mathcal{L}(a(x_i), y_i),$$

где $\mathcal{L}(a(x_i), y_i)$ — неотрицательная функция потерь, показывающая величину ошибки алгоритма $a(x_i)$ при истинном ответе y_i .

1.6.1. Процедура скользящего контроля

Рассмотрим выборку X^L , которая разбивается на N различных способов на две непересекающиеся подвыборки: $X^L = X_n^m \cup X_n^k$, где X_n^m — обучающая подвыборка размера m , а X_n^k — контрольная подвыборка длины $k = L - m$, при этом $n = 1, \dots, N$ обозначает номер разбиения.

Для каждого разбиения n рассчитывается алгоритм $a_n = \mu(X_n^m)$ и вычисляется значение функционала качества $Q_n = Q(a_n, X_n^k)$. Среднее арифметическое значений Q_n по всем разбиениям называют оценкой по методу скользящего контроля:

$$CV(\mu, X^L) = \frac{1}{N} \sum_{n=1}^N Q(\mu(X_n^m), X_n^k).$$

Разные методы скользящего контроля отличаются как видами функционала качества, так и способами разбиения выборки.

1.6.2. Доверительное оценивание

Кроме среднего значения качества на контроле, строятся также доверительные интервалы.

Непараметрическая оценка доверительного интервала.

Строится вариационный ряд значений $Q_n = Q(a_n, X_n^k)$, где $n = 1, \dots, N$:

$$Q^{(1)} \leq Q^{(2)} \leq \dots \leq Q^{(N)}.$$

Утверждение 1. Если разбиения осуществлялись случайно, независимо и равновероятно, то с вероятностью $\eta = \frac{t}{N+1}$ значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N-t+1)}$.

Следствие 1. Значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N)}$ с вероятностью $\eta = \frac{1}{N+1}$.

В частности, для получения верхней оценки с надёжностью 95% достаточно взять $N = 20$ разбиений.

Утверждение 2. Если разбиения осуществлялись случайно, независимо и равновероятно, то с вероятностью $\eta = \frac{2t}{N+1}$ значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы доверительного интервала $[Q^{(t)}, Q^{(N-t+1)}]$.

Следствие 2. Значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы вариационного ряда $[Q^{(1)}, Q^{(N)}]$ с вероятностью $\eta = \frac{2}{N+1}$.

В частности, для получения двусторонней оценки с надёжностью 95% достаточно взять $N = 40$ разбиений.

Параметрические оценки доверительного интервала основываются на априорном предположении о виде распределения случайной величины $Q(a(X^m), X^k)$. Если априорные предположения не выполняются, доверительный интервал может оказаться сильно смещённым. В частности, если предположения о нормальности распределения не выполнены, то нельзя использовать стандартное «правило двух сигм» или «трёх сигм». Джон Лангфорд в своей диссертации [Langford2002] указывает на распространённую ошибку, когда правило двух сигм применяется к функционалу частоты ошибок, имеющему на самом деле биномиальное распределение. Однако биномиальным распределением в общем случае тоже нельзя пользоваться, поскольку в результате обучения по случайным подвыборкам X^m вероятность ошибки алгоритма $a(X^m)$ оказывается случайной величиной. Следовательно, случайная величина $Q(a(X^m), X^k)$ описывается не биномиальным распределением, а (неизвестной) смесью биномиальных распределений. Аппроксимация смеси биномиальным распределением может приводить к ошибочному сужению доверительного интервала. Приведённые выше непараметрические оценки лишены этого недостатка.

1.6.3. Стратификация

Стратификация выборки представляет собой метод, направленный на уменьшение разброса (дисперсии) оценок, получаемых при скользящем контроле, что позволяет получить более узкие доверительные интервалы и более точные верхние оценки.

Процесс стратификации заключается в том, чтобы заранее разделить выборку на несколько частей (страты), и при разбиении на обучающую выборку длины m и контрольную выборку длины k обеспечить, чтобы каждая страта была представлена в обучении и контроле в одинаковых пропорциях $m : k$.

Стратификация классов в задачах классификации означает, что каждый класс делится между обучением и контролем в пропорции $m : k$.

Стратификация по вещественному признаку осуществляется следующим образом: объекты выборки сортируются по какому-либо критерию, например, по возрастанию значения одного из признаков. Затем выборка разбивается на k последовательных страт одинаковой длины (с точностью до 1). При формировании контрольных выборок из каждой страты выбирается по одному объекту — либо с заданным порядковым номером внутри страты, либо случайным образом.

1.7. Разновидности скользящего контроля

Существует несколько различных методов скользящего контроля, которые могут отличаться по способу разбиения выборки.

1.7.1. Полный скользящий контроль (complete CV)

Оценка скользящего контроля строится по всем возможным $N = C_L^k$ разбиениям. В зависимости от длины обучающей выборки k различают следующие варианты:

- **Частный случай при $k = 1$ — контроль по отдельным объектам (leave-one-out CV)**

Как было показано, контроль по отдельным объектам является асимптотически оптимальным при определённых условиях, а именно:

$$\frac{L_n(\hat{m})}{\inf_{m \in M_n} L_n(m)} \rightarrow 1 \quad \text{по вероятности,}$$

где:

- M_n — класс сравниваемых моделей;
- $L_n(m)$ — среднеквадратичная ошибка при выборе m -ой модели;

$$- \hat{m} = \arg \min_{m \in M_n} \text{CV}(m).$$

- **Общий случай при $k > 2$**

В этом случае число разбиений $N = C_L^k$ становится очень большим, даже для сравнительно малых значений k , что делает практическое применение данного метода затруднительным. Для такого случая полный скользящий контроль используется либо в теоретических исследованиях [Voronov2004], либо в редких ситуациях, когда удаётся вывести эффективную вычислительную формулу. Например, для метода k ближайших соседей существует такая формула, которая позволяет эффективно выбирать параметр k .

На практике чаще применяются другие варианты скользящего контроля.

1.7.2. Случайные разбиения

Разбиения $n = 1, \dots, N$ выбираются случайным образом, независимо и с одинаковой вероятностью из множества всех возможных C_L^k разбиений. Именно для такого случая верны приведённые выше оценки доверительных интервалов. На практике эти оценки обычно применяются без изменений и к другим методам разбиения выборки.

1.7.3. Контроль на отложенных данных (hold-out CV)

Оценка модели с использованием метода скользящего контроля основана на одном случайному разбиении выборки, где $N = 1$.

Однако этот метод имеет несколько значительных недостатков:

1. Часто приходится оставлять слишком много объектов в контрольной подвыборке, что приводит к сокращению обучающей выборки. Это уменьшение объема обучающей выборки вызывает смещение оценки (пессимистически завышенную вероятность ошибки).
2. Оценка модели значительно зависит от конкретного разбиения данных, в то время как более предпочтительно, чтобы она характеризовала исключительно алгоритм обучения, а не случайность разбиения.
3. Дисперсия оценки может быть высокой. Она может быть снижена путём усреднения оценок по нескольким разбиениям.

Важно различать два типа контроля по отложенным данным и по тестовой выборке:

- **Контроль по отложенными данным (Hold-out)** — оценка вероятности ошибки производится для классификатора, построенного по всей выборке.

- **Контроль по тестовой выборке (Test-set)** — оценка вероятности ошибки вычисляется для классификатора, обученного на обучающей подвыборке.

1.7.4. Контроль по отдельным объектам (leave-one-out CV)

Метод leave-one-out (LOO) является частным случаем полной кросс-валидации с использованием скользящего контроля при $k = 1$, что означает $N = L$, где L — количество объектов в выборке. Это один из наиболее популярных вариантов скользящей кросс-валидации.

Основное преимущество LOO заключается в том, что каждый объект участвует в процессе контроля ровно один раз, при этом размер обучающих подвыборок лишь на единицу меньше общей выборки.

Однако метод LOO имеет и ряд недостатков, основным из которых является высокая вычислительная нагрузка, поскольку для каждого объекта выборки требуется провести обучение модели заново. В некоторых случаях, когда используемые алгоритмы обучения позволяют быстро адаптировать внутренние параметры при замене одного объекта на другой, процесс LOO можно значительно ускорить.

1.7.5. Контроль по q блокам (q -fold CV)

В методе q -fold кросс-валидации выборка случайным образом делится на q непересекающихся блоков, каждый из которых имеет одинаковую (или почти одинаковую) длину k_1, \dots, k_q :

$$X^L = X_1^{k_1} \cup \dots \cup X_q^{k_q}, \quad k_1 + \dots + k_q = L.$$

Каждый блок поочередно используется в качестве контрольной подвыборки, а обучение модели проводится на оставшихся $q - 1$ блоках. Критерий ошибки определяется как среднее значение ошибки на контрольной подвыборке:

$$CV(\mu, X^L) = \frac{1}{q} \sum_{n=1}^q Q\left(\mu\left(X^L \setminus X_n^{k_n}\right), X_n^{k_n}\right),$$

где Q — функция потерь. Этот метод представляет собой компромисс между подходами LOO, hold-out и случайными разбиениями. С одной стороны, обучение проводится только q раз, а не L . С другой стороны, длина обучающих подвыборок, равная $L \frac{q-1}{q}$ с точностью до округления, практически не отличается от длины всей выборки L . Обычно выборку делят случайным образом на 10 или 20 блоков.

1.7.6. Контроль по $r \times q$ блокам ($r \times q$ -fold CV)

Контроль с использованием $r \times q$ -кратного разбиения (или $r \times q$ -fold кросс-валидации) представляет собой метод, при котором процедура q -кратной кросс-

валидации повторяется r раз. В каждом повторении данные случайным образом делятся на q непересекающихся блоков, обеспечивая полное покрытие выборки. Этот метод сохраняет все преимущества стандартной q -кратной кросс-валидации, добавляя при этом гибкость в выборе количества разбиений.

Данный подход стратифицированного скользящего контроля считается одной из базовых методик для оценки и сравнения производительности алгоритмов классификации. Он широко используется в таких системах, как WEKA и «Полигон алгоритмов».

Метод скользящего контроля применяется также в задачах прогнозирования.

1.8. Скользящий контроль в задачах прогнозирования

В задачах прогнозирования, динамического обучения, обучения с подкреплением и активного обучения примеры часто линейно упорядочены по времени их появления. В таких случаях возможности применения различных вариантов скользящего контроля ограничены.

1.8.1. Контроль с нарастающей длиной обучения

Предполагается, что обучающая подвыборка включает все предыдущие наблюдения: $X_n^m = \{x_1, \dots, x_n\}$. Контрольная подвыборка состоит из последующих наблюдений: $X_n^k = \{x_{n+\delta+1}, \dots, x_{n+\delta+k}\}$, где $\delta \geq 0$ — величина задержки прогноза (как правило, $\delta = 0$). Момент «текущего времени» n перемещается по выборке:

$$CV(\mu, X^L) = \frac{1}{T_2 - T_1 + 1} \sum_{n=T_1}^{T_2} Q(\mu(X_n^m), X_n^k),$$

где T_1 — минимальная длина обучающей выборки, необходимая для корректной работы алгоритма обучения μ , $T_2 = L - \delta - k$.

Так как длина обучающей подвыборки $m = n$ увеличивается со временем, точность прогнозов алгоритма может постепенно возрастать. Этот эффект может быть нежелательным, если цель скользящего контроля — объективная оценка качества алгоритма обучения.

1.8.2. Контроль с фиксированной длиной обучения

Отличается от метода с нарастающей длиной тем, что размер обучающей подвыборки m остаётся постоянным, включающим только последние m примеров временного ряда: $X_n^m = \{x_{n-m+1}, \dots, x_n\}$. В этом случае минимальная длина обучающей выборки принимается равной $T_1 = m$.

1.9. Недостатки скользящего контроля

- При использовании скользящего контроля обучение выполняется N раз, что связано с высокими вычислительными затратами.
- Оценка скользящего контроля предполагает наличие заранее заданного алгоритма обучения μ , но она не раскрывает, какими свойствами должны обладать "хорошие" алгоритмы обучения и как их можно построить. В этом смысле теоретические оценки обобщающей способности могут давать полезные рекомендации.
- Попытка применить скользящий контроль как критерий оптимизации в процессе обучения приводит к утрате его несмешённости, что увеличивает риск переобучения.
- Скользящий контроль предоставляет несмешённую точечную оценку риска, но не интервальную. На данный момент нет методов, которые позволяли бы на основе скользящего контроля строить точные доверительные интервалы для риска, то есть для математического ожидания потерь (включая вероятность ошибочной классификации).

1.10. Применение скользящего контроля

Скользящий контроль широко используется на практике для настройки некоторых ключевых параметров, которые, как правило, определяют структуру или сложность модели алгоритма и имеют ограниченное количество возможных значений.

Примеры применения:

- Выбор модели из ограниченного набора альтернативных алгоритмов.
- Оптимизация параметра регуляризации, включая:
 - параметр регуляризации в гребневой регрессии;
 - параметр весового затухания (weight decay) в нейронных сетях;
 - параметр C в методе опорных векторов.
- Настройка ширины окна в методах:
 - парзеновского окна;
 - ближайшего соседа;
 - ядерного сглаживания.

- Оптимизация количества нейронов в скрытом слое многослойной нейронной сети.
- Выбор информативного набора признаков.
- Сокращение решающего дерева.
- Минимизация структурного риска.

Список литературы

1. Воронцов К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов. — Математические вопросы кибернетики. М.: Физматлит, 2004.
2. Эфрон Б. Нетрадиционные методы многомерного статистического анализа. — М: Финансы и статистика. — 1988.
3. Langford J. Quantitatively Tight Sample Complexity Bounds. — Carnegie Mellon Thesis. — 2002. — 124 с.
4. Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. — 14th International Joint Conference on Artificial Intelligence, Palais de Congres Montreal, Quebec, Canada. — 1995. — С. 1137-1145.
5. Mullin M., Sukthankar R. Complete Cross-Validation for Nearest Neighbor Classifiers. — Proceedings of International Conference on Machine Learning. — 2000. — С. 1137-1145.

Задача: Cross-Validation (LOOCV)

У вас есть набор данных, состоящий из 6 объектов:

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6)\},$$

где y — целевая переменная, которая может принимать значение 0 или 1.

Простая линейная модель для предсказания y задана как:

$$y = a \cdot x + b,$$

где параметры a и b нужно подобрать с помощью обучения на обучающей выборке.

Требуется:

1. Используйте **leave-one-out cross-validation (LOOCV)** для оценки качества модели.
2. Для каждой итерации используйте метод наименьших квадратов для подбора параметров a и b по формулам:

$$a = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \quad b = \bar{y} - a \cdot \bar{x}.$$

3. Проведите LOOCV:

- На каждой итерации оставьте одно наблюдение для тестирования и обучайте модель на оставшихся данных.

- Рассчитайте параметры a и b для каждой подвыборки.
 - Используя найденные параметры, предскажите y для отложенного объекта.
4. Рассчитайте среднеквадратическую ошибку (MSE) на тестовых объектах:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

Данные:

Таблица с наблюдениями:

Объект i	x_i	y_i
1	1	0
2	2	0
3	3	1
4	4	1
5	5	1
6	6	0

Решение задачи LOOCV

1. Процедура LOOCV:

- На каждой итерации исключаем одно наблюдение из выборки для тестирования.
- Обучаем модель на оставшихся наблюдениях.
- Рассчитываем параметры линейной модели по формулам:

$$a = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \quad b = \bar{y} - a \cdot \bar{x},$$

где \bar{x} и \bar{y} — средние значения x и y на обучающей выборке.

- Предсказываем y для тестового объекта:

$$\hat{y} = a \cdot x + b.$$

2. Итерации:

Рассмотрим все 6 итераций:

- **Итерация 1:** Тестовый объект ($x_1 = 1, y_1 = 0$)
 - Обучающая выборка: $(x, y) = \{(2, 0), (3, 1), (4, 1), (5, 1), (6, 0)\}$.
 - Найденные параметры: a_1, b_1 .
 - Предсказание: \hat{y}_1 .
 - Ошибка: $(y_1 - \hat{y}_1)^2$.
- **Итерация 2:** Тестовый объект ($x_2 = 2, y_2 = 0$)
 - Обучающая выборка: $(x, y) = \{(1, 0), (3, 1), (4, 1), (5, 1), (6, 0)\}$.
 - Найденные параметры: a_2, b_2 .
 - Предсказание: \hat{y}_2 .
 - Ошибка: $(y_2 - \hat{y}_2)^2$.
- Аналогично повторяем для всех объектов $i = 3, 4, 5, 6$.

3. Среднеквадратическая ошибка:

Рассчитываем MSE:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

где $n = 6$.

4. Результат:

$$\text{MSE} \approx 0.653$$

Задача: Кросс-валидация для классификации (5-Fold)

Вам дан набор данных из 10 объектов:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_{10}, y_{10})\},$$

где x_i — признак (вещественное число), а y_i — метка класса (0 или 1).

Модель классификации представляет собой простое пороговое правило:

$$\hat{y} = \begin{cases} 1, & \text{если } x \geq t, \\ 0, & \text{если } x < t, \end{cases}$$

где t — порог, который нужно определить.

Требуется:

1. Разбейте данные на 5 фолдов для **5-Fold Cross-Validation**.
2. Для каждого фолда:
 - Используйте 4 фолда для обучения (подберите оптимальный порог t) и 1 фолд для тестирования.
 - Подберите t , чтобы минимизировать число ошибок классификации (ошибок предсказания).
3. Рассчитайте среднюю точность (accuracy) на тестовых фолдах:

$$\text{Accuracy} = \frac{\text{Число верных предсказаний}}{\text{Общее число объектов}}.$$

Данные:

Таблица с наблюдениями:

Объект i	x_i	y_i
1	1.2	0
2	2.3	0
3	2.8	0
4	3.4	1
5	4.1	1
6	4.8	1
7	5.5	1
8	6.0	1
9	6.7	0
10	7.5	0

Упрощение для решения:

- Разделите данные по порядку: первые 2 объекта в первый фолд, следующие 2 — во второй, и так далее.

- Для каждого обучающего фолда подберите порог t , перебрав средние значения между соседними объектами x_i .
- Для упрощения расчетов: оптимизация t и проверка классификации выполняются с помощью простых сравнений.

Решение задачи: Кросс-валидация для классификации (5-Fold)

Шаг 1: Разбиение данных на 5 фолдов

Каждый фолд состоит из двух объектов. Разбиение:

$$\begin{aligned} \text{Фолд 1: } & \{(1.2, 0), (2.3, 0)\}, \\ \text{Фолд 2: } & \{(2.8, 0), (3.4, 1)\}, \\ \text{Фолд 3: } & \{(4.1, 1), (4.8, 1)\}, \\ \text{Фолд 4: } & \{(5.5, 1), (6.0, 1)\}, \\ \text{Фолд 5: } & \{(6.7, 0), (7.5, 0)\}. \end{aligned}$$

Шаг 2: Выбор оптимального порога t для каждого фолда

Для обучения используется объединение 4 фолдов, а пятый фолд служит тестовым.

Фолд 1: Тестовый фолд $\{(1.2, 0), (2.3, 0)\}$

Обучающая выборка:

$$\{(2.8, 0), (3.4, 1), (4.1, 1), (4.8, 1), (5.5, 1), (6.0, 1), (6.7, 0), (7.5, 0)\}.$$

Проверяем пороги t (средние значения между x_i):

$$\begin{aligned} t_1 &= \frac{2.8 + 3.4}{2} = 3.1, & t_2 &= \frac{3.4 + 4.1}{2} = 3.75, \\ t_3 &= \frac{4.1 + 4.8}{2} = 4.45, & \dots, & t_7 &= \frac{6.7 + 7.5}{2} = 7.1. \end{aligned}$$

Оптимальный порог $t = 3.1$ минимизирует ошибки. Тестируем: обе точки $(1.2, 0)$ и $(2.3, 0)$ классифицируются верно.

Accuracy₁ = 1.0.

Фолд 2: Тестовый фолд $\{(2.8, 0), (3.4, 1)\}$

Обучающая выборка:

$$\{(1.2, 0), (2.3, 0), (4.1, 1), (4.8, 1), (5.5, 1), (6.0, 1), (6.7, 0), (7.5, 0)\}.$$

Оптимальный порог $t = 3.75$. Тестируем:

$$\hat{y}(2.8) = 0 \quad (\text{верно}), \quad \hat{y}(3.4) = 1 \quad (\text{верно}).$$

$\text{Accuracy}_2 = 1.0$.

Фолд 3: Тестовый фолд $\{(4.1, 1), (4.8, 1)\}$

Обучающая выборка:

$$\{(1.2, 0), (2.3, 0), (2.8, 0), (3.4, 1), (5.5, 1), (6.0, 1), (6.7, 0), (7.5, 0)\}.$$

Оптимальный порог $t = 4.5$. Тестируем:

$$\hat{y}(4.1) = 1 \quad (\text{верно}), \quad \hat{y}(4.8) = 1 \quad (\text{верно}).$$

$\text{Accuracy}_3 = 1.0$.

Фолд 4: Тестовый фолд $\{(5.5, 1), (6.0, 1)\}$

Обучающая выборка:

$$\{(1.2, 0), (2.3, 0), (2.8, 0), (3.4, 1), (4.1, 1), (4.8, 1), (6.7, 0), (7.5, 0)\}.$$

Оптимальный порог $t = 5.75$. Тестируем:

$$\hat{y}(5.5) = 1 \quad (\text{верно}), \quad \hat{y}(6.0) = 1 \quad (\text{верно}).$$

$\text{Accuracy}_4 = 1.0$.

Фолд 5: Тестовый фолд $\{(6.7, 0), (7.5, 0)\}$

Обучающая выборка:

$$\{(1.2, 0), (2.3, 0), (2.8, 0), (3.4, 1), (4.1, 1), (4.8, 1), (5.5, 1), (6.0, 1)\}.$$

Оптимальный порог $t = 6.35$. Тестируем:

$$\hat{y}(6.7) = 0 \quad (\text{верно}), \quad \hat{y}(7.5) = 0 \quad (\text{верно}).$$

$\text{Accuracy}_5 = 1.0$.

Шаг 3: Средняя точность

Средняя точность:

$$\text{Accuracy}_{\text{mean}} = \frac{\text{Accuracy}_1 + \text{Accuracy}_2 + \text{Accuracy}_3 + \text{Accuracy}_4 + \text{Accuracy}_5}{5} = 1.0.$$

Вывод

Модель с оптимальными порогами t классифицировала все объекты правильно на каждом из фолдов. Средняя точность составляет **100%**.

Задача: Кросс-валидация для оценки среднего значения

У вас есть набор данных, содержащий измеренные значения некоторой величины:

$$D = \{x_1, x_2, x_3, \dots, x_{12}\},$$

где x_i — вещественное число. Вам необходимо оценить среднее значение этой величины и одновременно оценить, насколько точно модель предсказывает новые значения, используя технику кросс-валидации.

—

Требуется:

1. Проведите разбиение данных на **4 фолда** для выполнения 4-Fold Cross-Validation. 2. Для каждого фолда:

- Используйте 3 фолда для расчета среднего значения \bar{x} .
 - Используйте 1 фолд для тестирования, чтобы рассчитать абсолютное отклонение предсказанного среднего \bar{x} от истинных значений.
3. Рассчитайте среднее абсолютное отклонение (MAE) по всем тестовым фолдам:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |x_i - \bar{x}|,$$

где n — количество объектов в тестовых фолдах.

—

Данные:

Таблица с измерениями:

Объект i	x_i
1	5.1
2	4.8
3	6.2
4	5.7
5	5.4
6	6.0
7	5.3
8	4.9
9	6.1
10	5.8
11	5.5
12	6.3

—

Упрощение для решения:

- Разделите данные на 4 фолда последовательно: первые 3 объекта в первый фолд, следующие 3 — во второй и так далее.
- Расчеты среднего \bar{x} и абсолютных отклонений выполняйте вручную для каждого фолда.
- Итоговая метрика МАЕ рассчитывается как среднее значение всех абсолютных отклонений.

Решение: Кросс-валидация для оценки среднего значения

Шаг 1. Разбиение данных на фолды

Разделим данные $D = \{5.1, 4.8, 6.2, 5.7, 5.4, 6.0, 5.3, 4.9, 6.1, 5.8, 5.5, 6.3\}$ на 4 фолда:

$$\begin{aligned} \text{Фолд 1: } & \{5.1, 4.8, 6.2\}, & \text{Фолд 2: } & \{5.7, 5.4, 6.0\}, \\ \text{Фолд 3: } & \{5.3, 4.9, 6.1\}, & \text{Фолд 4: } & \{5.8, 5.5, 6.3\}. \end{aligned}$$

Шаг 2. Расчёт среднего значения и абсолютных отклонений для каждого фолда

Для каждого фолда используем три других фолда для расчёта среднего значения \bar{x} и тестируем на оставшемся фолде.

Фолд 1 (тест): {5.1, 4.8, 6.2}

Среднее значение \bar{x} , рассчитанное по остальным фолдам:

$$\bar{x}_1 = \frac{5.7 + 5.4 + 6.0 + 5.3 + 4.9 + 6.1 + 5.8 + 5.5 + 6.3}{9} = 5.67.$$

Абсолютные отклонения:

$$|5.1 - 5.67| = 0.57, \quad |4.8 - 5.67| = 0.87, \quad |6.2 - 5.67| = 0.53.$$

Фолд 2 (тест): {5.7, 5.4, 6.0}

Среднее значение \bar{x} , рассчитанное по остальным фолдам:

$$\bar{x}_2 = \frac{5.1 + 4.8 + 6.2 + 5.3 + 4.9 + 6.1 + 5.8 + 5.5 + 6.3}{9} = 5.55.$$

Абсолютные отклонения:

$$|5.7 - 5.55| = 0.15, \quad |5.4 - 5.55| = 0.15, \quad |6.0 - 5.55| = 0.45.$$

Фолд 3 (тест): {5.3, 4.9, 6.1}

Среднее значение \bar{x} , рассчитанное по остальным фолдам:

$$\bar{x}_3 = \frac{5.1 + 4.8 + 6.2 + 5.7 + 5.4 + 6.0 + 5.8 + 5.5 + 6.3}{9} = 5.64.$$

Абсолютные отклонения:

$$|5.3 - 5.64| = 0.34, \quad |4.9 - 5.64| = 0.74, \quad |6.1 - 5.64| = 0.46.$$

Фолд 4 (тест): {5.8, 5.5, 6.3}

Среднее значение \bar{x} , рассчитанное по остальным фолдам:

$$\bar{x}_4 = \frac{5.1 + 4.8 + 6.2 + 5.7 + 5.4 + 6.0 + 5.3 + 4.9 + 6.1}{9} = 5.49.$$

Абсолютные отклонения:

$$|5.8 - 5.49| = 0.31, \quad |5.5 - 5.49| = 0.01, \quad |6.3 - 5.49| = 0.81.$$

Шаг 3. Итоговая метрика МАЕ

Среднее абсолютное отклонение рассчитывается как:

$$\text{MAE} = \frac{\sum_{i=1}^{12} |x_i - \bar{x}|}{12}.$$

Подставляем рассчитанные значения:

$$\text{MAE} = \frac{0.57 + 0.87 + 0.53 + 0.15 + 0.15 + 0.45 + 0.34 + 0.74 + 0.46 + 0.31 + 0.01 + 0.81}{12} =$$

Ответ:

Среднее абсолютное отклонение (МАЕ): 0.49

1.11. Вероятность переобучения

Рассмотрим объединенную выборку обучение + контроль:

$X^L = \{x_1, \dots, x_L\}$ – конечное генеральное множество объектов;

и

$A = \{a_1, \dots, a_D\}$ – конечное семейство алгоритмов;

т. е. те алгоритмы, среди которых мы выбираем (делаем Model Selection или делаем обучения по обучающей выборке). Пусть при этом функция потерь бинарна, т. е. у нас есть индикатор ошибки (условие того, что алгоритм a ошибается на объекте x):

$$\mathcal{L}(a, x) \equiv I(a, x) = [\text{алгоритм } a \text{ ошибается на объекте } x].$$

Тогда естественным образом возникает матрица ошибок (не путать с матрицей объект-признак) – у нее по строкам объекты, а по столбцам – наши алгоритмы (каждый столбец – бинарный вектор, на каких объектах ошибся наш алгоритм):

На столбце a_4 мы как раз видим переобучение – на обучающих объектах 0 ошибок, а на контрольных – все ошибки (идеально плохая ситуация). Фишка в том, что мы начнем разбивать генеральную выборку всеми возможными способами на обучение и контроль (их всего C_{l+k}^l). Обозначим

$$n(a, X) = \sum_{x \in X} I(a, x) – \text{число ошибок } a \in A \text{ на выборке } X \subset X^L;$$

$$\nu(a, X) = n(a, X)/|X| – \text{частота ошибок } a \text{ на выборке } X.$$

	a_1	a_2	a_3	a_4	a_5	a_6	\dots	a_D	
x_1	1	1	0	0	0	1	\dots	1	X^l — наблюдаемая
\dots	0	0	0	0	1	1	\dots	1	(обучающая) выборка
x_l	0	0	1	0	0	0	\dots	0	длины l
x_{l+1}	0	0	0	1	1	1	\dots	0	X^k — скрытая (кон-
\dots	0	0	0	1	0	0	\dots	1	трольная) выборка
x_L	0	1	1	1	1	1	\dots	0	длины $k = L - l$

Таблица 1.1. $L \times D$ – матрица ошибок с попарно различными столбцами

Нам придется сравнивать частоты ошибок на обучении и на контроле. Их разность – это и есть переобученность. Посмотрим на примере, откуда вообще берется матрица ошибок:

Пример 1.

Пусть у нас есть выборка из 10 объектов, 5 объектов одного класса и 5 другого. Строим линейные классификаторы. В матрице ошибок есть один нулевой вектор (линейный классификатор без ошибок см. рис. ??), 5 векторов с одной ошибкой (рис. ?? и т. д.). Вот откуда берется матрица ошибок. Максимально в ней может быть 2^l вектор-столбцов.

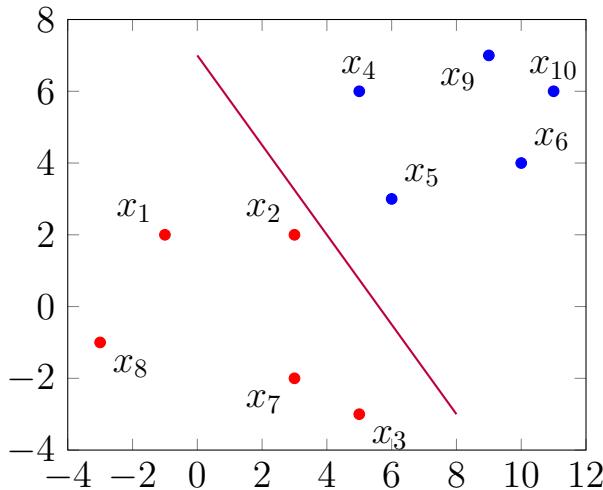


Рис. 1.1. линейный классификатор без ошибок

Наша задача будет оценивать вероятность переобучения. Для начала дадим несколько определений. Вероятностное пространство – это равновероятное разбиение выборки на 2 части (train X^l и test X^k). Интерпретация этому это гипотеза, что наша выборка IID (independent and identically distributed). Т. е. как бы мы эту выборку не упорядочивали, все ее упорядочивания равновероятны (как бы мы ее не разбивали на две части, все разбиения равновероятны). Это есть гипотеза простой выборки.

Замечание. С другой стороны на это можно смотреть, как на Complete Cross

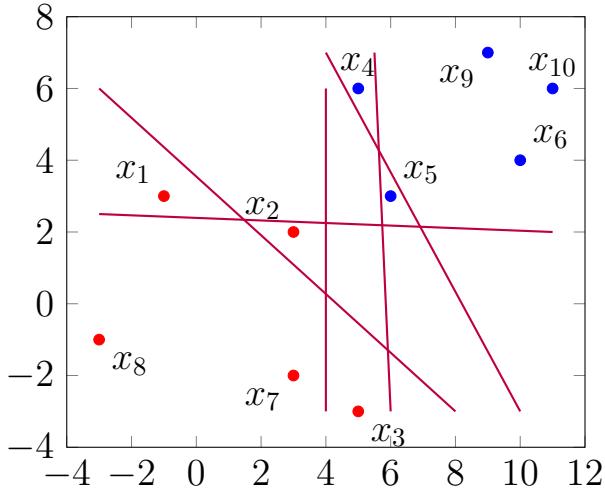


Рис. 1.2. линейный классификатор с 1 ошибкой

x_1	$ $	0	1	0	0	0	0	...
x_2	$ $	0	0	1	0	0	0	...
x_3	$ $	0	0	0	1	0	0	...
x_4	$ $	0	0	0	0	1	0	...
x_5	$ $	0	0	0	0	0	1	...
x_6	$ $	0	0	0	0	0	0	...
x_7	$ $	0	0	0	0	0	0	...
x_8	$ $	0	0	0	0	0	0	...
x_9	$ $	0	0	0	0	0	0	...
x_{10}	$ $	0	0	0	0	0	0	...

Таблица 1.2. матрица ошибок для примера 1

Validation (полный скользящий контроль). Всеми способами разбиваем выборку на train и test. Потом по всем этим способам вероятность любого события, зависящая от этого разбиения, усредняем по всем разбиениям. В этой вероятностной интерпретации очень удобно, что математическое ожидание есть просто усреднение по всем разбиениям выборки

$$P \equiv E \equiv \frac{1}{C_L^l} \sum_{X' \subset X^L} .$$

Переобученность — это разность частот ошибок на X^k и на X^l (обучились на X^l , частоту ошибок посчитали на X^k , частота ошибок нормирована от 0 до 1):

$$\delta(\mu, X^l, X^k) = \nu(\mu(X^l), X^k) - \nu(\mu(X^l), X^l).$$

Переобучение — это событие $\delta(\mu, X^l, X^k) \geq \varepsilon$ для фиксированного ε .

Основная наша задача — оценить вероятность переобучения (вероятность в смысле доли разбиения выборки, на которых δ оказалась $\geq \varepsilon$ при заданном ε):

$$R_\varepsilon(\mu, X^L) = P[\delta(\mu, X^l, X^k) \geq \varepsilon].$$

Теперь наша задача — научиться сверху оценивать эту величину. Рассмотрим простейший, но важный частный случай.

Пусть у нас нет никакого выбора алгоритма, научимся оценивать вероятность переобучения хотя бы для одного отдельно взятого алгоритма, т. е. для одного вектор-столбца матрицы ошибок. Пусть $A = a$ — одноэлементарное множество, $m = n(a, X^L)$. Тогда вероятность переобучения есть вероятность большого отклонения частот ошибок в двух подвыборках:

$$R_\varepsilon(a, X^L) = P[\nu(a, X^k) - \nu(a, X^l) \geq \varepsilon].$$

Теорема Для любой выборки X^l , любого $\varepsilon \in [0, 1]$ вероятность большого отклонения частот ошибок в двух подвыборках для фиксированного вектора ошибок:

$$R_\varepsilon(a, X^L) = \mathcal{H}_L^{l,m}\left(\frac{l}{L}(m - \varepsilon k)\right),$$

где $\mathcal{H}_L^{l,m}(x) = \sum_{s=0}^{|x|} \frac{C_m^s C_{L-m}^{l-s}}{C_L^l}$ — функция гипергеометрического распределения (ее левый хвост).

Доказательство. \square Обозначим число ошибок на обучении как $s = n(a, X^l)$. Аналогия со 2 задачей из упражнения, где m — количество объектов с ошибками, l — обучающая выборка (гипергеометрическое распределение):

$$P[n(a, X^l) = s] = C_m^s C_{L-m}^{l-s} / C_L^l.$$

Распишем R_ε , подставив в него $\nu(a, X^k) = \frac{m-s}{k}$, $\nu(a, X^l) = \frac{s}{l}$ (частота ошибок на обучении $\frac{s}{l}$, а на контроле $\frac{m-s}{k}$) и учитывая тот факт, что $\frac{m-s}{k} - \frac{s}{l}$ есть переобученность:

$$\begin{aligned} R_\varepsilon(a, X^L) &= P[\nu(a, X^k) - \nu(a, X^l) \geq \varepsilon] = \\ &= \sum_{s=0}^l \underbrace{\left[\frac{m-s}{k} - \frac{s}{l} \geq \varepsilon \right]}_{s \leq \frac{l}{L}(m - \varepsilon k)} \underbrace{P[n(a, X^l) = s]}_{C_m^s C_{L-m}^{l-s} / C_L^l} = \mathcal{H}_L^{l,m}\left(\frac{l}{L}(m - \varepsilon k)\right). \end{aligned}$$

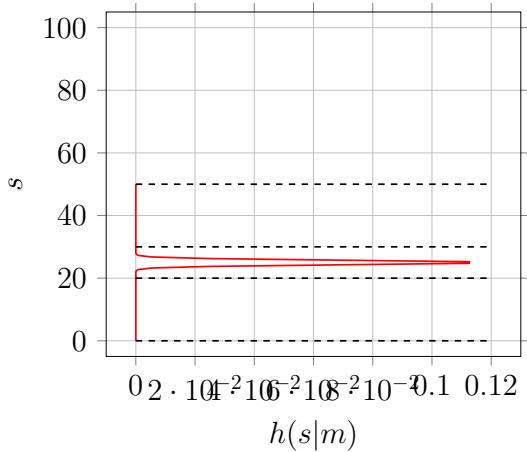
При этом мы получили ограничение на s , которое является следствием того, что переобученность $\leq \varepsilon$. Наша оценка — левый хвост гипергеометрического распределения. ■

Посмотрим как это выглядит на картинке. На оси x отложено m — общее число ошибок на полной выборке, по оси y откладывается число ошибок на обучающей выборке. На графике ?? видна узкая полоска — явление концентрации

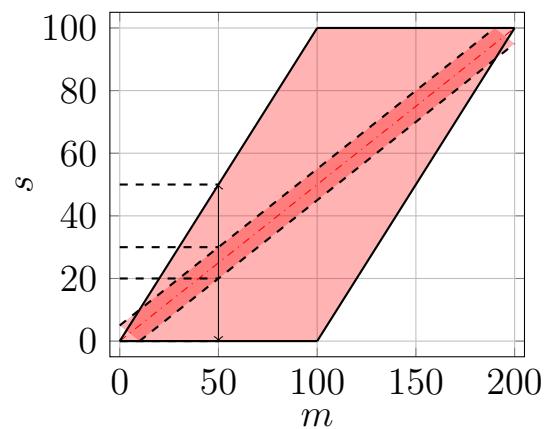
вероятностной меры. Это очень важное явление и в теории вероятности, и в машинном обучении, и в Computational learning theory. Это основной теоретический момент, из которого могут быть получены различные оценки (красная полоска). При том чем больше будет длина выборки, тем уже будет относительная ширина этой полоски.

Предсказание числа $m = n(a, X^L)$ по числу $s = n(a, X^l)$ возможно благодаря узости гипергеометрического пика, причем при $l, k \rightarrow \infty$ он сужается, и $\nu(a, X^l) \rightarrow \nu(a, X^k)$ (явление концентрации вероятности, закон больших чисел).

Гипергеометрическое распределение нам говорит, что если мы знаем m (общее число ошибок на полной выборке), то мы можем предсказать диапазон изменений для s , т. е. сколько у нас ошибок попадет в обучение. Значит будем знать $m - s$, сколько ошибок попадет в контроль. Но в реальной ситуации нам приходится обращать это гипергеометрическое распределение, т. к. мы знаем сколько ошибок у нас попало в обучение. Мы знаем что у нас по y и можем провести горизонтальную линию и сказать, в каком у нас диапазоне могло бы быть m , т. е. в каком диапазоне могло бы лежать общее число ошибок на объединенной выборке. А $m - s$ это число ошибок на контроле.



(a) $h(s|m)$ при $m = 50$



(b) $h(s|m)$ при $L = 200, k = 100$

Рис. 1.3. Гипергеометрическое распределение $h(s|m)$

Задача 1. Сколько линейных классификаторов с двумя ошибками из примера 1?

Решение.

Для решения задачи необходимо посмотреть сколько существует различных способов разделить объекты на рисунке ??, чтобы объектов одного класса было на 2 больше, чем другого. Приведем ответ в виде таблицы:

Задача 2. В урне L шаров, m из них черные, остальные белые; извлекаем l шаров наугад. Какова вероятность, что ровно s из них черные?

Решение.

x_1	1	0	0	0	0	1	1	0
x_2	1	1	0	0	0	0	0	0
x_3	0	1	1	0	0	0	0	1
x_4	0	0	1	1	0	0	0	0
x_5	0	0	0	1	1	1	0	0
x_6	0	0	0	0	1	0	1	0
x_7	0	0	0	0	0	0	0	1
x_8	0	0	0	0	0	0	0	0
x_9	0	0	0	0	0	0	0	0
x_{10}	0	0	0	0	0	0	0	0

Таблица 1.3. все возможные разбиения с 2 ошибками

Обозначим: $N = L$ – общее количество шаров в урне, m – количество черных шаров, $N - m$ – количество белых шаров, l – общее количество извлеченных шаров, s – количество черных шаров среди извлеченных.

Чтобы найти вероятность, нам нужно рассмотреть, сколько способов существует для выбора s черных шаров и $l - s$ белых шаров из общего количества.

Выбрать s черных шаров из m черных можно C_m^s способами.

А выбрать $l - s$ белых шаров из $N - m$ белых – C_{N-m}^{l-s} способами.

Общее количество способов выбрать l шаров из N есть C_N^l .

Таким образом, вероятность того, что из l извлеченных шаров ровно s будут черными, будет равна отношению числа благоприятных исходов к общему числу исходов:

$$P(X = s) = \frac{C_m^s \cdot C_{N-m}^{l-s}}{C_N^l}$$

Задача 3. Как посчитать количество рыб в пруду?

Решение. Выловим некоторое случайное подмножество рыб, пометим их, а потом выпустим обратно в пруд. Через некоторое время рыбы в пруду перемешаются (важно, чтобы они успели перемешаться), опять вылавливаем некоторое количество рыб и смотрим, какая доля среди них – меченная. По этим данным, очевидно, можно найти общее количество рыб в пруду.

1.12. Статистические критерии в машинном обучении

1.12.1. Теория

Статистические инструменты используются для отбора, извлечения и преобразования наиболее релевантных признаков. Например, коэффициент корреляции Пирсона, ранговый коэффициент Спирмена, коэффициент корреляции ANOVA,

ранговый коэффициент Кендалла и критерий Хи-квадрат

В данном обзоре рассмотрим приложения математической статистики в машинном обучении – точнее, в отборе признаков для обучения модели.

Мотивация: Преимущества хорошо подобранных признаков в датасете очевидны: это улучшает точность в обучении с учителем и без, уменьшает время и память, необходимые для корректной работы, помогает ослабить проклятие размерности (экспоненциальный рост необходимых данных).

Одним из основных методов, используемых для отбора признаков, является анализ корреляции. Корреляция позволяет определить, насколько сильно связаны между собой различные признаки и целевая переменная. Признаки с высокой корреляцией с целевой переменной могут быть отобраны для дальнейшего анализа, в то время как признаки, которые не имеют значительной связи, могут быть исключены. Это помогает избежать проблем многоколлинеарности, когда несколько признаков предоставляют одинаковую информацию.

Другим важным инструментом является тестирование гипотез. С помощью статистических тестов, таких как t-тест или ANOVA, можно оценить значимость отдельных признаков в контексте целевой переменной. Например, если мы имеем дело с задачей классификации, мы можем использовать тесты для определения того, какие признаки значительно различаются между классами. Это позволяет сосредоточиться на наиболее информативных признаках и исключить те, которые не вносят существенного вклада в модель.

Методы селекции на основе значимости также широко применяются в практике. Алгоритмы, такие как LASSO (Least Absolute Shrinkage and Selection Operator), используют регуляризацию для уменьшения коэффициентов менее значимых признаков до нуля, что автоматически исключает их из модели. Это не только упрощает модель, но и помогает избежать переобучения — проблемы, когда модель слишком точно подстраивается под обучающие данные и теряет способность обобщать на новых данных.

Методы оценки производительности модели также зависят от правильного отбора признаков. Кросс-валидация позволяет оценить, как изменения в наборе признаков влияют на общую производительность модели. Сравнение различных моделей с разными наборами признаков помогает выбрать оптимальный вариант, который обеспечивает наилучшие результаты.

Кроме того, современные подходы к машинному обучению, такие как ан-

самблевые методы (например, Random Forest), также используют статистические методы для оценки важности признаков. В таких методах важность каждого признака может быть оценена по тому, как сильно он влияет на уменьшение ошибки предсказания модели.

1.12.2. Задачи

1. Анализ корреляции

У вас есть набор данных с 5 признаками (X_1, X_2, X_3, X_4, X_5) и целевой переменной Y . Вы хотите выяснить, какие признаки имеют наибольшую корреляцию с Y .

Решение:

1. Рассчитайте коэффициенты корреляции Пирсона для каждого признака относительно Y .
2. Сравните полученные значения и выберите признаки с наибольшими абсолютными значениями коэффициента корреляции.

2. Тестирование гипотез

Вы хотите проверить, есть ли статистически значимая разница между двумя группами данных (группа А и группа В) по признаку X . У вас есть данные о значениях X для обеих групп.

Решение:

1. Проведите t -тест для независимых выборок.
2. Оцените p -значение для определения значимости.

3. Регуляризация LASSO

У вас есть набор данных с несколькими признаками и целевой переменной. Вы хотите использовать метод LASSO для отбора наиболее значимых признаков.

Решение:

1. Импортируйте библиотеку *Lasso* из *sklearn*.
2. Подготовьте данные и обучите модель LASSO.
3. Оцените важность признаков на основе полученных коэффициентов.

1.13. Кросс-валидация

1.13.1. Теория

Кросс-валидация — это процедура для оценки качества работы модели, которая позволяет более точно оценивать, как модель будет работать на новых, невидимых данных. Основная идея заключается в разбиении данных на несколько частей и обучении модели на разных комбинациях этих частей. При этом проверять качество модели необходимо на оставшихся данных.

Hold-out

В этом методе данные просто разделяются на train и test. После этого происходит обучение на train и проверка качества на test. При этом важно перемешивать данные, потому что в противном случае могут появиться неожиданные «спецэффекты». Например, в изначальных данных может быть, что первые 80% — это изображения с кошками, а остальные 20% — с собаками. Если при этом цель — научить модель различать кошек и собак, то очень важные всё перемешать, чтобы не оказалось, что в train попали только кошки.

hold_out.png

/2/2

Рис. 1.4. Hold-out

k-Fold

Это самый распространенный метод кросс-валидации, в нём данные разбиваются на k подмножеств. Модель обучается k раз, каждый раз используя одно подмножество в качестве тестовой выборки, а остальные $k-1$ как тренировочные. В конце ошибкой модели считается либо средняя полученная ошибка по всем выборкам тестового множества, либо вычисляется на отдельно отложенном валидационном множестве. Это позволяет использовать все доступные данные для обучения и тестирования, что особенно полезно в случаях, когда данные ограничены.

k-fold.png

/2/2

Рис. 1.5. k-Fold

Можно сделать вывод, что кросс-валидация помогает избежать переобучения и оптимизировать гиперпараметры моделей засчёт того, что модель много раз обучается и тестируется на разных подмножествах одних и тех же данных.

1.13.2. Задачи (источник)

Задача 1

Пример из практики Yandex.Research — как вы думаете, что не так с графиком обучения данной модели?

Решение На графике видна периодичность по числу итераций! По большим пикам можно вычислить места, где проход по данным начался заново. Кроме того, график в конце ползёт вниз, что означает, что модель уже начала переобучаться, выучив последовательность данных на трейне и используя эту информацию больше, чем сами данные.

Если данные перемешать, то график обучения станет таким:

Задача 2

Пусть нужно обучить модель, которая должна предсказывать погоду. Есть исторические данные за последние 10 лет. Можно ли как обычно разделять данные на train и test, произвольно перемешивая их? Если нет, то как нужно разделять данные?

Решение Нет, так делать нельзя, потому что в таком случае у модели при обучении будет доступ к данным из «будущего», которые он как раз и должна предсказывать, а это некорректно. Правильно будет разделить данные по времени: самые старые — train, самые новые — test.

Задача 3

А как сделать кросс-валидацию в условиях задачи 2?

Решение

Нужно сделать несколько пар (train, test), но так, чтобы модель никогда не получала лишних данных из будущего. С учётом особенностей фолды в кросс-валидации для временных рядов располагаются вдоль временной оси так, как показано на следующей картинке:

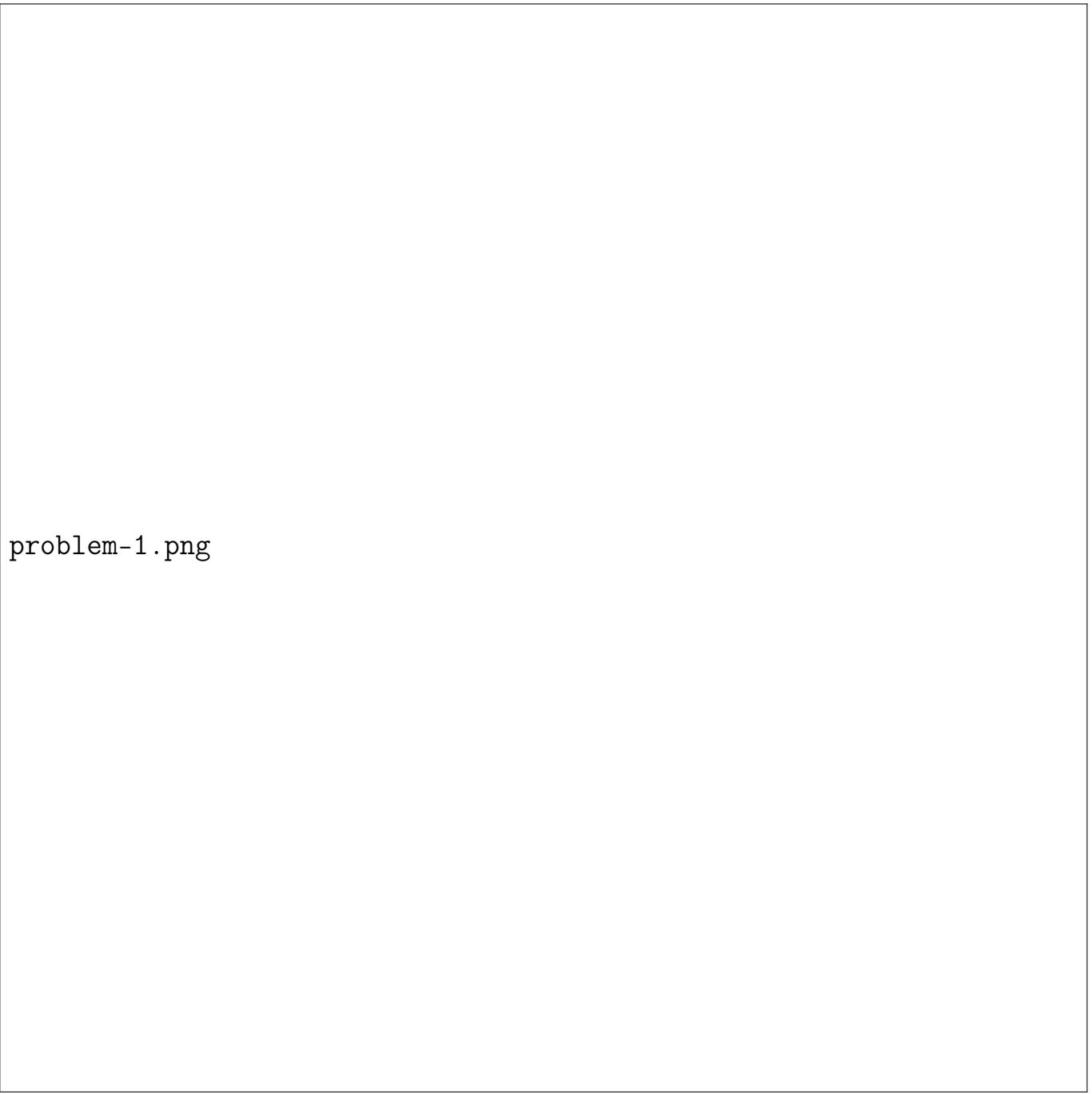


Рис. 1.6. Задача 1

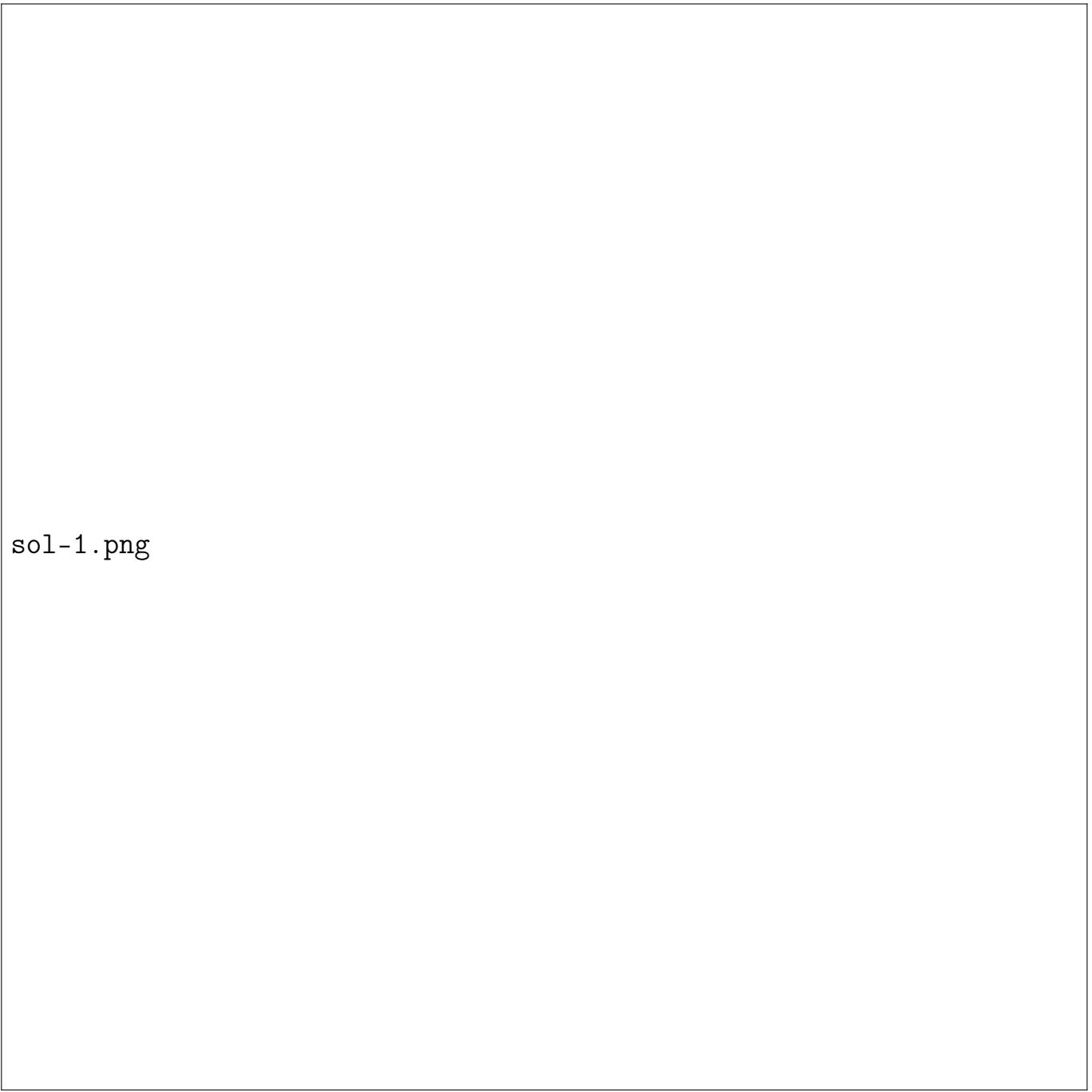


Рис. 1.7. Решение задачи 1

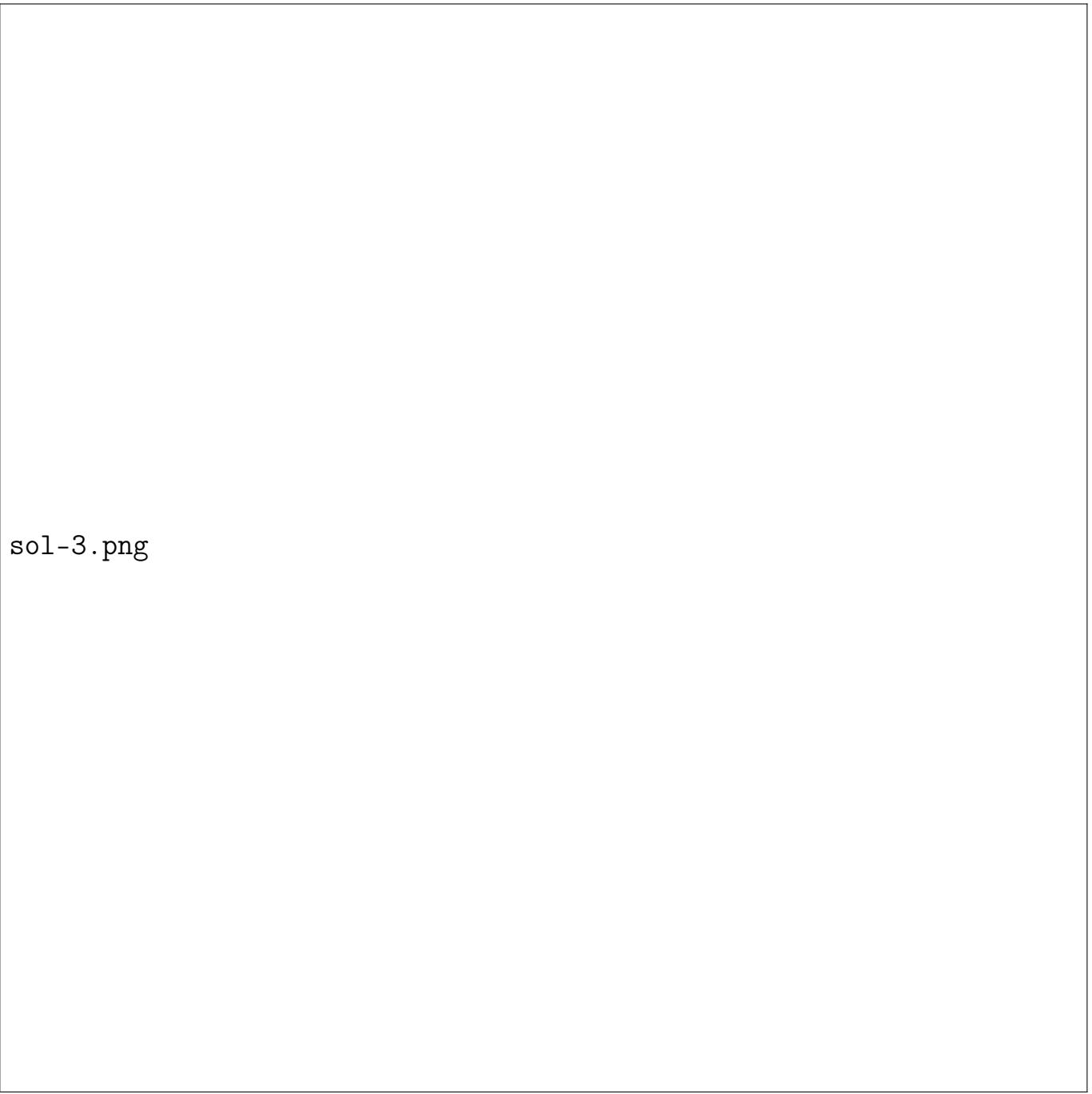


Рис. 1.8. Решение задачи 3

Глава 2

Линейные методы классификации и регрессии

2.1. О регуляризации

При решении задачи машинного обучения часто возникает проблема переобучения, при котором модель подстраивается под шум данных, что снижает ее обобщающую способность. Один из методов борьбы с этим — регуляризации, или же сокращение весов (англ.: weight decay). Данный метод добавляет штраф к функции потерь за сложность модели, в случае линейных моделей — штраф за большие веса коэффициентов. Регуляризация ограничивает пространство решений и делает модель более устойчивой к шуму, что увеличивает вероятность корректных предсказаний на новых данных. Пример, когда усложнение модели, путем добавления избыточных коэффициентов полиномиальной модели представлен на рис. ??.

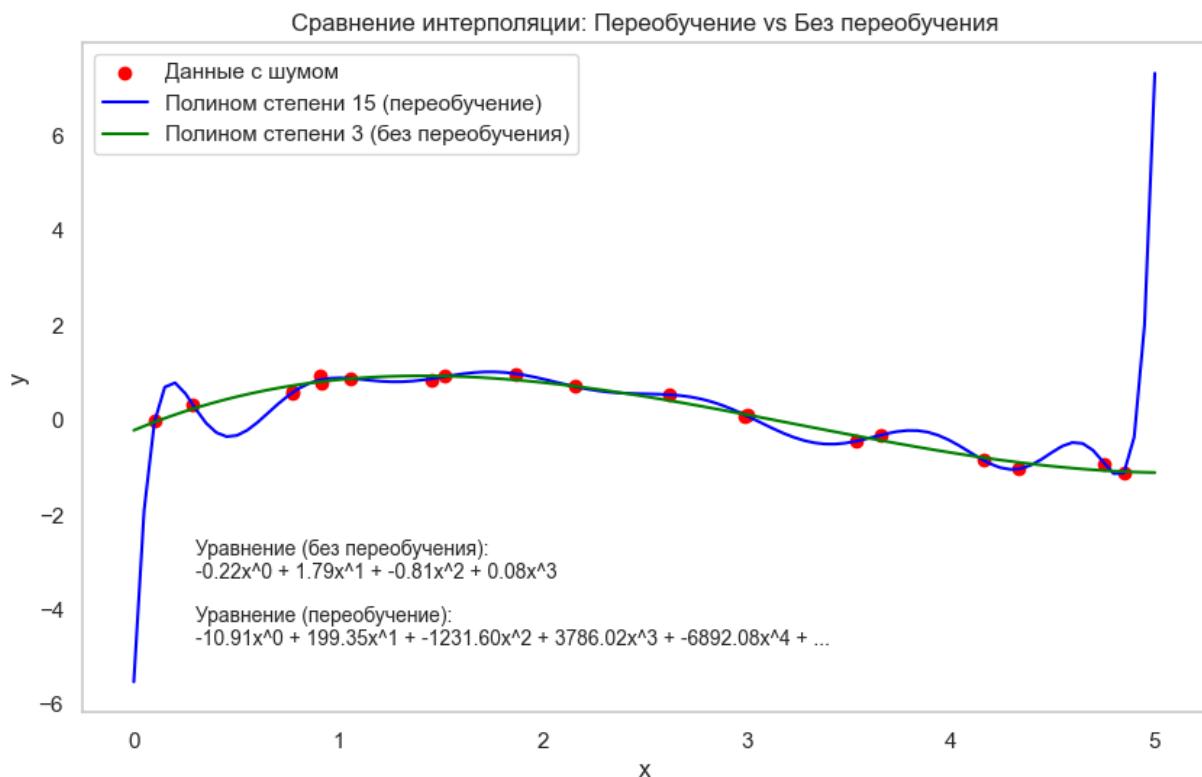


Рис. 2.1. Иллюстрация проблемы переобучения для решения линейной задачи

В разделе ?? рассмотрены основные методы регуляризации, применяемые в линейных моделях. Раздел ?? описывает вероятностную трактовку причин возникновения регуляризации. Завершается глава разделом ??, который содержит три теоретических задачи по теме регрессии.

2.1.1. Гауссовский и лапласовский регуляризаторы

Гауссовский и лапласовский регуляризаторы представляют собой две разные техники регуляризации, которые используются для управления сложностью моделей и предотвращения переобучения. Они основаны на различных подходах к штрафованию весов модели.

Лапласовский регуляризатор (L1 регуляризатор)

Лапласовский (L1) регуляризатор использует сумму модулей значений весов модели в качестве штрафа. Формально новую функцию потерь можно записать следующим образом: $L1 = L_0 + \lambda \sum_{i=1}^n |w_i|$, где L_0 — исходная функция потерь, $|w_i|$ — абсолютные значения весов модели, λ — коэффициент регуляризации. Преимущества:

- Лапласовский регуляризатор может приводить к обнулению некоторых весов, что делает модель более интерпретируемой и позволяет выделять наиболее важные признаки.
- Он может быть особенно полезен в задачах с высокой размерностью, где много признаков могут быть неинформативными.

Недостатки:

- Оптимизация с использованием L1 регуляризации может быть более сложной и требовать специальных алгоритмов (например, координатного спуска или методов, основанных на субградиенте).

Гауссовский регуляризатор (L2 регуляризатор)

Гауссовский (L2) регуляризатор использует штраф в виде суммы квадратов весов модели. Формально новую функцию потерь можно записать следующим образом: $L2 = L_0 + \lambda \sum_{i=1}^n w_i^2$, где L_0 — исходная функция потерь (например, среднеквадратичная ошибка для задачи регрессии), w_i — веса модели, λ — коэффициент регуляризации, который контролирует величину штрафа. Преимущества:

- Гауссовский регуляризатор помогает сгладить веса, что делает модель более устойчивой к шуму в данных.
- Он способствует распределению весов по всем признакам, уменьшая вероятность того, что некоторые признаки будут доминировать.

Недостатки:

- Может не приводить к полному обнулению весов, поэтому не всегда приводит к интерпретируемым моделям.

Сравнений L1 и L2 регуляризаторов

Выбор между гауссовским и лапласовским регуляризаторами зависит от конкретной задачи и целей (сравнение см. в таблице ??). Если важна интерпретируемость модели и выделение значимых признаков, то стоит рассмотреть L1 регуляризацию. Если же цель — улучшить общее качество модели без сильного сокращения количества признаков, то L2 регуляризация может быть предпочтительнее. Это связано с тем, что в L2 регуляризации за счет возведения в квадрат значений весов вклад нулевого веса и просто малого веса неразличим на при наличии других выделенных признаков с весами порядка или более единицы. Таким образом, L2 регуляризация, в отличии от L1 не стремится обнулить коэффициенты, что снижает интерпретируемость модели. Однако квадратичная функция является гладкой, поэтому лучше поддается вычислительным методам оптимизации.

Таблица 2.1. Сравнение L1 и L2 регуляризаторов

Характеристика	Лапласовский (L1)	Гауссовский (L2)
Штраф	Сумма абсолютных значений весов	Сумма квадратов весов
Эффект на веса	Обнуление (спарсность)	Сглаживание
Интерпретируемость	Выше	Меньше
Оптимизация	Сложнее	Легче

В ситуациях используется комбинация обоих методов регуляризации. Для этого вводится дополнительный гиперпараметр α — доля первой нормы в штрафе за вес. Получается так называемая эластичная сеть (англ.: elastic net). В таком случае: $L = L_0 + \lambda[\alpha \sum_{i=1}^n |w_i| + (1 - \alpha) \sum_{i=1}^n w_i^2]$. Даный подход позволяет учесть особенности двух подходов, однако усложняет модель.

2.1.2. Вероятностная интерпретация регуляризации

В первом рассмотрении регуляризацию можно рассматриваться как введение априорной информации о параметрах модели. Рассмотрим вводимые предположения:

- в случае L1 регуляризации мы предполагаем, что многие веса могут быть равны нулю, это соответствует идеи о том, что истинная модель простая и присутствуют лишние признаки;
- в случае L2 — веса распределены вблизи общего малого среднего значения, это отражает предположение о том, что все признаки вносят сопоставимый вклад в предсказание.

Описанным выше предположениям о весах соответствуют экспоненциальное и нормальное распределения (см. рис. ??). Рассмотрим их влияние на штраф и функцию потерь.

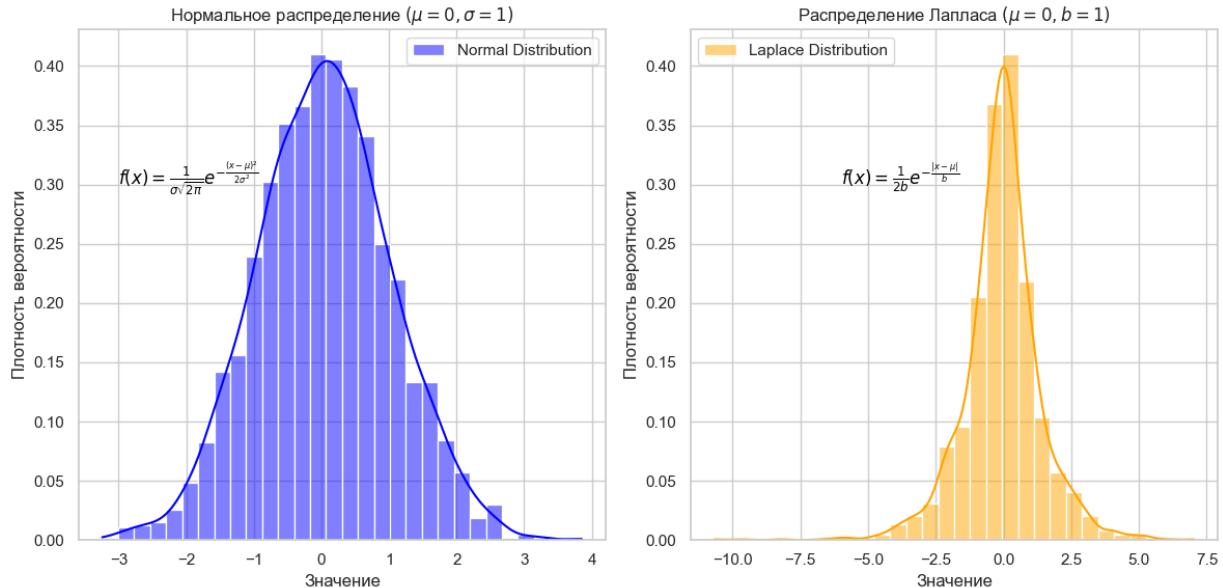


Рис. 2.2. Сравнение экспоненциального и нормального распределений

L1 регуляризация

предполагает, что веса w_j распределены по лапласовскому закону (или двойному экспоненциальному распределению), $w_j \sim Laplace(0, b)$.

L2 регуляризация

предполагает, что веса модели w_j имеют нормальное распределение с нулевым средним и некоторой дисперсией σ^2 . Это можно записать как $w_j \sim N(0, \sigma^2)$.

2.1.3. Задачи

Вопрос 1

Как изменение коэффициента λ влияет на величину весов w_j ?

L1 регуляризация

При увеличении λ происходит обнулению некоторых весов w_j . Это приводит к тому, что с увеличением λ количество ненулевых весов уменьшается, что может помочь в отборе признаков и упрощении модели.

L2 регуляризация

При увеличении λ происходит увеличение штрафа за большие значения весов. Это приводит к уменьшению величины весов w_j (все веса стремятся к нулю), что помогает избежать переобучения. В случае $\lambda = 0$ модель не имеет регуляризации, и веса могут принимать любые значения, что может привести к переобучению.

Вопрос 2

Какова геометрическая интерпретация регуляризации в пространстве весов?

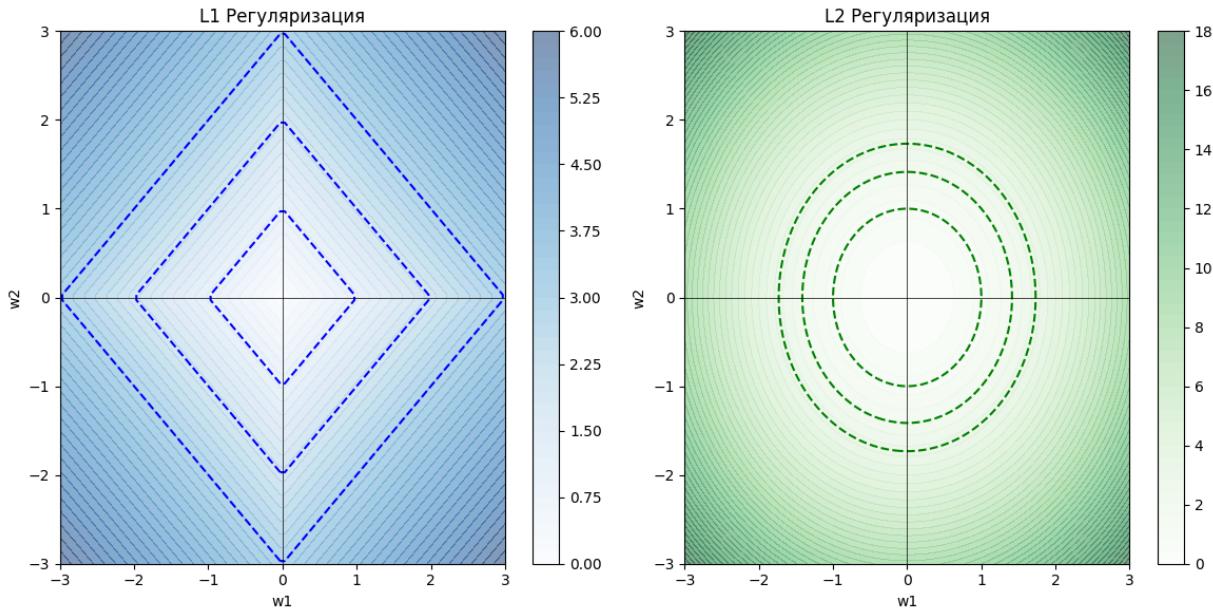


Рис. 2.3. Геометрическая интерпретация регуляризации в пространстве весов

L1 регуляризация

Геометрически L1 регуляризация создает ромбовидные (или параллелепипедные) области в пространстве весов. Оптимальные веса находятся на вершинах этих ромбов, что приводит к обнулению некоторых весов и, следовательно, к отбору признаков.

$$L1(w_1, w_2) = \text{const} \Leftrightarrow |w_1| + |w_2| = \text{const}$$

L2 регуляризация

Геометрически L2 регуляризация создает сферу (или гиперсферу) в пространстве весов, внутри которой минимизируется функция потерь. Это означает, что оптимальные веса будут находиться на поверхности этой сферы, что приводит к сглаживанию и уменьшению значений весов.

$$L2(w_1, w_2) = \text{const} \Leftrightarrow w_1^2 + w_2^2 = \text{const}$$

Вопрос 3

Какие особенности признаков компенсируют регуляризации?

L1 регуляризация

Используя L1 регуляризацию, можно обнулить веса для менее значимых признаков. После обучения модели можно проанализировать ненулевые веса и оставить только те признаки, которые имеют значимые коэффициенты, тем самым осуществляя отбор признаков.

L2 регуляризация

L2 регуляризация помогает сгладить веса при наличии мультиколлинеарности, уменьшая их величину и тем самым снижая влияние коррелирующих признаков. Это позволяет избежать чрезмерного увеличения весов для сильно коррелирующих признаков.

Основная идея

Градиентные методы — это широкий класс оптимизационных алгоритмов, используемых не только в машинном обучении. Здесь градиентный подход будет рассмотрен в качестве способа подбора вектора синаптических весов w в линейном классификаторе. Пусть $y^* : X \rightarrow Y$ — целевая зависимость, известная только на объектах обучающей выборки: $X^l = (x_i, y_i)_{i=1}^l$, где $y_i = y^*(x_i)$.

Найдём алгоритм $a(x, w)$, аппроксимирующий зависимость y^* . В случае линейного классификатора искомый алгоритм имеет вид:

$$a(x, w) = \varphi \left(\sum_{j=1}^n w_j x^j - w_0 \right),$$

где $\varphi(z)$ играет роль функции активации (в простейшем случае можно положить $\varphi(z) = \text{sign}(z)$).

Согласно принципу минимизации эмпирического риска, для этого достаточно решить оптимизационную задачу:

$$Q(w) = \sum_{i=1}^l L(a(x_i, w), y_i) \rightarrow \min_w,$$

где $L(a, y)$ — заданная функция потерь.

Для минимизации применим метод градиентного спуска (gradient descent). Это пошаговый алгоритм, на каждой итерации которого вектор w изменяется в направлении наибольшего убывания функционала Q (то есть в направлении антиградиента):

$$w := w - \eta \nabla Q(w),$$

где η — положительный параметр, называемый темпом обучения (learning rate).

Основные подходы к реализации градиентного спуска

- Пакетный (batch):** на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяется w . Этот подход требует больших вычислительных затрат.

2. **Стохастический (stochastic/online):** на каждой итерации из обучающей выборки случайным образом выбирается один объект. Таким образом, вектор w настраивается на каждый вновь выбираемый объект.

Алгоритм Stochastic Gradient (SG)

Вход:

- X^l — обучающая выборка;
- η — темп обучения;
- λ — параметр сглаживания функционала Q .

Выход: Вектор весов w

Тело алгоритма:

1. Инициализировать веса w_j , $j = 0, \dots, n$;
2. Инициализировать текущую оценку функционала: $Q := \sum_{i=1}^l L(a(x_i, w), y_i)$;
3. Повторять:
 - a. выбрать объект x_i из X^l (например, случайным образом);
 - b. вычислить выходное значение алгоритма $a(x_i, w)$ и ошибку: $\varepsilon_i := L(a(x_i, w), y_i)$;
 - c. сделать шаг градиентного спуска:

$$w := w - \eta L'_a(a(x_i, w), y_i) \varphi'(\langle w, x_i \rangle) x_i;$$

- d. оценить значение функционала:

$$Q := (1 - \lambda)Q + \lambda \varepsilon_i;$$

пока значение Q не стабилизируется и/или веса w не перестанут изменяться.

Порядок выбора объектов

В случае стохастического градиентного спуска объекты следует выбирать случайным образом, однако существуют эвристики, направленные на улучшение сходимости:

- Перемешивание (shuffling): случайно выбирать объекты, попеременно из разных классов. Идея в том, что объекты из разных классов менее "похожи" чем объекты одного класса, поэтому вектор w будет сильнее изменяться.
- Можно выбирать объект с вероятностью, обратно пропорциональной величине ошибки на объекте. Следует учитывать, что такая эвристика делает метод чувствительным к шумам.

Способы инициализации весов

1. Инициализация вектора w нулями.
2. $w_j := \text{rand}\left(-\frac{1}{n}, \frac{1}{n}\right)$, где n — размерность пространства признаков.
3. Решение исходной оптимизационной задачи при условии статистически независимых признаков, линейной функции активации (φ) и квадратичной функции потерь (L):

$$w_j := \frac{\langle y, f_j \rangle}{\langle f_j, f_j \rangle}.$$

Параметр сглаживания

Для оценки функционала Q на каждой итерации используется его приближённое значение по методу экспоненциального сглаживания, откуда λ лучше брать порядка $\frac{1}{l}$.

Известные частные случаи алгоритма

Метод SG (при соответствующем выборе функций активации и потерь) является обобщением следующих эвристик подбора w и алгоритмов классификации:

- Адаптивный линейный элемент (Adalines);
- Правило Хэбба;
- Алгоритм k -средних (K-Means);
- Learning Vector Quantization (LVQ).

Преимущества SG

- Метод подходит для динамического (online) обучения.
- Алгоритм способен обучаться на избыточно больших выборках.
- Различные стратегии обучения позволяют адаптировать алгоритм для задач с избыточной или небольшой выборкой.

Недостатки SG и способы их устранения

- Возможны проблемы сходимости. Для борьбы с этим применяют технику встрихивания коэффициентов.

- При высокой размерности пространства признаков n и/или малой длине выборки l возможно переобучение. Для борьбы с этим применяют метод сокращения весов:

$$Q_\tau(w) = Q(w) + \frac{\tau}{2} \|w\|^2.$$

Тогда правило обновления весов принимает вид:

$$w := w(1 - \eta\tau) - \eta\nabla Q(w).$$

- При больших значениях $\langle w, x_i \rangle$ значение φ' может становиться близким к нулю. Для предотвращения этого состояния вводят нормализацию признаков:

$$x^j := \frac{x^j - x_{\min}^j}{x_{\max}^j - x_{\min}^j}, \quad j = 1, \dots, n,$$

где x_{\min}^j, x_{\max}^j — минимальное и максимальное значения признака j -го признака. Регуляризация, такая как weight decay, также помогает избежать "паралича".

Сходимость алгоритма

Сходимость гарантируется при выпуклой функции $Q(w)$ и выполнении следующих условий:

$$\eta_t \xrightarrow{t \rightarrow \infty} 0, \quad \sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty.$$

Например, можно положить $\eta_t = \frac{\eta_0}{t}$, хотя на практике это не всегда удачно.

Задачи

Задача 1: Доказать сходимость алгоритма при условиях выше

Решение:

1. **Выпуклость функции:** Поскольку $Q(w)$ выпукла, мы можем использовать свойства выпуклых функций. Для любого w и w^* (где w^* — точка минимума функции Q) выполняется неравенство:

$$Q(w) \geq Q(w^*) + \nabla Q(w^*)^T (w - w^*).$$

2. **Итерация метода стохастического градиента:** Обновление весов в SGD задается следующим образом:

$$w_{t+1} = w_t - \eta_t \nabla Q(w_t; \xi_t),$$

где ξ_t — случайная переменная, представляющая выборку данных на итерации t .

3. Анализ изменения функции: Мы можем оценить изменение функции Q на каждой итерации:

$$Q(w_{t+1}) \leq Q(w_t) + \nabla Q(w_t; \xi_t)^T (w_{t+1} - w_t) + \frac{L}{2} \|w_{t+1} - w_t\|^2,$$

где L — константа Липшица для градиента ∇Q .

Подставляя обновление:

$$Q(w_{t+1}) \leq Q(w_t) - \eta_t \nabla Q(w_t; \xi_t)^T \nabla Q(w_t) + \frac{L}{2} \eta_t^2 \|\nabla Q(w_t; \xi_t)\|^2.$$

4. Суммирование изменений: Суммируя по всем итерациям, мы получаем:

$$\sum_{t=1}^T Q(w_{t+1}) - Q(w_1) \leq - \sum_{t=1}^T \eta_t \nabla Q(w_t; \xi_t)^T \nabla Q(w_t) + \sum_{t=1}^T \frac{L}{2} \eta_t^2 \|\nabla Q(w_t; \xi_t)\|^2.$$

5. Использование условий: Условия $\sum_{t=1}^{\infty} \eta_t = \infty$ и $\sum_{t=1}^{\infty} \eta_t^2 < \infty$ позволяют нам сделать вывод о том, что:

- Сумма шагов обучения стремится к бесконечности, что означает, что веса w_t будут продолжать обновляться.
- Сумма квадратов шагов обучения конечна, что позволяет контролировать величину изменений на каждой итерации.

6. Сходимость к минимуму: В результате, при условии, что $Q(w)$ выпукла, и учитывая условия на шаги обучения, мы можем утверждать, что последовательность w_t будет сходиться к некоторой точке w^* , которая является минимумом функции $Q(w)$.

Таким образом, метод стохастического градиента сходится к минимуму выпуклой функции $Q(w)$ при выполнении заданных условий.

Задача 2: Оценка вариации градиента

Условие: Пусть $\xi_1, \xi_2, \dots, \xi_n$ — независимые и одинаково распределённые (i.i.d.) случайные переменные, представляющие собой выборки из обучающего набора. Рассмотрим стохастический градиент $\nabla L(\theta; \xi_t)$.

Задача: Доказать, что математическое ожидание стохастического градиента совпадает с истинным градиентом функции потерь:

$$\mathbb{E}[\nabla L(\theta; \xi_t)] = \nabla L(\theta)$$

и оценить дисперсию $\text{Var}(\nabla L(\theta; \xi_t))$ в зависимости от размера выборки n .

Доказательство

- Определение стохастического градиента:** Пусть $L(\theta)$ — функция потерь, зависящая от параметров θ и от выборки ξ . Мы можем записать функцию потерь как среднее значение по всем данным:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n l(\theta; x_i, y_i),$$

где $l(\theta; x_i, y_i)$ — функция потерь для примера (x_i, y_i) .

- Истинный градиент:** Тогда истинный градиент функции потерь можно выразить как:

$$\nabla L(\theta) = \frac{1}{n} \sum_{i=1}^n \nabla l(\theta; x_i, y_i).$$

- Математическое ожидание стохастического градиента:** Теперь рассмотрим стохастический градиент:

$$\nabla L(\theta; \xi_t) = \nabla l(\theta; \xi_t),$$

где ξ_t — случайная выборка. Поскольку ξ_t выбирается из одного из n примеров, математическое ожидание стохастического градиента будет:

$$\mathbb{E}[\nabla L(\theta; \xi_t)] = \mathbb{E}[\nabla l(\theta; \xi_t)] = \frac{1}{n} \sum_{i=1}^n \nabla l(\theta; x_i, y_i) = \nabla L(\theta).$$

Таким образом, мы доказали, что:

$$\mathbb{E}[\nabla L(\theta; \xi_t)] = \nabla L(\theta).$$

- Оценка дисперсии:** Теперь найдём дисперсию стохастического градиента:

$$\text{Var}(\nabla L(\theta; \xi_t)) = \mathbb{E}[(\nabla L(\theta; \xi_t) - \mathbb{E}[\nabla L(\theta; \xi_t)])^2].$$

Подставим выражение для стохастического градиента:

$$\text{Var}(\nabla L(\theta; \xi_t)) = \mathbb{E}[(\nabla l(\theta; \xi_t) - \nabla L(\theta))^2].$$

Поскольку ξ_t является случайным выбором, мы можем использовать свойства дисперсии. Для n независимых и одинаково распределённых (i.i.d.) выборок дисперсия стохастического градиента будет уменьшаться с увеличением размера выборки:

$$\text{Var}(\nabla L(\theta; \xi_t)) = \frac{1}{n} \text{Var}(l(\theta; x, y)),$$

где (x, y) — случайная выборка из обучающего набора. Это означает, что дисперсия стохастического градиента уменьшается с увеличением размера выборки n .

Задача 3: Регуляризация и стохастический градиент

Условие

Рассмотрим функцию потерь с L2-регуляризацией:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n l(\theta; x_i, y_i) + \frac{\lambda}{2} \|\theta\|^2$$

где $l(\theta; x_i, y_i)$ — функция потерь для примера (x_i, y_i) , а λ — коэффициент регуляризации.

Задача

Обосновать, как регуляризация влияет на сходимость метода стохастического градиента, и показать, что использование регуляризации может помочь избежать переобучения, уменьшая значение функции потерь на валидационном наборе.

Влияние регуляризации на сходимость метода стохастического градиента

1. Сглаживание функции потерь:

- Добавление L2-регуляризации к функции потерь делает её более гладкой и выпуклой. Это связано с тем, что регуляризационный член $\frac{\lambda}{2} \|\theta\|^2$ добавляет "наказание" за большие значения параметров, что предотвращает резкие изменения градиента.
- Гладкость функции потерь способствует более стабильному обновлению параметров при использовании стохастического градиента. Это означает, что обновления параметров будут более предсказуемыми и менее подвержены шуму, что улучшает сходимость алгоритма.

2. Уменьшение переобучения:

- Регуляризация способствует уменьшению значений параметров модели, что, в свою очередь, снижает сложность модели. Это позволяет избежать переобучения, когда модель слишком точно подстраивается под тренировочные данные, включая шум.
- В результате, при использовании регуляризации, модель будет лучше обобщаться на новых данных, что выражается в меньшем значении функции потерь на валидационном наборе.

Доказательство эффекта регуляризации на валидационном наборе

1. Функция потерь на валидационном наборе:

- Пусть $L_{val}(\theta)$ — функция потерь на валидационном наборе. При использовании регуляризации, мы можем записать:

$$L_{val}(\theta) = \frac{1}{m} \sum_{j=1}^m l(\theta; x_j, y_j) + \frac{\lambda}{2} \|\theta\|^2$$

где m — количество примеров в валидационном наборе.

2. Сравнение значений функции потерь:

- Без регуляризации, модель может иметь высокую функцию потерь на валидационном наборе из-за переобучения. При добавлении L2-регуляризации, даже если функция потерь на тренировочном наборе остаётся низкой, регуляризация помогает поддерживать значение функции потерь на валидационном наборе на более низком уровне.

3. Кросс-валидация для выбора λ :

- Оптимальное значение λ можно выбрать с помощью кросс-валидации. Это позволяет находить компромисс между сложностью модели и её обобщающей способностью, что в конечном итоге приводит к меньшему значению функции потерь на валидационном наборе.

Заключение по задаче 3

Регуляризация, особенно L2-регуляризация, играет ключевую роль в улучшении сходимости метода стохастического градиента и в предотвращении переобучения модели. Она помогает сделать функцию потерь более гладкой и выпуклой, что способствует стабильности обновлений параметров. В результате, использование регуляризации приводит к лучшему обобщению модели и снижению значения функции потерь на валидационном наборе, что является важным аспектом при разработке надёжных моделей в машинном обучении.

Линейный дискриминантный анализ (LDA)

Линейный дискриминантный анализ (LDA) — это статистический метод для решения задач классификации, который используется для поиска линейных комбинаций признаков, наиболее эффективно разделяющих два или более классов.

Основной задачей LDA является минимизация внутриклассовой дисперсии и максимизация межклассовой дисперсии, что позволяет лучше различать классы на основе их характеристик.

Предположения LDA

LDA предполагает следующие условия для классов данных:

1. **Нормальность распределений.** Признаки $x \in \mathbb{R}^m$ для каждого класса C_k распределены нормально с параметрами: средним вектором $\mu_k \in \mathbb{R}^m$ и ковариационной матрицей $\Sigma_k \in \mathbb{R}^{m \times m}$.
2. **Однаковые ковариационные матрицы.** Все классы имеют одинаковую ковариационную матрицу Σ , что значительно упрощает задачу классификации, так как для построения линейной границы между классами используется одна и та же матрица ковариаций.

Согласно этим предположениям, распределение признаков в каждом классе можно выразить через многомерное нормальное распределение:

$$P(x|C_k) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right),$$

где μ_k — средний вектор признаков для класса C_k , а Σ — ковариационная матрица, одна и та же для всех классов.

Задача LDA

Цель LDA заключается в нахождении линейного классификатора, который разделяет классы с минимальной ошибкой. Для этого LDA ищет линейную функцию от признаков x вида:

$$\delta(x) = w^T x + b,$$

где w — вектор весов, b — смещение. Классы разделяются гиперплоскостью, заданной уравнением:

$$w^T x + b = 0.$$

Классификация основывается на выборе того класса, для которого значение $\delta(x)$ наибольшее:

$$\hat{y} = \text{sign}(w^T x + b).$$

Вывод классификатора LDA

Для решения задачи классификации методом LDA необходимо максимизировать правдоподобие для наблюдаемых данных, предполагая, что каждый класс имеет нормальное распределение с одинаковыми ковариационными матрицами. Рассмотрим два класса C_1 и C_2 . Обозначим средние векторы классов как μ_1 и μ_2 , а ковариационную матрицу как Σ .

В соответствии с теоремой Байеса, для каждого класса вероятность $P(C_k|x)$ вычисляется как:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}.$$

Для того чтобы классифицировать объект x , необходимо выбрать класс с наибольшей апостериорной вероятностью. Так как $P(x)$ не зависит от класса, задача сводится к сравнению правдоподобий:

$$P(C_1|x) > P(C_2|x) \quad \text{или} \quad \log P(C_1|x) > \log P(C_2|x).$$

Подставив выражения для $P(x|C_1)$ и $P(x|C_2)$, получаем:

$$-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1}(x - \mu_1) + \log P(C_1) > -\frac{1}{2}(x - \mu_2)^T \Sigma^{-1}(x - \mu_2) + \log P(C_2).$$

Упростив это выражение, мы приходим к линейному решению:

$$w^T x + b = 0,$$

где

$$w = \Sigma^{-1}(\mu_1 - \mu_2), \quad b = -\frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2) + \log \frac{P(C_1)}{P(C_2)}.$$

Таким образом, линейное правило классификации LDA основывается на разности средних μ_1 и μ_2 , взвешенных инвертированной ковариационной матрицей Σ^{-1} .

Оптимизация LDA

Задача оптимизации LDA заключается в том, чтобы найти параметры классификатора, минимизируя ошибку классификации. Это достигается путём минимизации внутриклассовой дисперсии и максимизации межклассовой дисперсии. Внутриклассовая дисперсия характеризует разброс объектов внутри одного класса, а межклассовая дисперсия — разброс между классами. В результате, LDA позволяет найти оптимальную гиперплоскость, которая максимизирует различие между классами.

Основные шаги алгоритма LDA

1. Рассчитываем среднее для каждого класса μ_k и ковариационную матрицу для всего набора данных Σ .
2. Вычисляем вектор весов $w = \Sigma^{-1}(\mu_1 - \mu_2)$ и смещение $b = -\frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2) + \log \frac{P(C_1)}{P(C_2)}$.
3. Классифицируем объект x , вычисляя $w^T x + b$ и присваивая метку класса, соответствующую наибольшему значению.

Задача 1: Основное правило классификации

Дано два класса данных C_1 и C_2 , каждый из которых представлен наборами векторов признаков $X_1, X_2 \in \mathbb{R}^m$. Пусть векторы признаков для каждого класса распределены нормально с одинаковыми ковариационными матрицами: $\Sigma_1 = \Sigma_2 = \Sigma \in \mathbb{R}^{m \times m}$ и средними μ_1 и μ_2 .

1. Используя принцип максимизации правдоподобия, выведите линейное правило классификации в LDA для двух классов C_1 и C_2 .
2. Докажите, что это правило эквивалентно выбору гиперплоскости, которая разделяет два класса, используя линейную комбинацию признаков. Определите, как вычисляется граница между классами.

Решение:

1. Для нахождения линейного классификатора мы предполагаем, что признаки x для каждого класса следуют нормальному распределению с различными средними μ_1, μ_2 , но одинаковыми ковариационными матрицами Σ . Согласно методу максимизации правдоподобия, логарифм правдоподобия для каждого класса выглядит следующим образом:

$$\log P(C_k|x) = -\frac{1}{2} \log |\Sigma| - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log P(C_k)$$

Для классификации выбираем класс с максимальным правдоподобием, что эквивалентно выбору гиперплоскости, которая разделяет два класса. После упрощений получаем линейное правило классификации:

$$\delta(x) = w^T x + b \quad \text{где} \quad w = \Sigma^{-1}(\mu_1 - \mu_2), \quad b = -\frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2) + \log \frac{P(C_1)}{P(C_2)}$$

Гиперплоскость, разделяющая классы, определяется по линейному выражению $w^T x + b = 0$.

2. Линейное правило классификации $\delta(x)$ позволяет разделить классы с помощью гиперплоскости, где весовой вектор w пропорционален разности средних $\mu_1 - \mu_2$, а смещение b зависит от ковариационной матрицы и вероятностей классов. Это утверждение доказывается тем, что линейный классификатор LDA минимизирует ошибку классификации для нормальных распределений с одинаковыми ковариациями.

Задача 2: Оптимизация классификатора

Дано два класса данных C_1 и C_2 , каждый из которых имеет нормальное распределение признаков с одинаковыми ковариационными матрицами Σ , но с различными средними значениями μ_1 и μ_2 .

- Используя предположения о нормальности распределений и одинаковости ковариационных матриц, выведите общее правило для классификатора LDA для разделения классов C_1 и C_2 .
- Покажите, что гиперплоскость, разделяющая классы, определяется разностью средних $\mu_1 - \mu_2$ и инвертированной ковариационной матрицей Σ^{-1} . Докажите, что правило классификации LDA можно записать как линейную функцию от признаков.

Решение:

- Используя предположения о нормальности распределений и одинаковости ковариационных матриц, максимизируем правдоподобие:

$$P(x|C_1) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1} (x - \mu_1)\right)$$

$$P(x|C_2) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_2)^T \Sigma^{-1} (x - \mu_2)\right)$$

Для классификации, принимаем решение на основе сравнения логарифмов правдоподобий:

$$\log P(C_1|x) - \log P(C_2|x)$$

Упрощая выражения, получаем линейное правило классификации:

$$w^T x + b = 0 \quad \text{где} \quad w = \Sigma^{-1}(\mu_1 - \mu_2), \quad b = -\frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2) + \log \frac{P(C_1)}{P(C_2)}$$

2. Гиперплоскость, разделяющая два класса, имеет уравнение $w^T x + b = 0$, где w пропорционален разности средних $\mu_1 - \mu_2$, а b зависит от ковариационной матрицы и вероятностей классов. Это доказательство основано на том, что для нормальных распределений с одинаковыми ковариационными матрицами оптимальное решение для классификации представляет собой линейную функцию от признаков.

Задача 3: Оценка ошибки классификации

Предположим, что у нас есть линейный классификатор, полученный методом LDA, который разделяет два класса C_1 и C_2 . Пусть обучающий набор данных состоит из n объектов: $\{(x_i, y_i)\}$, где $x_i \in \mathbb{R}^m$, а $y_i \in \{-1, 1\}$ — метки классов. Классификатор работает по правилу: если $w^T x + b \geq 0$, то класс C_1 , иначе класс C_2 .

1. Докажите, что ошибка классификации на обучающих данных для классификатора, построенного методом LDA, может быть выражена как сумма индикаторов неверной классификации для каждого примера.
2. Для случая, когда классы разделены линейной гиперплоскостью, выведите верхнюю границу ошибки классификации с использованием теоремы о обобщающей способности линейных классификаторов.

Решение:

1. Ошибка классификации на обучающих данных для классификатора $h(x) = \text{sign}(w^T x + b)$ выражается как сумма индикаторов неверной классификации:

$$\text{Ошибка} = \frac{1}{n} \sum_{i=1}^n \mathbb{I}(y_i \neq \text{sign}(w^T x_i + b))$$

где $\mathbb{I}(\cdot)$ — индикатор неверной классификации.

2. Для оценки ошибки на тестовых данных используется теорема о обобщающей способности, которая дает верхнюю границу ошибки классификации через максимальное расстояние от гиперплоскости до ближайших точек обучающей выборки. Для линейных классификаторов верхняя граница ошибки может быть выражена как:

$$\text{Ошибка} \leq \frac{1}{\sqrt{n}} \cdot \left(\max_i \|x_i\| \right)$$

Эта граница зависит от структуры обучающей выборки и обеспечивает оценку ошибки классификатора на новых данных.

2.2. Метод наименьших квадратов (МНК) в общем случае

2.2.1. Про линейную регрессию и МНК

Линейная регрессия — это метод анализа данных, который используется для определения линейной зависимости между зависимой переменной y и одной или несколькими независимыми переменными x_1, x_2, \dots, x_n . Цель метода заключается в построении модели, которая минимизирует ошибку предсказания.

Метод наименьших квадратов (МНК) — это наиболее распространённый способ нахождения коэффициентов линейной регрессии. Он минимизирует сумму квадратов отклонений предсказанных значений от наблюдаемых. Таким образом, МНК позволяет определить такие коэффициенты $\beta_0, \beta_1, \dots, \beta_n$, которые обеспечивают наилучшее соответствие модели данным.

Основная идея МНК: минимизация ошибки предсказания, заданной формулой:

$$Q(\beta) = \sum_{i=1}^N (y_i - \hat{y}_i)^2,$$

где y_i — наблюдаемые значения, а \hat{y}_i — предсказанные моделью значения.

2.2.2. МНК в общем случае

Определение: В общем случае задача линейной регрессии может быть представлена в матричной форме:

$$\mathbf{y} = X\beta + \epsilon,$$

где: - y — вектор целевых значений ($N \times 1$),

- X — матрица признаков ($N \times p$),

- β — вектор коэффициентов модели ($p \times 1$),

- ϵ — вектор ошибок ($N \times 1$).

Решение задачи МНК определяется как:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}.$$

Интерпретация: Этот результат минимизирует сумму квадратов остатков $\epsilon = \mathbf{y} - X\beta$. В случае, если матрица $X^T X$ вырожденная, решение может быть некорректным или недоступным, что требует применения регуляризации.

2.2.3. Проблемы и ограничения МНК

Несмотря на простоту и эффективность, метод МНК имеет ограничения:

1. Мультиколлинеарность

- **Определение:** мультиколлинеарность возникает, когда между независимыми переменными в матрице признаков X существует сильная линейная зависимость;
- **Почему это проблема:** при мультиколлинеарности матрица $X^T X$ становится плохо обусловленной (или даже вырожденной), что затрудняет нахождение её обратной матрицы. Это может привести к неустойчивым решениям, при которых малые изменения в данных существенно изменяют значения коэффициентов.

2. Чувствительность к выбросам

- **Определение:** МНК минимизирует сумму квадратов ошибок, что делает его очень чувствительным к выбросам (аномальным точкам);
- **Почему это проблема:** выбросы имеют большое влияние на значение целевой функции $Q(\beta)$, что может привести к сильному смещению коэффициентов регрессии.

3. Нарушение предположений: МНК предполагает линейность модели, гомоскедастичность (постоянную дисперсию ошибок) и отсутствие автокорреляции.

4. Высокая вычислительная сложность

- **Определение:** МНК требует вычисления матрицы $X^T X$ и её обратной, что имеет временную сложность $O(Np^2 + p^3)O(Np^2 + p^3)$, где N — количество наблюдений, p — количество признаков;
- **Почему это проблема:** для больших наборов данных с большим количеством признаков вычислительная сложность становится значительной, что может сделать процесс обучения долгим.

2.2.4. Задачи

Задача 1: Докажите, что решение системы нормальных уравнений для метода наименьших квадратов существует и единствено, если матрица $X^T X$ положительно определённая.

Решение: Метод наименьших квадратов минимизирует квадратичную функцию:

$$Q(\beta) = \|y - X\beta\|^2 = (y - X\beta)^T(y - X\beta)$$

Рассмотрим стационарные точки функции $Q(\beta)$, задаваемые системой нормальных уравнений:

$$X^T X \beta = X^T y$$

- Условие существования решения:

Решение существует, если матрица $X^T X$ невырожденная, то есть её детерминант $\det(X^T X) \neq 0$. Это выполняется, если признаки в X линейно независимы (нет мультиколлинеарности).

- Условие единственности решения:

Если $X^T X$ положительно определённая, то:

1. $(X^T X)$ симметрична.
2. Для любого ненулевого вектора v , $v^T (X^T X) v > 0$.

Положительная определённость гарантирует, что квадратичная форма $Q(\beta)$ строго выпуклая, а значит, имеет единственную точку минимума.

Задача 2: Опишите, как изменится целевая функция метода наименьших квадратов $Q(\beta) = \|y - X\beta\|^2$, если в данных присутствует выброс. Почему минимизация квадратичной ошибки делает метод чувствительным к таким точкам?

Решение: Выбросы увеличивают квадратичную ошибку, так как вклад отклонений от линии регрессии для таких точек пропорционален квадрату расстояния. Это означает, что несколько больших ошибок могут доминировать над множеством малых, и решение будет смещено в сторону выбросов. Например, если одна ошибка вдвое больше остальных, её вклад в целевую функцию будет в четыре раза больше. Это делает МНК очень чувствительным к выбросам и может привести к неверным коэффициентам модели.

Задача 3: Реализуйте простой случай МНК для одномерной линейной регрессии, где $X = [1, 2, 3]$, $y = [2, 4, 6]$. Найдите коэффициенты β_0 и β_1 .

Решение: Для одномерного случая формула нормальных уравнений имеет вид:

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

Подставляя значения:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \quad y = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}.$$

Рассчитаем:

$$X^T X = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}, \quad X^T y = \begin{bmatrix} 12 \\ 28 \end{bmatrix}.$$

Получаем:

$$\hat{\beta} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 12 \\ 28 \end{bmatrix}.$$

После вычислений $\hat{\beta} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$. Таким образом, уравнение линейной регрессии: $y = 2x$.

2.3. Вероятностные функции потерь

2.3.1. Принцип максимума правдоподобия

Предположим, что $X \times Y$ - вероятностное пространство с некоторой плотностью совместного распределения пары объект-ответ $p(x, y)$. Пусть X^l - простая (i.i.d., или independent identically distributed - независимо из одного и того же распределения) выборка: $(x_i, y_i)^l$, порожденная $p(x, y)$.

Задача состоит в том, чтобы оценить по выборке X^l плотность распределения $p(x, y)$. Получив оценку плотности, то с его помощью возможна классификация объекта x - мы сможем вычислять вероятность класса y для любого объекта x .

Далее введем **параметризацию** плотности, взяв за основу формулу условной вероятности: $p(x, y) = P(y|x, w)p(x)$, где $p(x, y)$ - искомая плотность; $P(y|x, w)$ - модель условной вероятности класса с параметром w ; $p(x)$ - непараметризуемое распределение в пространстве X . Возможны и другие способы введения параметризации, однако сейчас будет рассматриваться именно этот.

Так как выборка простая, то плотность порожденной выборки является произведением плотностей отдельных порожденных пар (x_i, y_i) . Таким образом, $\prod_{i=1}^l p(x_i, y_i)$ - **правдоподобие данных**.

В качестве критерия оптимизации возьмем один из фундаментальных методов в математической статистике - **принцип максимума правдоподобия**:

$$\prod_{i=1}^l p(x_i, y_i) = \prod_{i=1}^l P(y_i|x_i, w)p(x_i) \rightarrow \max_w$$

В силу того, что $p(x_i)$ - сомножитель, не зависящий от w , от него можно избавиться.

$$\prod_{i=1}^l P(y_i|x_i, w) \rightarrow \max_w$$

Поскольку стоит задача максимизации, критерий в виде произведения по всем объектам выборки неудобен. Чтобы избавиться от произведения, прологарифмируем критерий. Логарифм - монотонная функция, поэтому несущественно, что мы оптимизируем - функционал или его логарифм.

Логарифм правдоподобия (log-likelihood, log-loss):

$$L(w) = \sum_{i=1}^l \log P(y_i|x_i, w) \rightarrow \max_w$$

2.3.2. Связь правдоподобия и аппроксимации эмпирического риска

Посмотрим на одну и ту же задачу классификации на два класса $Y = \{+1; -1\}$ с двух сторон:

1. $P(y|x, w)$ - вероятностная модель классификации.
2. $g(x, w)$ - разделяющая (дискриминантная) функция - геометрический взгляд на задачу.

Критерии, возникающие в разных случаях:

1. *Максимизация правдоподобия* (Maximum Likelihood):

$$L(w) = \sum_{i=1}^l \log P(y_i|x_i, w) \rightarrow \max_w;$$

2. *Минимизация аппроксимированного эмпирического риска*:

$$Q(w) = \sum_{i=1}^l L(y_i g(x_i, w)) \rightarrow \min_w;$$

Здесь $L(M)$ - функция потерь; $y_i g(x_i, w)$ - значение отступа.

Оба критерия представляют собой оптимизацию некоторой величины, являющейся суммой по всем объектам выборки, по параметру. Каждое слагаемое суммы зависит только от одного объекта.

Эти два принципа **эквиваленты**, если положить:

$$-\log P(y_i|x_i, w) = L(y_i g(x_i, w))$$

2.3.3. Вероятностный смысл регуляризации

Рассматривается двухуровневая модель порождения данных:

1. $P(y|x, w)$ - вероятностная модель данных.
2. $p(w; y)$ - априорное распределение параметров модели.
3. γ - вектор гиперпараметров.

Теперь не только появление выборки X^l , но и модели w полагается стохастическим.

Совместное правдоподобие данных и модели по формуле условной плотности:

$$p(X^l, w) = p(X^l|w)p(w; \gamma)$$

Принцип максимума апостериорной вероятности (Maximum a Posteriori Probability, MAP):

$$L(w) = \log p(X^l, w) = \sum_{i=1}^l \log P(y_i|x_i, w) + \log p(w; \gamma) \rightarrow \max_w$$

Таким образом, слагаемое $-\log p(w; \gamma)$ является **регуляризатором** с вероятностной точки зрения.

2.3.4. Задачи

Задача 1.

Какому регуляризатору соответствует апостериорное распределение параметров модели, имеющее вид распределения Гаусса?

Решение:

Веса w_j независимы, $E[w_j] = 0$, $D[w_j] = C$.

$$p(w; C) = \frac{1}{(2\pi C)^{\frac{n}{2}}} \exp\left(-\frac{\|w^2\|}{2C}\right), \|w^2\| = \sum_{j=1}^n w_j^2$$

$$-\ln p(w; C) = \frac{1}{2C} \|w^2\| + const$$

От константы можно избавиться:

$$-\ln p(w; C) = \frac{1}{2C} \|w^2\|$$

Получаем квадратичный (L_2) регуляризатор. C является гиперпараметром, $\tau = \frac{1}{C}$ - коэффициент регуляризации.

Задача 2.

Какому виду регуляризации соответствует апостериорное распределение параметров модели, имеющее вид распределения Лапласа?

Решение:

Веса w_j независимы, $E[w_j] = 0$, $D[w_j] = C$.

$$p(w; C) = \frac{1}{(2C)^n} \exp\left(-\frac{\|w\|}{C}\right), \|w\| = \sum_{j=1}^n |w_j|$$

$$-\ln p(w; C) = \frac{1}{C} \|w\| + const$$

От константы можно избавиться:

$$-\ln p(w; C) = \frac{1}{C} \|w\|$$

Получаем абсолютный (L_1) регуляризатор. C является гиперпараметром, $\tau = \frac{1}{C}$ - коэффициент регуляризации.

Задача 3.

Найти вид апостериорного распределения параметров модели, соответствующий регуляризатору Elastic Net:

$$R(w; C_1; C_2) = \frac{1}{C_1} \sum_{i=1}^l |w_i| + \frac{1}{2C_2} \sum_{i=1}^l w_i^2$$

Решение:

$$-\ln p(w; C_1; C_2) = \frac{1}{C_1} \|w\| + \frac{1}{2C_2} \|w^2\|$$

С точностью до умножения на константу, получим:

$$p(w; C_1; C_2) = \exp\left(-\frac{\|w\|}{C_1}\right) \exp\left(-\frac{\|w^2\|}{2C_2}\right)$$

$$p(w; C_1; C_2) = \exp\left(-\frac{\|w\|}{C_1} - \frac{\|w^2\|}{2C_2}\right)$$

2.4. Методы оценки и проверки моделей

Оценка и проверка моделей являются важными этапами в построении машинного обучения. Эти методы позволяют определить, насколько хорошо модель обучается на данных и обобщает свои выводы на новых, ранее невиданных данных. Ниже представлены основные подходы к оценке и проверке моделей.

Кросс-валидация

Кросс-валидация — это метод проверки модели, который заключается в разбиении данных на несколько подвыборок (folds). Основные виды кросс-валидации:

- **K-блочная кросс-валидация** (K-fold cross-validation): данные делятся на K частей, и обучение проводится на $K - 1$ частях, а тестирование на оставшейся части. Процесс повторяется K раз, чтобы каждая часть данных использовалась для тестирования.
- **Leave-One-Out (LOO)**: частный случай кросс-валидации, где тестовая выборка состоит из одного наблюдения, а оставшиеся используются для обучения. Этот метод особенно полезен для небольших наборов данных, но может быть вычислительно затратным.
- **Stratified K-fold**: разновидность K-блочной кросс-валидации, где сохраняются пропорции классов в каждой из выборок, что особенно важно для несбалансированных данных.

Кросс-валидация помогает уменьшить риск переобучения и получить более надежные оценки качества модели.

Метрики качества

Для оценки модели применяются различные метрики, выбор которых зависит от задачи (регрессия или классификация):

- Для регрессии:

- Среднеквадратичная ошибка (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

Эта метрика чувствительна к выбросам, так как большие ошибки квадратично увеличивают значение MSE.

- Средняя абсолютная ошибка (MAE):

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|.$$

MAE более устойчива к выбросам, так как ошибки учитываются линейно.

- Коэффициент детерминации (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

Значение R^2 показывает, какую долю дисперсии целевой переменной объясняет модель.

- Для классификации:

- Точность (Accuracy):

$$\text{Accuracy} = \frac{\text{Количество верных предсказаний}}{\text{Общее количество наблюдений}}.$$

Используется для сбалансированных данных.

- Precision, Recall и F_1 -мера:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Эти метрики особенно важны для несбалансированных классов.

- ROC-кривые и площадь под кривой (AUC): ROC-кривая показывает соотношение TPR (True Positive Rate) и FPR (False Positive Rate) при различных порогах классификации, а AUC характеризует общую способность модели различать классы.

Разделение данных

Одним из простейших методов проверки модели является разделение данных на обучающую и тестовую выборки. Чаще всего используется пропорция 70%/30% или 80%/20%. Однако этот метод имеет недостаток: если данные случайно разделены неудачно, это может привести к неверным оценкам качества модели.

Для улучшения оценки часто используется тройное разбиение данных:

- **Обучающая выборка** (Train set): используется для обучения модели.
- **Валидационная выборка** (Validation set): используется для подбора гиперпараметров и предотвращения переобучения.
- **Тестовая выборка** (Test set): применяется для окончательной оценки модели.

Проблемы и рекомендации

- **Переобучение:** Если модель слишком сложная, она может хорошо работать на обучающих данных, но плохо обобщаться на тестовые. Регуляризация и кросс-валидация помогают бороться с этой проблемой.
- **Недообучение:** Слишком простые модели могут не учитывать важные зависимости в данных. Следует выбирать более сложные модели или добавлять новые признаки.
- **Сбалансированность данных:** Для несбалансированных данных рекомендуется использовать метрики, такие как Precision, Recall или AUC.

Задачи

- Для датасета (сгенерируйте сами) с 10 000 объектов примените 5-блочную кросс-валидацию. Опишите, как будут разделены данные на обучающие и тестовые выборки на каждом шаге. Рассчитайте среднее значение метрики Accuracy, если на каждом шаге она принимает значения: 0.85, 0.87, 0.86, 0.84, 0.88.
- Для задачи регрессии выберите подходящую метрику качества из MSE, MAE или R^2 и объясните, почему вы сделали такой выбор. Рассчитайте её значение для предсказаний $\hat{y} = [3, 5, 2, 7]$ и истинных значений $y = [3, 5, 4, 6]$.
- В задаче классификации модель предсказала 100 объектов, из которых 70 были классифицированы правильно, 20 — ложно положительными, а 10 — ложно отрицательными. Рассчитайте Precision, Recall и F_1 -меру.

2.5. Линейная классификация

2.5.1. Постановка задачи

Представим, что у нас есть множество объектов X и мы хотим каждому объекту сопоставить некоторый класс. Например, есть набор клиентов банков и хотим понять кому из них стоит выдавать кредит, а кому нет. Формализуя имеем отображение из множества объектов X в некоторое множество классов:

$$X \rightarrow \{0, 1, \dots, K\}, \text{ где } 0, \dots, K \text{ — номера классов.}$$

Такая задача называется задачей классификации.

Будем искать решения в виде некоторой линейной функции $y = w_1x_1 + \dots + w_nx_n + w_0$, где y — целевая переменная (target), (x_1, \dots, x_n) вектор признаков объекта выборки (features), вектор $w = (w_1, \dots, w_n)$ называют вектором весов (weights), а w_0 — свободный коэффициентом, или сдвигом (bias). Более компактно можно записать в виде: $y = \langle x, w \rangle + w_0$. Теперь наша задача свелась к подбору конкретного вектора (w_0, w_1, \dots, w_n) , задающего наше отображение.

Не сложно заметить, что сейчас наша функция имеет некоторое числовое значение, а мы бы хотели категориальное. Поправим это сказав, что если $y > 0$ это один класс, иначе — второй. $y = sign(\langle x, w \rangle + w_0)$. Многокатегориальный случай рассмотрим чуть позже.

Итого имеем некоторую разделяющую прямую (или гиперплоскость в случае больших размерностей) которая делит наше пространство на два класса.

В идеальном случае найдется плоскость, которая разделит классы, так чтобы первый оказался с одной стороны, а второй с другой. В таком случае выборка называется линейно разделимой, но чаще всего так получаться не будет.

2.5.2. Многоклассовая классификация

В случае если у нас больше двух классов, воспользуемся набором бинарных классификаторов. Разберем два самых популярных способа это сделать - one-vs-all и all-vs-all.

Один против всех

Обучим K линейных классификаторов $b_1(x), \dots, b_k(x)$, выдающих оценки принадлежности классам $1, \dots, K$ соответственно. В случае с линейными моделями эти классификаторы будут иметь вид:

$$b_k(x) = \text{sign}(\langle w_k, x \rangle + w_{0k}).$$

Каждый классификатор будем обучать отличать k -й класс от все остальных. Тогда логично, чтобы итоговый классификатор выдавал класс, соответствующий самому уверенному из бинарных алгоритмов. Уверенность в каком-то смысле можно измерить с помощью значений линейных функций:

$$y(x) = \text{argmax}_k (\langle w_k, x \rangle + w_{0k}).$$

Все против всех

Обучим C_K^2 классификаторов $b_{ij}(x)$, $i, j = 1, \dots, K, i \neq j$. Для линейной модели они будут иметь вид:

$$b_{ij}(x) = \text{sign}(\langle w_{ij}, x \rangle + w_{0,ij}).$$

Каждый классификатор будем обучать только на объектах классов i, j . Тогда наш классификатор будет выдавать для любого объекта либо класс i , либо класс j . Чтобы получить итоговый класс, пусть каждый классификатор проголосует за свой класс, а в качестве ответа выберем тот класс, за который будет больше голосов:

$$y(x) = \text{argmax}_k \sum_{i=1}^K \sum_{i \neq j} [b_{ij}(x) = k].$$

2.5.3. Задачи

Задача 1.

Почему существует линейный модели, но нет "полиномиальных" или "логарифмических" хотя зависимости между данными бывают довольно сложными?

Решение: На самом деле если мы подозреваем какую-то более сложную зависимость данных, то мы можем ввести новые признаки со сложной зависимостью, получив задачу в большей размерности, но все еще решаемую линейной моделью.

Задача 2.

Что делать если есть категориальный признаки, то есть он принимает какие-то значения, не обязательно являющиеся числами?

Решение: Линейная модель работает только с числовыми признаками, поэтому категориальный придется закодировать. Например, можно использовать технику one-hot encoding. Пусть наш признак может принимать k значений. Заменим его на k новых признаков, для которых один принимает единицу (в зависимости от значения категориального признака), а все остальные ноль.

Задача 2.

Каким еще может быть ответ у классифицирующей модели, помимо простой принадлежности какому-то классу?

Решение: Модель может выдавать не принадлежность классу, а вероятность с которой она бы отнесла его к соответствующему классу. Такой подход называется логистической регрессий и очень часто встречается, поскольку нам часто хочется знать насколько модель уверена в своем выборе.

2.6. Линейные методы для интерпретации моделей

Интерпретация моделей машинного обучения играет важную роль в задачах, где требуется понять, какие факторы влияют на предсказания модели. Линейные модели, благодаря своей простой структуре, являются одними из наиболее интерпретируемых алгоритмов. В этой главе мы рассмотрим основные подходы к интерпретации линейных моделей.

Значимость коэффициентов

В линейных моделях каждый коэффициент отражает вклад соответствующего признака в итоговый результат. Для интерпретации коэффициентов можно использовать следующие подходы:

- **Проверка знаков коэффициентов:** Знак коэффициента указывает на направление влияния признака. Положительный знак говорит о прямой зависимости, отрицательный — об обратной.
- **Нормализация признаков:** Перед обучением модели признаки часто стандартизируют (приводят к нулевому среднему и единичной дисперсии), чтобы значения коэффициентов можно было сравнивать между собой.

- **Статистическая значимость:** Для проверки важности коэффициентов можно использовать t-тесты, вычисляя р-значения для каждого коэффициента.

Доверительные интервалы

Для оценки надежности коэффициентов модели используются доверительные интервалы. Они позволяют определить диапазон значений, в котором с определённой вероятностью находится истинное значение коэффициента.

$$CI_j = \hat{\beta}_j \pm t_{\alpha/2} \cdot SE_j,$$

где $\hat{\beta}_j$ — оценка коэффициента, SE_j — стандартная ошибка, $t_{\alpha/2}$ — квантиль распределения Стьюдента.

Интервалы, включающие ноль, указывают на возможную незначимость соответствующего признака.

Можем на примере одномерного случая задачи линейной регрессии убедится в значимости главного параметра. Пусть в нашем распоряжении есть набор точек:

$$x = \{1, 2, 3, 4, 5\}; \quad y = \{2, 3, 5, 7, 11\}. \quad (2.1)$$

Задача заключается в построении линейной регрессии $y = \beta_0 + \beta_1 x$, а также определении доверительного интервала для коэффициента β_1 при $p = 0.05$. Используя формулы для коэффициентов МНК можно легко определить стандартное отклонение для β_1 ; подставив все в формулу выше получим границы доверительного интервала для β_1 :

$$\begin{aligned} CI_{\text{lower}} &= \beta_1 - t \cdot \text{SE}(\beta_1) = 2.2 - 3.182 \cdot 0.275 \approx 1.32, \\ CI_{\text{upper}} &= \beta_1 + t \cdot \text{SE}(\beta_1) = 2.2 + 3.182 \cdot 0.275 \approx 3.08. \end{aligned}$$

Сравнивая значение β_1 с границами доверительного интервала пониманием что β_1 является значимым параметром построенной линейной модели.

Методы визуализации влияния признаков

Для упрощения интерпретации линейных моделей часто используют визуальные методы. Некоторые из них:

- **Partial Dependence Plots (PDP):** Помогает визуализировать влияние одного или нескольких признаков на предсказания модели, усредняя влияние других признаков. Крутизна кривой в PDP покажет силу влияния этого признака.

Строгое математическое определение PD:

$$pd_{X_S}(x_S) \stackrel{\text{def}}{=} \mathbb{E}_{X_C} [f(x_S, X_C)] = \int f(x_S, x_C) p(x_C) dx_C, \quad (2.2)$$

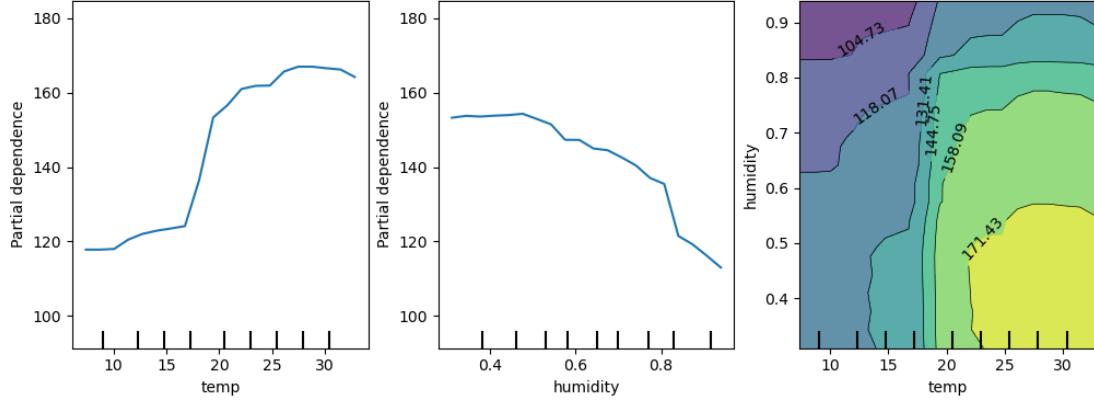


Рис. 2.4. Пример PDP в задаче числа суточной аренды велосипедов.

- **Влияние признаков (SHAP и LIME):** Современные методы, такие как SHAP (SHapley Additive exPlanations) и LIME (Local Interpretable Model-agnostic Explanations), позволяют объяснить предсказания модели для отдельных объектов.

Влияние мультиколлинеарности

Мультиколлинеарность возникает, когда между признаками существует сильная корреляция. Это может приводить к нестабильным оценкам коэффициентов. Для её диагностики используют:

- **Фактор инфляции дисперсии (VIF):**

$$VIF_j = \frac{1}{1 - R_j^2},$$

где R_j^2 — коэффициент детерминации при регрессии j -го признака на остальные.

Коэффициент детерминации R^2

Коэффициент детерминации (R^2) — это метрика, используемая для оценки качества модели регрессии. Он показывает, какую долю дисперсии зависимой переменной (Y) модель способна объяснить.

Формула

Коэффициент R^2 рассчитывается следующим образом:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}},$$

где:

- $SS_{\text{res}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ — остаточная сумма квадратов (сумма квадратов отклонений между реальными значениями y_i и предсказанными значениями \hat{y}_i);
- $SS_{\text{tot}} = \sum_{i=1}^n (y_i - \bar{y})^2$ — общая сумма квадратов (сумма квадратов отклонений реальных значений y_i от среднего значения \bar{y});
- \bar{y} — среднее значение зависимой переменной:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i.$$

Интерпретация

Значение R^2 находится в диапазоне от 0 до 1:

- $R^2 = 1$: модель идеально объясняет дисперсию данных.
- $R^2 = 0$: модель не объясняет дисперсию данных (эквивалентно случайному предсказанию среднего значения \bar{y}).
- $R^2 < 0$: модель объясняет дисперсию хуже, чем случайное предсказание среднего значения.

Для задачи определения значимого признака, каждый признак по очереди берется в качестве независимой переменной и для каждого формулируется задача регрессии:

$$X_j = \beta_0 + \sum_{i \neq j}^n \beta_i * X_i \tag{2.3}$$

Таким образом высокое значение VIF для отдельной переменной говорит об ее избыточности в обучаемой выборке, т.к. информация о ней (для линейной модели) почти полностью содержится в других признаках. На практике при значениях $VIF > 10$ говорят о мультиколлинеарности данных и избавляются от таких признаков или применяют L1, L2 регуляризации.

Практические примеры

Рассмотрим несколько примеров использования линейных методов для интерпретации:

- **Прогнозирование цен на недвижимость:** Вклад признаков, таких как площадь, расположение и возраст здания, можно интерпретировать напрямую через коэффициенты линейной модели.
- **Медицинская диагностика:** В задачах прогнозирования риска заболеваний линейные модели позволяют определить, какие факторы (например, возраст, уровень холестерина) оказывают наибольшее влияние.

Задача 1: Доверительный интервал β_1 .

Требуется найти доверительный интервал для β_1 , используя точки из этой главы.

Решение:

Для коэффициентов линейной регрессии используем формулы метода наименьших квадратов:

$$\begin{aligned}\beta_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \\ \beta_0 &= \bar{y} - \beta_1 \bar{x},\end{aligned}$$

где \bar{x} и \bar{y} — средние значения x и y соответственно.

Подставляя значения:

$$\begin{aligned}\bar{x} &= 3, \quad \bar{y} = 5.6, \\ \beta_1 &= \frac{\sum (x_i - 3)(y_i - 5.6)}{\sum (x_i - 3)^2} = 2.2, \quad \beta_0 = 5.6 - 2.2 \cdot 3 = -0.4.\end{aligned}$$

Средняя квадратная ошибка (MSE) определяется как:

$$\text{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - 2},$$

где \hat{y}_i — предсказанное значение для y_i . Подставляя значения:

$$\text{MSE} = \frac{(2 - 1.8)^2 + (3 - 3.6)^2 + (5 - 5.8)^2 + (7 - 8)^2 + (11 - 10.2)^2}{5 - 2} = 0.76.$$

Стандартная ошибка для β_1 :

$$\text{SE}(\beta_1) = \sqrt{\frac{\text{MSE}}{\sum_{i=1}^n (x_i - \bar{x})^2}} = \sqrt{\frac{0.76}{10}} = 0.275.$$

Квантиль распределения Стьюдента для уровня доверия 95% и $n - 2 = 3$ степеней свободы:

$$t_{0.025,3} \approx 3.182.$$

Границы доверительного интервала:

$$\begin{aligned} CI_{\text{lower}} &= \beta_1 - t \cdot \text{SE}(\beta_1) = 2.2 - 3.182 \cdot 0.275 \approx 1.32, \\ CI_{\text{upper}} &= \beta_1 + t \cdot \text{SE}(\beta_1) = 2.2 + 3.182 \cdot 0.275 \approx 3.08. \end{aligned}$$

Иллюстрация: На рисунке показана линия регрессии (красная), а также линии, соответствующие границам доверительного интервала (зелёная и оранжевая).

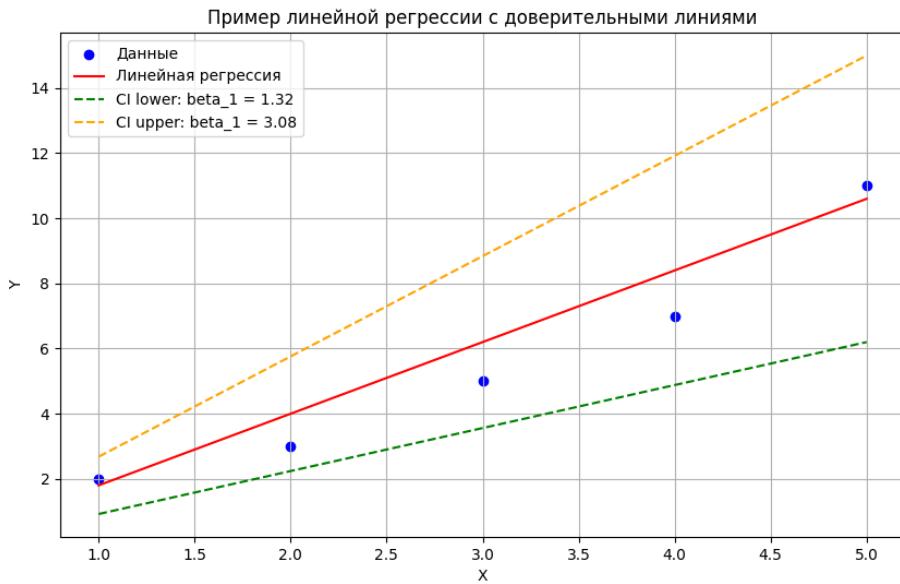


Рис. 2.5. Пример линейной регрессии с доверительными интервалами.

Задача 2: Интерпретация РДР.

Глядя на ??, проведите анализ значимости изображенных признаков - температуры и влажности воздуха на общее количество арендаемых в день велосипедов. Какое влияние оказывают эти признаки на предсказываемую величину?

Решение:

- **Температура** оказывает значительное положительное влияние на предсказываемую величину, особенно в диапазоне 20-30 градусов
- **Влажность** оказывает отрицательное влияние: по мере её увеличения предсказания снижаются.
- **Совместное влияние** показало, что оптимальная комбинация — высокая температура и низкая влажность.

Оба эти признака являются значимыми для построения предсказательной модели, т.к. линии графиков РД имеют крутой наклон. Также видно что наклон на графике не постоянный и целесообразно задуматься об использовании нелинейной регрессии или градиентного бустинга.

Задача 3: Определение мультиколлинеарности переменных по VIF.

Сделать вывод о значимости переменных X_2 и X_3

Таблица 2.2. Тренировочная выборка

Y	X_1	X_2	X_3
1	1	2	3
2	2	4	6
3	3	6	9
4	4	8	12
5	5	10	15

Решение:

Не трудно видеть что:

$$X_2 = 2 * X_1 \quad X_3 = 3 * X_1 \quad (2.4)$$

Таким образом оба этих признака линейно зависимы и никакой информации в себе не несут. При использовании формул для VIF, можно также убедится

$$VIF_2 - > \infty \quad VIF_3 - > \infty \quad (2.5)$$

2.7. Многоклассовая классификация

Задача многоклассовой классификации решает проблему нахождения принадлежности объекта к одному из K классов: $Y = \{1, 2, \dots, K\}$. В этом разделе будут разобраны некоторые из самых популярных подходов: one-vs-all, all-vs-all и многоклассовая логистическая регрессия. В качестве примера возьмем датасет из хендбука Яндекса, продемонстрированный на Рис. ??.

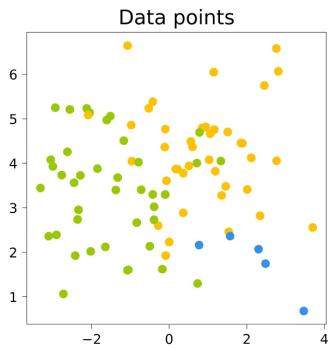


Рис. 2.6. Пример датасета задачи многоклассовой классификации

2.7.1. Один против всех (one-versus-all)

Обучим K линейных классификаторов $b_1(x), \dots, b_K(x)$, выдающих оценки принадлежности классам $1, \dots, K$ соответственно. В случае с линейными моделями эти классификаторы будут иметь вид $b_k(x) = \text{sign}(\langle w_k, x \rangle + w_{0k})$

Классификатор с номером k будем обучать по выборке $(x_i, 2\mathbb{I}[y_i = k] - 1)_{i=1}^\ell$; иными словами, мы учим классификатор отличать k -й класс от всех остальных.

Логично, чтобы итоговый классификатор выдавал класс, соответствующий самому уверененному из бинарных алгоритмов. Уверенность можно в каком-то смысле измерить с помощью значений линейных функций:

$$a(x) = \arg \max_{k \in Y} (\langle w_k, x \rangle + w_{0k})$$

Давайте посмотрим, что даст этот подход применительно к нашему датасету. Обучим три линейных модели, отличающих один класс от остальных (Рис. ??).

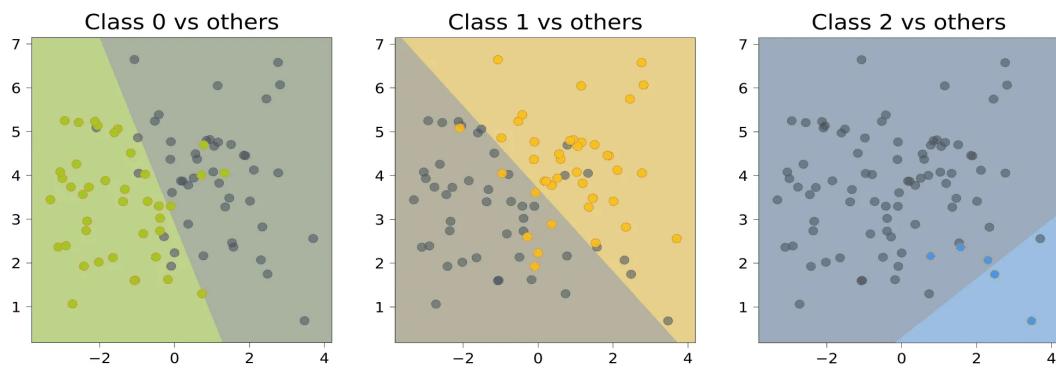


Рис. 2.7. Каждая из трех моделей, отделяющие свои классы

Теперь сравним значения линейных функций на Рис. ??.

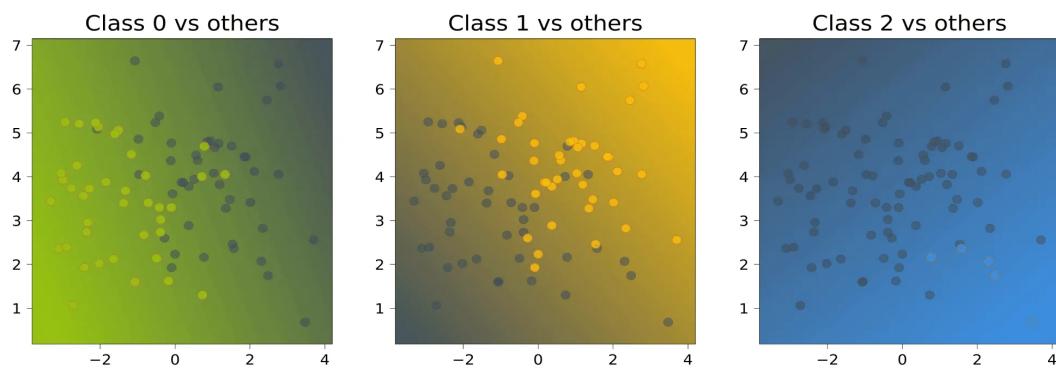


Рис. 2.8. Значения функций моделей

и для каждой точки выберем тот класс, которому соответствует большее значение, то есть самый «уверенный» классификатор, и изобразим это на Рис. ??

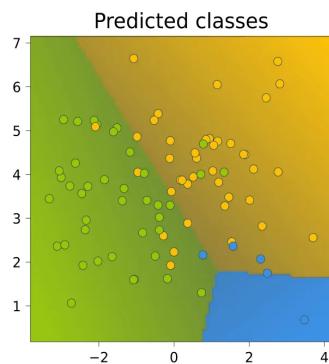


Рис. 2.9. Итог разделения пространства на классы

Хочется сказать, что самый маленький класс «обидели».

Проблема данного подхода заключается в том, что каждый из классификаторов $b_1(x), \dots, b_K(x)$ обучается на своей выборке, и значения линейных функций $\langle w_k, x \rangle + w_{0k}$ или, проще говоря, "выходы" классификаторов могут иметь разные масштабы. Из-за этого сравнивать их будет неправильно. Нормировать вектора весов, чтобы они выдавали ответы в одной и той же шкале, не всегда может быть разумным решением: так, в случае с SVM веса перестанут являться решением задачи, поскольку нормировка изменит норму весов.

Задача

Подумайте, какой простой двумерный датасет особенно плохо работает с данным подходом с линейным классификатором (что вызвано исключительно характером распределения классов, а не выбросами, количеством данных и т.п.).

Ответ

Примером такого датасета могут послужить три класса, такие что один находится между двумя другими, как на Рис. ???. В этом случае нельзя полагаться на значения разделяющей функции, так как, например, зеленый класс лежит сильно дальше границы желтого и синего классов, чем сам желтый класс. Поэтому, желто-синий классификатор будет относить зеленые точки к желтому цвету с большей уверенностью, чем желтые.

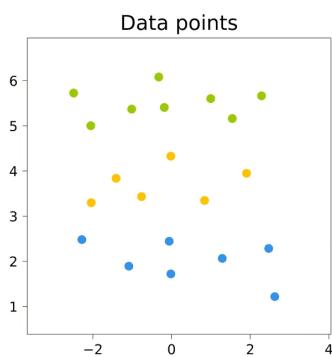


Рис. 2.10. Контрпример к one-vs-all

2.7.2. Все против всех (all-versus-all)

Обучим C_k^2 классификаторов $a_{ij}(x), i, j = 1, \dots, K, i \neq j$. Например, в случае с линейными моделями эти модели будут иметь вид

$$a_{ij}(x) = \text{sign}(\langle w_{ij}, x \rangle + w_{0,ij})$$

Классификатор $a_{ij}(x)$ будем настраивать по подвыборке $X_{ij} \subset X$, содержащей только объекты классов i и j . Соответственно, классификатор $a_{ij}(x)$ будет выдавать для любого объекта либо класс i , либо класс j . Проиллюстрируем это для нашей выборки на Рис. ??

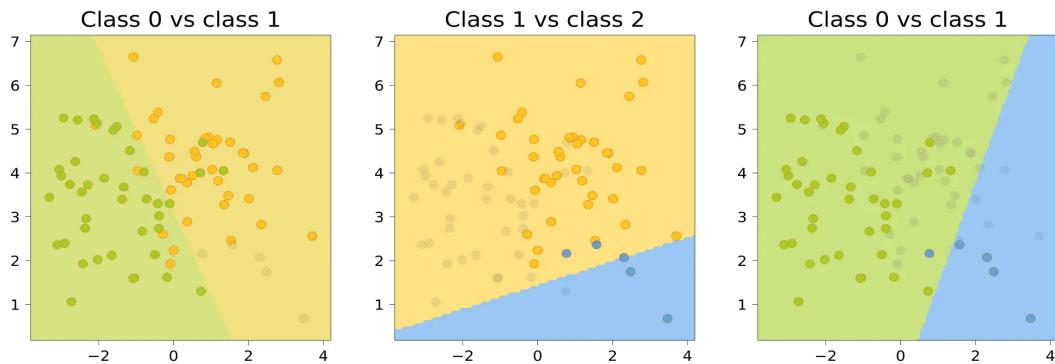


Рис. 2.11. Классификаторы, разделяющие по два класса

Чтобы классифицировать новый объект, подадим его на вход каждого из построенных бинарных классификаторов. Каждый из них проголосует за свой класс; в качестве ответа выберем тот класс, за который наберется больше всего голосов:

$$a(x) = \arg \max_{k \in Y} \sum_{i=1}^K \sum_{j \neq i} \mathbb{I}[a_{ij}(x) = k]$$

Для нашего датасета получается следующая картинка на Рис. ??

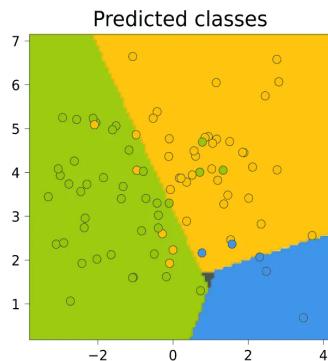


Рис. 2.12. Итоговый результат для all-vs-all

Обратите внимание на серый треугольник на стыке областей. Это точки, для которых голоса разделились (в данном случае каждый классификатор выдал какой-то свой класс, то есть у каждого класса было по одному голосу). Для этих точек нет явного способа выдать обоснованное предсказание.

2.7.3. Многоклассовая логистическая регрессия

Некоторые методы бинарной классификации можно напрямую обобщить на случай многих классов. Выясним, как это можно проделать с логистической регрессией.

В логистической регрессии для двух классов мы строили линейную модель $b(x) = \langle w, x \rangle + w_0$

а затем переводили её прогноз в вероятность с помощью сигмоидной функции $\sigma(M) = \frac{1}{1+\exp(-M)}$. Допустим, что мы теперь решаем многоклассовую задачу и построили K линейных моделей

$$b_k(x) = \langle w_k, x \rangle + w_{0k}$$

каждая из которых даёт оценку принадлежности объекта одному из классов. Как преобразовать вектор оценок $(b_1(x), \dots, b_K(x))$ в вероятности? Для этого можно воспользоваться оператором $\text{softmax}(z_1, \dots, z_k)$,

Задача

Подумайте, как должен выглядеть этот оператор, зная, что в случае двух классов он совпадает с сигмоидой, принимает значения от нуля до единицы и равен вектору вероятностей принадлежности каждому из классов.

Ответ

$$\text{softmax}(z_1, \dots, z_k) = \left(\frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)}, \dots, \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)}, \right)$$

В этом случае вероятность k -го класса будет выражаться как

$$P(y = k|x, w) = \frac{\exp(\langle w_k, x \rangle + w_{0k})}{\sum_{j=1}^K \exp(\langle w_j, x \rangle + w_{0j})}$$

Обучать эти веса предлагается с помощью метода максимального правдоподобия: так же, как и в случае с двухклассовой логистической регрессией:

$$\sum_{i=1}^{\ell} \log P(y = y_i|x_i, w) \rightarrow \max_{w_1, \dots, w_K}$$

Задача

Подумайте, какой существует простой многоклассовый (нелинейный) классификатор, использующий только расстояние между объектами и не требующий обучения (у него отсутствуют параметры, есть только гиперпараметр).

Ответ

Таким классификатором является KNN, или метод k ближайших соседей. Этот алгоритм причисляет объект к самому популярному классу среди его k ближайших соседей.

2.8. Логистическая регрессия

Что такое логистическая регрессия?

Логистическая регрессия — это статистический метод, используемый для классификации объектов на два или более классов. Она применяется, когда целевая переменная (y) принимает дискретные значения, чаще всего $y \in \{0, 1\}$. Основной задачей логистической регрессии является предсказание вероятности принадлежности объекта к определённому классу. Примеры таких задач: Классификация писем на спам и не спам, предсказание вероятности заболевания на основе медицинских данных, прогноз поступления студента на основе экзаменационных баллов.

Идея логистической регрессии

Логистическая регрессия основывается на линейной регрессии, но с одной важной модификацией: вместо того, чтобы предсказывать значения на бесконечном числовом отрезке, она ограничивает прогнозируемые значения интервалом $[0, 1]$. Это достигается с помощью логистической функции, которая преобразует линейное выражение в вероятность.

Математическая формулировка

Сначала вычисляется линейная комбинация входных признаков:

$$z = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n,$$

где: - b_0 — свободный член (интерсепт), - b_1, b_2, \dots, b_n — коэффициенты модели, - x_1, x_2, \dots, x_n — входные признаки.

Логистическая функция (или сигмоида) преобразует z в значение вероятности $P(y = 1)$:

$$P(y = 1) = \frac{1}{1 + e^{-z}} = \sigma(z)$$

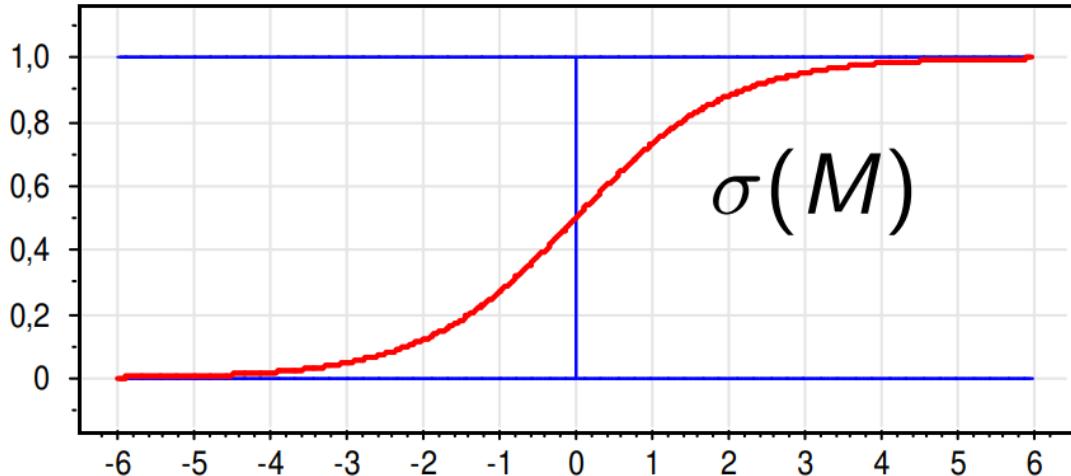


Рис. 2.13. График сигмоидной функции

Вероятность принадлежности к классу $y = 0$ вычисляется как:

$$P(y = 0) = 1 - P(y = 1).$$

Если вероятность $P(y = 1) \geq 0.5$, объект относят к классу $y = 1$; иначе — к классу $y = 0$.

Чтобы упростить обучение модели, применяется логарифмическая трансформация. Логарифмические шансы (логиты) определяются как:

$$\text{logit}(P) = \ln \left(\frac{P(y = 1)}{1 - P(y = 1)} \right).$$

Так как $P(y = 1) = \frac{1}{1+e^{-z}}$, подставим:

$$\ln \left(\frac{P(y = 1)}{1 - P(y = 1)} \right) = z = b_0 + b_1 x_1 + \dots + b_n x_n.$$

Обучение логистической регрессии

1. Функция правдоподобия

Вероятность $P(y)$ для одного наблюдения записывается как:

$$P(y) = P(y=1)^y \cdot (1 - P(y=1))^{1-y}.$$

Для всех наблюдений функция правдоподобия равна произведению вероятностей:

$$L(\mathbf{b}) = \prod_{i=1}^m P(y_i|x_i).$$

2. Логарифм правдоподобия

Чтобы упростить вычисления, берётся логарифм функции правдоподобия:

$$\ell(\mathbf{b}) = \sum_{i=1}^m [y_i \ln(P(y_i)) + (1 - y_i) \ln(1 - P(y_i))].$$

3. Максимизация логарифма правдоподобия

Коэффициенты $\mathbf{b} = [b_0, b_1, \dots, b_n]$ определяются так, чтобы логарифм правдоподобия был максимальен. Это достигается методами оптимизации, например, градиентным спуском.

Задача 1: Голосование на выборах

Условие: Политическая партия пытается спрогнозировать, проголосует ли человек за неё, исходя из возраста x_1 и политической активности x_2 (в часах участия в митингах за год). Известны данные:

Человек 1: $x_1 = 22, x_2 = 15$, проголосовал ($y = 1$).

Человек 2: $x_1 = 45, x_2 = 2$, не проголосовал ($y = 0$).

Человек 3: $x_1 = 30, x_2 = 10$, проголосовал ($y = 1$).

Определите, проголосует ли человек с $x_1 = 35$ и $x_2 = 8$. Используйте коэффициенты $b_0 = -8, b_1 = 0.2, b_2 = 0.5$.

Решение:

1. Записываем логистическую функцию:

$$P(y=1) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}}.$$

2. Подставляем значения $x_1 = 35$, $x_2 = 8$:

$$z = b_0 + b_1x_1 + b_2x_2 = -8 + 0.2 \cdot 35 + 0.5 \cdot 8 = -8 + 7 + 4 = 3.$$

3. Вычисляем вероятность:

$$P(y = 1) = \frac{1}{1 + e^{-3}} \approx 0.9526.$$

4. Так как $P(y = 1) > 0.5$, человек проголосует. Ответ: проголосует.

Задача 2: Магазин

Условие: Магазин оценивает вероятность покупки товара клиентом в зависимости от времени, которое клиент проводит на сайте. Для этого применяется логистическая регрессия. Её уравнение:

$$P(y = 1) = \frac{1}{1 + e^{-(b_0 + b_1x_1)}}.$$

Где x - время проведённое на сайте в минутах, $b_0 = -1.5$, $b_1 = 0.7$

Какова вероятность того, что клиент, проведший 3 минуты на сайте, сделает покупку?

Решение:

1. Записываем логистическую функцию:

$$P(y = 1) = \frac{1}{1 + e^{-(b_0 + b_1x_1)}}.$$

3. Подставляем значения $b_0 = -1.5$, $b_1 = 0.7$:

$$z = b_0 + b_1x_1 = -1.5 + 2.1 = 0.6$$

4. Вычисляем вероятность:

$$P(y = 1) = \frac{1}{1 + e^{-0.6}} \approx 0.645$$

Ответ: 64.5%

Задача 3: Утечка газа

Условие: На химическом заводе нужно спрогнозировать, есть ли утечка газа в системе на основе двух факторов: - x_1 : температура трубы (в градусах); - x_2 : давление внутри трубы (в барах).

Известные данные:

Ситуация 1: $x_1 = 500, x_2 = 100$, утечка ($y = 1$).

Ситуация 2: $x_1 = 300, x_2 = 80$, нет утечки ($y = 0$).

Ситуация 3: $x_1 = 400, x_2 = 90$, утечка ($y = 1$).

Определите, будет ли утечка при $x_1 = 450$ и $x_2 = 85$. Используйте коэффициенты $b_0 = -30, b_1 = 0.05, b_2 = 0.1$.

Решение:

1. Записываем логистическую функцию:

$$P(y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}}.$$

2. Подставляем значения $x_1 = 450, x_2 = 85$:

$$z = b_0 + b_1 x_1 + b_2 x_2 = -30 + 0.05 \cdot 450 + 0.1 \cdot 85 = -30 + 22.5 + 8.5 = 1.$$

3. Вычисляем вероятность:

$$P(y = 1) = \frac{1}{1 + e^{-1}} \approx 0.731.$$

4. Так как $P(y = 1) > 0.5$, утечка будет. Ответ: утечка будет.

Глава 3

Основы нейросетевых моделей

3.1. Полносвязная нейронная сеть (Персепtron).

3.1.1. Модель нейрона МакКаллока-Питтса

Рассмотрим модель нейрона МакКаллока-Питтса (1943) [2]. Множество объектов X (количество элементов множества M), множество ответов Y (количество элементов множества H_L). Признаки объектов задаются вектором функций $\overrightarrow{f(x)} = [f_1(x), f_2(x), \dots, f_n(x)]^T$ таких, что $f_j : X \rightarrow \mathbb{R}$. Для объекта $x_m \in X$ с помощью этих функций определяется вектор вещественных признаков $\overrightarrow{x_m} = [x_m^1, x_m^2, \dots, x_m^n]^T$, где $x_m^j = f_j(x_m)$. Нейрон вычисляет функцию активации σ от линейной комбинации вектора признаков $\overrightarrow{x_m}$ с весами \overrightarrow{w} . То есть выход нейрона $a(\overrightarrow{x_m}, \overrightarrow{w})$ вычисляется по формуле:

$$a(\overrightarrow{x_m}, \overrightarrow{w}) = \sigma\left(\sum_{i=1}^n x_m^i w_i - w_0\right) = \sigma(\langle \overrightarrow{x_m}, \overrightarrow{w} \rangle)$$

Математическая модель нейрона изображена на рисунке ???. Функцией активации σ может быть любая непрерывная нелинейная функция. Нелинейность функции нужна, чтобы иметь возможность приблизить любую непрерывную функцию с желаемой точностью (теорема Горбаня, 1998). В формуле вектор признаков $\overrightarrow{x_m}$ дополнен элементом -1 , который имеет вес w_0 , что соответствует физической интерпретации нейрона. Нейрон m получает на вход множество электрических сигналов x_m^i по дендритам i . Каждый дендрит i имеет свою толщину, и соответственно проводимость w_i . Чем выше проводимость w_i , тем больший вклад сигнала с данного дендрита в общую сумму. Нейрон суммирует сигналы с дендритов и если сумма выше порогового значения w_0 , то он передаёт информацию дальше по аксону. Сигнал на выходе нейрона, то есть аксоне, определяется функцией активации σ .

3.1.2. Реализация логических функций с помощью нейрона

Рассмотрим применение описанного выше нейрона для реализации простейших логических функций.

Логическая функция «и». Обозначение \wedge . На вход поступают признаки x^1, x^2 , результат функции $a = x^1 \wedge x^2$. Нейронной реализацией будет $a = [x^1 + x^2 - \frac{3}{2} > 0]$, где функция активации $[x > 0]$ равна 1, если $x > 0$, и равна 0 в противном случае.

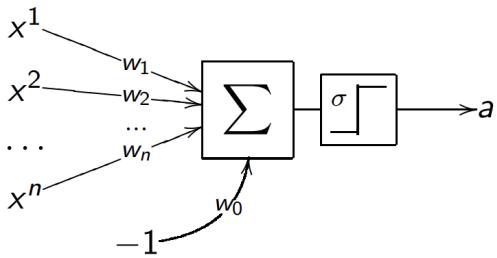
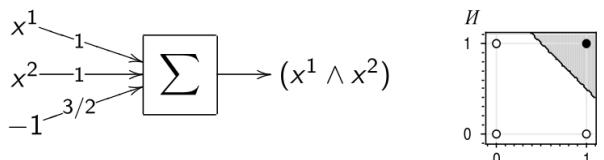


Рис. 3.1. Математическое описание модели нейрона МакКаллока-Питтса.

x^1	x^2	$a = x^1 \wedge x^2$
0	0	0
0	1	0
1	0	0
1	1	1

(a) Таблица истинности функции «и»

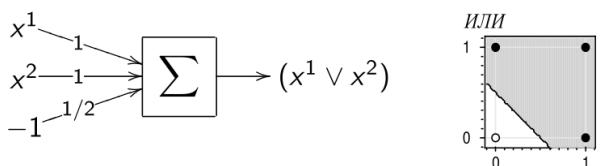


(b) Модель нейрона, описывающего функцию «и»

Логическая функция «или». Обозначение \vee . На вход поступают признаки x^1, x^2 , результат функции $a = x^1 \vee x^2$. Нейронной реализацией будет $a = [x^1 + x^2 - \frac{1}{2} > 0]$, где функция активации $[x > 0]$ равна 1, если $x > 0$, и равна 0 в противном случае.

x^1	x^2	$a = x^1 \vee x^2$
0	0	0
0	1	1
1	0	1
1	1	1

(a) Таблица истинности функции «или»



(b) Модель нейрона, описывающего функцию «или»

Задача 1. Привести модель нейрона, описывающую логическую функцию «не» $\neg x^1$.

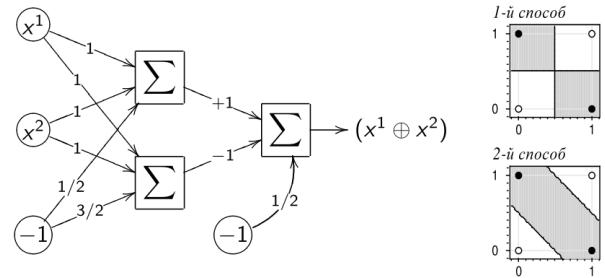
Решение:

$$a = [-x^1 + 1/2 > 0]$$

Логическая функция «исключающее или». Обозначение \oplus . Данная функция не реализуема с помощью одного нейрона. Но может быть реализована нейронной сетью $[n_1, n_2, n_3]$: $n_1 = (x^1 \vee x^2)$, $n_2 = (x^1 \wedge x^2)$, $n_3 = [n_1 - n_2 - \frac{1}{2} > 0]$.

x^1	x^2	$a = x^1 \oplus x^2$
0	0	0
0	1	1
1	0	1
1	1	0

(a) Таблица истинности функции «исключающее или»



(b) Модель нейрона, описывающего функцию «исключающее или»

Задача 2. Привести модель нейронной сети, описывающей логическую функцию $f(x^1, x^2, x^3)$, заданную таблицей истинности:

x^1	x^2	x^3	$f(x^1, x^2, x^3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Решение:

Используем ранее описанные нейроны «и» \wedge , «или» \vee , «не» \neg . Перепишем функцию, используя эти операции: $f = ((\neg x^1) \wedge ((\neg x^2) \wedge x^3)) \vee (x^1 \wedge ((\neg x^2) \vee x^3))$. Конструируем нейроны: $n_1 = \neg x^1$, $n_2 = \neg x^2$, $n_3 = n_2 \wedge x^3$, $n_4 = n_1 \wedge n_3$, $n_5 = n_2 \vee x^3$, $n_6 = x^1 \wedge n_5$, $n_7 = n_4 \vee n_6$. Итоговая многослойная нейронная сеть состоит из нейронов n_1, n_2, \dots, n_7 . Результат сети находится на выходе нейрона n_7 .

3.1.3. Область применимости многослойных нейронных сетей

Согласно [1]

1. Двухслойная сеть в $0, 1^n$ позволяет реализовать произвольную булеву функцию.
2. Линейный нейрон в \mathbb{R}^n отделяет полупространство признаков гиперплоскостью. Тогда двухслойная сеть позволяет отделить многогранную область, не обязательно выпуклую и связную.
3. Согласно теоремы Горбаня (1998) с помощью линейных операций и одной нелинейной функции активации можно приблизить любую непрерывную функцию с любой желаемой точностью.

3.1.4. Полносвязная нейронная сеть

Обобщением рассмотренной выше модели является полносвязная нейронная сеть (рис. ??). Сеть состоит из входного слоя (жёлтый цвет), скрытых слоёв (зелёный цвет), и выходного слоя (оранжевый цвет). Выход одного слоя, поступает на вход другого. Обозначим количество нейронов в слое l за H_l , в каждом слое оно может быть разным, $l \in \{1, 2, \dots, L\}$. Всего L слоёв. Вектор x^l – это выход l -ого слоя и вход $l + 1$, если $l \neq L$, то есть x^l – не последний слой. x^0 – это вход нейронной сети. Матрицы коэффициентов перехода между слоями обозначим за W^l . W^l – это матрица перехода между слоями $l - 1$ и l .

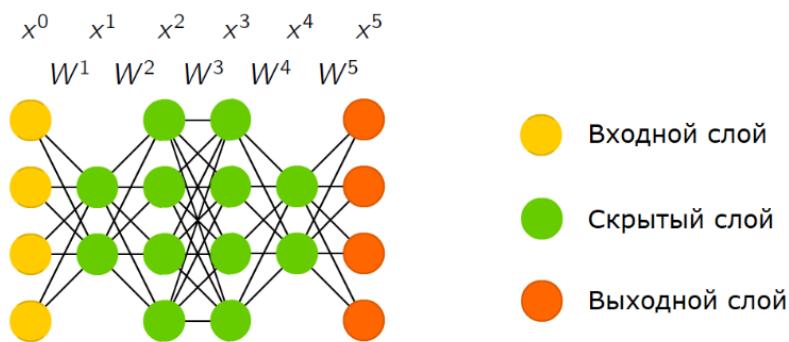


Рис. 3.5. Полносвязная нейронная сеть.

Задачей обучения нейронной сети является минимизация средних потерь на обучающей выборке X ($|X| = M$):

$$Q(\vec{W}, X) = \frac{1}{m} \sum_{m=1}^M \mathcal{L}(\vec{W}, x_m) \rightarrow \min_{\vec{W}}$$

Пусть объект x описывается вектором признаков $\vec{x} = [x_1, x_2, \dots, x_n]^T$. В задаче классификации множество ответов Y состоит из H_L элементов. Тогда последний слой нейронной сети x^L должен содержать H_L элементов. Нейронная сеть предсказывает ответ k , если на её выходе наблюдается вектор $x^L = [0, \dots, 0, 1, 0, \dots, 0]^T$, где единица стоит на k -ом месте.

Функция потерь может быть, например, квадратичной. Для объекта x функция потерь вычисляется по формуле $\mathcal{L}(\vec{W}^{(t)}, x) = \frac{1}{2} \sum_{i=1}^{H_L} (x_i^L((\vec{W}^{(t)}, x) - y_i)^2$, где \vec{y} – вектор из всех нулей кроме единицы на месте порядкового правильного ответа из множества ответов.

Рассмотрим различные методы обучения полносвязной нейронной сети.

3.1.5. Градиентный спуск

1. Выбираем какое-то начальное приближение вектора матриц перехода $\vec{W}^{(0)}$.
2. Итерационный процесс:

$$\vec{W}^{(t+1)} = \vec{W}^{(t)} - h \cdot \vec{\nabla} Q(\vec{W}^{(t)}, X)$$

где h – градиентный шаг или темп обучения.

Для рассмотренного ранее эмпирического риска:

$$\vec{W}^{(t+1)} = \vec{W}^{(t)} - \frac{h}{m} \cdot \vec{\nabla} \sum_{m=1}^M \mathcal{L}(\vec{W}^{(t)}, x_m)$$

3. Повторяем итерационный процесс, пока эмпирический риск $Q(\vec{W}^{(t)}, X)$ или вектор матриц перехода \vec{W} не сойдутся. Критерий сходимости может быть абсолютным, то есть когда модуль разности значений на последовательных шагах итерационного процесса меньше какого-то значения ε . Например, для эмпирического риска:

$$|Q^{(t+1)} - Q^{(t)}| < \varepsilon$$

Или может быть относительным, когда модуль отношения разности значений на последовательных шагах итерационного процесса к наименьшему, или наибольшему значению, или к первому меньше ε . Например, для эмпирического риска:

$$\left| \frac{Q^{(t+1)} - Q^{(t)}}{\min\{Q^{(t+1)}, Q^{(t)}\}} \right| < \varepsilon$$

При этом для эмпирического риска и для вектора матриц перехода значение ε может быть разным.

Для вектора матриц перехода можно предложить следующий способ:

$$\left\| \vec{W}^{(t+1)} - \vec{W}^{(t)} \right\|_1 = \sum_{l=1}^L \sum_{m=1}^{m_l} \sum_{n=1}^{n_l} |W_{mn}^{l(t+1)} - W_{mn}^{l(t)}|$$

где вектор матрица перехода рассматривается как вектор всех её элементов, и применяется первая норма. Размерность матрицы W^l равна $m_l \times n_l$. $W_{mn}^{l(t)}$ – это элемент в m -ой строке n -ого столбца матрицы перехода между слоями $l-1$ и l на шаге t итерационного процесса.

Недостатком данного метода является низкая скорость сходимости. Для ускорения применяется метод стохастического градиентного спуска.

3.1.6. Стохастический градиентный спуск

В отличие от обычного градиентного спуска, в котором вектор матриц перехода изменяется пропорционально градиенту эмпирического риска для всех объектов, в стохастическом градиенте вектор матриц перехода изменяется пропорционально функции потерь для одного объекта.

Алгоритм стохастического градиента:

1. Выбираем какое-то начальное приближение вектора матриц перехода $\vec{W}^{(0)}$. Вычисляем первое приближение эмпирического риска:

$$Q^{(0)}(\vec{W}^{(0)}, X) = \frac{1}{m} \sum_{m=1}^M \mathcal{L}(\vec{W}^{(0)}, x_m)$$

2. Итерационный процесс:

Выбираем какой-нибудь объект $x \in X$, например случайным образом или перебираем все элементы X по порядку. Корректируем вектор матриц перехода:

$$\vec{W}^{(t+1)} = \vec{W}^{(t)} - h \cdot \vec{\nabla} \mathcal{L}(\vec{W}^{(t)}, x)$$

где h – темп обучения.

Эмпирический риск оценивается по формуле:

$$Q^{(t+1)}(\vec{W}^{(t+1)}, X) = \lambda \mathcal{L}(\vec{W}^{(t)}, x) + (1 - \lambda) Q^{(t)}(\vec{W}^{(t)}, X)$$

где λ – темп забывания предыстории.

Рассмотрим, откуда взялась эта оценка $Q^{(t+1)}$.

Если $Q^{(t)}$ – среднее арифметическое объектов ε_i , $i = 1, 2, \dots, t$, то

$$Q^{(t)} = \frac{1}{t} \varepsilon_t + \frac{1}{t} \varepsilon_{t-1} + \frac{1}{t} \varepsilon_{t-2} + \dots + \frac{1}{t} \varepsilon_1$$

$$Q^{(t-1)} = \frac{1}{t-1} \varepsilon_{t-1} + \frac{1}{t-1} \varepsilon_{t-2} + \dots + \frac{1}{t-1} \varepsilon_1$$

$Q^{(t)}$ можно выразить через $Q^{(t-1)}$:

$$Q^{(t)} = \frac{1}{t} \varepsilon_t + \frac{t-1}{t} Q^{(t-1)} = \frac{1}{t} \varepsilon_t + \left(1 - \frac{1}{t}\right) Q^{(t-1)}$$

Если $Q^{(t)}$ – экспоненциальное скользящее среднее с параметром λ , то

$$Q^{(t)} = \lambda \varepsilon_t + \lambda(1-\lambda) \varepsilon_{t-1} + \lambda(1-\lambda)^2 \varepsilon_{t-2} + \dots + \lambda(1-\lambda)^{t-1} \varepsilon_1$$

$$Q^{(t-1)} = \lambda \varepsilon_{t-1} + \lambda(1-\lambda) \varepsilon_{t-2} + \lambda(1-\lambda)^2 \varepsilon_{t-3} + \dots + \lambda(1-\lambda)^{t-2} \varepsilon_1$$

Или $Q^{(t)}$ через $Q^{(t-1)}$:

$$Q^{(t)} = \lambda \varepsilon_t + (1-\lambda) Q^{(t-1)}$$

Пусть теперь $\lambda \sim \frac{1}{t}$. Тогда $(1-\lambda)^{t-1} = (1-\frac{1}{t})^{t-1} = \lim_{t \rightarrow \infty} (1-\frac{1}{t})^{t-1} = \frac{1}{e}$. По аналогии из радиотехники, где для экспоненциально убывающего сигнала характерное время затухания измеряется по уменьшению сигнала в e раз, тогда характерное количество членов, когда происходит затухание / «забывание» предыдущей истории ряда равно t .

3. Повторяем итерационный процесс, пока эмпирический риск $Q(\vec{W}^{(t)}, X)$ или вектор матриц перехода \vec{W} не сойдутся.

3.1.7. Метод обратного распространения ошибок (BackProp).

Для вычисления градиента функции потерь применяется метод обратного распространения ошибок. Для упрощения формулы индекс $^{(t)}$ будем опускать.

$$\vec{\nabla} \mathcal{L}(\vec{W}, x) = \left[\frac{\partial}{\partial W^0}, \frac{\partial}{\partial W^1}, \dots, \frac{\partial}{\partial W^L} \right]^T \cdot \mathcal{L}(\vec{W}, x)$$

Для рассмотренной ранее квадратичной функции потерь:

$$\vec{\nabla} \mathcal{L}(\vec{W}, x) = \left[\frac{\partial}{\partial W^0}, \frac{\partial}{\partial W^1}, \dots, \frac{\partial}{\partial W^L} \right]^T \cdot \frac{1}{2} \sum_{i=1}^{H_L} (x_i^L((\vec{W}, x) - y_i)$$

То есть выход нейронной сети \vec{x}^L является функцией от вектора матриц перехода \vec{W} .

Выведем формулу, по которой нейронная сеть вычисляет \vec{x}^L . На первом слое сети находится вектор \vec{x}^0 , который через матрицу перехода W^1 и вектор функций активации $\vec{\sigma}_1$ преобразуется во вход второго слоя x^1 :

$$\vec{x}^1 = \vec{\sigma}_1(W^1 \cdot \vec{x}^0)$$

Аналогично для последующих слоёв:

$$\vec{x}^2 = \vec{\sigma}_2(W^2 \cdot \vec{x}^1) = \vec{\sigma}_2(W^2 \cdot \vec{\sigma}_1(W^1 \cdot \vec{x}^0))$$

$$\vec{x}^L = \vec{\sigma}_L(W^L \cdot \vec{x}^{L-1}) = \vec{\sigma}_L(W^L \cdot \vec{\sigma}_{L-1}(W^{L-1} \cdot \vec{\sigma}_{L-1}(\dots (W^2 \cdot \vec{\sigma}_1(W^1 \cdot \vec{x}^0)) \dots)))$$

Чтобы получить формулы для обратного распространения ошибок необходимо найти $\vec{\nabla} \mathcal{L}$, рассматривая \vec{x}^L как функцию $\vec{x}^L(\vec{W})$.

Задача 3. Доказать рекуррентные формулы для метода обратного распространения ошибок в предположениях:

1. Функция потерь квадратичная $\mathcal{L}(\vec{W}, x) = \frac{1}{2} \sum_{i=1}^{H_L} (x_i^L((\vec{W}, x) - y_i)^2).$
2. Первый нейрон в слое нейронной сети всегда -1 , то есть $\forall i : x_0^i = -1$. H_l – количество нейронов в слое l без учёта нейрона -1 . Тогда индекс первого нейрона в слое, не всегда равного -1 , равен 1 , а индекс последнего нейрона равен H_l .
3. Вектор функций активации $\vec{\sigma}^l$ зависит от отступа \vec{M}^l : $\sigma_n^l = \sigma_n^l(M_n^l)$, где $M_n = \sum_{m=1}^{H_l} W_{nm}^l x_m^{l-1}$. $\vec{M}^l = W^l \vec{x}^{l-1}$.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_n^L} &= x_n^L - y_n, \\ \frac{\partial \mathcal{L}}{\partial W_{nm}^L} &= \frac{\partial \mathcal{L}}{\partial x_n^L} \frac{\partial \sigma_n^L}{\partial M_n^L} x_m^{L-1}, \\ \frac{\partial \mathcal{L}}{\partial x_n^l} &= \sum_{m=1}^{H_l} \frac{\partial \mathcal{L}}{\partial x_m^{l+1}} \frac{\partial \sigma_m^{l+1}}{\partial M_m^{l+1}} W_{mn}^{l+1}, \\ \frac{\partial \mathcal{L}}{\partial W_{nm}^l} &= \frac{\partial \mathcal{L}}{\partial x_n^l} \frac{\partial \sigma_n^l}{\partial M_n^l} x_m^{l-1}, \\ 1 \leq l < L. \end{aligned}$$

Решение:

Сначала докажем соотношения для $l = L$.

Рассматриваем функцию потерь как сложную функцию $\mathcal{L}(\vec{W}) = \mathcal{L}(\vec{x}^L(\vec{W}))$:

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^L} = \sum_{i=1}^{H_L} \frac{\partial \mathcal{L}}{\partial x_i^L} \frac{\partial x_i^L}{\partial W_{nm}^L}$$

Матрица W^L используется в уравнении: $\vec{x}^L = \vec{\sigma}^L(W^L \vec{x}^{L-1})$. В данном уравнении предполагается, что вектор \vec{x}^L имеет размерность $H_L \times 1$ (является столбцом, а не строкой) и не содержит -1, так как вес константы x_0^L не вычисляется по предыдущему слою, а определяется нулевым столбцом матрицы W^{L+1} , а вектор \vec{x}^{L-1} уже имеет длину $1 + H_{L-1}$, и содержит -1. Тогда размерность матрицы W^L равна $H_L \times (1 + H_{L-1})$. Дифференцировать \mathcal{L} по x_0^L не имеет смысла, так как всегда $x_0^L = -1$.

Так как $\mathcal{L}(\vec{x}^L) = \frac{1}{2} \sum_{n=1}^{H_L} (x_n^L - y_n)^2$, то

$$\frac{\partial \mathcal{L}}{\partial x_i^L} = x_i^L - y_i$$

Так как $\vec{x}^L = \vec{\sigma}^L(W^L \vec{x}^{L-1})$ или $x_i^L = \sigma_i^L \left(\sum_{k=0}^{H_{L-1}} W_{ik}^L x_k^{L-1} \right)$, тогда

$$\frac{\partial x_i^L}{\partial W_{nm}^L} = 0, \text{ если } n \neq i \text{ и}$$

$$\frac{\partial x_i^L}{\partial W_{nm}^L} = \frac{\partial \sigma_i^L}{\partial M_n^L} x_m^{L-1}, \text{ если } n = i.$$

То есть функция σ_i^L зависит от W_{nm}^l , если $n = i$. При дифференцировании суммы $\sum_{k=0}^{H_{L-1}} W_{ik}^L x_k^{L-1}$ остаётся только член с $k = m$, то есть x_m^{L-1} . Стоит отметить, что функция активации как функция от одного аргумента $\sigma_n^L = \sigma_n^L(M_n^L)$ известна при написании программы, аналогично будет известна и её производная $\left. \frac{\partial \sigma_n^L}{\partial M_n^L} \right|_{M_n^L}$.

Производная вычисляется в точке $M_n^L = \sum_{k=0}^{H_{L-1}} W_{nk}^L x_k^{L-1}$.

При использовании стохастического градиентного спуска отступ \vec{M}^l будет вычислен при прямом ходе, при вычислении \vec{x}^l от объекта $x \in X$. Поэтому для ускорения вычислений можно при прямом ходе вычислять и запоминать значения производных $\left. \frac{\partial \sigma_n^l}{\partial M_n^l} \right|_{M_n^l}$ в точке M_n^l , также на каждом слое нужно запоминать значения \vec{x}^l .

Теперь используя полученные соотношения запишем:

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^L} = \sum_{i=1}^{H_L} \frac{\partial \mathcal{L}}{\partial x_i^L} \frac{\partial x_i^L}{\partial W_{nm}^L} = \frac{\partial \mathcal{L}}{\partial x_n^L} \frac{\partial x_n^L}{\partial W_{nm}^L} = \frac{\partial \mathcal{L}}{\partial x_n^L} \frac{\partial \sigma_n^L}{\partial M_n^L} x_m^{L-1}$$

где $\frac{\partial \mathcal{L}}{\partial x_n^L} = x_n^L - y_n$.

Теперь докажем формулы для $1 \leq l < L$.

$x_n^l = \sigma_n^l \left(\sum_{k=0}^{H_{l-1}} W_{nk}^l x_k^{l-1} \right)$ или $x_n^l = x_n^l(\overrightarrow{W^l}, \overrightarrow{x^{l-1}})$. Тогда

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^l} = \sum_{i=1}^{H_l} \frac{\partial \mathcal{L}}{\partial x_i^l} \frac{\partial x_i^l}{\partial W_{nm}^l}$$

Найдём $\frac{\partial \mathcal{L}}{\partial x_i^l}$.

$$\frac{\partial \mathcal{L}}{\partial x_i^l} = \sum_{j=1}^{H_{l+1}} \frac{\partial \mathcal{L}}{\partial x_j^{l+1}} \frac{\partial x_j^{l+1}}{\partial x_i^l}$$

Производная $\frac{\partial \mathcal{L}}{\partial x_j^{l+1}}$ уже была вычислена на шаге для $l + 1$. Вычислим $\frac{\partial x_j^{l+1}}{\partial x_i^l}$. Так

как $x_j^{l+1} = \sigma_j^{l+1} \left(\sum_{k=0}^{H_l} W_{jk}^{l+1} x_k^l \right)$, то

$$\frac{\partial x_j^{l+1}}{\partial x_i^l} = \frac{\partial \sigma_j^{l+1}}{\partial M_j^{l+1}} W_{ji}^{l+1}$$

Итого

$$\frac{\partial \mathcal{L}}{\partial x_i^l} = \sum_{j=1}^{H_{l+1}} \frac{\partial \mathcal{L}}{\partial x_j^{l+1}} \frac{\partial \sigma_j^{l+1}}{\partial M_j^{l+1}} W_{ji}^{l+1}$$

Производная $\frac{\partial x_i^l}{\partial W_{nm}^l}$ вычисляется аналогично, как и для $l = L$:

$$\frac{\partial x_i^l}{\partial W_{nm}^l} = 0, \text{ если } n \neq i \text{ и}$$

$$\frac{\partial x_i^l}{\partial W_{nm}^l} = \frac{\partial \sigma_n^l}{\partial M_n^l} x_m^{l-1}, \text{ если } n = i.$$

Окончательно

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^l} = \frac{\partial \mathcal{L}}{\partial x_n^l} \frac{\partial \sigma_n^l}{\partial M_n^l} x_m^{l-1}$$

3.1.8. Алгоритм применения метода обратного распространения ошибки в стохастическом градиентном спуске.

Рассмотрим подробнее алгоритм применения метода обратного распространения ошибки в стохастическом градиентном спуске.

1. Выбираем какое-то начальное приближение вектора матриц перехода $\vec{W}^{(0)}$. Вычисляем первое приближение эмпирического риска:

$$Q^{(0)}(\vec{W}^{(0)}, X) = \frac{1}{m} \sum_{m=1}^M \mathcal{L}(\vec{W}^{(0)}, x_m)$$

2. Итерационный процесс:

Выбираем какой-нибудь объект $x \in X$, например случайным образом или перебираем все элементы X по порядку.

Прямой ход. Вычисляем отступ $\vec{M}^l = \vec{W}^l \cdot \vec{x}^{l-1}$. Вычисляем и запоминаем значения выходов нейронов $\vec{x}^l = \sigma^l(\vec{M}^l)$ и производные функций активации $\frac{\partial \sigma^l}{\partial \vec{M}^l} \Big|_{\vec{M}^l}$.

Вычисляем для последнего слоя производные от функции потерь:

$$\frac{\partial \mathcal{L}}{\partial x_n^L} = x_n^L - y_n$$

и

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^L} = \frac{\partial \mathcal{L}}{\partial x_n^L} \frac{\partial \sigma_n^L}{\partial M_n^L} x_m^{L-1}$$

Обратный ход для всех слоёв $1 \leq l < L$:

$$\frac{\partial \mathcal{L}}{\partial x_n^l} = \sum_{m=1}^{H_l} \frac{\partial \mathcal{L}}{\partial x_m^{l+1}} \frac{\partial \sigma_m^{l+1}}{\partial M_m^{l+1}} W_{mn}^{l+1}$$

и

$$\frac{\partial \mathcal{L}}{\partial W_{nm}^l} = \frac{\partial \mathcal{L}}{\partial x_n^l} \frac{\partial \sigma_n^l}{\partial M_n^l} x_m^{l-1}$$

В итоге был вычислен градиент функции потерь: $\vec{\nabla} \mathcal{L}(\vec{W}^{(t)}, x)$.

Корректируем вектор матриц перехода:

$$\vec{W}^{(t+1)} = \vec{W}^{(t)} - h \cdot \vec{\nabla} \mathcal{L}(\vec{W}^{(t)}, x)$$

где h – темп обучения.

Эмпирический риск оценивается по формуле:

$$Q^{(t+1)}(\vec{W}^{(t+1)}, X) = \lambda \mathcal{L}(\vec{W}^{(t)}, x) + (1 - \lambda) Q^{(t)}(\vec{W}^{(t)}, X)$$

где λ – темп забывания предыстории.

3. Повторяем итерационный процесс, пока эмпирический риск $Q(\vec{W}^{(t)}, X)$ или вектор матриц перехода \vec{W} не сойдутся.

3.2. Функции активации ReLU и PReLU. Проблема «паралича» сети.

3.2.1. Про функции активации

Функция активации в контексте нейронных сетей определяет выходной сигнал нейрона на основе входного сигнала или набора входных сигналов. Она играет важную роль в определении нелинейности модели и позволяет нейронной сети обучаться сложным зависимостям между входными данными и выходами, а также влиять на эффективность обучения модели.(грубо говоря определяет будет ли нейрон активирован - даст не нулевой выход)

В искусственных нейронных сетях используются различные типы функций активации, такие как тождественная функция, ступенчатые функции, сигмоидные функции и многие другие.

Важным требованием к функции активации является ее нелинейность, чтобы нейронная сеть могла эффективно справляться с нетривиальными задачами.

Примером широко используемой функции активации является ReLU.

3.2.2. ReLU (Rectified Linear Unit), проблема «паралича» сети

Определение: Функция активации ReLU определяется как: $f(x) = \max(0, x)$, где x – входное значение.

Не смотря на очевидную простоту функции активации ReLU, у неё есть одна существенная проблема: она может вызывать так называемый **«паралич» сети**.

Если градиент весов становится слишком большим, то некоторые нейроны могут получить очень большие отрицательные веса, что приведёт к тому, что их выходные значения всегда будут равны нулю независимо от входного сигнала. Эти нейроны становятся неактивными («замороженными») и больше не участвуют в процессе обучения. Если это происходит слишком часто, то это может привести к ухудшению результата.

Для решения этой проблемы была предложена модификация ReLU.

3.2.3. PReLU (Parametric Rectified Linear Unit)

Определение: Функция PReLU является обобщением ReLU и определяется как:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

В отличие от обычной ReLU, где все отрицательные значения обнуляются, PReLU вводит дополнительный параметр α , который управляет наклоном линии для отрицательных значений. Этот параметр α может быть обучен вместе с другими параметрами сети, что позволяет избежать полного отключения нейронов. То есть даже отрицательный входной сигнал будет вносить вклад в обучение модели, что уменьшает проблему паралича сети и улучшает получаемый результат.

3.2.4. Задачи

Задача 1

Представьте, что вы получили графики функций активации ReLU и PReLU (с заданным параметром α) для различных входных значений.

Опишите, как выглядит график функции активации ReLU и как он отличается от графика PReLU. Какова роль параметра α в PReLU?

Решение

График функции ReLU выглядит как угол, который начинается от нуля и идет вверх с углом 45 градусов для положительных значений.

График функции PReLU (Parametric ReLU) также имеет две части, но для отрицательных значений он не обрезается до нуля. Вместо этого, он наклонен под углом, определяемым параметром α .

Роль параметра α в PReLU:

Параметр α определяет наклон линии для отрицательных значений. Если α велико, то выход будет относительно большим даже для небольших отрицательных входов, что позволяет нейрону сохранять некоторую активность. Если α близко к нулю, то функция будет почти горизонтальной для отрицательных значений, что делает нейрон менее активным в этом диапазоне.

Задача 2.1

Рассмотрим простой случай использования функции активации ReLU в одном слое нейронной сети. Пусть входной сигнал равен -4 , а вес этого нейрона равен 10 . Каково будет значение на выходе этого нейрона после применения функции активации ReLU?

Решение:

Выходной сигнал до применения функции активации будет равен произведению входа и веса: $-4 \cdot 10 = -40$. После применения функции активации ReLU получаем: $f(-40) = \max(0, -40) = 0$. Таким образом, этот нейрон станет неактивным ("замороженным"), поскольку его выходной сигнал всегда будет нулевым независимо от изменения входного сигнала.

Задача 2.2

Пусть теперь используется функция активации PReLU с параметром $\alpha = 0.01$. Рассчитайте значение на выходе того же нейрона, что и в предыдущей задаче, но уже с использованием PReLU.

Решение:

После умножения входа и веса получаем тот же результат: $-4 \cdot 10 = -40$. Теперь применяем функцию активации PReLU:

$$f(-40) = \begin{cases} -40 & \text{if } (-40) > 0 \\ 0.01 \cdot (-40) & \text{if } (-40) \leq 0 \end{cases}$$

Так как $-40 \leq 0$, мы используем вторую часть определения функции: $f(-40) = 0.01 \cdot (-40) = -0.4$. В этом случае нейрон не становится полностью замороженным, так как его выходной сигнал остается отличным от нуля даже при отрицательном входе.

Задача 3

Входной сигнал x распределён равномерно на интервале $[-10, 10]$, а вес w нейрона равен 2, параметр $\alpha = 0.1$.

Какова вероятность, что нейрон заморозится при использовании каждой из функций ReLU и PReLU?

Решение:

Для ReLU:

Нейрон «замораживается», если выходной сигнал после применения функции активации всегда равен нулю. То есть, если $wx \leq 0$.

Поскольку $w = 2$, условие $wx \leq 0$ эквивалентно $x \leq 0$.

Вероятность того, что x окажется меньше или равно нулю, равна вероятности того, что x попадет в интервал $-10, 0$. Это составляет половину всего диапазона распределения, поэтому вероятность равна 0.5.

При использовании PReLU нейрон никогда не «замерзнет», потому что даже при отрицательных значениях x выходной сигнал будет отличен от нуля благодаря параметру α . Таким образом, вероятность того, что нейрон «заморозится», равна 0.

3.3. Drop Out

3.3.1. Теоретические сведения

Drop Out (метод отключения случайных нейронов) - это метод, который представляет собой эффективный подход к борьбе с переобучением в полносвязных нейронных сетях.

Переобучение может происходить, когда нейроны в сети начинают "запоминать" шум и особенности обучающего набора, вместо того чтобы извлекать общие закономерности. Во время обучения нейронной сети нейроны взаимодействуют друг с другом, и иногда происходит так, что один нейрон начинает исправлять ошибки другого. Это может привести к ситуации, когда в одном слое нейронов образуются большие веса с разными знаками, что делает модель менее стабильной и затрудняет ее обобщение. В таких случаях, даже если модель демонстрирует высокую точность на обучающих данных, она может оказаться неэффективной на тестовых данных.

Идея Drop Out заключается в том, что при обучении случайные нейроны отключаются (то есть возвращают всегда нулевое значение) и не участвуют в данном шаге обучения. В таком случае большие значения весов разных знаков не всегда участвуют в обучении одновременно, и модель стабилизируется.

Обучение с Dropout можно интерпретировать как обучение одновременно 2^N моделей (где N — количество нейронов) с разными архитектурами связей. При обучении выбирается модель, наиболее устойчивая к потере доли нейронов; разные части модели решают одну и ту же задачу вместо того, что бы компенсировать ошибки друг друга.

Недостатком Drop Out является более долгое обучение модели в связи со случайностью процесса обучения.

3.3.2. Реализация

При обучении выбирается параметр p — вероятность отключения. Модель обучается с отключением случайных нейронов. Можно отключать в каждом слое долю p нейронов, вместо независимого отключения, чтобы избежать полного отключения одного слоя.

$$x_i^l = \xi_i^l \sum_{j=1}^{K_{l-1}} w_{ij}^l x_j^{l-1}$$

l — номер текущего слоя, i — номер нейрона в этом слое, K_l — кол-во нейронов в слое l , $\xi \sim Be(1 - p)$ — случайная величина, отвечающая за отключение. $[\xi_i^l] = 1 - p$, поэтому на этапе применения вводится нормировка:

$$x_i^l = (1 - p) \sum_{j=1}^{K_{l-1}} w_{ij}^l x_j^{l-1}$$

Чаще используется **Inverted Drop Out** для простоты применения:

$$x_i^l = \frac{\xi_i^l}{1 - p} \sum_{j=1}^{K_{l-1}} w_{ij}^l x_j^{l-1}$$

— при обучении;

$$x_i^l = \sum_{j=1}^{K_{l-1}} w_{ij}^l x_j^{l-1}$$

— при применении.

3.3.3. Задачи

Задача 1 Какие по порядку величины должны быть ограничения на значения p , чтобы избежать отключения одного слоя целиком?

Решение: если в сети L слоёв размерности K , то $p \sim (\frac{c}{L})^{1/K}$. Тогда вероятность отключения одного слоя $p^K = \frac{c}{L}$; вероятность функционирования всех слоёв: $\sim (1 - p^K)^L = (1 - \frac{c}{L})^L \sim e^{-c}$.

Если взять $c \sim 0.01$, получим вероятность работы сети $\sim 99\%$.

Если $L = 4, K = 10$, то $p \sim 54\%$.

Задача 2 Сколько шагов обучения нужно провести, чтобы не осталось нейронов, которые были бы выкинуты на каждом шаге (то есть совсем не обучались)?

Решение: если в сети N нейронов, то вероятность одному нейрону совсем не обучиться в течение T итераций равна p^T . Тогда среднее число необученных нейронов равно Np^T . Из условия $Np^T < 1$ получаем $T > -\log_p N = \frac{\ln N}{\ln \frac{1}{p}}$.

3.4. CNN. Свёртки и пулинги для обработки изображений.

3.4.1. Стандартная схема свёрточной сети.

$x[i, j]$ — исходные признаки, пиксели $n \times m$ -изображения

w_{ab} — ядро свёртки, $a = -A, \dots, +A$, $b = -B, \dots, +B$

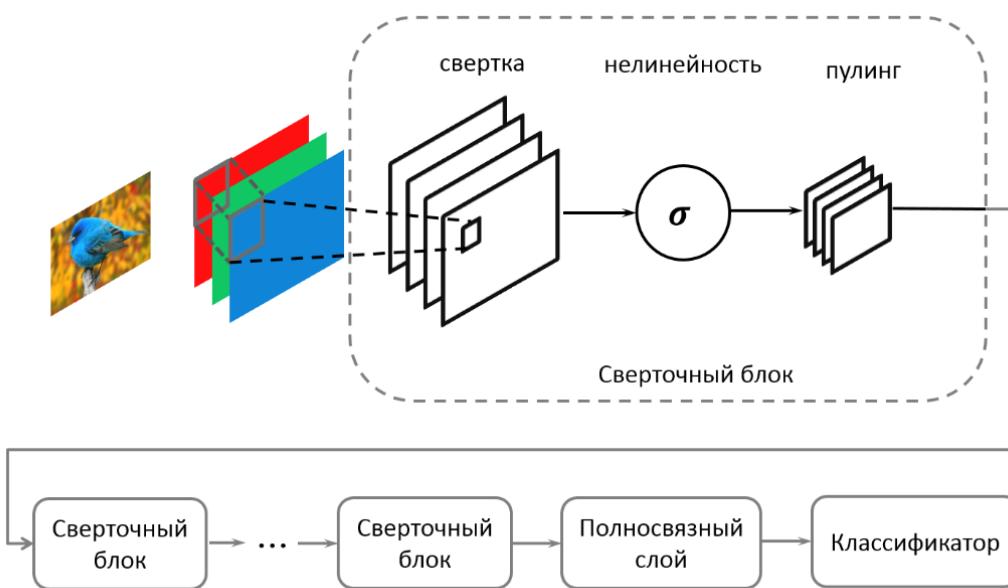
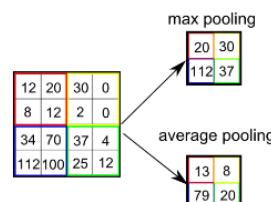
Неполносвязный свёрточный нейрон с $(2A + 1)(2B + 1)$ весами:

$$(x * w)[i, j] = \sum_{a=-A}^A \sum_{b=-B}^B w_{ab} x[i + a, j + b]$$

Объединяющий нейрон — это необучаемая свёртка с шагом $h > 1$, агрегирующая данные прямоугольной области $h \times h$ (объединяющий слой нейронов = пулинг слой):

$$y[i, j] = F(x[hi, hj], \dots, x[hi + h - 1, hj + h - 1])$$

где F — агрегирующая функция: max, average и т.п. Max-pooling позволяет обнаружить элемент в любой из ячеек.

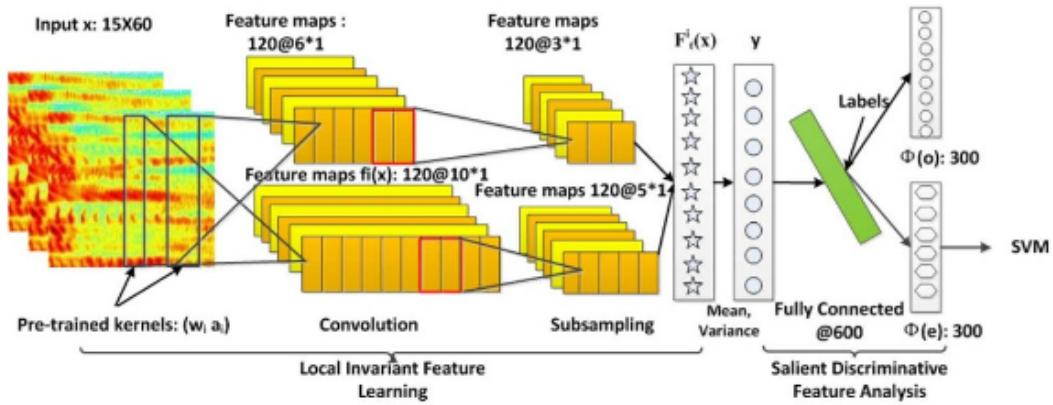


Свёрточная сеть обучается извлечению признаков
Чем выше слой, тем более крупные и сложные элементы изображений
он способен распознавать

3.4.2. Приложения CNN.

Классификация изображений(Свёрточная сеть AlexNet)

Распознавание речевых сигналов

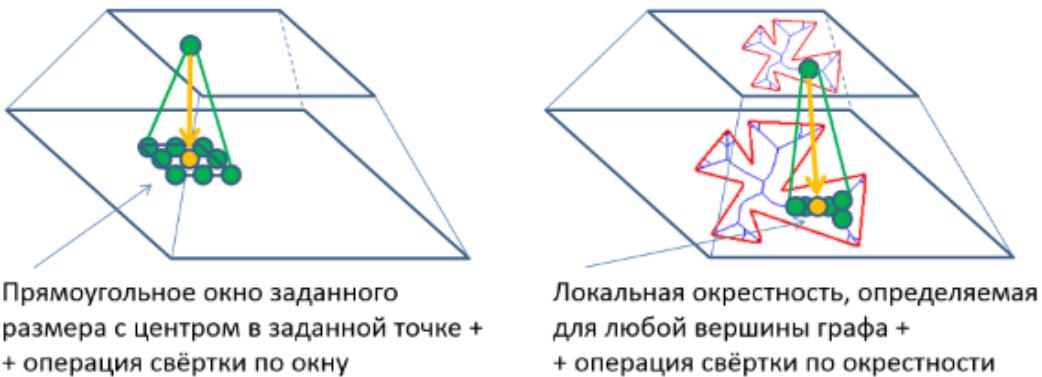


Классификация предложений в тексте:

Последовательные слова в тексте представляются векторами с помощью векторных представлений (word2vec и др.)

3.4.3. Обобщение CNN на любые структурированные данные.

Допустим, каждый объект имеет структуру, заданную графом
Свёртка определяется по локальной окрестности вершины
Пуллинг агрегирует векторы вершин локальной окрестности
Такая сеть обучается находить и классифицировать подграфы



3.4.4. Задачи

Задача 1

Дано изображение размером 5x5 пикселей и свёрточное ядро размером 3x3 пикселя

$$\left[\begin{array}{ccccc} 1 & 2 & 3 & 0 & 1 \\ 4 & 5 & 6 & 1 & 0 \\ 7 & 8 & 9 & 2 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 0 & 1 \end{array} \right] \quad \left[\begin{array}{ccc} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{array} \right]$$

Необходимо выполнить операцию свёртки

Решение

1. Результирующая матрица после свёртки будет размером

$$(5 - 3 + 1) \times (5 - 3 + 1) = 3 \times 3$$

2. Процесс свёртки:

Для каждого положения свёрточного ядра на изображении вычисляем сумму произведений.

Позиция (0,0): $(1 \times 1) + (2 \times 0) + (3 \times -1) +$
 $(4 \times 1) + (5 \times 0) + (6 \times -1) +$
 $(7 \times 1) + (8 \times 0) + (9 \times -1) =$
 $1 + 0 - 3 + 4 + 0 - 6 + 7 + 0 - 9 = -6$

Позиция (0,1): $(2 \times 1) + (3 \times 0) + (0 \times -1) +$
 $(5 \times 1) + (6 \times 0) + (1 \times -1) +$
 $(8 \times 1) + (9 \times 0) + (2 \times -1) =$
 $2 + 0 + 0 + 5 + 0 - 1 + 8 + 0 - 2 = 12$

Позиция (0,2): $(3 \times 1) + (0 \times 0) + (1 \times -1) +$
 $(6 \times 1) + (1 \times 0) + (2 \times -1) +$
 $(9 \times 1) + (2 \times 0) + (1 \times -1) =$
 $3 + 0 - 1 + 6 + 0 - 2 + 9 + 0 - 1 = 16$

Позиция (1,0): $(4 \times 1) + (5 \times 0) + (6 \times -1) +$
 $(7 \times 1) + (8 \times 0) + (9 \times -1) +$
 $(0 \times 1) + (1 \times 0) + (2 \times -1) =$
 $4 + 0 - 6 + 7 + 0 - 9 + 0 + 0 - 2 = -6$

Позиция (1,1): $(5 \times 1) + (6 \times 0) + (1 \times -1) +$
 $(8 \times 1) + (9 \times 0) + (2 \times -1) +$
 $(1 \times 1) + (0 \times 0) + (3 \times -1) =$
 $5 + 0 - 1 + 8 + 0 - 2 + 1 + 0 - 3 = 8$

Позиция (1,2): $(6 \times 1) + (1 \times 0) + (0 \times -1) +$
 $(9 \times 1) + (2 \times 0) + (1 \times -1) +$
 $(2 \times 1) + (3 \times 0) + (4 \times -1) =$
 $6 + 0 + 0 + 9 + 0 - 1 + 2 + 0 - 4 = 12$

Позиция (2,0): $(7 \times 1) + (8 \times 0) + (9 \times -1) +$
 $(0 \times 1) + (1 \times 0) + (2 \times -1) +$
 $(1 \times 1) + (0 \times 0) + (3 \times -1) =$
 $7 + 0 - 9 + 0 + 0 - 2 + 1 + 0 - 3 = -4$

Позиция (2,1): $(8 \times 1) + (9 \times 0) + (2 \times -1) +$
 $(1 \times 1) + (2 \times 0) + (3 \times -1) +$
 $(0 \times 1) + (1 \times 0) + (4 \times -1) =$
 $8 + 0 - 2 + 1 + 0 - 3 + 0 + 0 - 4 = 0$

Позиция (2,2): $(9 \times 1) + (2 \times 0) + (1 \times -1) +$
 $(2 \times 1) + (3 \times 0) + (4 \times -1) +$
 $(1 \times 1) + (0 \times 0) + (1 \times -1) =$
 $9 + 0 - 1 + 2 + 0 - 4 + 1 + 0 - 1 = 6$

После выполнения всех операций свёртки
мы получаем следующую матрицу:

$$\begin{vmatrix} -6 & 12 & 16 \\ -6 & 8 & 12 \\ -4 & 0 & 6 \end{vmatrix}$$

Задача 2

Даны изображения продукции с производственной линии.

К сожалению, некоторые с дефектами.

Предоставьте алгоритм построения CNN для распознавания дефектов на изображениях.

Подойдёт такое решение:

1. Создать CNN с тремя свёрточными слоями и двумя pooling-слоями.
2. Использовать dropout для предотвращения переобучения.
3. Добавить полносвязный слой перед выходным слоем.
4. Взять сигмоидный выходной слой для бинарной классификации.
5. Обучить модель на размеченных изображениях, используя функцию потерь бинарная кросс-энтропия.

Задача 3

Каковы основные компоненты CNN и их функции?

Решение:

- Свёрточные слои: применяют свёртку к входным данным, чтобы выделить важные признаки.
- Слои подвыборки(Pooling): уменьшают размерность данных, сохранивая наиболее значимую информацию.
- Полносвязные слои: используются для окончательной классификации на основе извлечённых признаков.

Оптимальное прореживание нейронных сетей

Введение

Прореживание нейронных сетей (англ. optimal brain damage) — метод упрощения структуры регрессионной модели, например, нейронной сети. Основная идея прореживания (англ. pruning) заключается в том, что те элементы модели или те нейроны сети, которые оказывают малое влияние на ошибку аппроксимации, можно исключить из модели без значительного ухудшения качества аппроксимации [.] Такой подход позволяет достичь следующих целей:

- **Сжатие нейросети.** Уменьшается количество параметров, которые нужно хранить, что важно для устройств с ограниченными ресурсами.
- **Ускорение вычислений.** Меньшее количество параметров требует меньше операций умножения, что ускоряет работу модели.

- **Регуляризация.** Снижение числа параметров уменьшает склонность модели к переобучению, делая её более устойчивой к шуму данных.
- **Повышение качества.** Часто итоговая модель после прореживания показывает лучшие результаты, чем исходная. Это связано с тем, что избыточно сложная модель склонна к переобучению, а последовательное исключение избыточных параметров позволяет оптимально адаптировать её сложность к задаче.

История метода

Метод второго порядка был предложен Яном ЛеКуном в 1990 году [lecun1990optimal] и получил название *Optimal Brain Damage*. На тот момент этот подход был одним из лучших для уменьшения размеров нейронных сетей и улучшения их качества. Позднее Хассиби и Штурм [hassibi1993optimal] разработали его улучшение *Optimal Brain Surgery*, основываясь на анализе вторых производных. Ранее существовали методы нулевого порядка, где исключались элементы с малыми весами [mozer1989skeletonization]. В 1990 году А. Н. Горбань предложил метод, использующий первые производные, что позволило обойтись без вычисления вторых производных. Этот подход получил название *контрастирование нейронных сетей*. Е. М. Миркес развил идеи Горбаня, создав библиотеку функций и язык описания для проекта «Идеального нейрокомпьютера».

Впоследствии появились более современные методы, например Dropout, L_2 -регуляризации и другие, которые вытеснили этот подход из основного арсенала. Тем не менее, метод прореживания всё ещё может быть полезным инструментом, особенно в ситуациях, требующих компактности модели.

Математическая постановка задачи

Рассмотрим регрессионную модель

$$y_n = f(\mathbf{w}, \mathbf{x}_n) + \nu,$$

где $\mathbf{w} \in \mathbb{R}^d$ — вектор параметров, $\mathbf{x}_n \in \mathbb{R}^p$ — вектор независимых переменных, $y_n \in \mathbb{R}$ — зависимая переменная, ν — случайная ошибка.

Задана выборка $D = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Для минимизации функции ошибки

$$E_D(\mathbf{w}) = \sum_{n=1}^N \ell(y_n, f(\mathbf{w}, \mathbf{x}_n)),$$

где $\ell(\cdot, \cdot)$ — функция потерь, требуется найти $\mathbf{w}^{MP} = \arg \min_{\mathbf{w}} E_D(\mathbf{w})$.

Прореживание параметров

Цель метода — исключение параметров w_i , которые оказывают наименьшее влияние на ошибку E_D . Для этого используется квадратичная аппроксимация E_D в окрестности \mathbf{w}^{MP} :

$$E_D(\mathbf{w} + \Delta\mathbf{w}) \approx E_D(\mathbf{w}^{MP}) + \frac{1}{2}\Delta\mathbf{w}^T H \Delta\mathbf{w},$$

где $H = \nabla_{\mathbf{w}}^2 E_D(\mathbf{w}^{MP})$ — матрица Гессе.

Исключение параметра w_i эквивалентно наложению ограничения $\Delta w_i + w_i = 0$. Для выполнения данного условия минимизируется функция:

$$\Delta E_D = \frac{1}{2}\Delta\mathbf{w}^T H \Delta\mathbf{w},$$

при ограничении $\mathbf{e}_i^T \Delta\mathbf{w} + w_i = 0$, где \mathbf{e}_i — единичный вектор с i -м элементом, равным 1.

Оптимизация

Используем метод множителей Лагранжа для минимизации:

$$S = \frac{1}{2}\Delta\mathbf{w}^T H \Delta\mathbf{w} - \lambda(\mathbf{e}_i^T \Delta\mathbf{w} + w_i).$$

Дифференцируя S по $\Delta\mathbf{w}$ и λ и приравнивая производные к нулю, получаем:

$$\Delta\mathbf{w} = -\frac{w_i}{[H^{-1}]_{ii}} H^{-1} \mathbf{e}_i.$$

Подставляя это выражение в ΔE_D , находим:

$$L_i = \frac{w_i^2}{2[H^{-1}]_{ii}}.$$

Параметр i , минимизирующий L_i , выбирается для исключения.

Итеративный процесс прореживания

Вышенаписанный процесс можно повторять несколько раз для повышения эффективности модели:

1. Определяется вес, который оказывает минимальное влияние на ошибку, и он исключается из модели. Удаление выполняется до тех пор, пока ошибка не превышает заданный порог.

2. После удаления части параметров модель дообучается, чтобы восстановить её качество.
3. Процесс повторяется, пока не будет достигнута желаемая компактность или качество модели или пока не пройдет заданное число итераций.

Задачи

Выпишите Лагранжиан для решения задачи нахождения веса, зануление которого ведет к минимальной ошибке для методов 3 и 4 порядков. Попробуйте решить эту оптимизационную задачу.

Мы раскладываем функции ошибки до второй производной, хотя приращение $\Delta w_i = -w_i$ может быть достаточно большим. Оцените применимость данного метода в зависимости от величины весов w_i .

Рассчитайте L_i для модели, в которой $H = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix}$, $w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$.

3.5. Методы оптимизации с использованием Autograd: SGD, Adam, RMSProp

3.5.1. Автоматическое дифференцирование с Autograd

Autograd – это инструмент для автоматического дифференцирования, который позволяет автоматически вычислять градиенты функций. В контексте методов оптимизации, Autograd упрощает процесс нахождения производных для сложных функций потерь. Благодаря Autograd, пользователю не нужно вручную выводить градиенты – они вычисляются программно с использованием обратного распространения (backpropagation).

Autograd используется во многих популярных библиотеках, таких как PyTorch и JAX. На семинарах были продемонстрированы примеры использования Autograd для автоматического вычисления градиентов при оптимизации нейронных сетей.

Пример использования Autograd

Предположим, у нас есть функция потерь $L(\theta) = (\theta^2 + 3\theta + 2)$. С помощью Autograd можно вычислить градиент этой функции по параметру θ :

```
import autograd.numpy as np
from autograd import grad
```

```

def loss(theta):
    return theta**2 + 3 * theta + 2

grad_loss = grad(loss)
print(grad_loss(1.0)) # Выводит 5.0

```

3.5.2. Стохастический градиентный спуск (SGD)

Стохастический градиентный спуск (SGD) представляет собой метод оптимизации, который используется для минимизации функции потерь $L(\theta)$, где θ – параметры модели. На каждом шаге обновления параметров SGD использует оценку градиента по одному или нескольким случайно выбранным объектам:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L_i(\theta_t), \quad (3.1)$$

где:

- θ_t – параметры на шаге t ;
- η – коэффициент обучения;
- $\nabla_{\theta} L_i(\theta_t)$ – градиент функции потерь по i -му примеру на шаге t .

SGD часто используется благодаря своей простоте, но он может быть неустойчивым и медленным при выборе неподходящего коэффициента обучения.

3.5.3. RMSProp

Метод RMSProp (Root Mean Square Propagation) предназначен для адаптивного изменения шага обучения. В отличие от SGD, RMSProp учитывает среднеквадратичное значение градиентов для каждого параметра. Формулы для обновления параметров имеют вид:

$$g_t = \nabla_{\theta} L_i(\theta_t), \quad (3.2)$$

$$v_t = \gamma v_{t-1} + (1 - \gamma) g_t^2, \quad (3.3)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t, \quad (3.4)$$

где:

- v_t – скользящее среднее квадратов градиентов;

- γ – коэффициент сглаживания (обычно 0.9);
- ϵ – небольшая константа для избежания деления на ноль.

3.5.4. Adam

Метод Adam (Adaptive Moment Estimation) сочетает идеи из SGD и RMSProp, используя как первую, так и вторую моменты градиентов. Алгоритм Adam вычисляется по следующим формулам:

$$g_t = \nabla_{\theta} L_i(\theta_t), \quad (3.5)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad (3.6)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \quad (3.7)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad (3.8)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \quad (3.9)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \quad (3.10)$$

где:

- m_t – скользящее среднее градиентов (первая производная);
- v_t – скользящее среднее квадратов градиентов (вторая производная);
- β_1 и β_2 – коэффициенты экспоненциального сглаживания (обычно $\beta_1 = 0.9$ и $\beta_2 = 0.999$);
- \hat{m}_t и \hat{v}_t – корректированные моменты с учётом смещения;
- η – шаг обучения.

3.5.5. Задачи

Задача 1

Рассмотрим функцию потерь $L(\theta) = \theta^2$. Используя SGD с шагом обучения $\eta = 0.1$, выполните два шага оптимизации, начиная с $\theta_0 = 1$. Найдите θ_1 и θ_2 .

Решение:

$$\begin{aligned}\nabla_{\theta} L(\theta_0) &= 2 \cdot \theta_0 = 2 \cdot 1 = 2, \\ \theta_1 &= \theta_0 - 0.1 \cdot 2 = 1 - 0.2 = 0.8, \\ \nabla_{\theta} L(\theta_1) &= 2 \cdot 0.8 = 1.6, \\ \theta_2 &= \theta_1 - 0.1 \cdot 1.6 = 0.8 - 0.16 = 0.64.\end{aligned}$$

Задача 2

Покажите, что в RMSProp, если градиент на каждом шаге постоянен и равен g , значение v_t стремится к g^2 при $t \rightarrow \infty$.

Решение:

$$v_t = \gamma v_{t-1} + (1 - \gamma)g^2.$$

В установившемся режиме $v_t = v_{t-1} = v$, поэтому:

$$\begin{aligned}v &= \gamma v + (1 - \gamma)g^2, \\ v(1 - \gamma) &= (1 - \gamma)g^2, \\ v &= g^2.\end{aligned}$$

Задача 3

Докажите, что для Adam корректированные моменты \hat{m}_t и \hat{v}_t стремятся к m_t и v_t соответственно при $t \rightarrow \infty$.

Решение:

- $\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$. При $t \rightarrow \infty$, $\beta_1^t \rightarrow 0$, поэтому $1 - \beta_1^t \rightarrow 1$ и $\hat{m}_t \rightarrow m_t$.
- Аналогично, $\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$, и при $t \rightarrow \infty$, $\hat{v}_t \rightarrow v_t$.

Глава 4

Метрические методы классификации и регрессии

4.1. Обобщенный метрический классификатор

4.1.1. Гипотеза непрерывности и компактности

Прежде, чем строить какой-либо обобщенный метрический метод для задачи классификации или регрессии важно подчеркнуть исходные.

Для задачи регрессии предполагается, что зависимость, которую мы восстанавливаем непрерывная. Т.е. "близким" объектам соответствуют "близкие" ответы.

Для задач классификации предполагается гипотеза компактности в ее бытовом смысле. Классы - это компактные сгустки точек, т.е. "близкие" объекты, как правило, лежат в одном классе.

4.1.2. Формализация расстояния

Теперь ответим на вопрос, а какие объекты считать "близки" в строгом математическом смысле?

Для определения расстояния используются классические метрики вроде:

- Евклидовой метрики

$$\rho(x, x_i) = \left(\sum_{j=1}^n |x^j - x_i^j|^2 \right)^{\frac{1}{p}} \quad (4.1)$$

- Обобщенной метрики Миновского

$$\rho(x, x_i) = \left(\sum_{j=1}^n w_j |x^j - x_i^j|^p \right)^{\frac{1}{p}} \quad (4.2)$$

$x = (x^1, \dots, x^n)$ - вектор признаков соответствующего объекта;

w_1, \dots, w_n - обучаемые веса признаков, отвечающие за приведение к общему масштабу, задание степени информативности признаков.

Для произвольного $x \in X$ отранжируем объекты обучающей выборки x_1, \dots, x_l следующим образом:

$$\rho(x, x^{(1)}) \leq \rho(x, x^{(2)}) \leq \dots \leq \rho(x, x^{(l)}) \quad (4.3)$$

$x^{(i)}$ - i -ый сосед объекта x среди объектов обучающей выборки x_1, \dots, x_l ;
 $y^{(i)}$ - ответ на i -ом соседе объекта x .

Метрический алгоритм классификации:

$$a(x; X^l) = \operatorname{argmax}_{y \in Y} \underbrace{\sum_{i=1}^l [y^{(i)} = y] w(i, x)}_{\Gamma_y(x)} \quad (4.4)$$

$w(i, x)$ - вес, степень близости к объекту x его i -го соседа, неотрицателен, не возрастает по i ;

$\Gamma_y(x)$ - оценка близости объекта x к классу y .

Стоит отметить, что обобщенный метрический алгоритм особенный среди ML алгоритмов. Обычно любой алгоритм формулируется в формате оптимизационной задачи, решая которую находятся параметры модели. В метрическом алгоритме не ставится задача явной оптимизации.

Фактически алгоритму будут различаться тем, как мы выберем вес $w(i, x)$, характеризующий степень близости.

4.1.3. Метод k ближайших соседей (nearest neighbours, kNN)

Если определить вес как $w(i, x) = [i \leq k]$, то и получим метод kNN. Фактически для объекта x проходятся по k его ближайшим соседям, объектов какого класса среди них будет больше, такую метку класса и присваивают объекту x .

Преимущества метода:

- Простота реализации. Никакой оптимизационной задачи решать не надо.
- Единственный гиперпараметр метода k можно оптимизировать с помощью leave-one-out:

$$LOO(k, X^l) = \sum_{i=1}^l [a(x_i; X^l \setminus x_i, k) \neq y_i] \Rightarrow \min_k \quad (4.5)$$

Недостатки метода:

- Возможно неоднозначная классификация, в случае если среди соседей будет равное число объектов двух разных классов.
- Не учитываются значения расстояний. Например, пусть $k = 5$. Первые четыре соседа находятся действительно рядом, а пятый где-то далеко и фактически "ближайшим" соседом не является, но будет учитываться при классификации.

4.1.4. Задача 1

Докажите, что метрика, определенная в \mathbb{R}^2 :

$$p(x, x_i) = \left((x^1 - x_i^1)^2 + \alpha (x^2 - x_i^2)^2 \right)^{\frac{1}{2}}, \alpha > 0 \quad (4.6)$$

является метрикой, соответствующей строгому математическому определению этого понятия. Как в зависимости от величины α будут меняться линии уровня $p(x, x_0) = r$ для данной метрики? Докажите, что при $\alpha = 1$ линиями уровня будут окружностями. Когда введение такого параметра может быть оправдано на практике?

4.1.5. Задача 2

Что будет происходить с качеством классификации методом kNN при экстремально высокой размерности пространства, в котором находятся классифицируемые объекты? Почему все точки в таком пространстве будут фактически "равноудаленными"?

4.1.6. Задача 3

Пусть осуществляется поиск оптимального числа соседей с помощью критерия leave-one-out. Какое будет оптимальное число соседей, если при его подсчете забыть исключить сам объект из обучающей выборки? Как будут выглядеть графики частоты ошибок от числа соседей в смещенном и несмещенном случае?

Часто используемые ядра $K(r)$

$\Pi(r) = [|r| \leq 1]$ — прямоугольное

$T(r) = (1 - |r|)[|r| \leq 1]$ — треугольное

$E(r) = (1 - r^2)[|r| \leq 1]$ — квадратичное (Епанечникова)

$Q(r) = (1 - r^2)^2[|r| \leq 1]$ — квартическое

$G(r) = \exp(-2r^2)$ — гауссовское

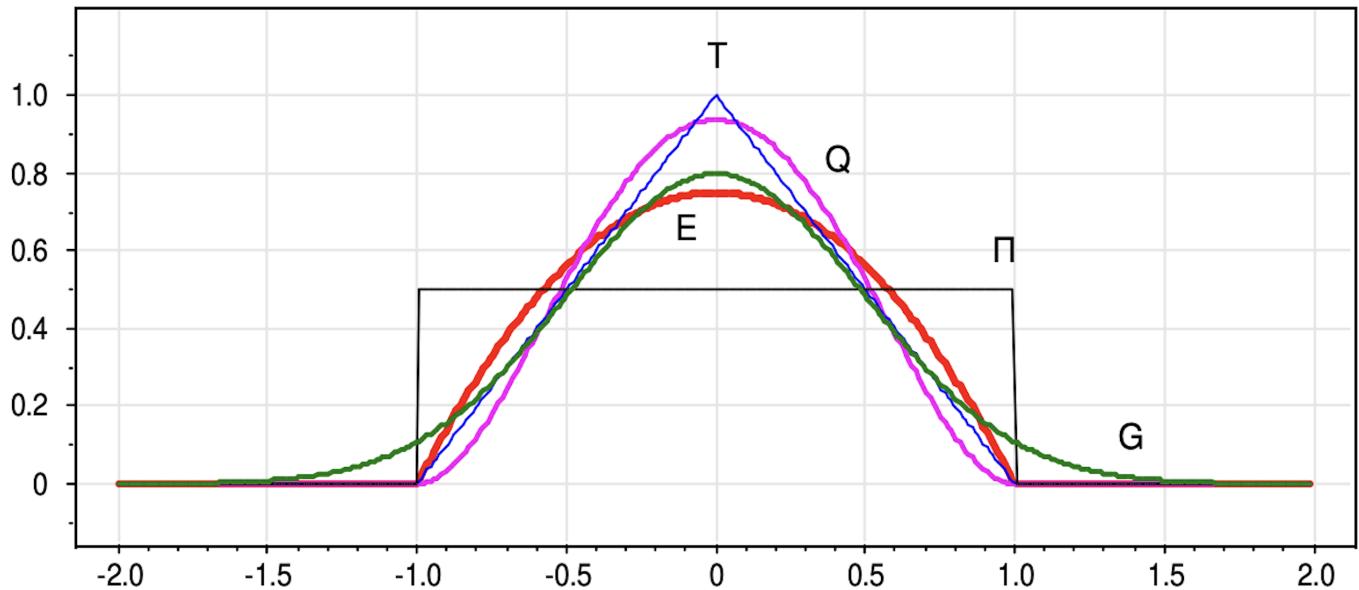
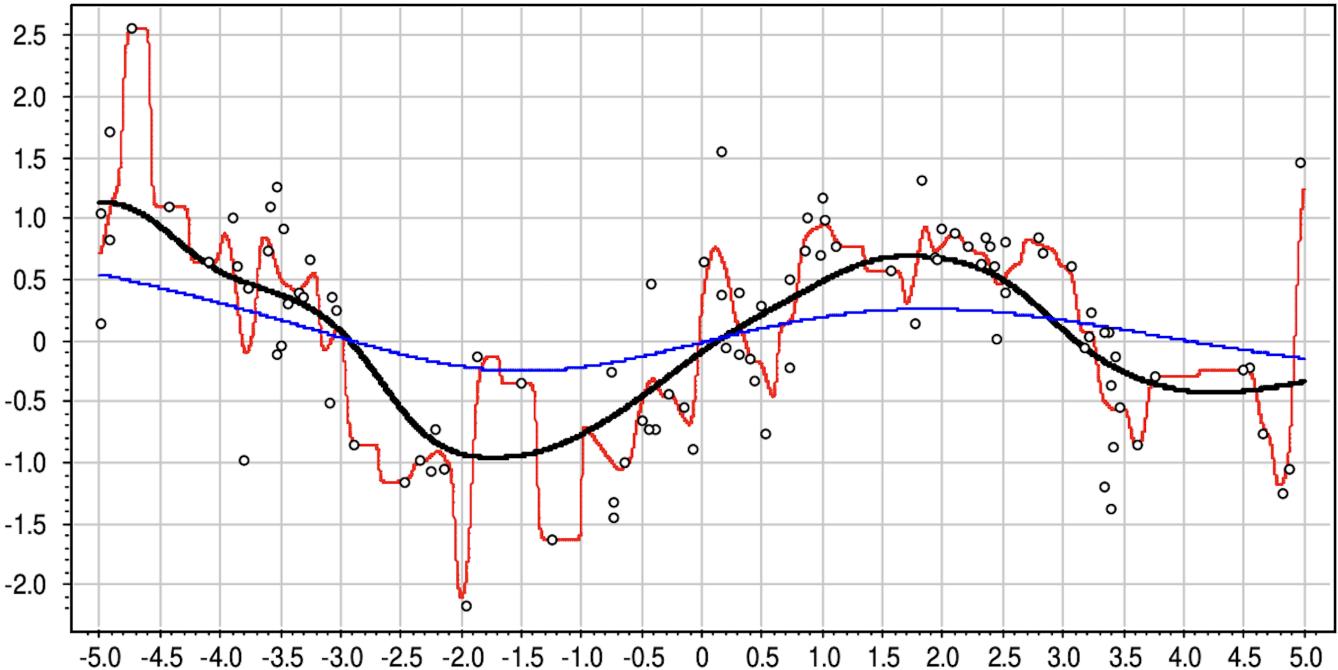


Рис. 4.1. Графики ядер

Выбор ядра K и ширины окна h

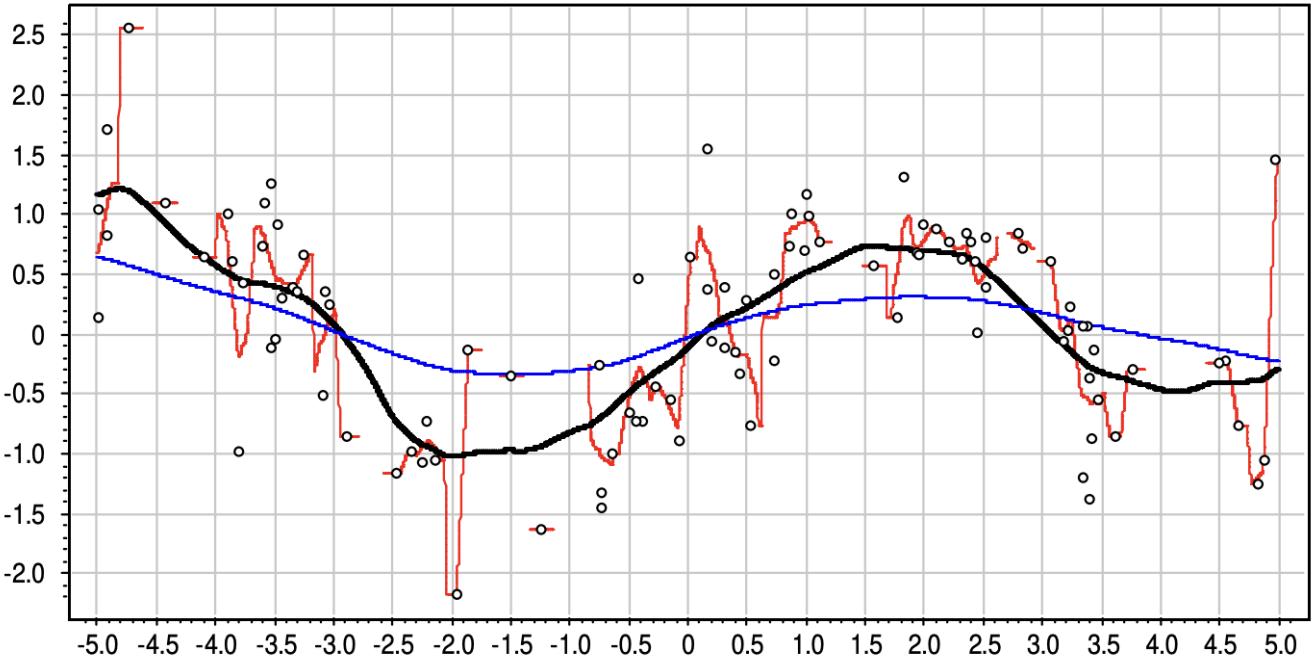
$h \in \{0.1, 1.0, 3.0\}$, гауссовское ядро $K(r) = \exp(-2r^2)$. Графики с различной шириной окна h :



- Гауссовское ядро \Rightarrow гладкая аппроксимация
- Ширина окна существенно влияет на точность аппроксимации

Выбор ядра K и ширины окна h

$h \in \{0.1, 1.0, 3.0\}$, треугольное ядро $K(r) = (1 - |r|)[|r| \leq 1]$. Графики с разными значениями h при треугольном ядре:



- Треугольное ядро \Rightarrow кусочно-линейная аппроксимация
- Аппроксимация не определена, если в окне нет точек выборки

Выбор ядра K и ширины окна h

- Ядро $K(r)$
 - * существенно влияет на гладкость функции $a_h(x)$,
 - * слабо влияет на качество аппроксимации.
- Ширина окна h
 - * существенно влияет на качество аппроксимации.
- Переменная ширина окна по k ближайшим соседям:

$$w_i(x) = K \left(\frac{\rho(x, x_i)}{h(x)} \right), \quad h(x) = \rho(x, x^{(k+1)})$$

где $x^{(k)}$ — k -й сосед объекта x .

- Оптимизация ширины окна по скользящему контролю:

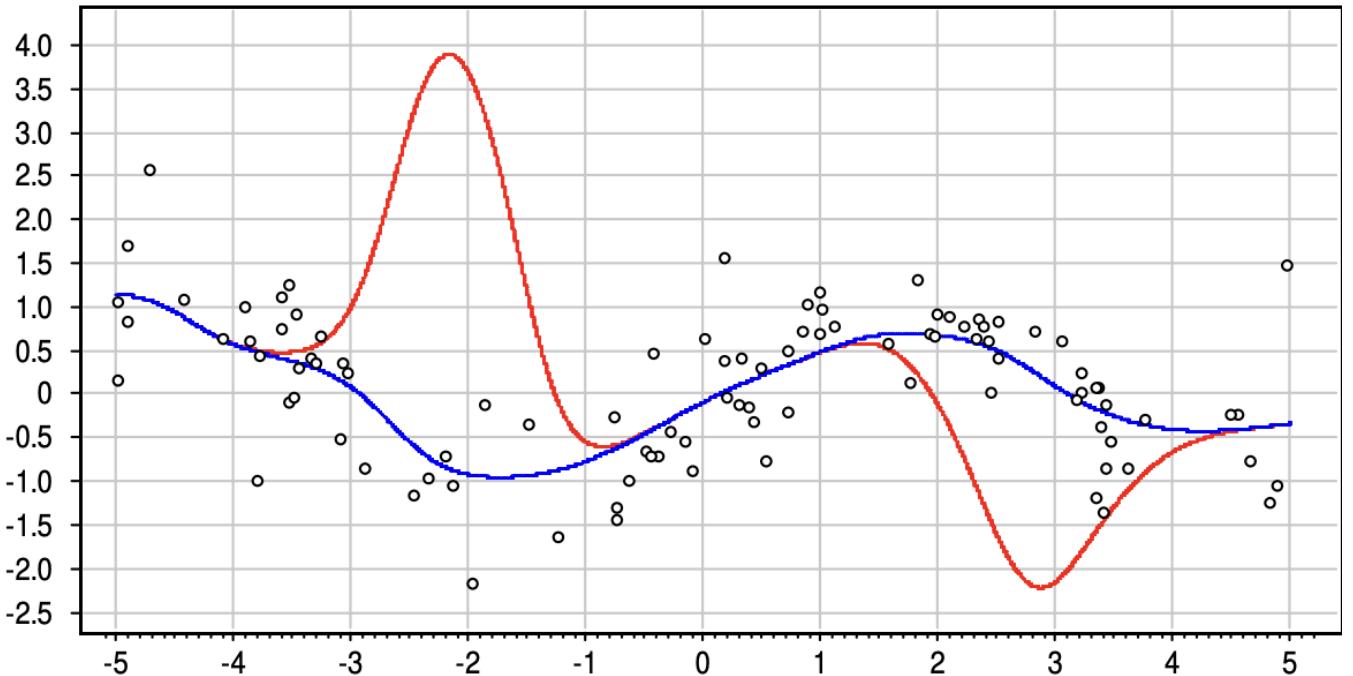
$$\text{LOO}(h, X^\ell) = \sum_{i=1}^{\ell} (a_h(x_i; X^\ell \setminus \{x_i\}) - y_i)^2 \rightarrow \min_h$$

Проблема выбросов (эксперимент на синтетических данных)

$\ell = 100, h = 1.0$, гауссовское ядро $K(r) = \exp(-2r^2)$

Две из 100 точек — выбросы с ординатами $y_i = 40$ и -40

Синяя кривая — выбросов нет



Проблема выбросов и локально взвешенное сглаживание

Проблема выбросов: точки с большими случайными ошибками y_i сильно иска- жают функцию $a_h(x)$

Основная идея:

чем больше величина ошибки $\varepsilon_i = |a_h(x_i; X^\ell \setminus \{x_i\}) - y_i|$,

тем больше прецедент (x_i, y_i) похож на выброс,
тем меньше должен быть его вес $w_i(x)$.

Эвристика:

домножить веса $w_i(x)$ на коэффициенты $\gamma_i = \tilde{K}(\varepsilon_i)$,
где \tilde{K} — ещё одно ядро, вообще говоря, отличное от $K(r)$.

Рекомендация:

квартическое ядро $\tilde{K}(\varepsilon) = K_Q\left(\frac{\varepsilon}{6 \operatorname{med}\{\varepsilon_i\}}\right)$,
где $\operatorname{med}\{\varepsilon_i\}$ — медиана вариационного ряда ошибок.

Алгоритм LOWESS (LOcally WEighted Scatter plot Smoothing)

Вход: X^ℓ — обучающая выборка;

Выход: коэффициенты γ_i , $i = 1, \dots, \ell$;

инициализация: $\gamma_i := 1$, $i = 1, \dots, \ell$;

повторять

— оценки скользящего контроля в каждом объекте:

$$a_i := a_h(x_i; X^\ell \setminus \{x_i\}) = \frac{\sum_{j=1, j \neq i}^{\ell} y_j \gamma_j K\left(\frac{\rho(x_i, x_j)}{h(x_i)}\right)}{\sum_{j=1, j \neq i}^{\ell} \gamma_j K\left(\frac{\rho(x_i, x_j)}{h(x_i)}\right)}, \quad i = 1, \dots, \ell;$$

— $\gamma_i := \tilde{K}(|a_i - y_i|)$, $i = 1, \dots, \ell$;

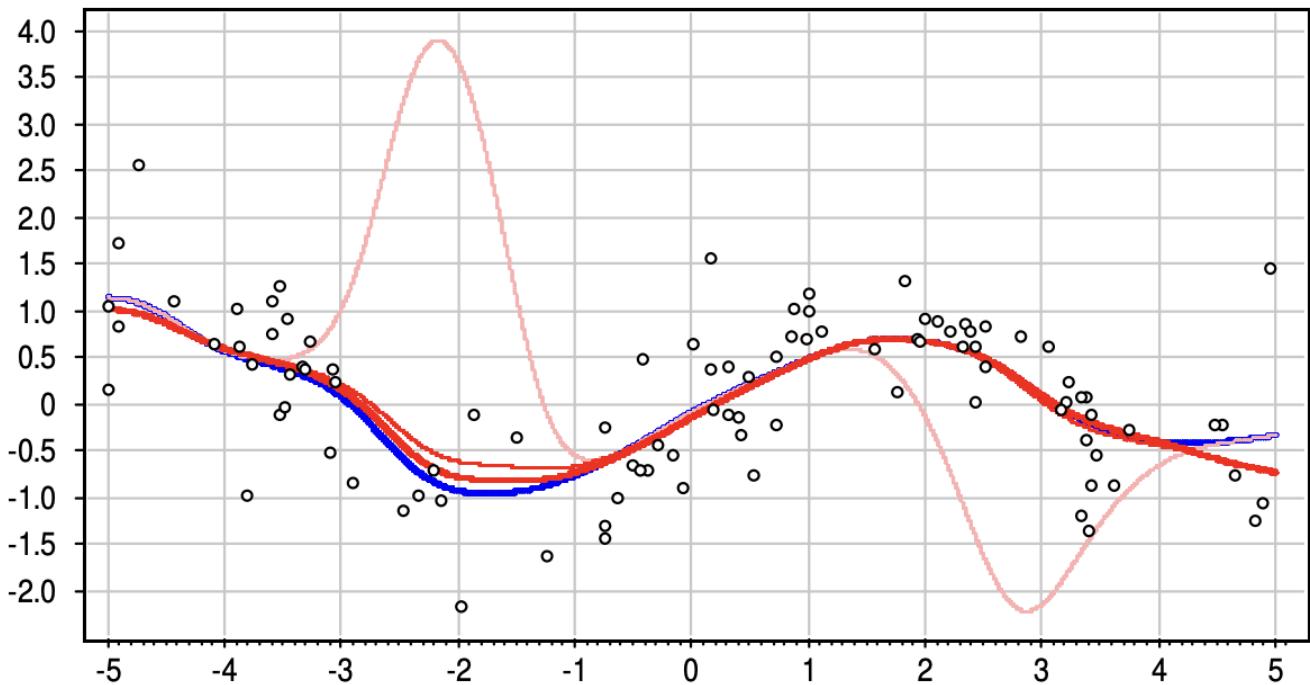
пока коэффициенты γ_i не стабилизируются;

Пример работы LOWESS на синтетических данных

$\ell = 100$, $h = 1.0$, гауссовское ядро $K(r) = \exp(-2r^2)$

Две из 100 точек — выбросы с ординатами $y_i = 40$ и -40

В данном случае LOWESS сходится за несколько итераций:



4.2. Задачи

4.2.1. Задача 1

Объясните, как выбор ядра $K(r)$ влияет на гладкость регрессионной функции $a_h(x)$. Приведите примеры различных ядер и их влияния.

4.2.2. Ответ:

Ядро $K(r)$ определяет форму и вес окрестности точки, используемой для оценки регрессионной функции. Гладкие ядра, такие как гауссовское, дают более плавные оценки, в то время как менее гладкие, такие как прямоугольное, приводят к менее сглаженным функциям. Например:

- Гауссовское ядро $K(r) = \exp(-r^2)$ — даёт плавные, гладкие оценки.
- Треугольное ядро $K(r) = 1 - |r|$ (при $|r| \leq 1$) — более резкое, но все ещё гладкое.
- Прямоугольное ядро $K(r) = 0.5$ (при $|r| \leq 1$) — приводит к кусочно-постоянной функции.

4.2.3. Задача 2

Приведите пример оптимального веса $w_i(x)$ в алгоритме LOWESS, если известно распределение ошибок в данных. Поясните, как это распределение должно влиять на выбор веса, и почему весовое ядро \tilde{K} должно учитывать распределение ошибок.

4.2.4. Ответ:

Если ошибка ε_i распределена согласно некоторому известному распределению, например нормальному, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, то весовой коэффициент должен минимизировать дисперсию предсказания. Весовая функция $\tilde{K}(\varepsilon_i)$ должна убывать, когда ε_i отклоняется от некоторого центрального значения (0 для нормального распределения), чтобы уменьшить влияние выбросов:

$$w_i(x) = \exp\left(-\frac{\varepsilon_i^2}{2\sigma^2}\right)$$

Этот вес сильнее подавляет ошибки, отклоняющиеся от средней, что уменьшает их влияние на итоговую модель. Выбор весового ядра \tilde{K} должен учитывать распределение ошибок, чтобы учсть типичные вариации данных.

4.2.5. Задача 3

На основе метода LOWESS предложите способ оценки доверительного интервала для предсказаний. Выведите формулу для доверительного интервала и объясните, как она может быть использована для оценки надежности модели.

4.2.6. Ответ:

1. Построение модели:
 - Применим LOWESS для построения основной модели, получив предсказанные значения \hat{y}_i для каждого x_i .
2. Оценка остаточной дисперсии:
 - Вычислим остаточные отклонения $e_i = y_i - \hat{y}_i$.
 - Оценим дисперсию ошибок σ^2 как среднеквадратическое отклонение:

$$\sigma^2 = \frac{1}{n-k} \sum_{i=1}^n e_i^2$$

где n — число точек данных, k — число параметров (в случае LOWESS, это скорее степень полинома в локальных регрессиях).

3. Построение доверительного интервала:

- Для каждого предсказанного значения \hat{y}_i построим доверительный интервал, используя стандартное отклонение остаточных ошибок и критические значения из t-распределения:

$$\hat{y}_i \pm t_{\alpha/2, n-k} \cdot \sqrt{\frac{\sigma^2}{n_i}}$$

где $t_{\alpha/2, n-k}$ — квантиль t-распределения с уровнем значимости α , и n_i — эффективное число точек в окрестности x_i (окрестность, которая использовалась для регрессии, может быть выражена размером окна или числом соседей).

4. Использование доверительных интервалов:

- Доверительные интервалы позволяют пользователю оценить, насколько "надёжны" предсказанные значения. Узкие интервалы свидетельствуют о высокой уверенности.

- Визуализация доверительных интервалов на графиках помогает выявлять области, где модель может быть неопределённой или подверженной ошибкам.

4.3. Формула Надарая-Ватсона.

X -объекты Y -ответы $X = (x_i, y_i)_{i=1}^l$ -обучающая выборка Будем обозначать $x_i = (x_i^1, \dots, x_i^n)$ - вектор признаков объекта x_i .

$a(x) = f(x, \theta)$ - параметрическая модель зависимости θ -вектор параметров модели

Метод наименьших квадратов (МНК):

$$Q(\theta; X^l) = \sum_{i=1}^l (f(x_i, \theta) - y_i)^2 w_i \longrightarrow \min_{\theta},$$

где $w(i, x)$ - некоторый вес, отражающий степень важности i-ого объекта. Вес неотрицателен и не возрастает по i .

Сложно определить, какую стоит взять параметрическую модель.

Идея: Будем считать, что $f(x_i, \theta) = \theta$, а w_i зависит от x - т. е. веса объектов задаются в зависимости от того, на каком объекте будет получен объект. Мы будем обучаться для каждого объекта, на котором хотим получить y .

Тогда:

$$Q(\theta; X^l) = \sum_{i=1}^l (\theta - y_i)^2 w(i, x) \longrightarrow \min_{\theta},$$

Наконец, положим

$$w_i(x) = K\left(\frac{\rho(x, x^{(i)})}{h}\right),$$

где $K(r)$ - невозрастающая функция, определённая на неотрицательных числах, называемая ядром, положительная на отрезке $[0, 1]$, h - ширина окна. То есть, чем больше расстояние до $x^{(i)}$, тем меньше оно влияет на x .

Продифференцируем $Q(\theta; X^l)$ по θ и приравняем получившееся значение 0, и отсюда получим формулу ядерного сглаживания Надарая-Ватсона:

$$\theta(x, X^l) = \frac{\sum_{i=1}^l y_i w_i(x)}{\sum_{i=1}^l w_i(x)} = \frac{\sum_{i=1}^l y_i K\left(\frac{\rho(x, x^{(i)})}{h}\right)}{\sum_{i=1}^l K\left(\frac{\rho(x, x^{(i)})}{h}\right)}$$

Средневзвешенное значение y на тех объектах, которые близки к x .

Мы видим, что такой метод для каждого объекта x рассматривает только объекты обучающей выборки, находящиеся на расстоянии не больше h от x . Причём, чем дальше объект от x , тем меньший вклад он даёт в оценку близости.

Обоснование формулы Надарая-Ватсона:

Теорема: пусть выполнены следующие условия:

1. Выборка X^l простая из совместного распределения $p(x, y)$
2. ядро $K(r)$ ограничено $\int_0^{+\infty} K(r) dr < \infty$, $\lim_{r \rightarrow \infty} r K(r) = 0$. (из этого условия следует, что ядро приводит большие расстояния в 0)
3. зависимость $E(y|x)$ не имеет вертикальных асимптот
- $E(y^2|x) = \int_Y y^2 p(y|x) dy < \infty$ при любом $x \in X$
4. последовательность h_l (ширина окна) убывает с ростом выборки, но не слишком быстро и не слишком медленно $\lim_{l \rightarrow \infty} h_l = 0$, $\lim_{l \rightarrow \infty} l h_l = 0$

Тогда имеет место сходимость по вероятности: $\theta(x, X^l) \xrightarrow{P} E(y|x)$
(здесь E -мат. ожидание y при условии x)

в любой точке $x \in X$, в которой $E(y|x)$, $p(x)$, $D(y|x)$ -непрерывны, $p(x) > 0$

По теореме даже если распределение $p(x, y)$ и $E(y|x)$ не известны, θ будет стремится к нему с ростом длины обучающей выборки

Замечание. Ширина окна отвечает за пере или недообучение, следовательно сильно влияет на качество аппроксимации.

Для подбора наилучшей модели оптимизируются параметры:

- ширина окна h ;
- ядро K .

Приведём примеры наиболее часто используемых ядер:

- $K_1(r) = (1 - r^2)[r \leq 1]$ - ядро Епанечникова;
- $K_2(r) = (1 - r^2)^2[r \leq 1]$ - квартическое ядро;
- $K_3(r) = (1 - |r|)[r \leq 1]$ - треугольное ядро;
- $K_4(r) = [r \leq 1]$ - прямоугольное ядро;
- $K_5(r) = e^{-2r^2}$ - гауссовское ядро.

Замечание. Существует проблема выбросов, при которой точки с большими случайными y_i сильно искажают итоговую функцию. С этим можно справиться, домножив веса w_i на коэффициенты $\gamma_i = K^*(\epsilon_i)$, где $\epsilon_i = |\theta(x_i, X^l) - y_i|$ подразумевается, что из X^l исключили x_i , K^* -другое ядро

Задача 1. Объяснить, почему не стоит брать финитное ядро на неизвестных данных при малой ширине окна h . Объяснить, почему нельзя просто взять большое h , чтобы решить проблему из предыдущего пункта.

Ответ: При малой ширине окна h и финитном ядре у некоторых точек может не оказаться соседей, попадающих в радиус ядра K . На случай такого события в формулу Надарая-Ватсона в знаменатель добавляют малый член, чтобы не делить на 0.

Если увеличить ширину окна h , то для всех точек с большей вероятностью смогут найти соседи, но в таком случае точность аппроксимации ухудшится.

Задача 2. как ядро влияет на гладкость функции? Ответ обосновать.

Ответ: Будем считать, что ядро не обращается в 0. Вспомним, что функция, являющаяся результатом деления двух гладких функций, является гладкой. Сумма

гладких функций, умноженных на число- также гладкая функция Пусть ядро K - гладкое, тогда $\sum_{i=1}^l y_i K\left(\frac{\rho(x, x^{(i)})}{h}\right)$ -гладкое, и функция $\frac{\sum_{i=1}^l y_i K\left(\frac{\rho(x, x^{(i)})}{h}\right)}{\sum_{i=1}^l K\left(\frac{\rho(x, x^{(i)})}{h}\right)}$ будет гладкой

Следовательно, выбор ядра сильно влияет на гладкость.

Задача 3. Имеются следующие данные: $x = [1.2, 2.7, 3.1, 4.5, 5.3]$, $y = [3.5, 1.8, 5.2, 2.1, 4.7]$. Необходимо предсказать значение y для $x = 2.5$, используя формулу Надарая-Ватсона с квартическим ядром и $h = 1$.

Решение: Для каждой точки из набора данных вычисляем вес $K\left(\frac{x-x_i}{h}\right)$, где $x = 2.5$, $h = 1$, Суммируем все вычисленные веса и делим каждый вес на эту сумму, чтобы получить нормированные веса.

веса не равны 0 только у точек $x=2.7$ $x=3.1$, их веса равны соответственно 0.9216 и 0.4096.

При нормировке получим веса 0.6923 и 0.3077 Предсказание $\hat{y}(2.5)$ вычисляется как взвешенная сумма значений по уже известной формуле
примерный $\hat{y}(2.5) = 2.85$

4.4. Влияние выбора метрики на качество работы kNN

Метод k -ближайших соседей (kNN) является одним из базовых алгоритмов машинного обучения и широко применяется в задачах классификации и регрессии. Этот алгоритм относится к метрическим методам, так как выбор ближайших соседей основывается на измерении расстояний между объектами. Главным фактором, определяющим качество модели, является выбор метрики расстояния, так как она определяет, какие объекты считаются «похожими».

4.4.1. Сущность метода k -ближайших соседей

Алгоритм k -ближайших соседей работает следующим образом:

1. Для нового объекта вычисляется расстояние до всех объектов обучающей выборки.
2. Выбираются k ближайших объектов в соответствии с выбранной метрикой.
3. Класс нового объекта определяется на основе классов выбранных соседей (например, по принципу большинства для задач классификации).

4. В задачах регрессии целевое значение нового объекта вычисляется как среднее (или взвешенное среднее) значений соседей.

Ключевым аспектом метода является расстояние, вычисляемое на основе метрики, которая влияет на работу алгоритма, в том числе на точность и устойчивость модели.

4.4.2. Популярные метрики расстояния

1. **Евклидова метрика (L_2 -норма):** Одна из самых популярных метрик, измеряющая геометрическое расстояние между точками в пространстве. Формула:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Особенности:

- Подходит для данных, где все признаки одинаково масштабированы.
- Чувствительна к выбросам, так как квадрат отклонения значительно увеличивает вклад большого расхождения.

2. **Манхэттенская метрика (L_1 -норма):** Измеряет расстояние как сумму абсолютных разностей координат. Формула:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

Особенности:

- Лучше подходит для разреженных данных.
- Менее чувствительна к выбросам по сравнению с Евклидовой метрикой.

3. **Метрика Минковского:** Обобщение Евклидовой и Манхэттенской метрик. Формула:

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Особенности:

- При $p = 2$ совпадает с Евклидовой метрикой, при $p = 1$ — с Манхэттенской.

- Позволяет гибко настраивать степень влияния больших отклонений через параметр p .

4. Косинусное расстояние: Измеряет угол между векторами, игнорируя их длину. Формула:

$$d(x, y) = 1 - \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|}$$

Особенности:

- Часто применяется для текстовых данных (например, векторов TF-IDF).
- Устойчиво к изменениям масштаба векторов.

5. Метрика Чебышёва: Учитывает максимальное различие по одной из координат. Формула:

$$d(x, y) = \max_i |x_i - y_i|$$

Особенности:

- Удобна для задач, где критически важно учитывать наибольшее расхождение.
- Формирует кубические границы ближайших соседей.

4.4.3. Влияние выбора метрики на качество работы модели

Выбор метрики может значительно изменить поведение алгоритма k -ближайших соседей. Основные факторы влияния:

- 1. Геометрия данных:** Разные метрики формируют разные границы классов. Например, Евклидова метрика создает окружные границы, а Манхэттенская — прямоугольные. В задачах с нелинейными разделяющими гиперплоскостями может потребоваться комбинированный подход или нестандартные метрики.
- 2. Масштаб признаков:** Метрики, такие как Евклидова, чувствительны к различиям в масштабе признаков. Например, если один признак имеет диапазон $[0, 1]$, а другой — $[0, 1000]$, последний будет доминировать. Для решения этой проблемы применяется нормализация или стандартизация данных.
- 3. Шум и выбросы:** Метрики по-разному реагируют на шум. Евклидова метрика чувствительна к выбросам, так как квадратичное расстояние

значительно увеличивается для больших расхождений. Метрики, такие как Манхэттенская или косинусное расстояние, менее чувствительны к шуму.

4. **Высокая размерность:** В задачах с большим количеством признаков (проблема «проклятия размерности») расстояния между всеми объектами становятся примерно одинаковыми. Это снижает различимость ближайших соседей, что делает выбор метрики критически важным.
5. **Специфика задачи:** Например, для задач обработки текстов косинусная метрика часто оказывается лучше, так как она учитывает только направление векторов, игнорируя их длину. Для задач с пространственными данными чаще применяются Евклидова или Манхэттенская метрика.

4.4.4. Примеры задач

1. **Теоретическая задача:** Рассмотрите два набора данных из двух классов, представленных точками на плоскости. Постройте границы разделения классов при использовании:
 - Евклидовой метрики,
 - Манхэттенской метрики.
 Объясните, как выбор метрики влияет на форму границ.
2. **Практическая задача:** Используя набор данных `Iris`, обучите модель k -ближайших соседей с Евклидовой, Манхэттенской и косинусной метриками. Сравните метрики качества (точность, F1-меру) и сделайте вывод о том, какая метрика работает лучше и почему.
3. **Исследовательская задача:** Для синтетических данных с перекрывающимися классами разработайте алгоритм выбора оптимальной метрики с использованием кросс-валидации. Постройте графики зависимости точности от параметра k для разных метрик.

4.4.5. Заключение

Выбор метрики расстояния является ключевым фактором, определяющим качество работы метода k -ближайших соседей. Оптимальная метрика зависит от природы данных, задачи и требований к точности и устойчивости модели. Для улучшения результатов рекомендуется проводить предварительный анализ данных, нормализацию признаков и тестирование нескольких метрик с использованием кросс-валидации.

4.5. Более быстрые оптимизации kNN.

Описанный в пункте 4.3 тип KNN называется Brute-Force, поскольку в нём используется метод полного перебора для поиска ближайших соседей, что делает его простым в реализации, но слишком медленным при работе с большим объемом данных. Для решения данной проблемы в реализации scikit-learn предусмотрены более продвинутые методы, основанные на бинарных деревьях, что позволяет получить значительный прирост в производительности.

4.5.1. BallTree

BallTree — это древовидная структура, в основе которой лежит разбиение исходного пространства данных на вложенные гиперсфераe, что позволяет более эффективно отсекать большие области пространства, в которых отсутствуют ближайшие соседи для точек. В большинстве случаев такой алгоритм подходит для данных с произвольной метрикой расстояния.

Построение BallTree состоит из следующих шагов:

1. из множества точек выбирается одна случайным образом и для неё находится самая дальняя точка;
2. далее все точки разбиваются на гиперсфераe (узлы) по ближайшему расположению к двум точкам из шага 1;
3. затем данный процесс повторяется рекурсивно для каждой гиперсфераe, пока в ней не останется определённое количество точек или не будет достигнута заданная глубина дерева.

При поиске k-ближайших соседей для новой точки, алгоритм сравнивает расстояние от заданной точки до центра каждого дочернего узла и оставляет лишь те, в которых данное расстояние меньше радиуса узлов.

Для оценки качества полученной древовидной структуры и её дальнейшей оптимизации очень полезной будет информация о пересекающихся гиперсфераe (узлах) A и B в метрике M, расстояние между которыми можно определить следующим образом:

$$d_M(A, B) = \max(0, d_M(c_A, c_B) - r_A - r_B)$$

где c_A и c_B — центры сфер, а r_A и r_B — их радиусы.

В данном случае оптимизация BallTree с учётом пересекающихся гиперсфераe (узлов) может включать в себя следующие подходы:

балансировка дерева: поскольку пересечение гиперсфер может указывать на несбалансированность дерева, перебалансировка его узлов позволяет улучшить эффективность поиска, минимизируя количество посещаемых узлов;

выбор оптимального размера листа: в случае сильного пересечения гиперсфер, увеличение размера листа может уменьшить количество узлов, что также ускорит поиск;

слияние узлов: полезно при значительном пересечении, что также уменьшает их общее количество и используется в предыдущих пунктах;

выбор порядка обхода: информация о структуре пересечения может сделать более эффективным порядок посещаемых узлов, начиная проверку с наиболее вероятных кандидатов.

Стоит отметить, что описанные методы оптимизации плюс-минус похожим образом могут быть применимы и для алгоритма ниже.

4.5.2. KD-Tree

KD-Tree (*k*-dimensional tree) — ещё одна древовидная структура, отдалённо напоминающая BallTree, однако в данном случае используются гиперплоскости для разбиения точек вместо гиперсфер, что позволяет также эффективно оставлять лишь те области пространства данных, в которых могут присутствовать ближайшие соседи. Обычно KD-Tree больше подходит для данных с евклидовой или манхэттенской метрикой расстояния.

Построение KD-Tree состоит из следующих шагов:

- 1) из множества точек выбирается одна из координат (обычно поочередно для каждого уровня дерева, но можно и случайным образом) и по ней вычисляется медиана;
- 2) далее все точки разбиваются на два узла (подмножества) по отношению к медиане: на те, у которых значение выбранной координаты меньше либо равно медиане, и на те, у которых больше;
- 3) данный процесс повторяется рекурсивно для каждого узла, пока в нём не останется определённое количество точек или не будет достигнута заданная глубина дерева.

При поиске ближайших соседей для новой точки, алгоритм сравнивает значение заданной точки с медианой в каждом узле, выбирая таким образом ближайшее подпространство, которое будет листом с ближайшими соседями. Возвращаясь обратно к корню, алгоритм будет сравнивать точки в текущем узле с ближайшими соседями и обновлять их значения в случае нахождения более близких к заданной точке.

4.5.3. Примеры задач

Задача 1. У вас есть два набора данных:

Набор данных А: 10 миллионов точек в 5-мерном пространстве с использованием Евклидовой метрики.

Набор данных В: 1 миллион точек в 10-мерном пространстве с кастомной метрикой расстояния. Вопросы:

Какую структуру данных (KD-Tree или BallTree) вы выберете для каждого набора данных? Объясните ваш выбор. Как влияет размерность пространства на эффективность KD-Tree?

Ответ:

Для набора данных А оптимально использовать KD-Tree, так как он хорошо работает с Евклидовой метрикой, особенно в пространствах с небольшой размерностью. Для набора данных В лучше подходит BallTree, так как он поддерживает произвольные метрики и может быть эффективнее в высоких размерностях.

С увеличением размерности эффективность KD-Tree значительно снижается из-за проклятия размерности: гиперплоскости разбиения теряют свою полезность, и увеличивается количество узлов, которые нужно проверять.

Задача 2. Имеется 2D-набор данных с точками:

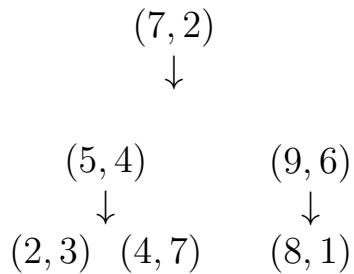
(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)

1. Постройте KD-Tree для этих точек.
2. Какие оптимизации можно применить к этому дереву, если оно оказалось несбалансированным?

Ответ:

1. Построение дерева:
 - Выбор корня: координата x , медиана по $x \rightarrow$ точка (7, 2).
 - Левое поддерево ($x < 7$): медиана по $y \rightarrow$ точка (5, 4).
 - * Левый потомок: (2, 3).
 - * Правый потомок: (4, 7).
 - Правое поддерево ($x > 7$): медиана по $y \rightarrow$ точка (9, 6).
 - * Левый потомок: (8, 1).

Итоговое дерево:



2. Оптимизации:

- Если дерево несбалансировано, можно:
 - * Перестроить дерево, используя медиану всех точек на каждом уровне.
 - * Увеличить размер листов (количество точек в узле) для уменьшения глубины дерева.

Задача 3: Поиск ближайшего соседа с использованием BallTree

Дано: гиперсфера в 3D-пространстве, центры $(1, 1, 1)$, $(4, 4, 4)$, радиусы $r_1 = 2$, $r_2 = 3$. Найдите ближайшего соседа для точки $(2, 2, 2)$, используя алгоритм BallTree.

Решение:

1. Вычисляем расстояние от точки до центров гиперсфер:

$$d((2, 2, 2), (1, 1, 1)) = \sqrt{(2-1)^2 + (2-1)^2 + (2-1)^2} = \sqrt{3} \approx 1.73$$

$$d((2, 2, 2), (4, 4, 4)) = \sqrt{(2-4)^2 + (2-4)^2 + (2-4)^2} = \sqrt{12} \approx 3.46$$

2. Сравниваем с радиусами:

- Точка $(2, 2, 2)$ находится внутри гиперсферы с центром $(1, 1, 1)$, так как $1.73 < 2$.
- Точка не входит в гиперсферу с центром $(4, 4, 4)$, так как $3.46 > 3$.

Ответ: Ближайший сосед — точка в гиперсфере с центром $(1, 1, 1)$.

4.6. Расстояние Махalanобиса в метрических методах классификации и регрессии

Метрики являются фундаментальными инструментами в задачах классификации и регрессии, основанных на сходстве или расстоянии между объектами. Они опре-

деляют, как мы измеряем "близость" между парами объектов в пространстве признаков.

Расстояние Махalanобиса является обобщением Евклидова расстояния, которое учитывает не только разности значений признаков, но и статистические свойства данных, такие как дисперсии и корреляции между признаками. Это делает его особенно полезным в случае многомерных данных с неоднородной структурой.

4.6.1. Определение:

Расстояние Махalanобиса между вектором признаков объекта \mathbf{x} и средним вектором μ определяется как:

$$d(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{S}^{-1} (\mathbf{x} - \mu)}$$

где:

- \mathbf{x} — вектор признаков объекта,
- μ — вектор средних значений признаков (например, центра кластера или класса),
- \mathbf{S} — ковариационная матрица признаков,
- \mathbf{S}^{-1} — обратная матрица к \mathbf{S} ,
- $(\mathbf{x} - \mu)^T$ — транспонированный вектор разностей.

4.6.2. Преимущества расстояния Махalanобиса:

1. Расстояние Махalanобиса не зависит от масштаба признаков, что устраняет необходимость в предварительной нормализации или стандартизации данных.
2. Метрика учитывает корреляции между признаками, что позволяет более точно измерять расстояние в пространстве, где признаки взаимосвязаны.
3. Она адаптируется к распределению данных, отражая их внутреннюю структуру и дисперсию, что может улучшить результаты в задачах классификации и кластеризации.

4.6.3. Недостатки расстояния Махalanобиса:

1. Необходимо вычислять ковариационную матрицу \mathbf{S} и обратную матрицу \mathbf{S}^{-1} , что является дорогостоящей операцией, особенно при высокой

размерности данных.

2. Для точной оценки ковариационной матрицы требуется достаточно большой объём данных, число наблюдений должно значительно превышать число признаков.
3. Если ковариационная матрица не обратима (например, из-за линейной зависимости между признаками или недостаточного числа наблюдений), то невозможно напрямую вычислить \mathbf{S}^{-1} .
4. Выбросы могут сильно искажать оценку ковариационной матрицы и средних значений, что приводит к неправильному вычислению расстояний.
5. Сложности в интерпретации: Поскольку расстояние учитывает сложные связи в данных, интерпретация значений может быть менее интуитивной по сравнению с Евклидовым расстоянием.

4.6.4. Практические аспекты использования:

1. Регуляризация ковариационной матрицы: Для решения проблем с вырожденностью часто применяется добавление небольшого значения к диагональным элементам ковариационной матрицы (тирихоновская регуляризация).
2. Понижение размерности: Можно использовать методы понижения размерности (например, анализ главных компонентов) для уменьшения вычислительной нагрузки и устранения мультиколлинеарности.
3. Отбор признаков: Исключение избыточных или некоррелированных признаков может улучшить оценку ковариационной матрицы и качество метрики.

4.6.5. Применение в методах классификации и регрессии

1. Классификация по ближайшим соседям (k-NN): Использование расстояния Махalanобиса вместо Евклидова может улучшить качество классификации в случаях, когда признаки имеют различные масштабы или коррелированы.
2. Дискриминантный анализ: В линейном дискриминантном анализе (LDA) расстояние между классами измеряется с помощью расстояния Махalanобиса, что позволяет учитывать разброс и ориентацию классов в пространстве признаков.
3. Обнаружение выбросов: Объекты с большим расстоянием Махalanобиса от центра распределения могут рассматриваться как выбросы.

4.6.6. Задачи

Задача 1: Использование расстояния Махalanобиса в классификации k-NN

Вы применяете метод классификации k-близайших соседей (k-NN) для разделения объектов на два класса: Класс А и Класс В. У каждого класса известны средние векторы признаков и ковариационные матрицы:

Класс А:

$$\mu_A = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad \mathbf{S}_A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Класс В:

$$\mu_B = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \mathbf{S}_B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Новый объект имеет признаки $\mathbf{x} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$.

Вычислите расстояния Махalanобиса от объекта \mathbf{x} до каждого класса и определите, к какому классу следует отнести данный объект.

Решение:

1. Расстояние до Класса А:

- Вычисляем разность:

$$\mathbf{x} - \mu_A = \begin{bmatrix} 4 - 2 \\ 4 - 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

- Находим обратную ковариационную матрицу \mathbf{S}_A^{-1} :

$$\mathbf{S}_A^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

- Вычисляем расстояние:

$$d_A = \sqrt{(\mathbf{x} - \mu_A)^\top \mathbf{S}_A^{-1} (\mathbf{x} - \mu_A)} = \sqrt{\begin{bmatrix} 2 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}} = \sqrt{(2)(1) + (2)(1)} = \sqrt{4} = 2$$

2. Расстояние до Класса В:

- Вычисляем разность:

$$\mathbf{x} - \mu_B = \begin{bmatrix} 4 - 5 \\ 4 - 5 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

- Обратная ковариационная матрица \mathbf{S}_B^{-1} уже известна, так как \mathbf{S}_B — единичная матрица:

$$\mathbf{S}_B^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Вычисляем расстояние:

$$d_B = \sqrt{(\mathbf{x} - \mu_B)^\top \mathbf{S}_B^{-1} (\mathbf{x} - \mu_B)} = \sqrt{(-1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} \approx 1,414$$

3. В итоге:

Поскольку $d_B < d_A$, объект \mathbf{x} принадлежит Классу В.

Задача 2: Обнаружение аномалий с помощью расстояния Махаланобиса

В задаче обнаружения мошеннических транзакций используются два признака: сумма покупки (X_1) и количество товаров (X_2). Для нормальных транзакций известно:

- Средний вектор:

$$\mu = \begin{bmatrix} 100 \\ 10 \end{bmatrix}$$

- Ковариационная матрица:

$$\mathbf{S} = \begin{bmatrix} 400 & 0 \\ 0 & 4 \end{bmatrix}$$

Новая транзакция имеет значения $\mathbf{x} = \begin{bmatrix} 160 \\ 14 \end{bmatrix}$.

Вычислите расстояние Махаланобиса и определите, является ли эта транзакция аномальной, используя пороговое значение $d_{\text{порог}} = 3$.

Решение:

1. Вычисляем разность:

$$\mathbf{x} - \mu = \begin{bmatrix} 160 - 100 \\ 14 - 10 \end{bmatrix} = \begin{bmatrix} 60 \\ 4 \end{bmatrix}$$

2. Находим обратную ковариационную матрицу \mathbf{S}^{-1} :

$$\mathbf{S}^{-1} = \begin{bmatrix} \frac{1}{400} & 0 \\ 0 & \frac{1}{4} \end{bmatrix}$$

3. Вычисляем расстояние:

$$d = \sqrt{(\mathbf{x} - \mu)^\top \mathbf{S}^{-1} (\mathbf{x} - \mu)} = \sqrt{(60)^2 \times \frac{1}{400} + (4)^2 \times \frac{1}{4}} = \sqrt{\frac{3600}{400} + \frac{16}{4}} = \sqrt{9 + 4} = \sqrt{13} \approx$$

4. Поскольку $d = 3,606 > d_{\text{порог}} = 3$, транзакция считается аномальной.

Задача 3: Выбор метрики расстояния при наличии коррелированных признаков

Вы работаете с набором данных, содержащим два признака: X_1 и X_2 . При анализе данных вы обнаружили, что признаки X_1 и X_2 имеют сильную положительную корреляцию.

Вы планируете использовать метод k-ближайших соседей (k-NN) для классификации новых объектов. Возникает вопрос: какую метрику расстояния следует использовать — Евклидово расстояние или расстояние Махalanобиса? Объясните свой выбор.

Решение:

Предполагаемые рассуждения:

- Поскольку признаки X_1 и X_2 сильно коррелированы, это означает, что они содержат избыточную информацию. Изменения в одном признаке сопровождаются изменениями в другом.
- При использовании Евклидова расстояния каждый признак рассматривается независимо, без учета корреляции. Это может привести к тому, что влияние коррелированных признаков будет преувеличено, и объекты будут казаться более далекими друг от друга вдоль направления корреляции.
- Расстояние Махalanобиса учитывает ковариацию между признаками. За счет включения ковариационной матрицы оно корректирует влияние коррелированных признаков, "сжимая" пространство вдоль направления корреляции.

Вывод:

В данном случае целесообразнее использовать расстояние Махalanобиса, так как оно учитывает корреляцию между признаками и обеспечивает более точное измерение сходства между объектами. Это улучшит качество классификации методом k-NN, так как позволит корректно определять ближайших соседей основываясь на истинной структуре данных.

4.7. Профиль компактности и оценка обобщающей способности

Пусть стоит задача улучшения алгоритма 1NN. Этого можно добиться, оставив из выборки только нужные элементы, то есть произвести выбор эталонных элементов. Для такого выбора необходимо выполнить минимизацию CCV.

4.7.1. CCV

Полный скользящий контроль (complete cross-validation, CCV):

$$\text{CCV}(X^L) = \frac{1}{C_L^\ell} \sum_{X^\ell \sqcup X^k} \frac{1}{k} \sum_{x_i \in X^k} [a(x_i; X^\ell) \neq y_i]$$

- частота ошибок алгоритма на контрольной выборке X^k , усреднённая по всем C_L^ℓ разбиениям выборки $X^L = X^\ell \sqcup X^k$ на обучающую подвыборку X^ℓ и контрольную X^k .

Для дальнейших рассуждений введем понятие профиля компактности.

4.7.2. Профиль компактности

Профиль компактности выборки X^L - это функция доли объектов x_i , у которых m -й сосед $x_i^{(m)}$ лежит в другом классе:

$$\Pi(m) = \frac{1}{L} \sum_{i=1}^L [y_i \neq y_i^{(m)}]; \quad m = 1, \dots, L-1,$$

$x_i^{(m)}$ – m -й сосед объекта x_i среди X^L ;
 $y_i^{(m)}$ – ответ на m -м соседе объекта x_i .

4.7.3. CCV для метода 1NN

С учетом введенного $\Pi(m)$ справедлива следующая теорема о точном выражении CCV для метода 1NN:

$$\text{CCV}(X^L) = \sum_{m=1}^k \Pi(m) \frac{C_{L-1-m}^{\ell-1}}{C_{L-1}^\ell}.$$

Нетрудно заметить, что CCV при длине выборки равной 1 совпадает с LOO (leave-one-out).

4.7.4. Пример профилей компактности

Проанализируем графики.

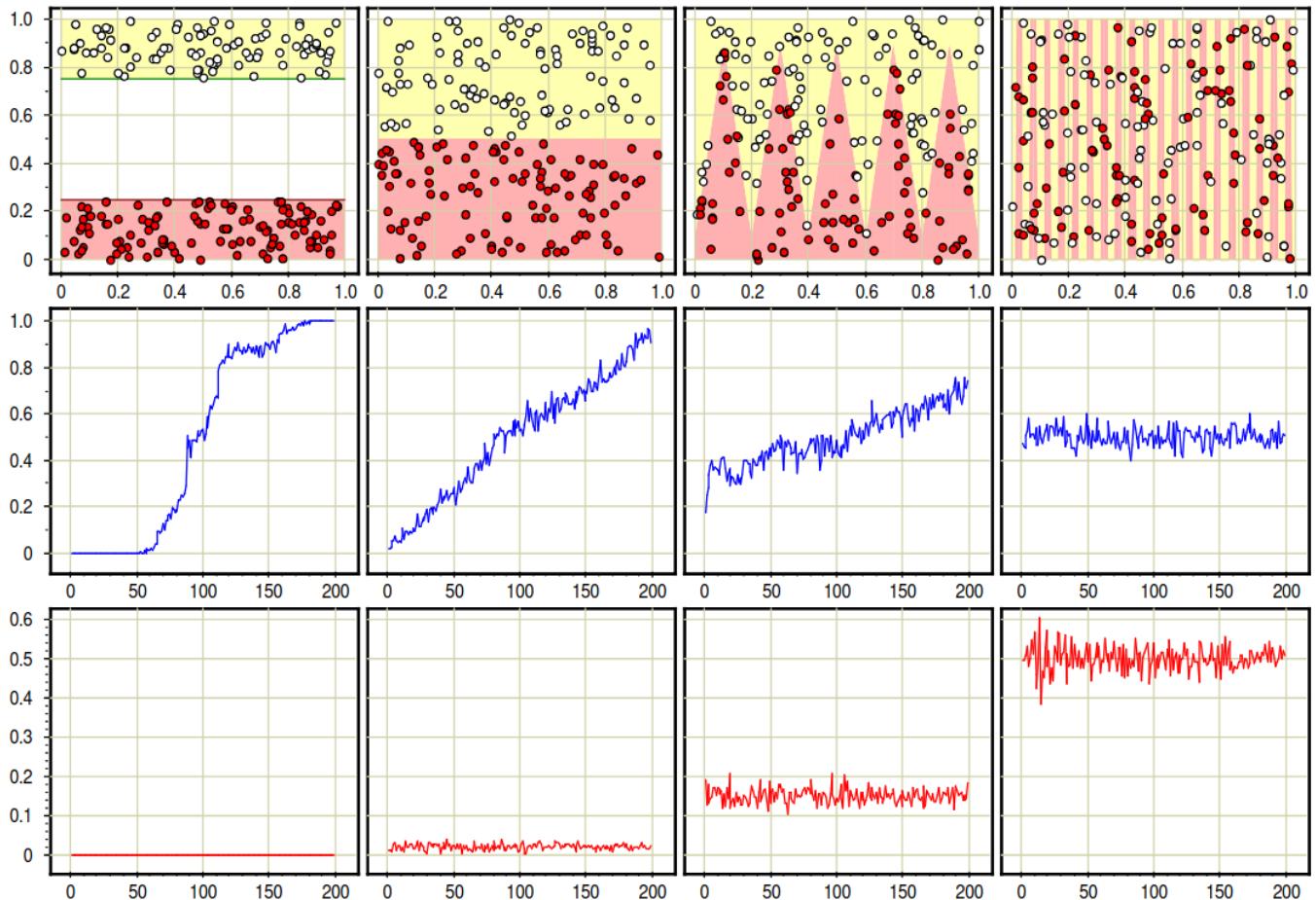


Рис. 4.2. Верхний ряд - два класса, средний ряд - профиль компактности $\Pi(m)$, нижний ряд - зависимость CCV от длины контроля.

Первый набор: классы разнесены далеко друг от друга. Из профиля компактности видно, что у всех объектов 50 ближайших соседей лежат в своем классе. Значит, у этой задачи будет почти 100% качество при использовании kNN алгоритма.

Второй набор: классы расположены так, что между ними проходит граница. Профиль компактности линейно растет при увеличении числа соседей, причем для малого количества соседей имеем ненулевой $\Pi(m)$.

Третий набор: классы проникают друг в друга. Начальный участок профиля компактности выше нуля, значит, у алгоритма kNN будет больше ошибок.

Четвертый набор: классы равномерно распределены. Профиль компактности находится на уровне 0.5, и у kNN нет возможности адекватно решить данную задачу: имеем случайное гадание.

Замечание: из нижнего ряда имеем, что CCV почти не зависит от длины контроля, то есть нет причины выделять в контрольную выборку более одного объекта и достаточно использовать CCV(1).

4.7.5. Задачи

Задача 1: доказать точное выражение CCV для метода 1NN.

Решение:

$$\begin{aligned}
 \text{CCV} &= \frac{1}{C_L^\ell} \sum_{X^\ell \sqcup X^k} \frac{1}{k} \sum_{i=1}^L [x_i \in X^k] [a(x_i; X^\ell) \neq y_i] = \\
 &= \sum_{X^\ell \sqcup X^k} \sum_{i=1}^L \sum_{m=1}^k \frac{[y_i^{(m)} \neq y_i]}{k C_L^\ell} [x_i^{(m)} \in X^\ell] [x_i, x_i^{(1)}, \dots, x_i^{(m-1)} \in X^k] = \\
 &= \sum_{m=1}^k \sum_{i=1}^L \frac{[y_i^{(m)} \neq y_i]}{k C_L^\ell} \sum_{X^\ell \sqcup X^k} [x_i^{(m)} \in X^\ell] [x_i, x_i^{(1)}, \dots, x_i^{(m-1)} \in X^k] = \\
 &= \sum_{m=1}^k \sum_{i=1}^L \frac{[y_i^{(m)} \neq y_i]}{k C_L^\ell} C_{L-1-m}^{\ell-1} = \underbrace{\sum_{m=1}^k \frac{1}{\sum_{i=1}^L [y_i^{(m)} \neq y_i]} \frac{C_{L-1-m}^{\ell-1}}{C_{L-1}^\ell}}_{\Pi(m)}.
 \end{aligned}$$

Задача 2: получить выражения для CCV при малой длине контроля.

Решение:

$$\text{CCV}(1) = \Pi(1) = \text{LOO}$$

$$\text{CCV}(2) = \Pi(1) \frac{\ell}{\ell+1} + \Pi(2) \frac{1}{\ell+1}$$

$$\text{CCV}(3) = \Pi(1) \frac{\ell}{\ell+2} + \Pi(2) \frac{2\ell}{(\ell+1)(\ell+2)} + \Pi(3) \frac{2}{(\ell+1)(\ell+2)}$$

При достаточно гладком распределении $\Pi(m)$ CCV слабо зависит от длины контроля.

Задача 3: получить скорость спада множителя при $\Pi(m)$ в выражении для CCV.

Решение:

$$R(m) = \frac{C_{L-1-m}^{\ell-1}}{C_{L-1}^{\ell}};$$

$$\frac{R(m+1)}{R(m)} = 1 - \frac{\ell-1}{L-1-m} < \frac{k}{L-1}.$$

То есть $R(m)$ стремится к нулю быстрее геометрической прогрессии.

4.7.6. Минимизация CCV

Из рассмотренных графиков и приведенных задач получили, что CCV тем меньше, чем чаще близкие объекты лежат в одном классе. Значит, минимизируя CCV, можно добиться лучшего результата работы алгоритма k ближайших соседей. CCV почти не зависит от длины контроля для kNN, поэтому достаточно использовать CCV(1) равный LOO.

Приближенные методы поиска ближайших соседей (ANN)

Иногда методы точного поиска ближайших соседей могут быть достаточно затратными и вычислительно сложными. Но зачастую реальные задачи не требуют вычислять именно самых близких соседей, и достаточно найти всего несколько наиболее близких. Для этой цели нам подойдут приближенные методы поиска, рассмотренные ниже.

4.7.7. Random projection trees - Annoy

Ряд алгоритмов приближенного поиска основан на деревьях, которые часто применяются для поиска соседей. Одним из наиболее известных и зарекомендовавших

себя методов из данного семейства является алгоритм **Annoy**. Опишем принцип его работы.

Пусть у нас есть обучающая выборка. Наша цель — построить структуру данных, которая позволит нам находить ближайшие точки к любой точке запроса. Для начала выберем из нее два объекта случайным образом. Между ними симметрично проводится разделяющая гиперплоскость. Далее в каждом из полученных полу-пространств снова выбирается два случайных элемента из набора данных, и уже между ними проводятся разделяющие гиперплоскости.

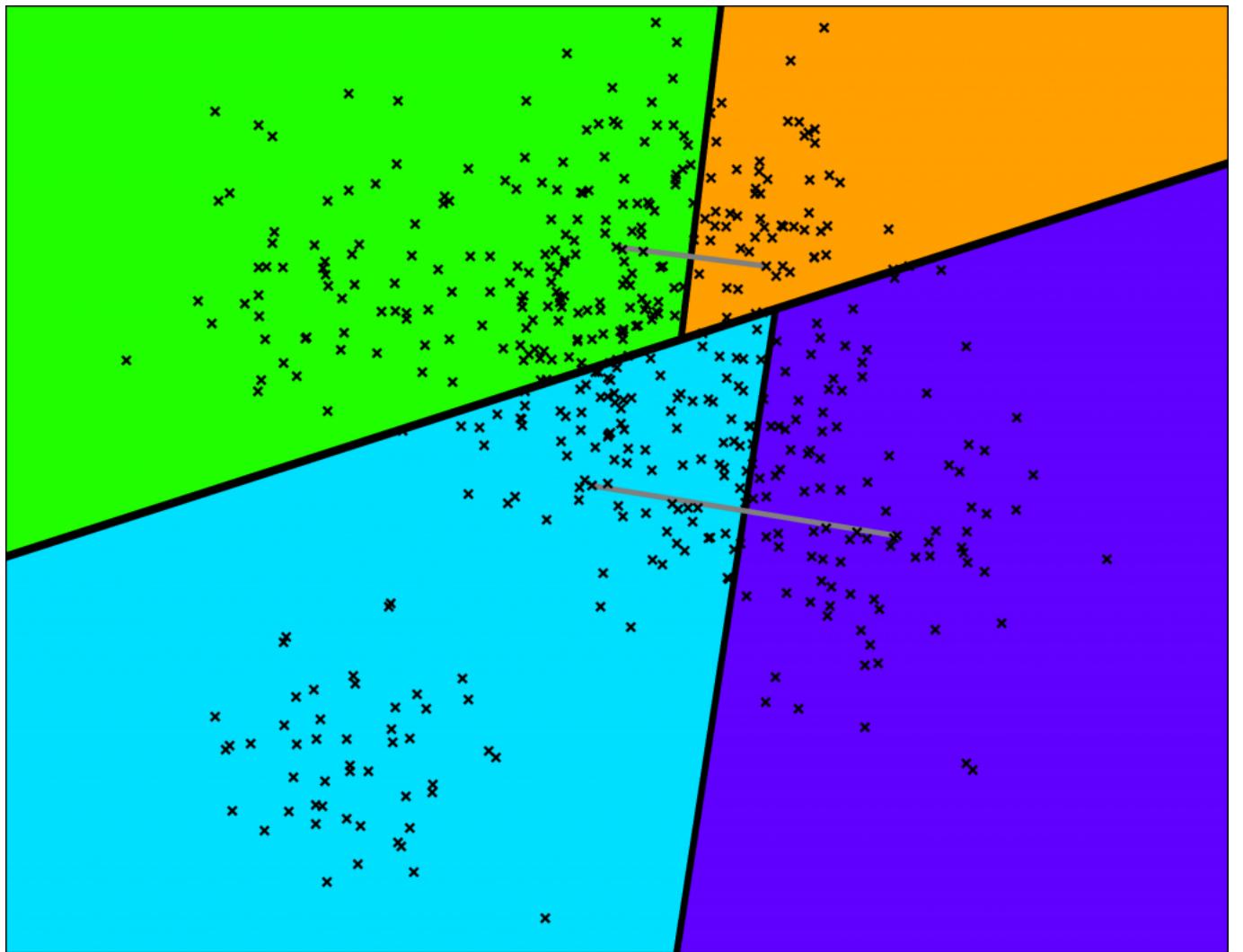


Рис. 4.3. Разделение пространства гиперплоскостями в алгоритме Annoy.

Данная процедура продолжается итеративно, пока в каждой области останется не более K объектов. Здесь K является подбираемым гиперпараметром. На самом деле, данная процедура чем-то похожа на то, как работают k-d-деревья.

В результате работы вышеописанного алгоритма мы получим бинарное дерево (рис. 3) (глубина порядка $O(\log N)$), спускаясь по которому, найдем область с

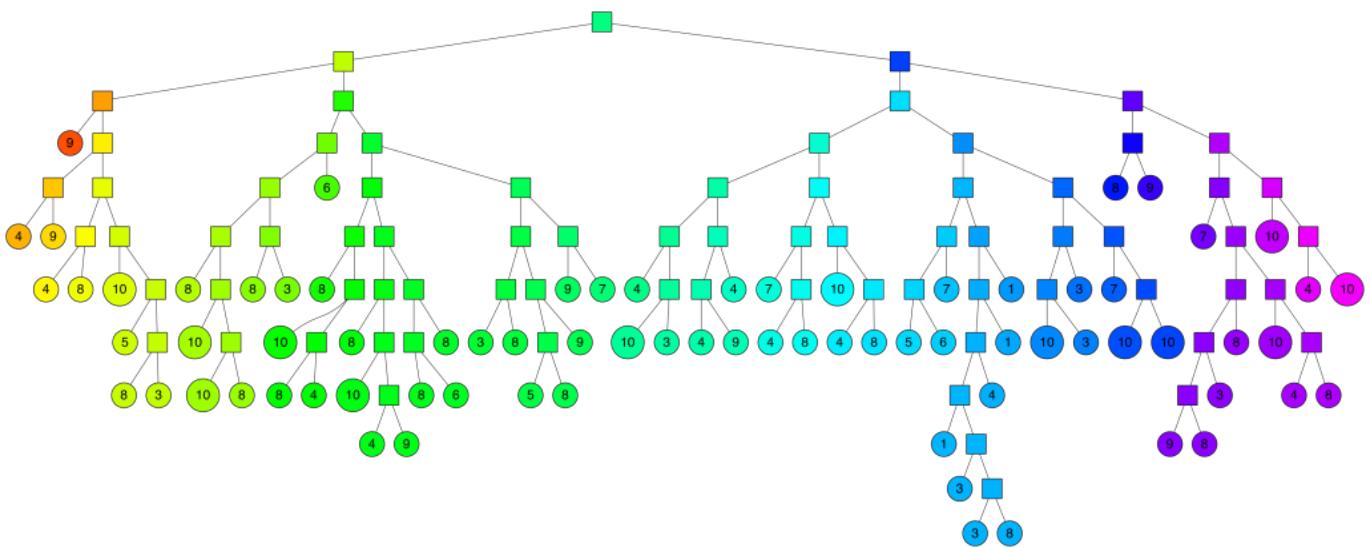


Рис. 4.4. В результате работы алгоритма получаем бинарное дерево.

целевым объектом и некоторым количеством близких к нему элементов. Давайте попробуем найти точку, обозначенную красным крестом на рис. 4.

То, как будет выглядеть путь вниз по бинарному дереву в данном случае, показано на рис. 5.

Таким образом, мы получаем 7 ближайших соседей. Уже хороший результат, но достаточно ли нам этого?

Задача 1

Мы проделали весь алгоритм, описанный выше, и получили для некоторой точки 7 ближайших соседей. На самом деле, нам этого недостаточно, попробуйте подумать, почему.

Решение. Во-первых, может произойти ситуация, что нам нужно не 7 ближайших соседей, а больше. Например, методы приближенного поиска соседей используются для подбора рекомендаций фильмов, и в такой задаче нам необходимо найти около 10-15 наиболее похожих картин для пользователя. Во-вторых, некоторые из ближайших соседей на самом деле могут не попасть в итоговую область пространства и остаться за его пределами.

Необходимо увеличить точность алгоритма при помощи составления леса из таких деревьев. Мы можем выполнить поиск по всем деревьям одновременно, и взять объединение соответствующих целевому объекту областей. Именно так работает алгоритм Annoy.

Задача 2

Давайте подумаем, почему же методы приближенного поиска (ANN - approximate nearest neighbor) дают выигрыш по времени и являются менее ресурсозатратными по сравнению с точными методами?

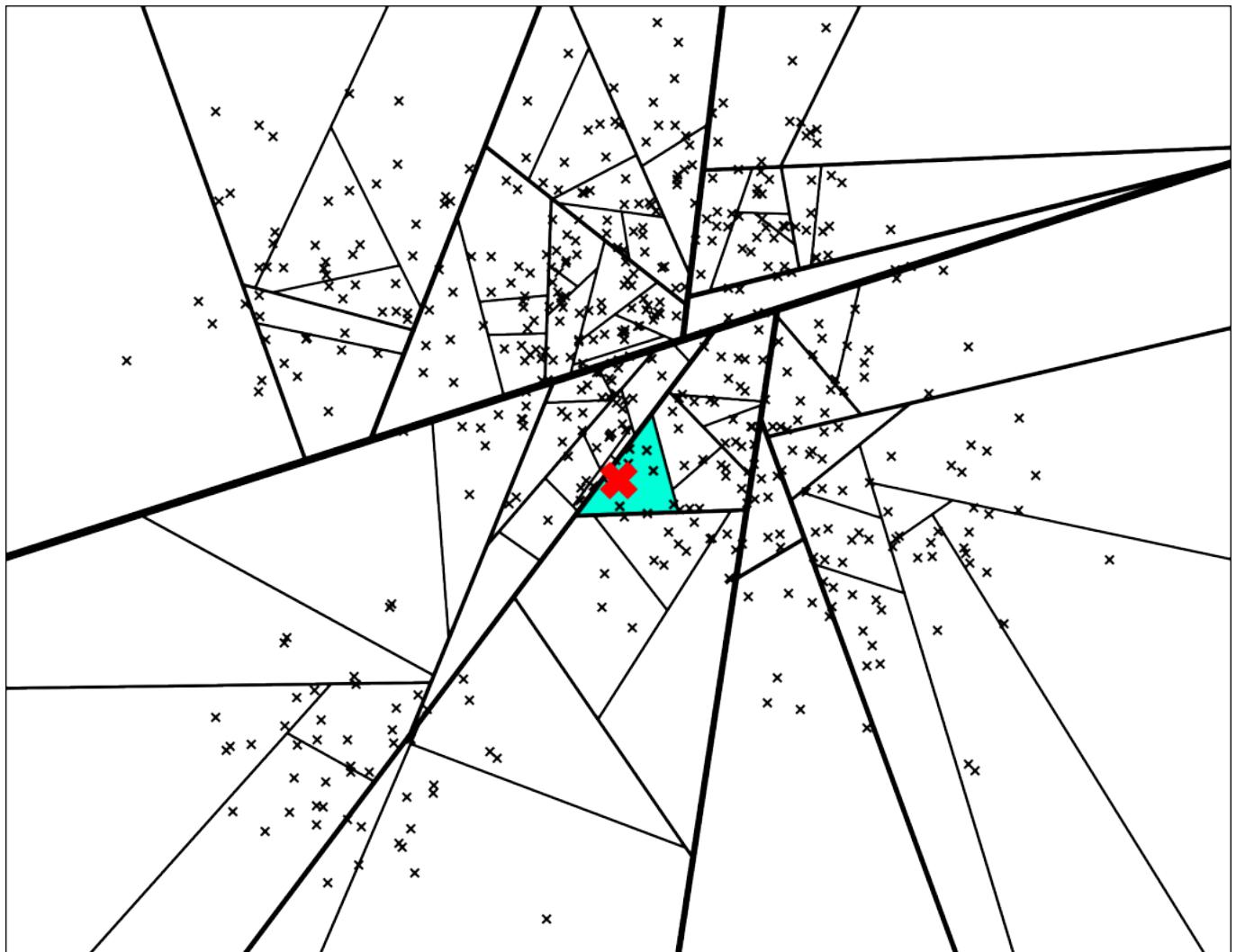


Рис. 4.5. Поиск для точки, обозначенной красным крестом.

Решение. Если объем анализируемых данных становится слишком большим, что нередко встречается в реальной жизни, то точные методы поиска ближайших соседей будут просматривать всю выборку данных, что займет огромное количество времени. А алгоритмы ANN не просматривают все объекты из датасета, следовательно, являются более быстрыми и эффективными. ANN — это алгоритм, который находит точку данных в наборе данных, которая очень близка к заданной точке запроса, но не обязательно является абсолютно ближайшей. Алгоритм точного поиска выполняет исчерпывающий поиск по всем данным, чтобы найти идеальное совпадение, тогда как алгоритм ANN остановится на совпадении, которое достаточно близко.

Задача 3

Оцените время построения модели Annoy для базы данных из 500 объектов.

Решение. Время поиска ближайших соседей для одного запроса в модели Annoy — $O(\log N)$, где N — количество объектов в базе данных. А время построения

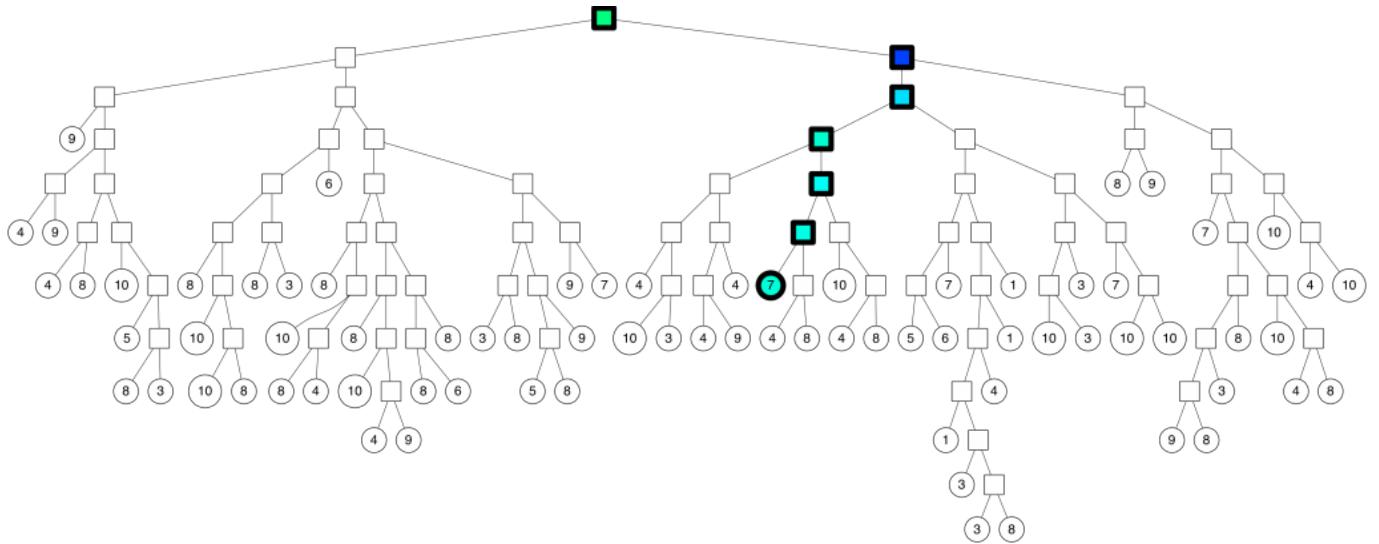


Рис. 4.6. Поиск точки.

модели Annoy для базы данных из 500 объектов составляет $O(N \cdot \log N)$. Для $N = 500$ будет :

$$T = O(500 \cdot \log 500)$$

$$\log 500 \approx 2.7$$

$$T = O(500 \cdot 2.7) = O(1350)$$

То есть время, необходимое для построения модели Annoy для 500 объектов, будет порядка 1350 операций.

4.8. Отбор эталонных объектов

Обучающие объекты делятся на три категории: эталоны (типичные представители классов), неинформативные (окружены объектами того же класса) и шумовые выбросы (расположены среди объектов чужого класса). Исключение шумовых и неинформативных объектов позволяет улучшить классификацию, сократить объём данных и ускорить поиск ближайших эталонов.

Алгоритм STOLP реализует эту идею, основываясь на весовой функции $w(i, u)$. Он строит метрический алгоритм $a(u; \Omega)$, где $\Omega \subset X_\ell$ — множество эталонов.

Отступ объекта x_i относительно алгоритма $a(x_i; \Omega)$, обозначенный как $M(x_i, \Omega)$, используется для классификации объектов:

- $M(x_i, \Omega) < 0$ — объект окружён чужими классами и считается выбросом;

- $M(x_i, \Omega) > 0$ — объект окружён своими классами и является либо эталоном, либо неинформативным.

4.9. Компактность и профиль компактности

Определение 1. Профиль компактности относительно множества эталонов $\Omega \subseteq X^L$ определяется как:

$$\Pi(m, \Omega) = \frac{1}{L} \sum_{i=1}^L \left[y_i^{(m|\Omega)} \neq y_i \right],$$

где $x_i^{(m|\Omega)}$ — m -й сосед объекта x_i из множества Ω , а y_i — истинная метка класса объекта x_i .

Теорема 1. Компактность множества эталонов Ω вычисляется как:

$$CCV(\Omega) = \frac{1}{L} \sum_{i=1}^L \sum_{m=1}^k \left[y_i^{(m|\Omega)} \neq y_i \right] \cdot \frac{C_{L-1}^{L-1-m}}{C_L^{L-1}},$$

где $T(x_i, \Omega)$ — вклад объекта x_i в CCV .

Жадный отбор эталонов по критерию $CCV(\Omega) \rightarrow \min$

Жадная стратегия удаления не-эталонов:

1. Инициализация: $\Omega := X^L$.
 2. Повторять:
 - a. Найти $x \in \Omega$, при котором $CCV(\Omega \setminus \{x\}) \rightarrow \min$.
 - b. Удалить x : $\Omega := \Omega \setminus \{x\}$.
 - c. Обновить $T(x_i, \Omega)$ для всех x_i , где $x \in kNN(x_i)$.
- пока CCV уменьшается или практически не увеличивается.

Жадная стратегия добавления эталонов:

1. Инициализация: $\Omega := \{\text{по одному объекту от каждого класса}\}$.
 2. Повторять:
 - a. Найти $x \in X^L \setminus \Omega$, при котором $CCV(\Omega \cup \{x\}) \rightarrow \min$.
 - b. Добавить x : $\Omega := \Omega \cup \{x\}$.
 - c. Обновить $T(x_i, \Omega)$ для всех x_i , где $x \in kNN(x_i)$.
- пока CCV уменьшается.

Алгоритм: Отбор эталонных объектов STOLP**Вход:**

- X_ℓ — обучающая выборка;
- δ — порог фильтрации выбросов;
- ℓ_0 — допустимая доля ошибок.

Выход:

- Множество опорных объектов $\Omega \subset X_\ell$.

Алгоритм:

1. Для всех $x_i \in X_\ell$ проверить, является ли x_i выбросом:

$$\text{если } M(x_i, X_\ell) < \delta, \text{ то } X_{\ell-1} := X_\ell \setminus \{x_i\}; \quad \ell := \ell - 1;$$

2. Инициализация: взять по одному эталону от каждого класса:

$$\Omega := \arg \max_{x_i \in X_\ell, y \in Y} M(x_i, X_\ell);$$

3. Пока $\Omega \neq X_\ell$:

- Выделить множество объектов, на которых алгоритм $a(u; \Omega)$ ошибается:

$$E := \{x_i \in X_\ell \setminus \Omega : M(x_i, \Omega) < 0\};$$

- Если $|E| < \ell_0$, то выход.
- Присоединить к Ω объект с наименьшим отступом:

$$x_i := \arg \min_{x \in E} M(x, \Omega); \quad \Omega := \Omega \cup \{x_i\};$$

Результаты работы алгоритма

Алгоритм делит обучающие объекты на три категории:

- Шумовые выбросы, которые удаляются.
- Этапонные объекты, которые формируют подмножество Ω .
- Неинформативные объекты, которые также удаляются.

Если гипотеза компактности верна, то большая часть обучающих объектов окажется неинформативной и будет отброшена, обеспечивая сжатие данных.

Оценка эффективности алгоритма STOLP

Алгоритм STOLP имеет относительно низкую эффективность: добавление каждого эталона требует перебора объектов $X_\ell \setminus \Omega$ и вычисления отступов относительно Ω , что приводит к сложности $O(|\Omega|^2 \ell)$. Для ускорения можно добавлять несколько эталонов одновременно, выбирая их на большом расстоянии друг от друга, чтобы минимизировать влияние на отступы.

На этапе отсея выбросов допустимо вычислить отступы один раз и отбросить объекты с $M(x_i, \Omega) < \delta$. Эффективная реализация включает процедуру обновления отступов $M_i = M(x_i, \Omega)$, что позволяет гибко управлять вычислениями.

Задачи

Задача 1 Докажите, что при использовании алгоритма STOLP отступ объекта $M(x_i, \Omega)$ не зависит от порядка перебора эталонных объектов в Ω .

Решение. Рассмотрим отступ объекта x_i относительно множества эталонов Ω :

$$M(x_i, \Omega) = \sum_{x_j \in \Omega} y_j w(\rho(x_i, x_j)).$$

Порядок перебора объектов $x_j \in \Omega$ не влияет на результат вычисления, так как операция суммирования коммутативна:

$$\sum_{x_j \in \Omega} y_j w(\rho(x_i, x_j)) = \sum_{x_j \in \text{перестановке } \Omega} y_j w(\rho(x_i, x_j)).$$

Таким образом, $M(x_i, \Omega)$ остается неизменным независимо от порядка объектов в Ω . Это доказывает инвариантность отступа относительно перестановки эталонов.

Задача 2 В выборке X_ℓ используется алгоритм STOLP для классификации объектов по двум классам. Выбросы имеют отступы $M(x_i, \Omega) < \delta$, где $\delta = 0$. Рассмотрим набор из трёх объектов:

- $x_1 : M(x_1, \Omega) = -0.5$,
- $x_2 : M(x_2, \Omega) = 0.8$,
- $x_3 : M(x_3, \Omega) = -0.2$.

Определите, какие объекты будут исключены из обучающей выборки X_ℓ на первом этапе алгоритма STOLP.

Решение. На этапе отсея выбросов STOLP удаляет объекты, для которых отступ $M(x_i, \Omega) < \delta$. Учитывая $\delta = 0$, удаляются объекты с отрицательными отступами.

Проверяем отступы:

- $M(x_1, \Omega) = -0.5 < 0$: x_1 будет удалён,
- $M(x_2, \Omega) = 0.8 > 0$: x_2 останется в выборке,
- $M(x_3, \Omega) = -0.2 < 0$: x_3 будет удалён.

Ответ: из обучающей выборки будут удалены объекты x_1 и x_3 .

Задача 3 Докажите, что алгоритм STOLP допускает использование любых метрических функций $\rho(x, x')$, если они удовлетворяют свойствам метрики.

Решение. Метрическая функция $\rho(x, x')$ используется для вычисления отступов $M(x_i, \Omega)$ и определяется через весовую функцию $w(\rho)$. Для корректной работы алгоритма требуется, чтобы $\rho(x, x')$ удовлетворяла следующим свойствам метрики:

1. **Неотрицательность:** $\rho(x, x') \geq 0$,
2. **Равенство нулю только при совпадении точек:** $\rho(x, x') = 0 \iff x = x'$,
3. **Симметричность:** $\rho(x, x') = \rho(x', x)$,
4. **Неравенство треугольника:** $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.

Эти свойства гарантируют, что отступы $M(x_i, \Omega)$ корректно отражают относительное положение объекта x_i относительно эталонов.

Кроме того, функция $w(\rho)$ должна быть убывающей, чтобы больший вклад в отступ вносили ближайшие эталоны, что не зависит от конкретной метрики, а лишь от её свойств.

Вывод: любой выбор функции $\rho(x, x')$, удовлетворяющей свойствам метрики, допустим в алгоритме STOLP.

4.10. Метод окна Парзена.

Напомним идею метрического классификатора. Будем обозначать $x = (x^1, \dots, x^n)$ – вектор признаков объекта x , $x_i = (x_i^1, \dots, x_i^n)$ – вектор признаков объекта x_i . Пусть на пространстве признаков задана метрика ρ . Для произвольного объекта x отранжируем объекты обучающей выборки x_1, x_2, \dots, x_l :

$$\rho(x, x^{(1)}) \leq \rho(x, x^{(2)}) \leq \dots \leq \rho(x, x^{(l)}).$$

Таким образом, $x^{(i)}$ – i -й ближайший сосед объекта x среди обучающей выборки, обозначим через $y^{(i)}$ ответ на нём.

Метрический классификатор предлагает следующую модель зависимости:

$$a(x; X^l) = \arg \max_{y \in Y} \sum_{i=1}^l [y^{(i)} = y] w(i, x),$$

где $w(i, x)$ - некоторый вес, отражающий степень близости к объекту x его i -го соседа. Вес неотрицателен и не возрастает по i .

Кроме того, введём обозначение:

$$\Gamma_y(x) = \sum_{i=1}^l [y^{(i)} = y] w(i, x) - \text{оценка близости объекта } x \text{ к классу } y.$$

Напомним также, что метод k ближайших соседей заключается в выборе в качестве весовой функции $w(i, x) = [i \leq k]$. Среди недостатков этого метода выделим следующие:

- в силу того, что функция близости дискретнозначная (принимает не более k значений), часто попадаем в ситуацию неоднозначности классификации, когда $\Gamma_y(x) = \Gamma_z(x)$, $y \neq z$;
- метод не учитывает значение расстояний от объекта до ближайших соседей. Естественным кажется использование меньшего веса для далёкого объекта, даже если он попадает в k ближайших.

Для борьбы с этими недостатков модифицируем веса следующим образом:

$$w(i, x) = [i \leq k] w_i, \text{ где } w_i \text{ зависит только от номера соседа.}$$

Тем самым, получим метод k взвешенных ближайших соседей. В качестве w_i можно брать, например, линейно убывающие веса $w_i = \frac{k+1-i}{k}$ или экспоненциально убывающие веса $w_i = q^i$, $0 < q < 1$.

Однако линейно убывающие веса всё ещё допускают неоднозначность классификации. Кроме того, метод по-прежнему не учитывает расстояния между объектами.

Наконец, положим

$$w(i, x) = K \left(\frac{\rho(x, x^{(i)})}{h} \right),$$

где $K(r)$ - невозрастающая функция, определённая на неотрицательных числах, называемая ядром, положительная на отрезке $[0, 1]$ и равная нулю вне его, h - ширина окна.

Таким образом, получим метод окна Парзена фиксированной ширины:

$$a(x; X^l, h, K) = \arg \max_{y \in Y} \sum_{i=1}^l [y^{(i)} = y] K\left(\frac{\rho(x, x^{(i)})}{h}\right).$$

Мы видим, что такой метод для каждого объекта x рассматривает только объекты обучающей выборки, находящиеся на расстоянии не больше h от x . Причём, чем дальше объект от x , тем меньший вклад он даёт в оценку близости.

Замечание. Вообще говоря, в качестве ядра можно брать функции, которые принимают ненулевые значения вне отрезка $[0, 1]$, однако тогда они подбираются быстро убывающими (см. далее гауссовское ядро).

Заметим, что если расстояние от объекта x до всех объектов обучающей выборки больше h , то построенный метод не может классифицировать x , так как в

таком случае $\sum_{i=1}^l [y^{(i)} = y] K\left(\frac{\rho(x, x^{(i)})}{h}\right) \equiv 0$. Это соображение наталкивает на использование метода окна Парзена переменной ширины:

$$a(x; X^l, h, K) = \arg \max_{y \in Y} \sum_{i=1}^l [y^{(i)} = y] K\left(\frac{\rho(x, x^{(i)})}{\rho(x, x^{(k+1)})}\right).$$

Замечание. В знаменателе в ядре стоит расстояние до $k + 1$ -го соседа, так как на практике зачастую берутся ядра, которые в точке 1 равны 0. Таким образом, метод окна Парзена переменной ширины учитывает именно k ближайших соседей объекта.

Для подбора наилучшей модели оптимизируются параметры:

- ширина окна h или количество сосеседей k ;
- ядро K .

Выше мы считали, что ядро определено на неотрицательной полуоси. Эквивалентно, можно считать, что ядро определено на всей действительной оси, является чётной функцией, невозрастающей на отрезке $[0, 1]$ (и, как правило, равна 0 вне отрезка $[-1, 1]$). В таком случае расстояние между объектами можно понимать как ориентированное. Однако, нетрудно видеть, что ориентация ни на что не влияет. Приведём примеры наиболее часто используемых ядер:

- $K_1(r) = \frac{3}{4}(1 - r^2)[r \leq 1]$ - ядро Епанечникова;
- $K_2(r) = \frac{15}{16}(1 - r^2)^2[r \leq 1]$ - квартическое ядро;

- $K_3(r) = (1 - |r|)[r \leq 1]$ - треугольное ядро;
- $K_4(r) = \frac{1}{2}[r \leq 1]$ - прямоугольное ядро;
- $K_5(r) = \frac{1}{\sqrt{2\pi}}e^{-r^2/2}$ - гауссовское ядро.

Числовые коэффициенты выбираются из соображений нормировки: $\int_{-\infty}^{+\infty} K_i(r)dr = 1$.

Задача 1. Докажите, что метод окна Парзена фиксированной ширины можно переписать следующим образом:

$$a(x; X^l, h, K) = \arg \max_{y \in Y} \sum_{i=1}^l [y_i = y] K\left(\frac{\rho(x, x_i)}{h}\right),$$

т.е. объекты обучающей выборки не обязательно ранжировать по расстоянию до объекта.

Доказательство. Заметим, что при любом фиксированном $y \in Y$ справедливо равенство $\sum_{i=1}^l [y_i = y] K\left(\frac{\rho(x, x_i)}{h}\right) = \sum_{i=1}^l [y^{(i)} = y] K\left(\frac{\rho(x, x^{(i)})}{h}\right)$, поскольку каждый объект обучающей выборки в обеих суммах участвует лишь единожды и значение соответствующего слагаемого не зависит от номера объекта. Отметим, что метод окна Парзена переменной ширины также можно переписать аналогичным образом.

Задача 2. Вася решает методом окна Парзена фиксированной ширины задачу классификации объектов на 8 классов по 3 вещественным признакам. Все признаки объектов в обучающей выборке принимают значения, по модулю не меньшие 1, и объекты i -го класса попадают в i -й октант пространства признаков. Вася использует евклидову метрику на пространстве признаков и берёт ширину окна $h = 1$. К какому из классов может его классификатор отнести точку 0?

Ответ. Ни к какому.

Решение. По условию расстояние между 0 и объектом обучающей выборки $\rho(x_i, 0) \geq \sqrt{(1-0)^2 + (1-0)^2 + (1-0)^2} = \sqrt{3} > 1 = h$. Таким образом, ни одна точка обучающей выборки не попадает в нужное окно с центром в нуле.

Задача 3. Докажите, что в качестве ядер в методе окна Парзена фиксированной ширины также можно брать выпуклую комбинацию ядер K_1, K_2, K_3, K_4 из примеров выше.

Доказательство. Заметим, что выпуклая комбинация также будет положительна при $|r| \leq 1$ и нулевая вне этого отрезка, чётна и не убывает на отрезке $[0, 1]$. Более того, также сохраняется нормировка:

$$\int_{-1}^1 [\alpha_1 K_1(r) + \alpha_2 K_2(r) + \alpha_3 K_3(r) + \alpha_4 K_4(r)] dr = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1.$$

4.11. Метод потенциальных функций

Метод потенциальных функций - метрический классификатор, частный случай метода ближайших соседей. Позволяет с помощью простого алгоритма оценивать вес («важность») объектов обучающей выборки при решении задачи классификации.

В общем виде, алгоритм ближайших соседей есть:

$$a(x; X^l) = \arg \max_{y \in Y} \sum_{i=1}^l [y^{(i)} = y] w(i, x),$$

где $w(i, x)$ — вес, степень близости к объекту x его i -го соседа.

Метод потенциальных функций заключается в выборе в качестве веса $w(i, x)$, функции следующего вида:

$$w(i, x) = \gamma^{(i)} K\left(\frac{\rho(x, x^{(i)})}{h^{(i)}}\right),$$

где:

- $K(r)$ – ядро, не возрастает и положительно на $[0, 1]$,
- $\gamma^{(i)} \geq 0$ – «заряд» объекта $x^{(i)}$,
- $h^{(i)} > 0$ – параметр, задающий «ширину потенциала» объекта $x^{(i)}$.

Основная идея метода была навеяна электростатическим взаимодействием элементарных частиц. Известно, что потенциал («мера воздействия») электрического поля элементарной заряженной частицы в некоторой точке пространства пропорционален отношению заряда частицы (Q) к расстоянию до частицы (r): $\varphi(r) \sim \frac{Q}{r}$. Из-за этого в качестве $K(r)$ часто выступают функции: $\frac{1}{r}$ или $\frac{1}{r+a}$.

Метод потенциальных функций реализует полную аналогию указанного выше примера. При классификации объект проверяется на близость к объектам из обучающей выборки. Считается, что объекты из обучающей выборки «заряжены»

своим классом, а мера «важности» каждого из них при классификации зависит от его «заряда» и расстояния до классифицируемого объекта.

4.11.1. Подбор параметров

Отметим, что параметрическую модель зависимости метода потенциальных функций можно существенно упростить (см. задачу ниже):

$$a(x; X^l) = \arg \max_{y \in Y} \sum_{i=1}^l [y_i = y] \gamma_i K \left(\frac{\rho(x, x_i)}{h_i} \right).$$

Из выражения выше следует, что в методе потенциальных функций используются две группы параметров: $\{h_i\}$ и $\{\gamma_i\}$.

«Ширина окна потенциала» h_i выбирается для каждого объекта из эмпирических соображений. «Заряд» γ_i объектов выборки можно подобрать, исходя из информации, содержащейся в выборке. Ниже приведен алгоритм, который позволяет «обучать» параметры $\gamma_1, \dots, \gamma_n$, то есть подбирать их значения по обучающей выборке X^l .

Алгоритм подбора параметров $\{\gamma_i\}$

Вход: обучающая выборка из l пар «объект-ответ» – $X^l = ((x_1, y_1), \dots, (x_l, y_l))$.

Выход: значения параметров γ_i для всех $i = \overline{1, l}$.

Описание:

1. Инициализация: $\gamma_i := 0$ для всех $i = \overline{1, l}$;
2. Повторять пункты 3 – 4, пока эмпирический риск $Q(a, X^l) > \varepsilon$ (то есть пока процесс не стабилизируется):
3. Выбрать очередной объект x_i из выборки X^l ;
4. Если $a(x_i) \neq y_i$, то $\gamma_i := \gamma_i + 1$;
5. Вернуть значения γ_i для всех $i = \overline{1, l}$.

4.11.2. Преимущества и недостатки

Преимущества метода потенциальных функций:

- Метод прост для понимания и алгоритмической реализации;

- Порождает потоковый алгоритм;
- Хранит лишь часть выборки, следовательно, экономит память.

Недостатки метода:

- Порождаемый алгоритм медленно сходится;
- Параметры $\{\gamma_i\}$ и $\{h_i\}$ настраиваются слишком грубо;
- Значения параметров $\gamma_1, \dots, \gamma_l$ зависят от порядка выборки из выборки X^l .

4.11.3. Задачи для самопроверки

Задача 1.

Покажите, что параметрическую модель зависимости метода потенциальных функций можно переписать в следующем виде:

$$a(x; X^l) = \arg \max_{y \in Y} \sum_{i=1}^l [y_i = y] \gamma_i K \left(\frac{\rho(x, x_i)}{h_i} \right).$$

Решение.

Ответ следует из независимости суммы от перестановки слагаемых. $\forall y \in Y$:

$$\sum_{i=1}^l [y^{(i)} = y] \gamma^{(i)} K \left(\frac{\rho(x, x^{(i)})}{h^{(i)}} \right) = \sum_{i=1}^l [y_i = y] \gamma_i K \left(\frac{\rho(x, x_i)}{h_i} \right).$$

Задача 2.

Покажите, что метод потенциальных функций можно свести к линейному классификатору в случае $Y = \{-1, +1\}$ (бинарная классификация).

Решение.

Обозначим $\Gamma_y(x) = \sum_{i=1}^l [y^{(i)} = y] w(i, x)$ - оценка близости объекта x к классу y .

Тогда получим:

$$a(x; X^l) = \arg \max_{y \in Y} \Gamma_y(x) = \text{sign} (\Gamma_{+1}(x) - \Gamma_{-1}(x)) = \text{sign} \sum_{i=1}^l \gamma_i y_i K \left(\frac{\rho(x, x_i)}{h_i} \right),$$

что совпадает с линейной моделью классификации

$$a(x; X^l) = \operatorname{sign} \sum_{j=1}^n \gamma_j f_j(x),$$

где

- $f_j(x) = y_j K\left(\frac{\rho(x, x_j)}{h_j}\right)$ – новые признаки объекта x ,
- γ_j – веса линейного классификатора,
- $n = l$ – число признаков равно числу объектов обучения.

Задача 3.

Привести пример выборки $(x_1, y_1), \dots, (x_l, y_l)$, такой, что метод потенциальных функций построит модель с эмпирическим риском $Q(a, X^l) = \frac{1}{l} \sum_{i=1}^l [a(x_i) \neq y_i] = \frac{1}{l}$.

Решение.

Рассмотрим l -мерное пространство признаков $X = R^l$. В качестве выборки можно взять набор объектов, геометрически находящихся в вершинах l -симплекса. Тогда $\rho(x_i, x_j) = \operatorname{const}$ (для простоты положим $\rho(x_i, x_j) = 1$). Остается лишь определить множество ответов: $y_1 = +1, y_2 = +1, \dots, y_{l-1} = +1, y_l = -1$. Используя результат задачи 2 (положив $h_i = 1$), получим:

$$a(x; X^l) = \operatorname{sign} \left(\sum_{i=1}^{l-1} \gamma_i - \gamma_l \right).$$

Алгоритм подбора параметров $\{\gamma_i\}$ приведет к следующему набору (уже на второй итерации): $\gamma_1 = 1, \dots, \gamma_{l-1} = 0, \gamma_l = 1$. Т.е. $\forall x \in X^l$

$$a(x; X^l) = 1.$$

Воспользовавшись формулой для эмпирического риска получаем требуемое.

4.12. Полный скользящий контроль (CCV, Complete Cross-Validation)

Формула для частоты ошибок алгоритма, усреднённой по всем разбиениям выборки:

$$CCV(X^l) = \frac{1}{C_L^l} \sum_{X^l \sqcup X^k} \frac{1}{k} \sum_{x_i \in X^k} (a(x_i; X^l) \neq y_i),$$

где:

- C_L^l — число разбиений выборки X^L на обучающую и контрольную части,
- X^k, X^l — контрольная и обучающая подвыборки,
- $a(X^k; X^l)$ — алгоритм на обучающей подвыборке X^l ,

Интуиция Это метод, в котором мы оцениваем качество модели, проверяя её на всех возможных разбиениях данных. Пусть у нас есть выборка из n объектов. Мы выбираем фиксированный размер тестовой выборки k , а остальные ($n-k$) объектов идут в обучающую выборку. Суть CCV в том, чтобы перебрать все возможные комбинации такого разбиения, посчитать качество модели для каждого случая и усреднить результат. Число разбиений вычисляется с помощью биномиального коэффициента. Например, если $n = 5$, а $k = 2$, то $C_5^2 = \frac{5!}{2!(5-2)!} = 10$. Это значит, что модель будет протестирована на 10 разных разбиениях.

Очевидно, этот метод очень не эффективен с точки зрения вычислений особенно при больших k , однако при малых k можем заметить интересный факт:

Замечание: При $k = 1$:

$$CCV(X^l) = LOO(X),$$

где $LOO(X)$ — Leave-One-Out Cross-Validation.

Задача: 1. От чего зависит обобщающая способность CCV метода 1NN? 2. Возможно ли улучшить алгоритм 1NN, минимизируя CCV?

Решение задачи:

1.

- Компактность классов(если объекты одного класса расположены близко друг к другу и далеко от других классов, ошибки минимальны)
- Размер выборки(на малых выборках результаты CCV хуже из-за высокой дисперсии)

- Шумы в данных (наличие шумов увеличивает ошибку, так как ближайший сосед может быть некорректным)
- Метрики (выбор метрики сильно влияет на качество классификации)

2. Да, можно. Например, поможет:

- Удалить шумы
- Выбрать другую метрику
- Использовать взвешенный kNN(вместо одного ближайшего соседа можно рассмотреть несколько соседей с учётом их расстояний)
- Понизить размерность(PCA, t-SNE)

4.12.1. Понятие профиля компактности

Профиль компактности выборки X^L — это функция, показывающая долю объектов x_i , у которых m-й ближайший сосед $x_i^{(m)}$ принадлежит другому классу:

$$\Pi(m) = \frac{1}{L} \sum_{i=1}^L [y_i \neq y_i^{(m)}], \quad m = 1, \dots, L-1$$

где:

- $x_i^{(m)}$ — m-й сосед объекта x_i среди выборки X^L ,
- $y_i^{(m)}$ — метка класса m-го соседа объекта x_i .

Теорема (точное выражение CCV для метода 1NN)

$$CCV(X^L) = \sum_{m=1}^k \Pi(m) \frac{C_{L-1-m}^{l-1}}{C_{L-1}^l},$$

Доказательство(точное выражение CCV для метода 1NN)

$$\begin{aligned} CCV &= \frac{1}{C_L^l} \sum_{X^k \sqcup X^l} \frac{1}{k} \sum_{i=1}^L [x_i \in X^k] [a(x_i; X^l) \neq y_i] = \\ &= \sum_{X^k \sqcup X^l = X} \sum_{i=1}^L \sum_{m=1}^l \frac{[y_i^{(m)} \neq y_i]}{k C_L^k} [x_i^{(m)} \in X^l] [x_i; x_i^{(1)}, \dots, x_i^{(m-1)} \in X^k] = \end{aligned}$$

$$\begin{aligned}
&= \sum_{m=1}^k \sum_{i=1}^L \frac{[y_i^{(m)} \neq y_i]}{k C_L^l} \sum_{X^k \sqcup X^l} [x_i^{(m)} \in X^l] [x_i; x_i^{(1)}, \dots, x_i^{(m-1)} \in X^k] = \\
&= \sum_{m=1}^k \sum_{i=1}^L \frac{[y_i^{(m)} \neq y_i]}{k C_L^l} C_{L-1-m}^{l-1} = \sum_{m=1}^k \frac{1}{L} \sum_{i=1}^L [y_i^{(m)} \neq y_i] \frac{C_{L-1-m}^{l-1}}{C_{L-1}^l} \\
&= \sum_{m=1}^k \Pi(m) \frac{C_{L-1-m}^{l-1}}{C_{L-1}^l}
\end{aligned}$$

4.12.2. Свойства профиля компактности и CCV

Свойства профиля компактности и оценки CCV

Обозначим $R(m)$ как: $R(m) = C_{L-1-m}^{l-1} \overline{C_{L-1}^l \rightarrow 0}$ быстрее геометрической прогрессии

Формализации гипотезы компактности

$CCV = \sum_{m=1}^k \Pi(m) R(m)$ тем меньше чем чаще близкие объекты лежат в одном классе

Интуиция: Как мы заметили выше, $R(m)$ очень быстро убывает по m , то есть в нашем профиле компактности для оценки важны только $\Pi(m)$ при маленьких m .
При малых k :

$k = 1: CCV = \Pi(1) = LOO;$

$k = 2: CCV = \Pi(1) \frac{\ell}{\ell+1} + \Pi(2) \frac{1}{\ell+1}$

$k = 3: CCV = \Pi(1) \frac{\ell}{\ell+2} + \Pi(2) \frac{2\ell}{(\ell+1)(\ell+2)} + \Pi(3) \frac{2}{(\ell+2)(\ell+2)}$

Заметим, что при уже $k = 3$ большие Π входят с маленькими коэффициентами

4.12.3. Задача отбора эталонов $\Omega \subseteq X^L$ (prototype learning)

Задача заключается в построении классификатора $a(x; X^l \cap \Omega)$, который использует только объекты из Ω в качестве ближайших соседей. Требуется найти оптимальное подмножество эталонов Ω

Определение (профиль компактности относительно Ω)

$$\Pi(m, \Omega) = \frac{1}{L} \sum_{i=1}^L [y_i^{(m, \Omega)} \neq y_i], \quad m = 1, \dots, |\Omega|,$$

где $y_i^{(m, \Omega)}$ — m -й сосед объекта x_i из множества Ω

Теорема

Полный скользящий контроль для множества эталонов Ω имеет вид:

$$CCV(\Omega) = \frac{1}{L} \sum_{i=1}^L \sum_{m=1}^k [y_i^{(m,\Omega)} \neq y_i] \frac{C_{L-1-m}^{l-1}}{C_{L-1}^l}$$

, где вклад объекта x_i в ССВ обозначается как: $T(x_i, \Omega)$ = вклад объекта x_i в ССВ

4.12.4. Жадный отбор эталонов Ω по критерию $CCV(\Omega) \rightarrow \min$

Жадная стратегия удаления не-эталонов

Начальное множество эталонов: $\Omega := X^L$

повторять:

- **найти:** $x \in \Omega$, при котором $CCV(\Omega \setminus \{x\}) \rightarrow \min$;
- **обновить:** $\Omega := \Omega \setminus \{x\}$;
- **обновить** вклад объектов: $T(x_i, \Omega)$ для всех $x_i : x \in kNN(x_i)$

пока ССВ уменьшается или почти не увеличивается

Жадная стратегия добавления эталонов

Начальное множество эталонов: $\Omega := \{\text{по одному объекту от каждого класса}\}$

повторять:

- **найти** $x \in X^L \setminus \Omega$, при котором $CCV(\Omega \cup \{x\}) \rightarrow \min$;
- **обновить:** $\Omega := \Omega \cup \{x\}$;
- **обновить** вклад объектов: $T(x_i, \Omega)$ для всех $x_i : x \in kNN(x_i)$.

пока ССВ уменьшается.

4.12.5. Задачи

Вопрос 0: Объясните, как профиль компактности формализует гипотезу компактности?

Решение: Если классы компактны, то у каждого объекта большинство его соседей лежат в том же классе, значит, что $\Pi(m)$ близко к 0 при маленьких m , то есть нет ошибок при классификации по соседям

Вопрос 1:

Каково значение CCV при $k = 1$ и как оно связано с методом LOO? Какое у этого факта практическое значение?

Решение: при $k = 1$: $CCV(X^L) = LOO(X^L)$

Практическая значимость: CCV при $k = 1$ можно использовать как эталон для проверки качества других классификаций. Также есть CCV показывает высокое количество ошибок в этом случае, то можно понять, что данных много выбросов и шумов. К тому же данные легче интерпретировать

Вопрос 2 Как можно минимизировать CCV для метода 1NN, если в данных присутствуют шумы или высокое количество избыточных признаков? Какие методы могут улучшить качество алгоритма?

Решение:

- Удаление шумов
- Снижение размерности
- Изменение метрики
- Использование взвешенного k-NN

Вопрос 3 Опишите, как жадная стратегия отбора эталонов Ω по критерию $CCV(\Omega) \rightarrow \min$ работает для удаления не-эталонов. Почему эта стратегия уменьшает CCV?

Решение:

Начальное множество эталонов: $\Omega := X^L$

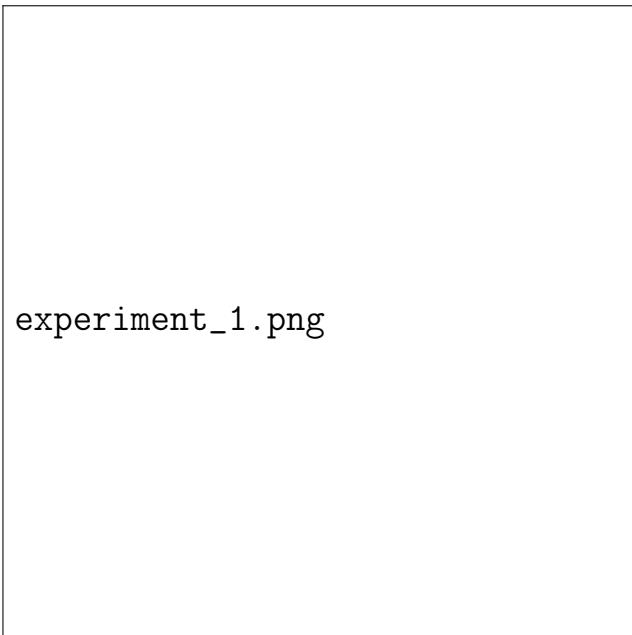
повторять:

- **найти:** $x \in \Omega$, при котором $CCV(\Omega \setminus \{x\}) \rightarrow \min$;
- **обновить:** $\Omega := \Omega \setminus \{x\}$;
- **обновить** вклад объектов: $T(x_i, \Omega)$ для всех $x_i : x \in kNN(x_i)$

Удаление объектов, которые не являются «эталонами», снижает вероятность ошибок классификации, так как остаются только те объекты, которые наиболее точно представляют классы. Это делает границы между классами более четкими и уменьшает вклад шумных или неинформативных объектов в CCV.

Вопрос 5: Мы изучили два жадных алгоритма отбора эталонов. Сравните их. Составьте эксперимент

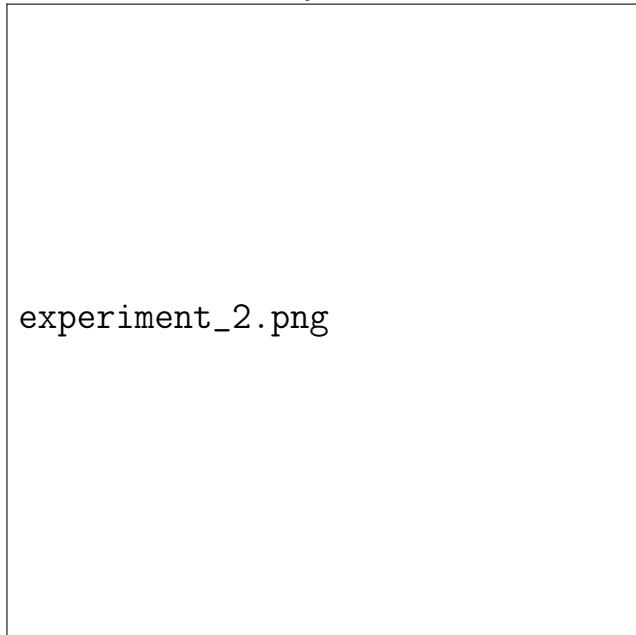
Решение:



experiment_1.png

Возьмем синтетические данные по 500 объектов двух классов(оба из двухмерной гауссовой плотности), добавим 30 шумовых объектов

Последовательно удаление:

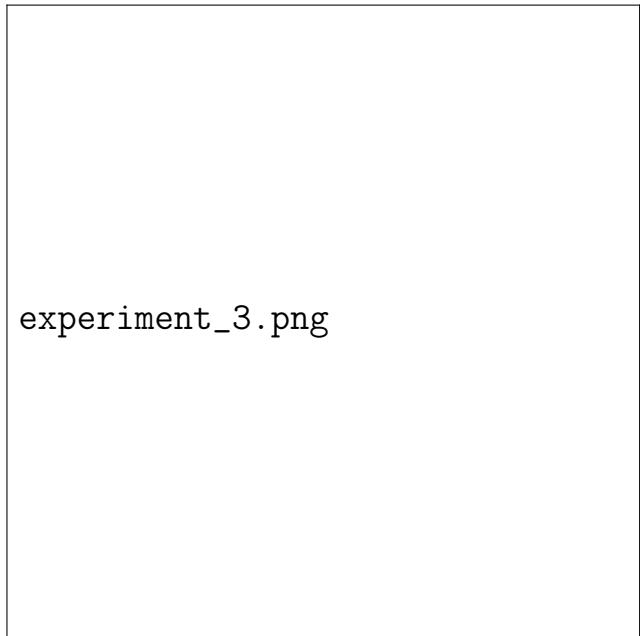


Возникают 3 типа объекта:

- шумовые выбросы(30 шумов)(будут выброшены в первую очередь, тк их валидация больше всего уменьшает ССВ)
- основная масса(окло 987)(бесполезны для классификации, так как других таких же соседей вокруг много)

– эталоны (стоят вдоль границы классов)

Теперь будем жадно добавлять объекты:



experiment_3.png

Тут же сначала возникают эталонные точки далеко от границы класса и классификация дальше будет опираться на эти точки

Глава 5

Метод опорных векторов

5.1. SVM-классификация

5.1.1. Постановка задачи

Метод опорных векторов (Support Vector Machine, SVM) решает задачу бинарной классификации, где требуется найти оптимальную гиперплоскость, разделяющую два класса в пространстве признаков. Оптимальность понимается как максимизация ширины разделяющей полосы между классами.

5.1.2. Математическая формализация

Пусть дана обучающая выборка $\{(x_i, y_i)\}_{i=1}^{\ell}$, где $x_i \in \mathbb{R}^n$ - векторы признаков, $y_i \in \{-1, +1\}$ - метки классов. Разделяющая гиперплоскость описывается уравнением:

$$\langle w, x \rangle - w_0 = 0,$$

где $w \in \mathbb{R}^n$ - вектор весов, $w_0 \in \mathbb{R}$ - порог.

5.1.3. Условия разделимости

Для корректной классификации должны выполняться условия:

$$\begin{cases} \langle w, x_i \rangle - w_0 \geq +1, & \text{если } y_i = +1 \\ \langle w, x_i \rangle - w_0 \leq -1, & \text{если } y_i = -1 \end{cases}$$

Эти условия можно объединить:

$$y_i(\langle w, x_i \rangle - w_0) \geq 1, \quad i = 1, \dots, \ell$$

5.1.4. Оптимизационная задача

Ширина разделяющей полосы равна $\frac{2}{\|w\|}$. Задача максимизации ширины эквивалентна задаче минимизации:

$$\frac{1}{2} \|w\|^2 \rightarrow \min_{w, w_0}$$

при ограничениях $y_i(\langle w, x_i \rangle - w_0) \geq 1$.

5.1.5. Двойственная задача

Применяя метод множителей Лагранжа, получаем двойственную задачу:

$$L(w, w_0, \lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{\ell} \lambda_i (y_i (\langle w, x_i \rangle - w_0) - 1)$$

Условия оптимальности:

$$\begin{cases} w = \sum_{i=1}^{\ell} \lambda_i y_i x_i \\ \sum_{i=1}^{\ell} \lambda_i y_i = 0 \end{cases}$$

5.1.6. Ядра

Для нелинейной классификации используется переход в пространство признаков большей размерности через отображение $\phi(x)$. Скалярное произведение заменяется на ядро:

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

Популярные ядра:

- Линейное: $K(x, z) = \langle x, z \rangle$
- Полиномиальное: $K(x, z) = (\langle x, z \rangle + 1)^d$
- RBF: $K(x, z) = \exp(-\gamma \|x - z\|^2)$

5.1.7. Дискриминантная функция в ядовом пространстве

После применения ядрового преобразования классификация новых точек осуществляется с помощью дискриминантной функции, которая принимает вид:

$$f(x) = \sum_{i=1}^{\ell} \lambda_i y_i K(x_i, x) - w_0$$

где:

- x_i - опорные векторы из обучающей выборки
- λ_i - множители Лагранжа (двойственные переменные)
- y_i - метки классов опорных векторов
- $K(x_i, x)$ - значение ядровой функции между опорным вектором и классифицируемой точкой

- w_0 – порог, определяющий сдвиг разделяющей гиперплоскости

Важно отметить, что в этой формуле суммирование происходит только по опорным векторам, так как для остальных точек обучающей выборки $\lambda_i = 0$. Это свойство обеспечивает эффективность вычислений при классификации новых точек.

Решающее правило для определения класса новой точки:

$$\text{class}(x) = \text{sign}(f(x)) = \begin{cases} +1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) < 0 \end{cases}$$

5.1.8. Мягкие границы

Для случая линейно неразделимой выборки вводятся переменные ослабления $\xi_i \geq 0$:

$$y_i(\langle w, x_i \rangle - w_0) \geq 1 - \xi_i$$

Целевая функция модифицируется:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\ell} \xi_i \rightarrow \min_{w, w_0, \xi}$$

где $C > 0$ – параметр регуляризации.

5.1.9. Задача 1

Условие:

Дан набор точек в двумерном пространстве с метками классов:

$$\begin{aligned} x_1 &= (0, 0), & y_1 &= -1 \\ x_2 &= (2, 0), & y_2 &= +1 \\ x_3 &= (0, 2), & y_3 &= +1 \\ x_4 &= (2, 2), & y_4 &= +1 \end{aligned}$$

В результате обучения SVM с линейным ядром получены следующие значения двойственных переменных:

$$\lambda_1 = 0.5, \quad \lambda_2 = 0, \quad \lambda_3 = 0.5, \quad \lambda_4 = 0$$

Найдите вектор весов w и определите, является ли точка $x_{\text{new}} = (1, 1)$ опорным вектором, если известно, что она лежит точно на разделяющей гиперплоскости.

Решение:

1. Найдем вектор весов w через опорные векторы:

$$w = \sum_{i=1}^4 \lambda_i y_i x_i$$

2. Подставляем известные значения:

$$\begin{aligned} w &= 0.5 \cdot (-1) \cdot (0, 0) + 0 \cdot (+1) \cdot (2, 0) + \\ &\quad + 0.5 \cdot (+1) \cdot (0, 2) + 0 \cdot (+1) \cdot (2, 2) \\ &= (0, 0) + (0, 0) + (0, 0.6) + (0, 0) \\ &= (0, 1) \end{aligned}$$

3. Для точек на разделяющей гиперплоскости выполняется:

$$\langle w, x \rangle - w_0 = 0$$

4. Подставляя координаты $x_{\text{new}} = (1, 1)$:

$$\langle (0, 1), (1, 1) \rangle - w_0 = 0$$

$$w_0 = 1$$

5. Чтобы точка была опорным вектором, она должна лежать на границе разделяющей полосы, то есть:

$$y_{\text{new}}(\langle w, x_{\text{new}} \rangle - w_0) = \pm 1$$

6. Проверяем это условие:

$$\langle (0, 1), (1, 1) \rangle - 1 = 0$$

Получаем 0, что не равно ± 1 .

Ответ: $w = (0, 1)$. Точка x_{new} не является опорным вектором.

5.1.10. Задача 2

Условие:

Дано множество точек в двумерном пространстве:

$$\begin{aligned} x_1 &= (1, 1), & y_1 &= +1 \\ x_2 &= (2, 2), & y_2 &= +1 \\ x_3 &= (0, 0), & y_3 &= -1 \\ x_4 &= (-1, 1), & y_4 &= -1 \end{aligned}$$

После обучения SVM получена разделяющая гиперплоскость $2/3 \cdot x_{i1} + 1/3 \cdot x_{i2} - 1 = 0$. Определите, какие из точек являются опорными векторами.

Решение:

Опорными векторами являются точки, лежащие на границе разделяющей полосы. Для их определения нужно:

1. Запишем вектор весов из уравнения гиперплоскости:

$$w = (2/3, 1/3) \quad w_0 = 1$$

2. Вычислим отступ для каждой точки по формуле:

$$M(x) = y(\langle w, x \rangle - w_0) = y(2x_{i1} + x_{i2} - 3)$$

3. Проверяем каждую точку:

$$\begin{aligned} M(x_1) &= (+1)(2/3 \cdot 1 + 1/3 \cdot 1 - 1) = 0 \\ M(x_2) &= (+1)(2/3 \cdot 2 + 1/3 \cdot 2 - 1) = 1 \\ M(x_3) &= (-1)(2/3 \cdot 0 + 1/3 \cdot 0 - 1) = 1 \\ M(x_4) &= (-1)(2/3 \cdot (-1) + 1/3 \cdot 1 - 1) = 4/3 \end{aligned}$$

4. Опорными векторами являются точки с отступом, равным единице, а также точки, лежащие на разделяющей гиперплоскости (отступ равен нулю).

Ответ: Точки x_2 и x_3 являются опорными векторами.

5.1.11. Задача 3

Условие:

При обучении SVM с полиномиальным ядром второй степени $K(x, z) = (\langle x, z \rangle + 1)^2$ получены следующие опорные векторы:

$$\begin{aligned} x_1 &= (2, 0), \quad \lambda_1 = 0.2, \quad y_1 = +1 \\ x_2 &= (0, 1), \quad \lambda_2 = 0.3, \quad y_2 = -1 \\ x_3 &= (1, 1), \quad \lambda_3 = 0.1, \quad y_3 = +1 \end{aligned}$$

К какому классу будет отнесена точка $x_{\text{new}} = (1, 0)$, если $w_0 = 0.5$?

Решение:

Для определения класса точки нужно найти знак дискриминантной функции:

$$f(x) = \sum_{i=1}^3 \lambda_i y_i K(x_i, x) - w_0$$

Вычислим значения ядра для x_{new} и каждого опорного вектора:

$$\begin{aligned} K(x_1, x_{\text{new}}) &= (2 \cdot 1 + 0 \cdot 0 + 1)^2 = 9 \\ K(x_2, x_{\text{new}}) &= (0 \cdot 1 + 1 \cdot 0 + 1)^2 = 1 \\ K(x_3, x_{\text{new}}) &= (1 \cdot 1 + 1 \cdot 0 + 1)^2 = 4 \end{aligned}$$

Подставляем в дискриминантную функцию:

$$\begin{aligned} f(x_{\text{new}}) &= 0.2 \cdot (+1) \cdot 9 + 0.3 \cdot (-1) \cdot 1 + 0.1 \cdot (+1) \cdot 4 - 0.5 \\ &= 1.8 - 0.3 + 0.4 - 0.5 = 1.4 \end{aligned}$$

Так как $f(x_{\text{new}}) > 0$, точка относится к классу +1.

Ответ: Точка x_{new} будет отнесена к классу +1.

5.2. SVM-регрессия

5.2.1. Постановка задачи

В задаче регрессии требуется найти функцию $f(x) = w^T \phi(x) + b$, которая аппроксимирует целевые значения y на основе входных данных x , минимизируя ошибки предсказания.

В SVM-регрессии вводится допустимая область погрешностей — ϵ -окрестность. Это означает, что отклонения $|f(x_i) - y_i|$ в пределах ϵ считаются несущественными, и модель игнорирует их. Цель — минимизировать сложность модели, связанную с $\|w\|$, штрафуя при этом за отклонения, выходящие за пределы ϵ .

5.2.2. Прямая задача

Функция потерь. Для задачи SVM-регрессии используется ϵ -чувствительная функция потерь:

$$L_\epsilon(f(x), y) = \begin{cases} 0, & \text{если } |f(x) - y| \leq \epsilon, \\ |f(x) - y| - \epsilon, & \text{иначе.} \end{cases}$$

Прямая постановка задачи:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L_\epsilon(f(x_i), y_i),$$

где:

- $\|w\|^2$ — регуляризационный член, минимизирующий сложность модели,
- C — коэффициент, регулирующий баланс между штрафами за ошибки и сложностью модели,
- $L_\epsilon(f(x_i), y_i)$ — штраф за выход за пределы ϵ -окрестности.

5.2.3. Преобразование задачи

Для учёта отклонений выше ϵ вводятся штрафные переменные ξ_i и ξ_i^* :

- ξ_i — превышение сверху ($y_i > f(x_i) + \epsilon$),
- ξ_i^* — превышение снизу ($y_i < f(x_i) - \epsilon$).

Задача минимизации принимает вид:

$$\min_{w, b, \xi, \xi^*} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*),$$

при ограничениях:

$$\begin{aligned} y_i - (w^T \phi(x_i) + b) &\leq \epsilon + \xi_i, \\ (w^T \phi(x_i) + b) - y_i &\leq \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* &\geq 0. \end{aligned}$$

5.2.4. Метод Лагранжа

Для решения задачи вводится лагранжиан, который включает:

- Целевую функцию,
- Ограничения через множители Лагранжа $(\alpha, \alpha^*, \eta, \eta^*)$.

Лагранжиан записывается как:

$$\begin{aligned} L(w, b, \xi, \xi^*, \alpha, \alpha^*, \eta, \eta^*) = & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) - \\ & - \sum_{i=1}^n \alpha_i [\epsilon + \xi_i - y_i + w^T \phi(x_i) + b] - \\ & - \sum_{i=1}^n \alpha_i^* [\epsilon + \xi_i^* + y_i - w^T \phi(x_i) - b] - \\ & - \sum_{i=1}^n (\eta_i \xi_i + \eta_i^* \xi_i^*). \end{aligned}$$

Для нахождения двойственной задачи необходимо минимизировать L по w, b, ξ, ξ^* и максимизировать по множителям Лагранжа.

5.2.5. Условия оптимальности

1. Производная по w :

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^n (\alpha_i - \alpha_i^*) \phi(x_i) = 0 \implies w = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \phi(x_i).$$

2. Производная по b :

$$\frac{\partial L}{\partial b} = \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0.$$

3. Производные по ξ_i и ξ_i^* :

$$\alpha_i + \eta_i = C, \quad \alpha_i^* + \eta_i^* = C, \quad 0 \leq \alpha_i, \alpha_i^* \leq C.$$

5.2.6. Двойственная задача

Подставляя условия оптимальности в лагранжиан, исключаем w, b, ξ_i, ξ_i^* . Получаем двойственную задачу:

$$\max_{\alpha, \alpha^*} -\frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) - \epsilon \sum_{i=1}^n (\alpha_i + \alpha_i^*) + \sum_{i=1}^n y_i (\alpha_i - \alpha_i^*),$$

где $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ — ядровая функция.

Ограничения:

$$\sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad 0 \leq \alpha_i, \alpha_i^* \leq C.$$

Для решения двойственной задачи используется метод квадратичного программирования.

5.2.7. Построение финальной модели

После решения двойственной задачи оптимальные α_i и α_i^* определяют параметры модели:

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) K(x_i, x) + b.$$

Смещение b вычисляется через опорные векторы — точки, где выполняется одно из условий:

$$y_i - (w^T \phi(x_i) + b) = \epsilon, \quad \text{или} \quad y_i - (w^T \phi(x_i) + b) = -\epsilon.$$

Опорные векторы ($\alpha_i > 0$ или $\alpha_i^* > 0$) определяют форму модели.

5.2.8. Выбор ядра

Выбор ядра играет ключевую роль в качестве работы модели SVM-регрессии. Различные ядра по-разному преобразуют входные данные, что может существенно повлиять на точность предсказаний и обобщающую способность модели.

Выбор ядра зависит от особенностей данных, структуры зависимости и доступных вычислительных ресурсов. Экспериментальная проверка нескольких типов ядер и последующая оценка метрик качества модели — это ключевой этап в процессе выбора оптимального ядра.

На рисунке ниже представлена SVM-регрессия с тремя типами ядер: RBF, линейным и полиномиальным.

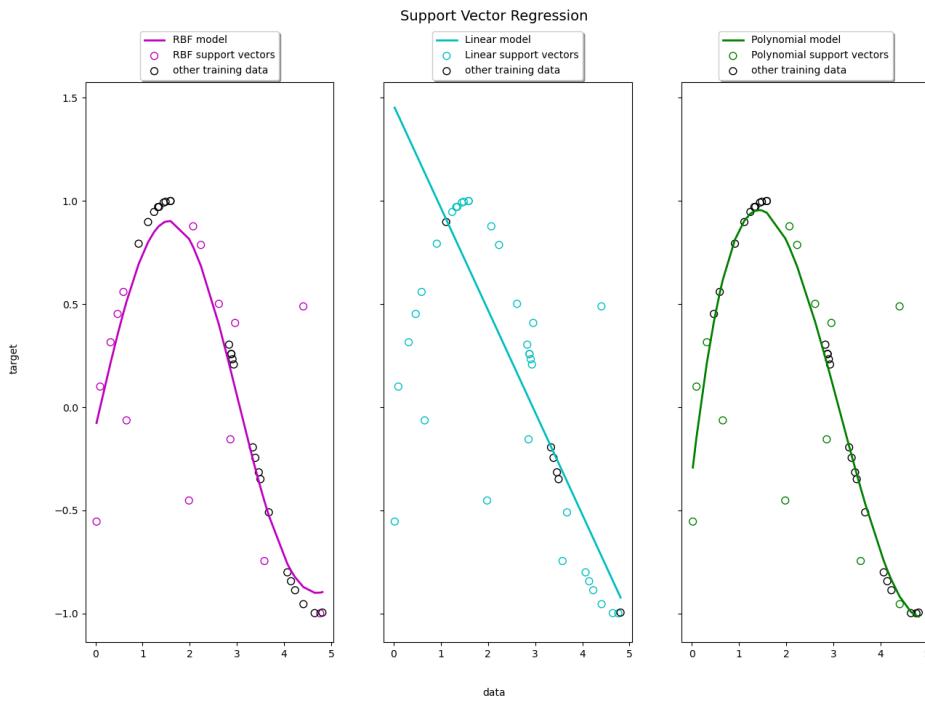


Рис. 5.1. Сравнение SVM-регрессия с разными типами ядер

5.2.9. Задача 1

Условие:

Рассмотрим следующий набор точек, лежащих на границе ϵ -окрестности для SVM-регрессии с линейным ядром:

$$\{(1, 2), (2, 3), (3, 5), (4, 6)\}$$

Постройте регрессионную модель и найдите смещение b , если $w = 2$, $\epsilon = 1$.

Решение:

Для нахождения смещения b необходимо использовать точки, которые лежат на границах ϵ -окрестности. Мы знаем, что для таких точек выполняется равенство:

$$y_i - (wx_i + b) = \epsilon \quad \text{или} \quad y_i - (wx_i + b) = -\epsilon$$

Найдем для каждой точки b , подставив их координаты в эти уравнения:

– Для точки $(1, 2)$:

$$2 - (2 \cdot 1 + b) = 1 \Rightarrow 2 - (2 + b) = 1 \Rightarrow -b = 1 \Rightarrow b = -1$$

– Для точки $(2, 3)$:

$$3 - (2 \cdot 2 + b) = 1 \Rightarrow 3 - (4 + b) = 1 \Rightarrow -b = 2 \Rightarrow b = -2$$

– Для точки $(3, 5)$:

$$5 - (2 \cdot 3 + b) = 1 \Rightarrow 5 - (6 + b) = 1 \Rightarrow -b = 2 \Rightarrow b = -2$$

– Для точки $(4, 6)$:

$$6 - (2 \cdot 4 + b) = 1 \Rightarrow 6 - (8 + b) = 1 \Rightarrow -b = 3 \Rightarrow b = -3$$

Для вычисления окончательного значения смещения b , усредняем найденные значения:

$$b_{\text{avg}} = \frac{-1 + (-2) + (-2) + (-3)}{4} = \frac{-8}{4} = -2$$

Таким образом, смещение $b = -2$.

Регрессионная модель для SVM с линейным ядром имеет следующий вид:

$$f(x) = wx + b$$

Подставляем данное в условии значения $w = 2$ и найденное значение $b = -2$:

$$f(x) = 2x - 2$$

Это и есть наша линейная регрессионная модель.

Ответ: $f(x) = 2x - 2$.

5.2.10. Задача 2

Условие:

У нас есть два набора данных для задачи регрессии:

- Набор 1: $\{(1, 2), (2, 3), (3, 4), (4, 5)\}$
- Набор 2: $\{(1, 1), (2, 4), (3, 9), (4, 16)\}$

Предположим, что мы используем SVM-регрессию с различными типами ядер (линейное, полиномиальное, RBF). Определите, какое ядро будет оптимальным для каждого набора данных.

Решение:

- Набор 1: данные имеют линейную зависимость, следовательно, линейное ядро будет лучшим выбором.
- Набор 2: данные имеют квадратичную зависимость, следовательно, оптимально будет использовать полиномиальное ядро второй степени.

Ответ: для первого набора данных оптимальным ядром будет линейное, для второго набора данных - полиномиальное второй степени.

5.2.11. Задача 3

Условие:

Дан набор данных для SVM-регрессии с линейным ядром:

$$\{(1, 2), (2, 2.8), (3, 5.2), (4, 8)\}.$$

Параметры модели: $w = 1.5$, $b = 0.5$, $\epsilon = 0.5$.

1. Определите, какие из точек набора данных находятся вне ϵ -окрестности (требуют штрафных переменных ξ или ξ^*).
2. Вычислите значения штрафных переменных для этих точек.

Решение:

1. Определение границ ϵ -окрестности: Уравнение модели SVM-регрессии с линейным ядром:

$$f(x) = wx + b.$$

Подставляем данные в условии значения:

$$f(x) = 1.5x + 0.5.$$

Границы ϵ -окрестности:

$$f(x) - \epsilon \leq y \leq f(x) + \epsilon.$$

2. Проверка точек:

– Для точки $(1, 2)$:

$$f(1) = 1.5 \cdot 1 + 0.5 = 2.0, \quad 2 - 0.5 \leq 2 \leq 2 + 0.5 \quad (\text{в окрестности}).$$

– Для точки $(2, 2.8)$:

$$f(2) = 1.5 \cdot 2 + 0.5 = 3.5, \quad 3.5 - 0.5 \not\leq 2.8 \leq 3.5 + 0.5 \quad (\text{вне окрестности}).$$

– Для точки $(3, 5.2)$:

$$f(3) = 1.5 \cdot 3 + 0.5 = 5.0, \quad 5.0 - 0.5 \leq 5.2 \leq 5.0 + 0.5 \quad (\text{в окрестности}).$$

– Для точки $(4, 8)$:

$$f(4) = 1.5 \cdot 4 + 0.5 = 6.5, \quad 6.5 - 0.5 \leq 8.0 \not\leq 6.5 + 0.5 \quad (\text{вне окрестности}).$$

3. Штрафные переменные:

– Для точки $(2, 2.8)$:

$$\xi_i = f(2) - y - \epsilon = 3.5 - 2.8 - 0.5 = 0.2.$$

– Для точки $(4, 8)$:

$$\xi_i^* = y - f(4) - \epsilon = 8 - 6.5 - 0.5 = 1.0.$$

Таким образом, штрафные переменные:

$$\xi_2^* = 0.2, \quad \xi_4 = 1.0.$$

Ответ: $\xi_2^* = 0.2, \quad \xi_4 = 1.0.$

1-norm SVM (LASSO SVM)

Аппроксимация эмпирического риска с L_1 -регуляризацией

$$\sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \mu \sum_{j=1}^n |w_j| \rightarrow \min_{w, w_0}$$

Плюс: отбор признаков с параметром селективности μ

- чем больше μ , тем меньше признаков останется

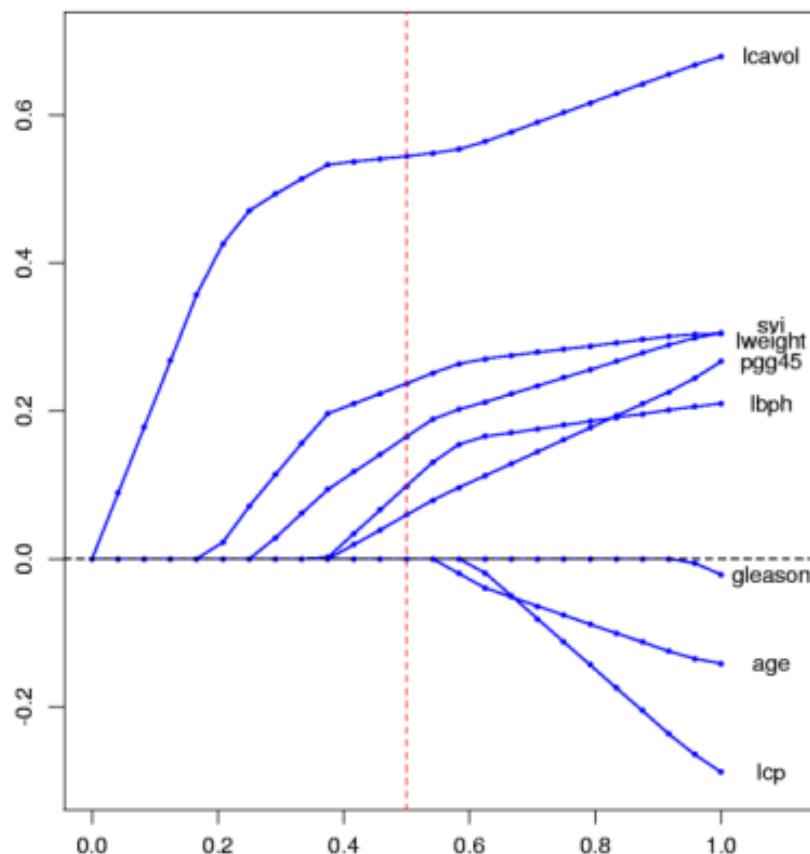
Минус: слишком агрессивный отбор признаков

- по мере увеличения μ признак может быть отброшен, хотя y существенно зависит от него (даже когда ещё не все шумовые признаки отброшены)

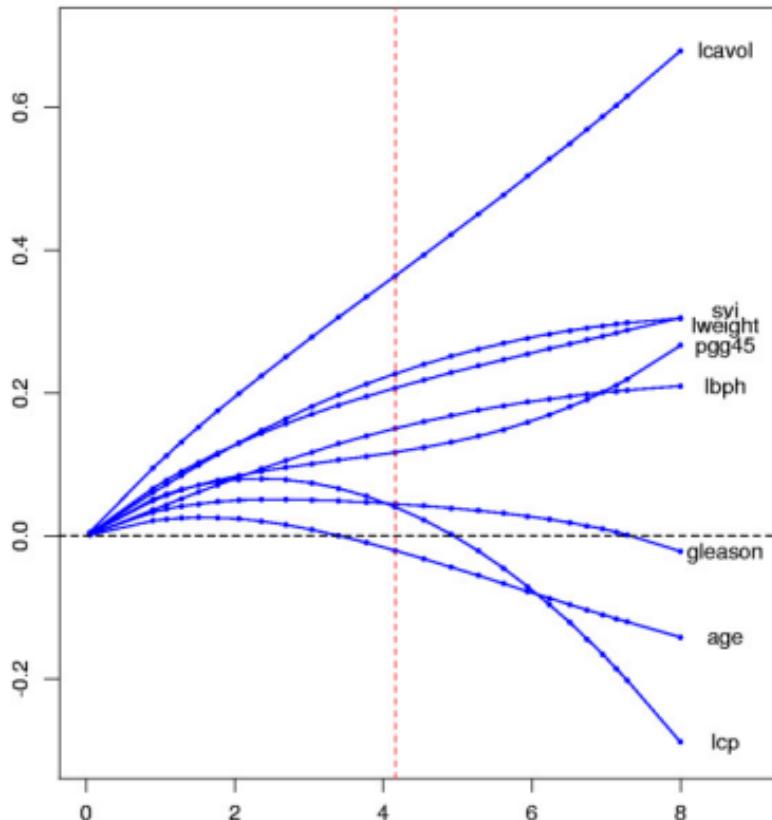
Сравнение L2 и L1 регуляризации

Зависимость весов w_j от коэффициента $\frac{1}{\mu}$:

- L_1 регуляризатор: $\mu \sum_j |w_j|$



- L_2 регуляризатор: $\mu \sum_j w_j^2$



Doubly Regularized SVM (Elastic Net SVM)

$$C \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \mu \sum_{j=1}^n |w_j| + \frac{1}{2} \sum_{j=1}^n w_j^2 \rightarrow \min_{w, w_0}$$

Плюсы:

- Параметр селективности μ управляет отбором признаков: чем больше μ , тем меньше признаков останется
- Есть эффект группировки (grouping effect): значимые зависимые признаки отбираются вместе

Минусы:

- Шумовые признаки также группируются и могут вместе оставаться в модели
- Приходится подбирать два параметра регуляризации μ, τ (есть специальные методы, например, regularization path)

Elastic Net Analysis

Elastic Net менее жёстко отбирает признаки.

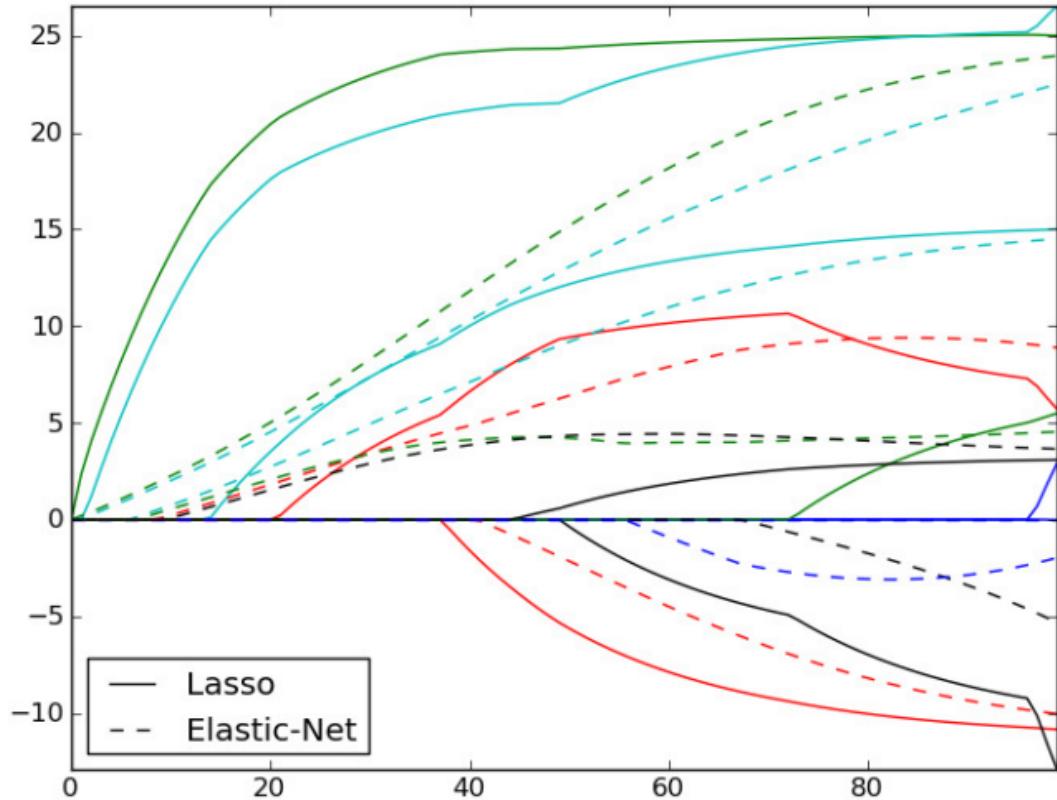


Рис. 5.2. Зависимости весов w_j от коэффициента $\log \frac{1}{\mu}$

Support Features Machine (SFM)

$$C \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \sum_{j=1}^n R_\mu(w_j) \rightarrow \min_{w, w_0}$$

$$R_\mu(w_j) = \begin{cases} 2\mu|w_j|, & |w_j| \leq \mu \\ \mu^2 + w_j^2, & |w_j| \geq \mu \end{cases}$$

Плюсы

- Только один параметр регуляризации μ

- Отбор признаков с параметром селективности μ
- Эффект группировки: значимые зависимые признаки ($|w_j| > \mu$) входят в решение совместно (как в *Elastic Net*)
- Шумовые признаки ($|w_j| < \mu$) не группируются и подавляются независимо друг от друга (как в *LASSO*)

Relevance Features Machine (RFM)

$$C \sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \sum_{j=1}^n \ln(w_j^2 + \frac{1}{\mu}) \rightarrow \min_{w, w_0}$$

$$R(w) = \ln(w^2 + \frac{1}{\mu}), \quad \mu = 0.1, 1, 100$$

Плюсы

- Только один параметр регуляризации μ
- Отбор признаков с параметром селективности μ
- Есть эффект группировки
- Лучше отбирает набор значимых признаков, когда они только совместно обеспечивают хорошее решение

Задачи

Задача 1

Качественно объяснить, почему L_1 -регуляризатор приводит к отбору признаков

Ответ:

Аппроксимация эмпирического риска с L_1 -регуляризацией:

$$\sum_{i=1}^{\ell} (1 - M_i(w, w_0))_+ + \mu \sum_{j=1}^n |w_j| \rightarrow \min_{w, w_0}$$

Почему L_1 -регуляризатор приводит к отбору признаков?

Замена переменных:

$$u_j = \frac{1}{2}(|w_j| + w_j), \quad v_j = \frac{1}{2}(|w_j| - w_j).$$

Тогда

$$w_j = u_j - v_j \quad |w_j| = u_j + v_j.$$

$$\begin{aligned} \sum_{i=1}^{\ell} (1 - M_i(u - v, w_0))_+ + \mu \sum_{j=1}^n (u_j + v_j) &\rightarrow \min_{u,v}, \\ u_j \geq 0, \quad v_j \geq 0, \quad j = 1, \dots, n. \end{aligned}$$

чем больше μ , тем больше индексов j таких, что $u_j = v_j = 0$, но тогда $w_j = 0$, значит, **признак не учитывается**.

Задача 2

Привести пример нежелательного эффекта в процессе обучения, с которым поможет справиться регуляризация

Ответ:

Регуляризация помогает в случае линейной зависимости (мультиколлинеарности) признаков:

Пусть построен классификатор: $a(x, w) = \text{sign}\langle w, x \rangle$

Мультиколлинеарность: $\exists u \in \mathbb{R}^n: \forall x \in X \langle u, x \rangle = 0$

Неединственность решения и рост нормы вектора весов: $\forall \gamma \in \mathbb{R} a(x, w) = \text{sign}\langle w, x \rangle = \text{sign}\langle w + \gamma u, x \rangle$

Проявления переобучения:

- слишком большие веса $|w_j|$ разных знаков
- неустойчивость дискриминантной функции $\langle w, x \rangle$
- $Q(X^\ell) \ll Q(X^k)$

Способ уменьшить переобучение:

регуляризация $\|w\| \rightarrow \min$ (сокращение весов, *weight decay*)

Задача 3

Дана задача оптимизации:

$$\frac{1}{2}(wx - b)^2 + \lambda|w| \rightarrow \min_w,$$

где $x, b \in \mathbb{R}$; $\lambda \geq 0$

При каких λ данная задача имеет решение $w_0 \neq 0$?

Ответ:

Находим правую и левую односторонние производные в нуле и рассматриваем, когда они больше и меньше 0 соответственно:

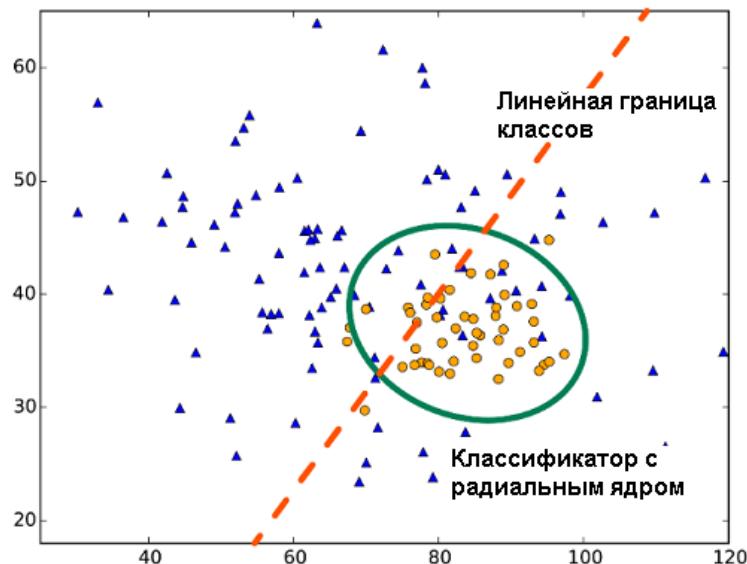
$$\begin{cases} -xb + \lambda > 0 \\ -xb - \lambda < 0 \end{cases} \Leftrightarrow \lambda > |xb|$$

$$\lambda \geq 0$$

Это условие на λ , при котором задача имеет решение $w_0 = 0$, поэтому нам подходит $\lambda \in [0; |xb|)$.

Ядерные функции машины опорных векторов

При наличии нелинейной связи между признаками и откликом качество линейных классификаторов, как показано на рисунке ниже, часто может оказаться неудовлетворительным.



Для учета нелинейности обычно расширяют пространство переменных, включая различные функциональные преобразования исходных предикторов (полиномы, экспоненты и проч.). Машину опорных векторов SVM (Support Vector Machine) можно рассматривать как нелинейное обобщение линейного классификатора, полученного при решении двойственной задачи (прямая задача является задачей бинарной классификации):

$$\begin{cases} w = \sum_{i=1}^{\ell} \lambda_i y_i x_i; \\ w_0 = \langle w, x_i \rangle - y_i, \quad \text{для любого } i : \lambda_i > 0, M_i = 1. \end{cases}$$

Линейный классификатор с признаками $f_i(x) = \langle x, x_i \rangle$:

$$a(x) = \operatorname{sign} \left(\sum_{i=1}^{\ell} \lambda_i y_i \langle x, x_i \rangle - w_0 \right).$$

Введя здесь новую нелинейную функцию $K(x, x')$, вместо $\langle x, x' \rangle$, можно строить модели с разделяющих поверхностями самой различной формы.

Дадим формальное определение:

Определение:

Функция $K : X \times X \rightarrow \mathbb{R}$ – ядро, если $K(x, x') = \langle \psi(x), \psi(x') \rangle$ при некотором $\psi : X \rightarrow H$, где H – гильбертово пространство.

В качестве таких функций чаще всего используют следующие:

- линейное ядро: $K(x, x') = \langle x, x' \rangle$, что соответствует классификатору на опорных векторах в исходном пространстве (см. рис. 6.3);
- полиномиальное ядро со степенью p : $K(x, x') = (1 + \langle x, x' \rangle)^p$;
- гауссово ядро с радиальной базовой функцией (RBF): $K(x, x') = \exp(\gamma \langle x - x' \rangle^2)$;
- сигмоидное ядро: $K(x, x') = \tanh(\gamma \langle x, x' \rangle + \beta_0)$.

Для того, чтобы определять является ли функция ядром, существует

Теорема Мерсера:

Функция двух переменных $K(x, x')$ является ядром тогда и только тогда, когда она

- симметрична, то есть $K(x, x') = K(x', x)$;
- неотрицательно определена, то есть $\int_X \int_X K(x, x') g(x)g(x') dx dx' \geq 0$ для любой функции $g : X \rightarrow \mathbb{R}$;

Нужно отметить, что на практике проверка неотрицательной определенности функции $K(x, x')$ часто является нелегкой задачей. Вместо этого обычно используют какие-то конструктивные методы синтеза ядер, например достаточно очевидное утверждение: $K(x, x') = \alpha_1 K_1(x, x') + \alpha_2 K_2(x, x')$ при $\alpha_1, \alpha_2 > 0$ – ядро.

Задача 1

Найти пространство H и преобразование $\psi : X \rightarrow H$, при которых $K(x, x') = \langle \psi(x), \psi(x') \rangle$, где $X = \mathbb{R}^n$, $K(x, x') = \langle x, x' \rangle^2$, $x = (x_1, x_2, \dots, x_n)$, $x' = (x'_1, x'_2, \dots, x'_n)$

Решение

$$\begin{aligned} K(x, x') &= \langle x, x' \rangle^2 = \langle (x_1, x_2, \dots, x_n), (x'_1, x'_2, \dots, x'_n) \rangle^2 = (x_1 x'_1 + \dots + x_n x'_n)^2 \\ &= x_1^2 x'_1^2 + \dots + x_n^2 x'_n^2 + 2x_1 x'_1 x_2 x'_2 + \dots + 2x_1 x'_1 x_n x'_n + \dots + 2x_{n-1} x'_{n-1} x_n x'_n = \\ &= \langle (x_1^2, x_2^2, \dots, x_n^2, \sqrt{2}x_1 x_2, \dots, \sqrt{2}x_{n-1} x_n), (x'_1^2, x'_2^2, \dots, x'_n^2, \sqrt{2}x'_1 x'_2, \dots, \sqrt{2}x'_{n-1} x'_n) \rangle \end{aligned}$$

Подсчетом слагаемых можно убедиться, что $\dim H = \frac{1}{2}n(n-1)$, а $\psi : (x_1, x_2, \dots, x_n) \mapsto (x_1^2, x_2^2, \dots, x_n^2, \sqrt{2}x_1 x_2, \dots, \sqrt{2}x_{n-1} x_n)$

Задача 2

Доказать, что $\forall \psi : X \rightarrow \mathbb{R}$ произведение $K(x, x') = \psi(x)\psi(x')$ – ядро.

Решение

Действительно, произведение $\psi(x)\psi(x')$ есть скалярное произведение в одномерном пространстве \mathbb{R} , а в силу свойств скалярного произведения (положительная определенность и симметричность) по теореме Мерсера получаем, что $K(x, x')$ ядро.

Задача 3

Доказать, что $K(x, x') = \exp(-\langle x - x' \rangle^2)$ ядро (гауссово ядро с радиальной базовой функцией).

Решение

Для доказательства нужно воспользоваться утверждением: композиция произвольного ядра K_0 и произвольной функции $f : \mathbb{R} \rightarrow \mathbb{R}$, представимой в виде сходящегося степенного ряда с неотрицательными коэффициентами $K(x, x') = f(K_0(x, x'))$, является ядром. $K_0(x, x') = \langle x - x' \rangle^2$ является ядром по свойствам

скалярного произведения. Как известно, экспонента представима в виде степенного ряда с положительными коэффициентами, поэтому выполнены все условия утверждения, что доказывает, что $K(x, x')$ ядро.

Relevance Vector Machine(RVM)

Идея метода релевантных векторов

Берем за основу структуру решения как в SVM:

$$\sum_{i=1}^{\ell} \lambda_i y_i x_i$$

(опорным векторам x_i соответствуют $\lambda_i \neq 0$)

Проблема: Какие из коэффициентов λ_i лучше обнулить?

Делаем так, что регуляризатор зависит не от w , а от λ_i .

$$\rho(\lambda) = \frac{1}{(2\pi)^{l/2} \sqrt{\alpha_1 \dots \alpha_l}} \exp\left(-\sum_{i=1}^{\ell} \frac{\lambda_i^2}{2\alpha_i}\right)$$

- т.е. λ_i - независимые, гауссовые и с дисперсиями α_i

Будем решать задачу оптимизации с регуляризатором (логарифмируем плотность

- дисперсию не считаем константой):

$$\frac{1}{2} \sum_{i=1}^{\ell} \ln \alpha_i + \frac{\lambda_i^2}{\alpha_i}$$

Задача оптимизации:

$$\sum_{i=1}^{\ell} (1 - M_i(w(\lambda), w_0))_+ + \frac{1}{2} \sum_{i=1}^{\ell} \ln \alpha_i + \frac{\lambda_i^2}{\alpha_i} \rightarrow \min_{\lambda, \alpha}$$

Если одновременно оптимизируем и λ и α , то при уменьшении дисперсию, λ будет маленькая - то есть "обнуляться" и соответствующие объекты не будут опорными.

Плюсы

- Опорных векторов, как правило меньше (более "разреженное" решение)
- Шумовые вбросы уже не входят в число опорных
- Не надо искать параметр регуляризации (вместо этого α_i оптимизируется в процессе обучения)
- Аналогично SVM, можно использовать ядра

Минусы

- Не всегда есть преимущество по качеству классификации

Задачи по RVM

Задача 1:

Объяснить, как метод релевантных векторов (Relevance Vector Machine, RVM) достигает разреженности в параметрах модели.

Ответ:

Метод релевантных векторов (RVM) достигает разреженности через байесовскую структуру, устанавливая независимые нормальные априорные распределения с нулевым средним для весов модели, каждый с собственной гиперпараметром точности (обратная дисперсия). Конкретно, для каждого веса w_i существует связанный гиперпараметр точности α_i . Априорное распределение весов задается как:
Априорное распределение весов задается как:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^N \mathcal{N}(w_i|0, \alpha_i^{-1})$$

В процессе обучения RVM определяет гиперпараметры $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_N]$, максимизируя маржинальное правдоподобие данных. Многие из этих гиперпараметров стремятся к бесконечности, что приводит к обнулению соответствующих весов w_i .

Задача 2:

Объяснить, в чем заключается основное отличие метода релевантных векторов (RVM) от метода опорных векторов (SVM), если рассматривать их с точки зрения подхода к регуляризации и использования априорных предположений.

Ответ:

- Основное отличие заключается в том, что RVM использует байесовский подход, тогда как SVM — детерминированный метод.

- В RVM вводятся априорные распределения на веса модели, что позволяет автоматически выполнять регуляризацию, оставляя наиболее значимые параметры.
- В результате RVM достигает разреженности (аналогично SVM) без использования параметра регуляризации, а лишь за счет максимизации апостериорной вероятности.

Задача 3:

Объяснить, в чем метод релевантных векторов может быть быстрее и медленнее метода опорных векторов

Ответ:

Быстрее, потому что:

- В RVM гораздо быстрее применение модели к новым точкам, потому что опорных векторов гораздо меньше
- В SVM необходимо оптимизировать параметр регуляризации C (или аналогичный параметр для других ядер), что требует дополнительных затрат времени. В RVM регуляризация происходит автоматически через байесовский подход (гиперпараметры α_i), устраняя необходимость выбора таких параметров вручную.

Медленнее, потому что:

- Байесовский подход в RVM делает процесс обучения более ресурсоемким, чем у SVM. Это связано с итеративными расчетами для апостериорных вероятностей и гиперпараметров - обучение происходит дольше

Резюме по линейным классификаторам

- SVM - лучший метод линейной классификации
- SVM изящно обобщается для нелинейной классификации, для линейной и нелинейной регрессии
- Апроксимация пороговой функции потерь $L(M)$ увеличивает зазор и повышает качество классификации
- Регуляризация устраняет мультиколлинеарность и уменьшает переобучение
- Негладкость функции потерь приводит к отбору объектов
- Негладкость регуляризатора приводит к отбору признаков

Комментарий по последним двум пунктам:

Отбор опорных объектов возник из-за того, что возникает негладкая функция потерь, аналогично (но смотрим с другой стороны) отбор признаков так же связан с негладкостью регуляризатора.

Что мы можем отбирать в задаче классификации?

У нас есть матрица объекты - признаки, то есть либо строки, либо столбцы являются лишними.

Соответственно и возникает подход: решаем задачу классификации, отфильтровав матрицу по строкам и по столбцам.

Фильтрация может быть реализована:

- путем выбора негладких функций потерь, которые приводят к фильтрации строк (набора объектов)
- путем выбора негладких регуляризаторов, которые приводят к фильтрации столбцов (набора признаков)

Роль регуляризации возникает благодаря выписыванию оптимизационной задачи с ограничениями-неравенствами → требуется применять условие Каруша-Куна-Таккера (применимы только к гладким функциям)

Если функция негладкая → вводим дополнительные переменные, неотрицательные и могут обращаться в ноль → происходит отбор

Обобщения линейного SVM. Ядра и спрямляющие пространства. SVM как двухслойная нейронная сеть.

Нелинейное обобщение SVM

Определение. Функция $K : X \rightarrow X$ - ядро, если $K(x, \tilde{x}) = \langle \psi, \tilde{\psi} \rangle$ при некотором $\psi : X \rightarrow H$, где H - гильбертово пространство

Теорема. Функция $K(x, \tilde{x})$ является ядром тогда и только тогда, когда она симметрична: $K(x, \tilde{x}) = K(\tilde{x}, x)$ и неотрицательно определена: $\iint_{XX} K(x, \tilde{x})g(x)g(\tilde{x})dxdy \geq 0$ для любой $g : X \rightarrow \mathbb{R}$

Конструктивные методы синтеза ядер:

- $K(x, \tilde{x}) = \langle x, \tilde{x} \rangle$ - ядро
- $K(x, \tilde{x}) = 1$ - ядро
- $K(x, \tilde{x}) = K_1(x, \tilde{x}) \times K_2(x, \tilde{x})$ - ядро, если K_1, K_2 - ядра

- $K(x, \tilde{x}) = \psi(x)\psi(\tilde{x})$ - ядро при $\psi : X \rightarrow \mathbb{R}$
- $K(x, \tilde{x}) = \alpha_1 K_1(x, \tilde{x}) + \alpha_2 K_2(x, \tilde{x}), \alpha_1 > 0, \alpha_2 > 0$
- $K(x, \tilde{x}) = \iint_X s(x, z)s(\tilde{x}, z)dz$ - ядро, если $s : X \times X \rightarrow \mathbb{R}$ - симметричная интегрируемая функция
- $K(x, \tilde{x}) = f(K_0(x, \tilde{x}))$ - ядро, если K_0 - ядро и $f : \mathbb{R} \rightarrow \mathbb{R}$ представима в виде сходящегося степенного ряда с неотрицательными коэффициентами

Задача 1

Найти пространство H и преобразование $\psi : X \rightarrow H$, при которых $K(x, \tilde{x}) = \langle \psi(x), \psi(\tilde{x}) \rangle$, где $X = \mathbb{R}^2$, $K(u, v) = \langle u, v \rangle^2$, $u = (u_1, u_2)$, $v = (v_1, v_2)$

Решение

$$\begin{aligned} K(u, v) = \langle u, v \rangle^2 &= \langle (u_1, u_2), (v_1, v_2) \rangle^2 = (u_1 v_1 + u_2 v_2)^2 = u_1^2 u_2^2 + 2u_1 v_1 u_2 v_2 = \\ &= \langle (u_1^2, u_2^2, \sqrt{2}u_1 u_2), (v_1^2, v_2^2, \sqrt{2}v_1 v_2) \rangle \end{aligned}$$

То есть $H = \mathbb{R}^3$, $\psi : (u_1, u_2) \rightarrow (u_1^2, u_2^2, \sqrt{2}u_1 u_2)$

Задача 2

Решите предыдущую задачу при условии, что $X = \mathbb{R}^n$, $K(u, v) = \langle u, v \rangle^d$

Решение

Заметим, что в таком случае компонентами вектора $\psi(u)$ будут различные произведения $(u_1)^{d_1}, (u_2)^{d_2}, \dots, (u_n)^{d_n}$ при $d_1, d_2, \dots, d_n : d_1 \geq 0, d_2 \geq 0, \dots, d_n \geq 0$ и $d_1 + d_2 + \dots + d_n = d$. Число мономов и есть размерность пространства H : $\dim H = C_{n+d-1}^d$ - число сочетаний с повторением.

Задача 3

Представьте нелинейное обобщение SVM для классификатора в виде двухслойной нейронной сети. Считать, что опорные объекты из множества $X = \mathbb{R}^n$

Решение

Обозначим x_1, x_2, \dots, x_h как опорные объекты. Тогда

$$a(x) = \text{sign}\left(\sum_{i=1}^h \lambda_i y_i K(x, \tilde{x}) - w_0\right) \quad (5.1)$$

Двухслойная нейронная сеть представлена на рисунке ниже.

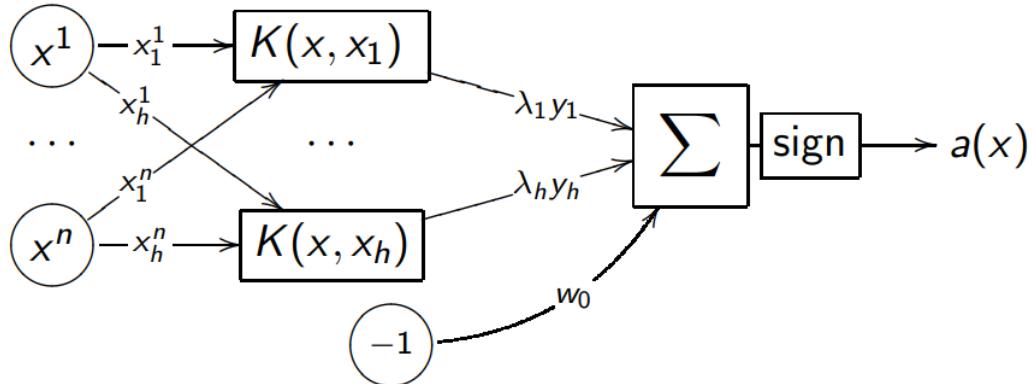


Рис. 5.3. SVM в виде двухслойной нейросети

Глава 6

Метод главных компонент

Введение в метод главных компонент(PCA)

Метод главных компонент (Principal Component Analysis, PCA) — это статистический метод, используемый для снижения размерности данных с сохранением наиболее значимой информации. PCA находит новые признаки (главные компоненты), которые представляют собой линейные комбинации исходных признаков, причем эти компоненты ортогональны и ранжированы по величине объясняемой дисперсии. **Основные этапы метода:**

1. Центрирование данных:

Данные центрируются так, чтобы среднее значение каждой переменной было равно нулю:

$$X_c = X - \bar{X},$$

где X - исходная матрица данных (размер $n \times p$), \bar{X} - вектор средних значений по столбцам.

2. Построение ковариационной матрицы:

Вычисляется ковариационная матрица:

$$\Sigma = \frac{1}{n-1} X_c^T X_c$$

где Σ - симметричная матрица размером $p \times p$.

3. Собственные значения и собственные векторы:

Решается задача нахождения собственных значений и собственных векторов ковариационной матрицы:

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

где λ_i - собственные значения, \mathbf{v}_i - соответствующие им собственные векторы.

4. Ранжирование главных компонент:

Собственные значения упорядочиваются по убыванию:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$$

Первые несколько компонент, соответствующие самым большим собственным значениям, объясняют большую часть дисперсии данных.

5. Проекция данных:

Данные проецируются на главные компоненты:

$$Z = X_c V_k,$$

где V_k — матрица k собственных векторов, соответствующих k наибольшим собственным значениям, Z — матрица данных в пространстве главных компонент.

Свойства метода:

- Главные компоненты ортогональны:

$$\mathbf{v}_i^T \mathbf{v}_j = 0, \quad i \neq j$$

- Дисперсия объясняется последовательностью собственных значений:

$$\text{Объясненная дисперсия} = \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}.$$

Задачи на использование метода главных компонент

Задача 1: Вклад признаков в главные компоненты

Пусть $X \in R_{n \times p}$ набор данных с n образцами (строками) и p признаками (столбцами). PCA стремится найти набор собственных векторов (главных компонент), которые максимизируют дисперсию данных при проецировании на эти векторы.

Задача состоит в том, чтобы математически оценить, какой вклад вносит каждый признак в главные компоненты, и проранжировать признаки в зависимости от их вклада.

Решение: Метод PCA ищет собственные векторы \mathbf{v}_i и собственные значения λ_i удовлетворяющие:

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i,$$

где λ_i — величина дисперсии данных вдоль \mathbf{v}_i . Собственные векторы \mathbf{v}_i формируют матрицу $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p]$, где каждый столбец \mathbf{v}_i указывает направления главных компонент.

Вклад признака в главные компоненты:

Каждый признак в X вносит вклад в главные компоненты через веса собственных векторов \mathbf{v}_i . Элементы v_{ij} (где v_{ij} — j -й элемент i -го собственного вектора) определяют значимость j -го признака для i -й главной компоненты.

Вклад j -го признака в i -ю главную компоненту оценивается как квадрат соответствующего элемента v_{ij}^2 .

Общий вклад j -го признака во все главные компоненты можно найти, суммируя его взвешенные вклады с учётом дисперсий (λ_i):

$$j = \sum_{i=1}^p \lambda_i v_{ij}^2.$$

Этот показатель учитывает как значимость признака для каждой компоненты (v_{ij}^2), так и долю дисперсии, объясняемую компонентой (λ_i).

На конкретном примере: Пусть собственные вектора образуют матрицу V :

$$V = \begin{bmatrix} 0.5 & 0.6 & 0.3 & 0.1 \\ 0.4 & -0.7 & 0.2 & 0.5 \\ -0.6 & 0.2 & 0.7 & -0.4 \\ 0.5 & 0.3 & -0.6 & -0.6 \end{bmatrix}$$

Собственные значения:

$$\Lambda = \text{diag}(4.0, 2.5, 1.2, 0.3)$$

Посчитаем вклад признака 1 ($j = 1$): Для этого берём первую строчку V :

$$v_{1,.} = [0.5, 0.6, 0.3, 0.1]$$

Считаем:

$$\begin{aligned} \text{Contribution}_1 &= (0.5^2 \cdot 4.0) + (0.6^2 \cdot 2.5) + (0.3^2 \cdot 1.2) + (0.1^2 \cdot 0.3) \\ &= 1.0 + 0.9 + 0.108 + 0.003 = 2.011 \end{aligned}$$

То же самое повторяем для остальных строк и находим максимальное значение.

Задача 2: Ошибка "реконструкции" РСА

Пусть $X \in R_{n \times p}$ набор данных с n образцами (строками) и p признаками (столбцами), с помощью метода главных компонент нужно:

- Спроецируйте данные в более низкоразмерное пространство, определяемое k главными компонентами.
- Реконструируйте исходные данные из пространства пониженной размерности.
- Вычислите ошибку реконструкции и оцените, как она меняется в зависимости от количества сохраняемых компонент k .

Решение: Для реконструкции данных из k -мерного подпространства используется обратная проекция:

$$\hat{X} = ZV_k^T + \bar{X}$$

Здесь: - ZV_k^T возвращает проекцию данных в исходное p -мерное пространство.

- Добавление \bar{X} восстанавливает исходное смещение данных.

Ошибка реконструкции должна показывать какую часть информации мы потеряли при использовании только k компонент при репрезентации данных.

1. Определим ошибку реконструкции для одного объекта x_i :

$$E_i = \|x_i - \hat{x}_i\|^2 = \|(x_i - \bar{X}) - (z_i V_k^\top)\|^2$$

2. Обобщим на весь набор данных:

$$E = \frac{1}{n \times p} \sum_{i=1}^n \|x_i - \hat{x}_i\|^2$$

3. Заменим на выражение для \hat{x}_i :

$$E = \frac{1}{n \times p} \sum_{i=1}^n \|x_i - \bar{X} - ZV_k^\top\|^2$$

Мы получили выражение для ошибки реконструкции. Теперь докажем, что ошибка реконструкции E уменьшается монотонно с k , и когда $k = p$, $E = 0$.

1. Общая дисперсия данных - это след ковариационной матрицы, которая представляет собой сумму всех собственных значений:

$$\text{Total Variance} = \sum_{j=1}^p \lambda_j$$

2. Дисперсия, которую уловили k компонент это:

$$\text{Captured Variance} = \sum_{j=1}^k \lambda_j$$

3. Ошибка реконструкции по сути является дисперсией, которую не удалось уловить, то есть просто:

$$E = \text{Total Variance} - \text{Captured Variance} = \sum_{j=k+1}^p \lambda_j$$

4. Так как $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$, добавление большего числа компонент ($k \rightarrow k + 1$) уменьшает E :

$$\sum_{j=k+1}^p \lambda_j > \sum_{j=k+2}^p \lambda_j$$

5. В тот момент, когда $k = p$, $\sum_{j=k+1}^p \lambda_j = 0 \Rightarrow E = 0$.

Задача 3: Построить критерий D-оптимальности для выбора лучших k-компонент

Набор данных представляет собой матрицу $n \times p$, где n - число образцов (строк), а p - число признаков (столбцов). Введём критерий D-оптимальности, используемый для выбора подмножества точек из набора данных, которое максимизирует детерминант информационной матрицы.

$$D_{opt} : \max \det(X^T X)$$

Ключевым свойством критерия D-оптимальности является то, что он максимизирует объём многомерной фигуры, которая получается из рассматриваемых признаков.

Нужно построить критерий D-оптимальности для выбора лучших k главных компонент, которые максимизируют детерминант объясненной дисперсии (или объём фигуры) в k -мерном подпространстве РСА.

Решение:

Критерий D-оптимальности для подпространства k задаётся максимизацией детерминанта информационной матрицы Λ_k :

$$D_{opt}(k) = \max \det(\Lambda_k)$$

Так как Λ_k является диагональной матрицей, её детерминант равен произведению собственных значений:

$$\det(\Lambda_k) = \prod_{i=1}^k \lambda_i$$

Итак, наша задача сводится к выбору k -мерного подпространства (т.е. первых k главных компонент), которые максимизируют произведение $\lambda_1 \cdot \lambda_2 \cdots \lambda_k$, что эквивалентно решению следующей задачи:

$$\max_{V_k} \prod_{i=1}^k \lambda_i$$

где λ_i - собственные значения матрицы ковариации Σ . Для вычисления $D_{\text{opt}}(k)$:

1. Центрируем данные:

$$X_c = X - \bar{X},$$

где \bar{X} - матрица средних значений.

2. Вычисляем ковариационную матрицу:

$$\Sigma = \frac{1}{n-1} X_c^T X_c$$

3. Находим собственные значения $\lambda_1, \lambda_2, \dots, \lambda_p$ и соответствующие собственные векторы $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p$.

4. Выбираем первые k собственных значений $\lambda_1, \lambda_2, \dots, \lambda_k$, которые максимизируют:

$$\prod_{i=1}^k \lambda_i$$

Максимизация $\prod_{i=1}^k \lambda_i$ эквивалентна максимизации объёма **k -мерного эллипсоида**, описывающего данные в пространстве первых k главных компонент. Это позволяет отобрать k измерений, которые сохраняют максимальную дисперсию данных.

Денойзинг данных с помощью метода главных компонент

Метод главных компонент (PCA) — это один из самых распространенных и эффективных методов для снижения размерности данных, который также может быть применен для денойзинга. Денойзинг данных — это процесс удаления шума из наблюдений для выделения более чистых и значимых сигналов. Во многих

областях, таких как обработка изображений, анализ звука, необходимо избавляться от шумов, которые могут искажать результаты анализа и ухудшать качество моделей.

Благодаря своей способности выявлять скрытые структуры в многомерных данных PCA может быть использован для денойзинга. При анализе данных PCA проецирует данные в пространство главных компонент.

Основные моменты применения PCA для денойзинга включают:

- Выделение главных компонент: PCA позволяет выделить компоненты, вдоль которых данные наиболее рассеяны.
- Реконструкция данных: Уменьшение уровня шума и восстановление более "чистого" сигнала могут быть произведены с помощью удаления компонент, у которых меньше дисперсия.
- Снижение размерности: PCA снижает размерность данных, что делает их более вычислительно эффективными. Это особенно полезно в контексте обработки больших объемов данных.

PCA выбирают по следующим причинам:

- Линейность: PCA представляет собой линейный метод, что делает его простым для интерпретации и анализа. Однако основной недостаток заключается в том, что он может не эффективно обрабатывать данные с нелинейными взаимосвязями.
- Простота реализации: PCA является относительно простым в реализации. Многие библиотеки для анализа данных, такие как scikit-learn в Python, имеют встроенные инструменты для выполнения PCA.
- Характеристики контекста: PCA позволяет не только проводить денойзинг, но и выявлять основные характеристики и структуры в данных, что часто полезно при анализе образцов.

PCA может значительно упростить сложные многомерные наборы данных, обеспечивая при этом сохранение наиболее важной информации. При этом снижение размерности данных может помочь в построении более легких и интерпретируемых моделей, что особенно важно в машинном обучении. Однако у PCA есть и минусы. Так, отбрасывание компонент может привести к потере важной информации, если не удается точно оценить, какие компоненты следует сохранять. Ограниченнность линейности также может быть недостатком, так как в данных со сложными и нелинейными зависимостями PCA может не обнаружить все важные структуры. Хотя PCA упрощает данные, интерпретировать полученные главные компоненты может быть непросто, поскольку они являются линейными комбинациями исходных переменных.

Таким образом, метод главных компонент является мощным инструментом для денойзинга данных, особенно в контексте многомерных наборов данных. Его способность выявлять значимую информацию и удалять шум делает его предпочтительным выбором в различных областях. Однако важно учитывать плюсы и минусы метода, чтобы правильно применять его в соответствии с конкретными задачами и свойствами данных. Выбор подходящего количества компонент и тщательная интерпретация результатов остаются ключевыми шагами, которые могут существенно повлиять на успех применения РСА для денойзинга.

Задачи про денойзинг данных

Задача 1

Пусть I — это набор изображений, состоящий из n изображений, каждое из которых имеет m пикселей. Изображения могут содержать шум, например, из-за помех во время съемки. Требуется устраниить этот шум, сохраняя основные детали изображения с помощью РСА.

Решение: Данные можно представить в виде матрицы $X \in \mathbb{R}^{n \times m}$, где строки соответствуют изображениям, а столбцы — пикселям. Далее необходимо центрировать данные, вычитая среднее значение по каждому столбцу (пикселю):

$$X_{cen} = X - \mu$$

где μ — вектор среднего значения по всем изображениям. Вычислим ковариационную матрицу:

$$C = \frac{1}{n-1} X_{cen}^T X_{cen}$$

Далее необходимо найти собственные значения λ_i и собственные векторы v_i матрицы C и упорядочить собственные значения по убыванию. Отберем первые k собственных векторов, которые обеспечивают максимальную дисперсию, где k выбирается в зависимости от дисперсии. Запишем выбранные векторы в матрицу V_k . Спроектируем центрированные данные на выбранные главные компоненты:

$$Z = X_{cen} V_k.$$

Реконструируем уменьшенную версию изображений, используя только k основных компонент:

$$\hat{X} = Z V_k^T + \mu.$$

Задача 2

Пусть D — оригинальные данные, которые содержат как полезную информацию, так и шум. После применения РСА к данным были получены очищенные данные

(денойзинг) D' . Оценить, насколько эффективно PCA справилось с устраниением шума, используя метрику RMSE (Root Mean Square Error, RMSE).

Решение: Вычислим разницу (ошибку) между оригиналыми и очищенными данными:

$$E_i = D_i - D'_i, \quad \forall i = 1, 2, \dots, n$$

Затем вычисляем RMSE для получения общих значений ошибок:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n E_i^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (D_i - D'_i)^2}$$

Задача 3

Пусть D — оригиналые данные, которые содержат как полезную информацию, так и шум. После применения PCA к данным были получены очищенные данные (денойзинг) D' . Оценить, насколько эффективно PCA справилось с устраниением шума, используя коэффициент детерминации R^2 .

Решение:

$$R^2 = 1 - \frac{\sum_{i=1}^n (D_i - D'_i)^2}{\sum_{i=1}^n (D_i - \bar{D})^2}$$

где \bar{D} — среднее значение оригиналых данных.