Линейная регрессия

Линейная регрессия. Общее определение

Регрессионная модель, которая имеет вид:

$$y = f(x, b) + \varepsilon,\ E(\varepsilon) = 0,$$

где b - параметры модели, а ε - случайная ошибка модели, а функция f(x,b) имеет вид:

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k,$$

где b_i - параметры модели, k - количество параметров, x_i - признаки (фичи) исследуемых объектов.

Линейная регрессия. Пример

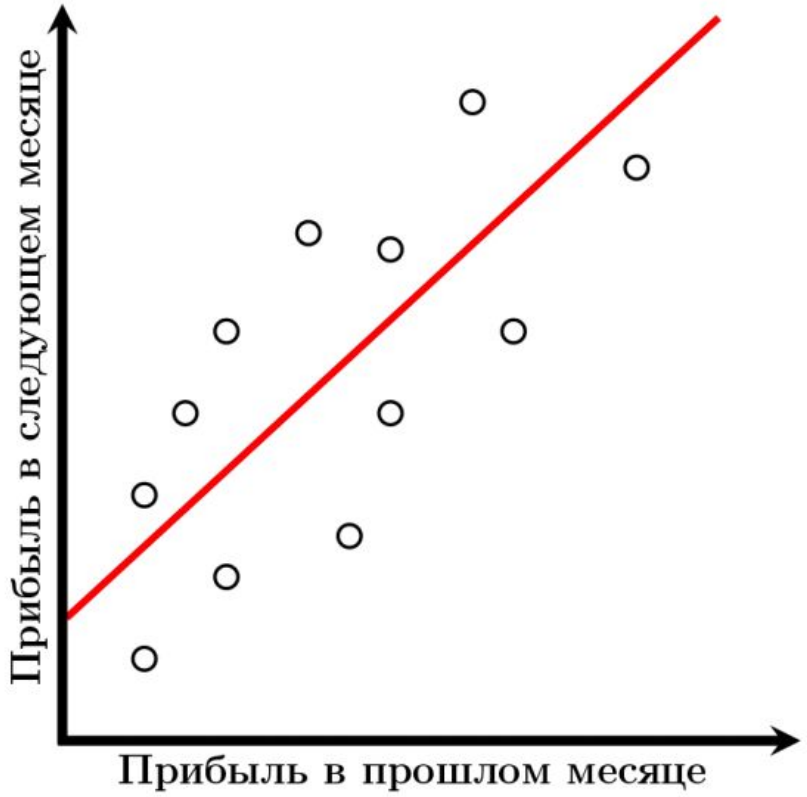
Попытаемся предсказать прибыль магазина в следующем месяце на основании прибыли в предыдущем.

Из имеющейся истории продаж построим обучающую выборку:

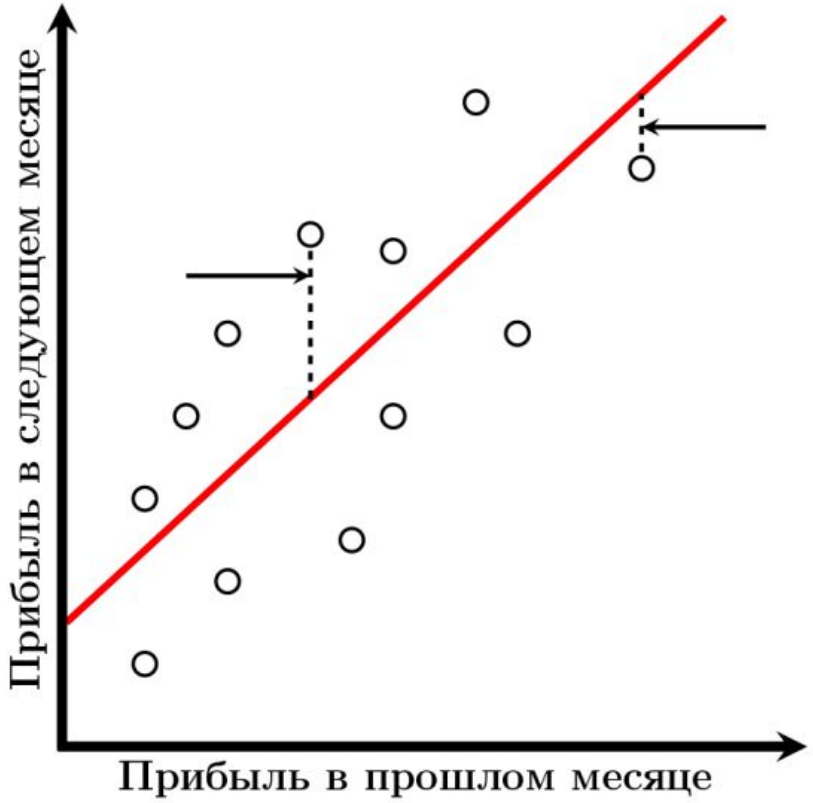
- значение в текущем месяце - результирующее значение
- значение в предыдущем месяце - единственный признак

Линейная регрессия. Пример

Линейная регрессия. Пример

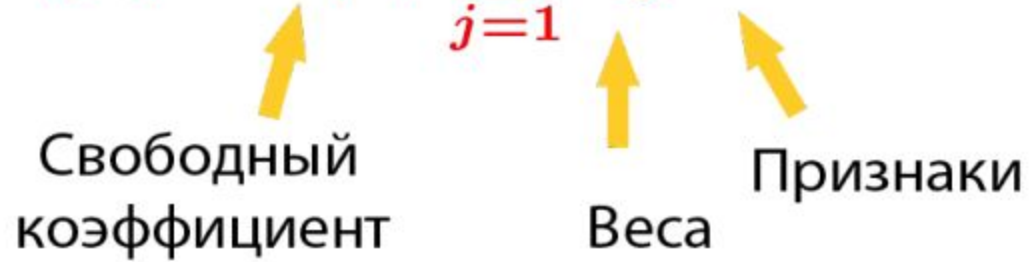


Линейная регрессия. Пример



Линейная регрессия

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



Линейная регрессия

- Добавим константный признак (например, число 1). Тогда сумму можно объединить и заменить на скалярное произведение векторов:

$$a(x) = \sum_{j=1}^{d+1} \mathbf{w}_j \, x^j = \langle \mathbf{w}, x \rangle$$

Линейная регрессия

Как определить качество регрессионной модели?

- Разность
- Модуль разности
- Квадрат разности

Вопрос?

Почему нельзя пойти дальше и взять, например, куб разности или даже четвёртую степень?

Среднеквадратичная ошибка

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

Среднеквадратичная ошибка

- Заменим предсказание на произведение вектора признаков на вектор параметров:

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

Вещественный вектор

Обучение линейной регрессии

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

- d неизвестных, где d - количество параметров
- один из признаков константный
- выпуклая функций (как сумма выпуклых функций)

Матричня запись

Матричня запись

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{\ell 1} & \cdots & x_{\ell d} \end{pmatrix}$$

Признак

Матричня запись

Вектор ответов:

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_\ell \end{pmatrix}$$

Матричня запись

$$Q(\mathrm{w}, X) = \frac{1}{\ell}\|X\,\mathrm{w} - y\|^2 \to \min_{\mathrm{w}}$$

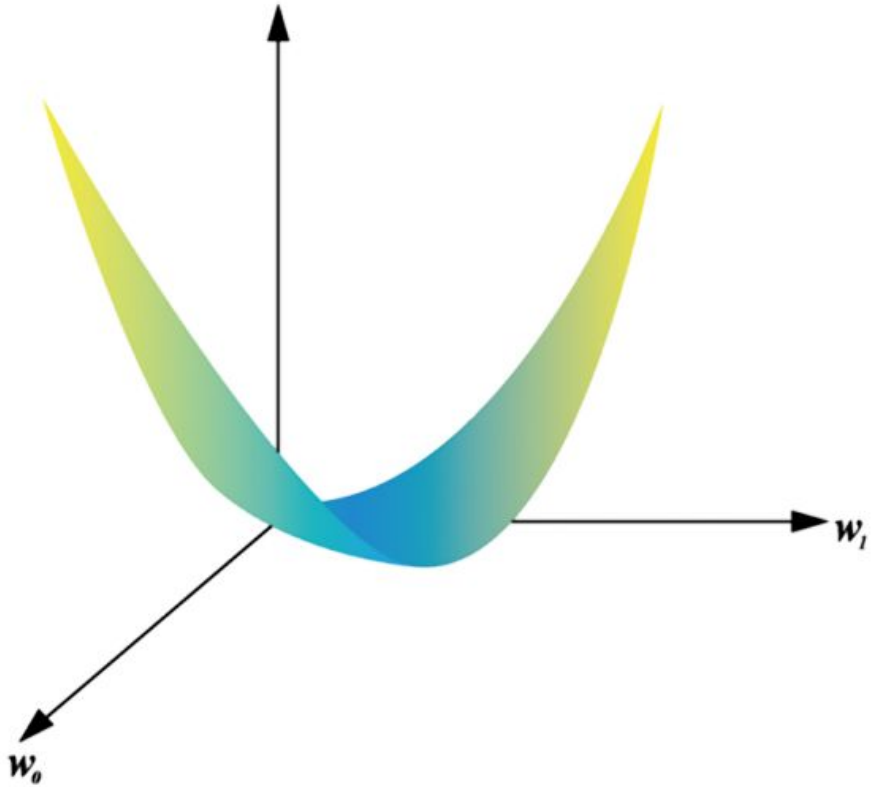
Точное решение

$$\mathbf{w}_* = (X^T X)^{-1} X^T y$$

- Могут возникнуть проблемы с точностью
- Требуется обращать матрицу размерности d*d, на что требуется порядка d^3 операций

Градиентный спуск

- Функция ошибки гладкая и выпуклая



Простейший случай для 1 признака

- Модель:

$$a(x) = w_1 x + w_0$$

- Функционал:

$$Q(w_0, w_1, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i)^2$$

Простейший случай для 1 признака

- Инициализация: $\mathbf{w}^0 = 0$
- Для итераций t=1,2,...

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta_t \nabla Q(\mathbf{w}^{t-1}, X)$$

- Условие остановки:

$$\| \mathbf{w}^t - \mathbf{w}^{t-1} \| < \varepsilon$$

Простейший случай для 1 признака

Частные производные:

$$\frac{\partial Q}{\partial w_1} = \frac{2}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i) x_i$$

$$\frac{\partial Q}{\partial w_0} = \frac{2}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i)$$

Многомерный случай

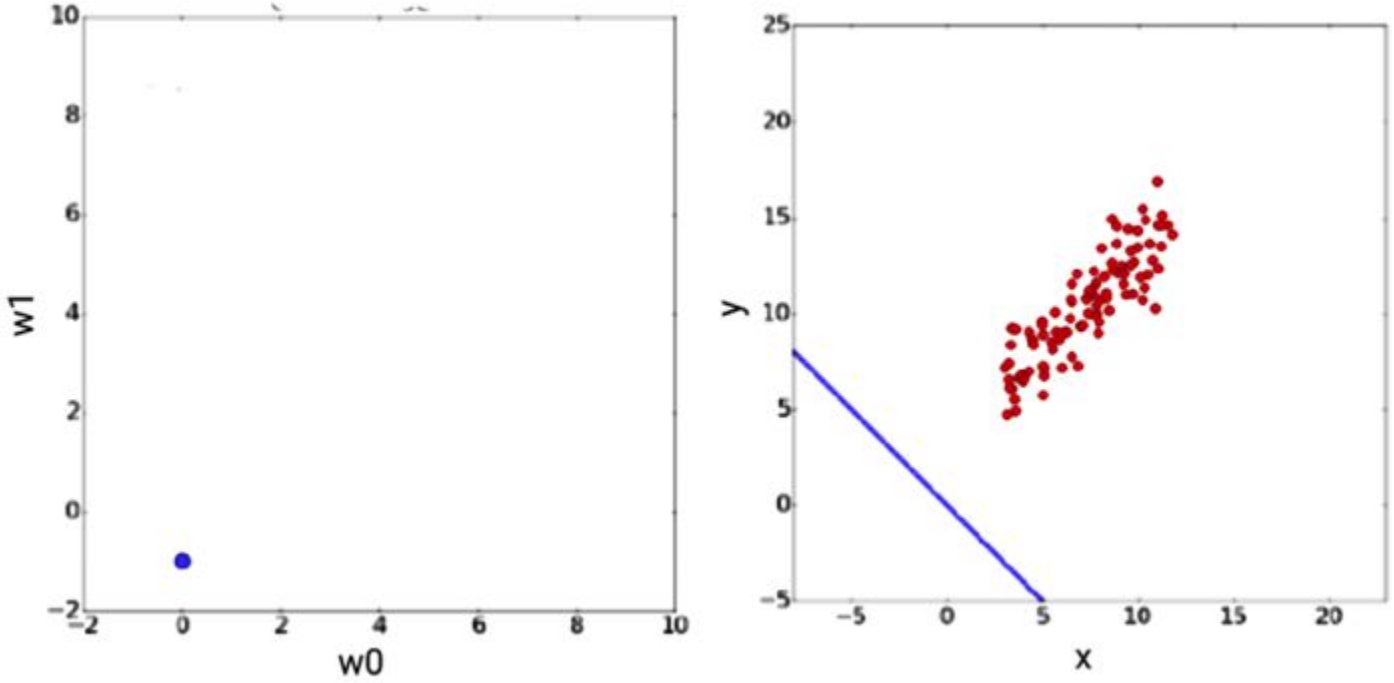
- Функционал:

$$Q(\mathrm{w}, X) = \frac{1}{\ell}\|X\,\mathrm{w} - y\|^2 \to \min_{\mathrm{w}}$$

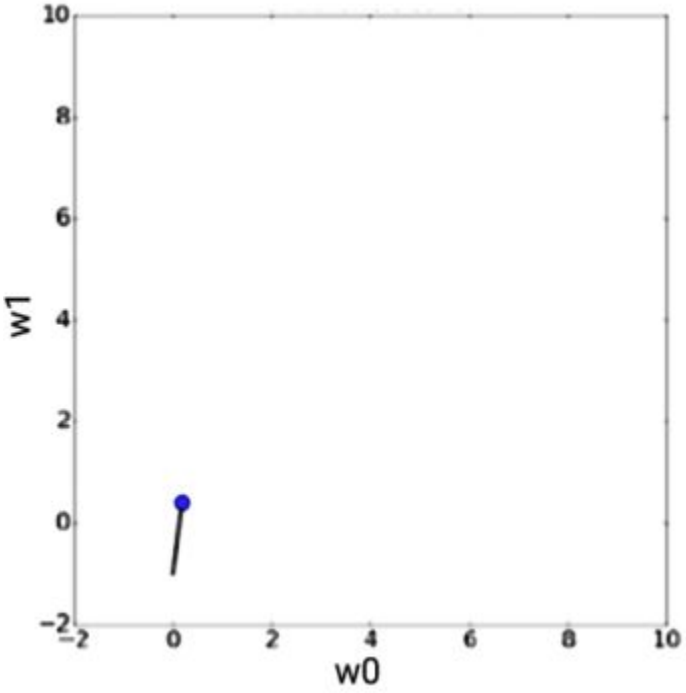
- Градиент:

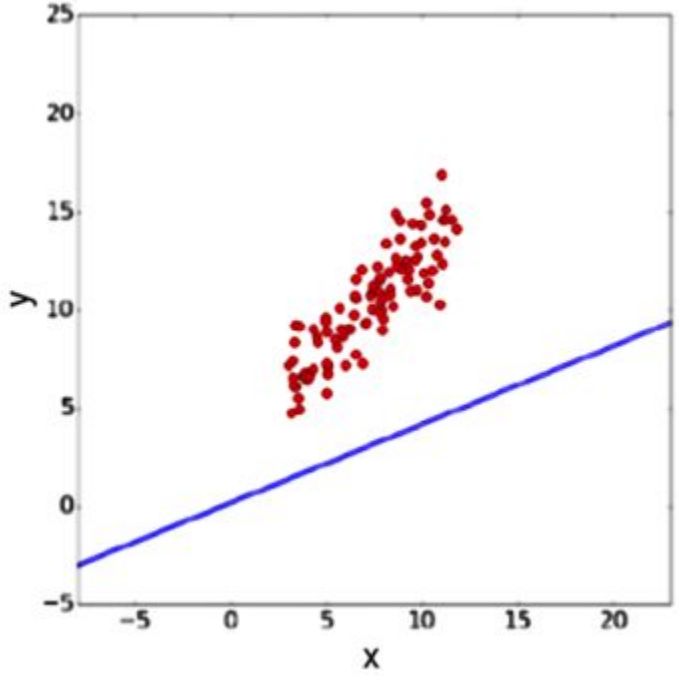
$$\nabla_{\mathrm{w}} Q(\mathrm{w}, X) = \frac{2}{\ell} X^T (X\,\mathrm{w} - y)$$

Градиентный спуск

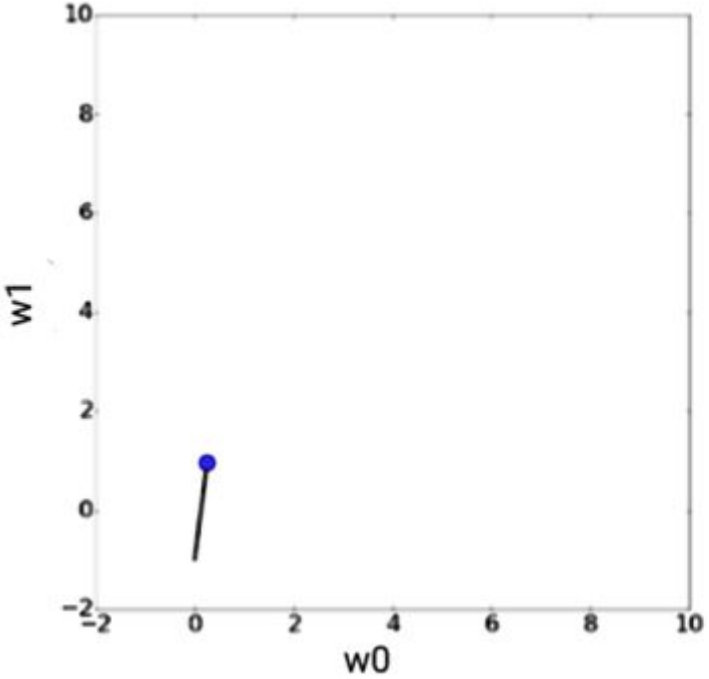


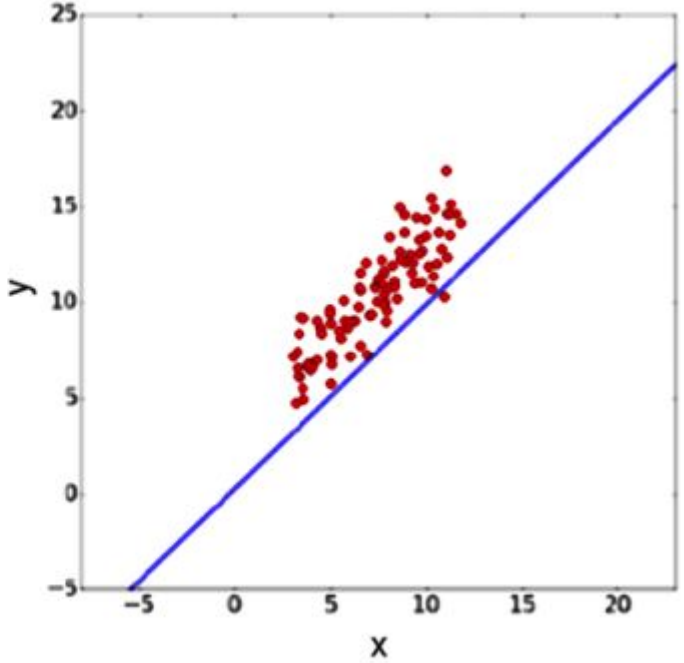
Градиентный спуск



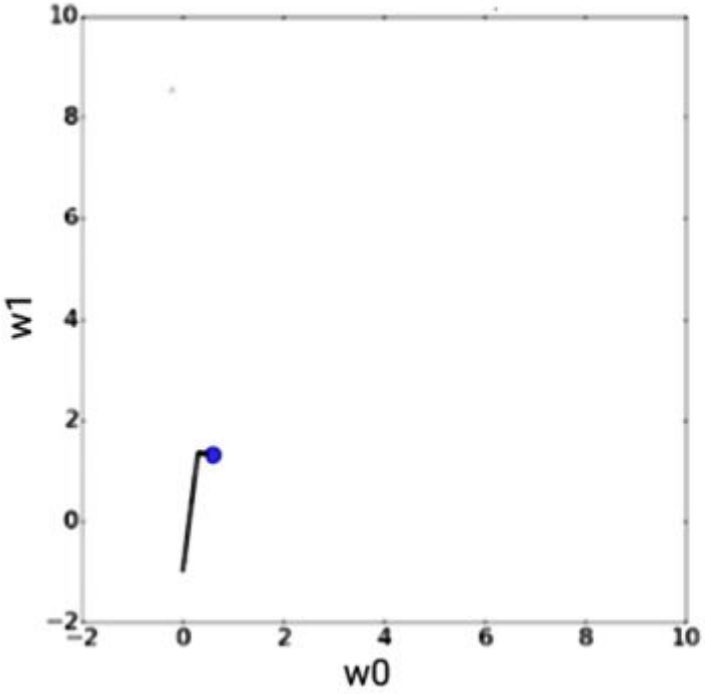


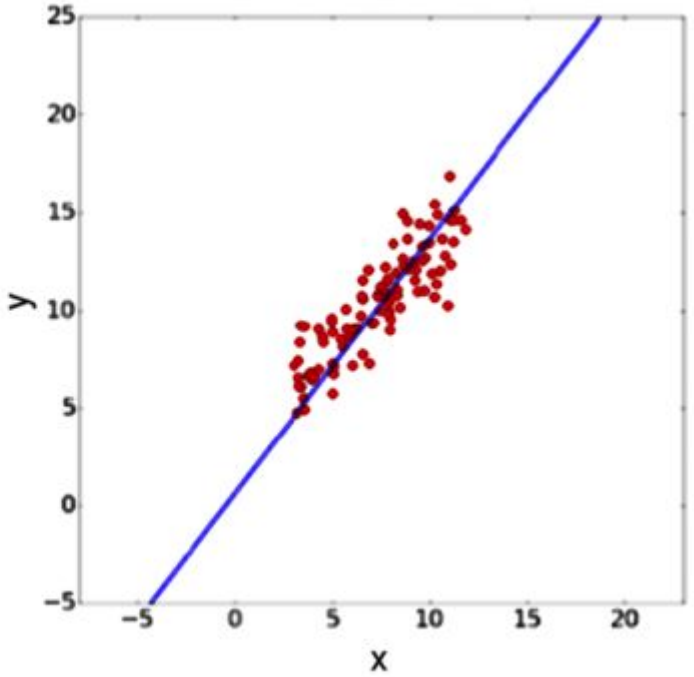
Градиентный спуск



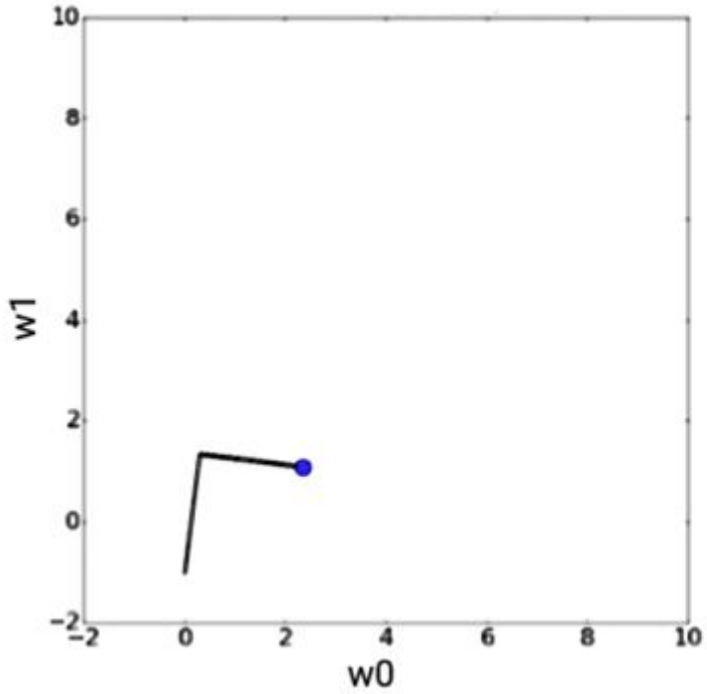


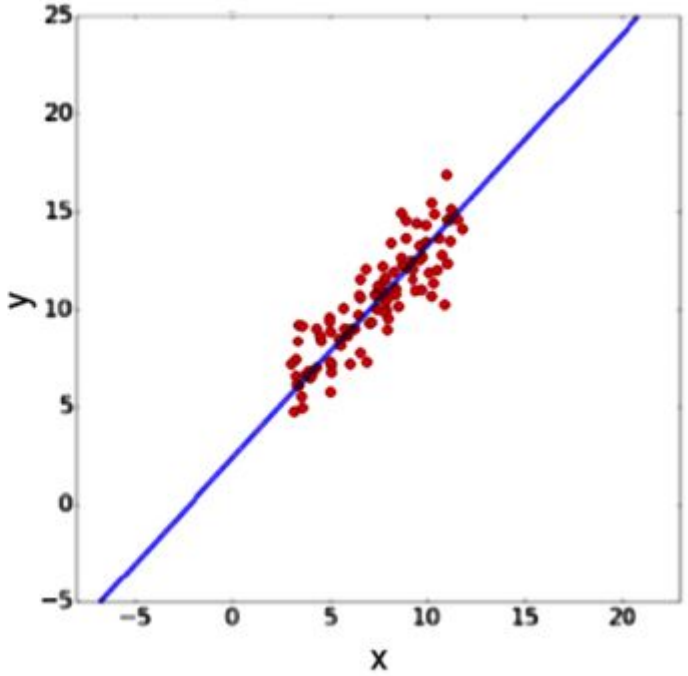
Градиентный спуск



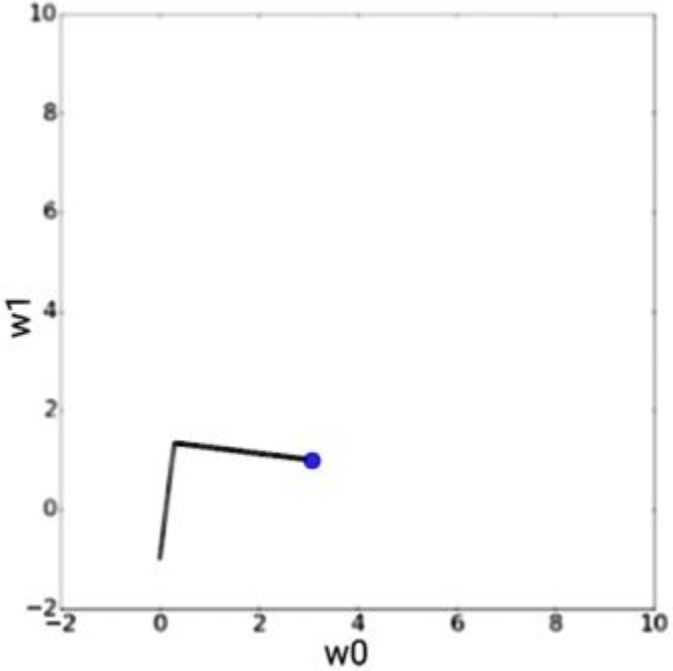


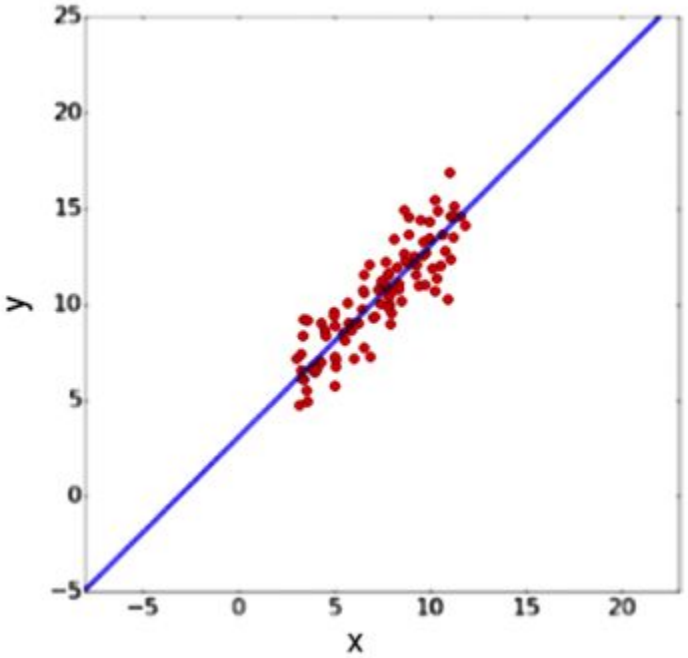
Градиентный спуск



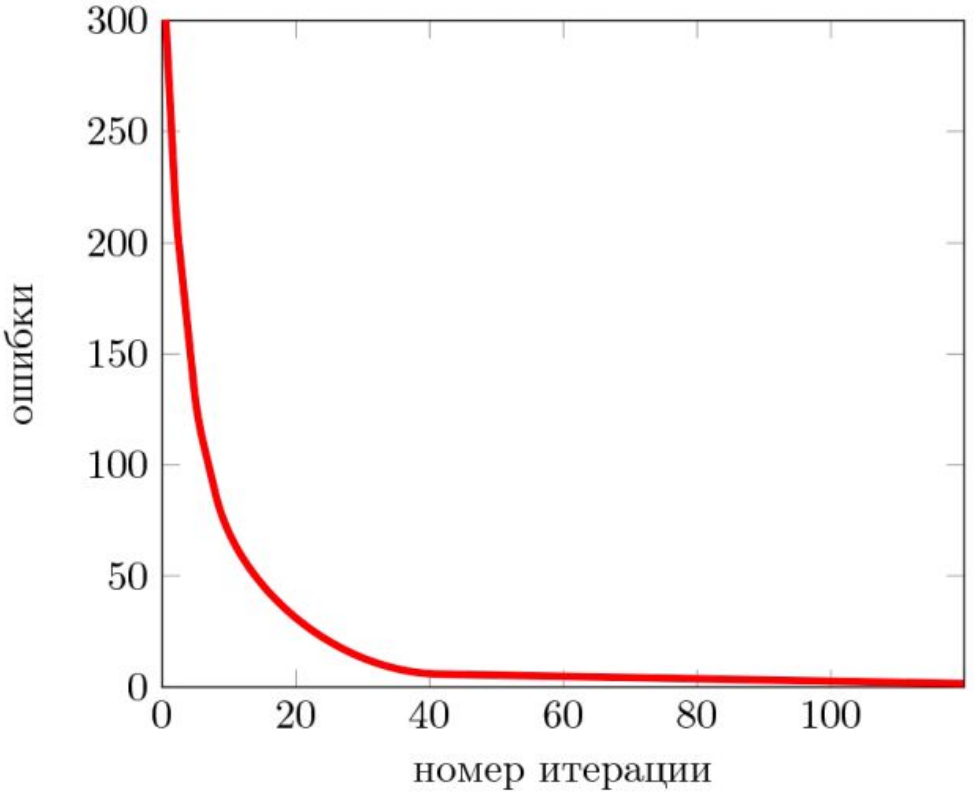


Градиентный спуск



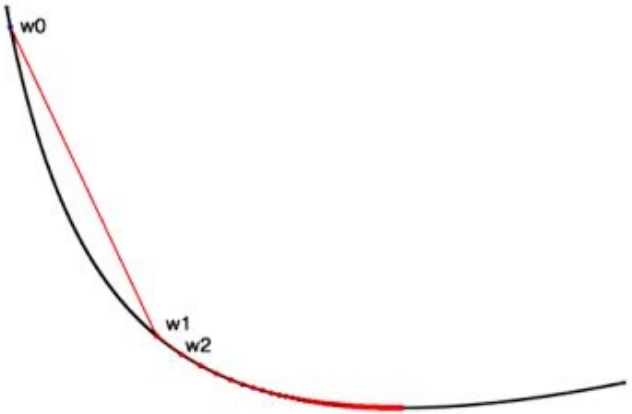


Градиентный спуск

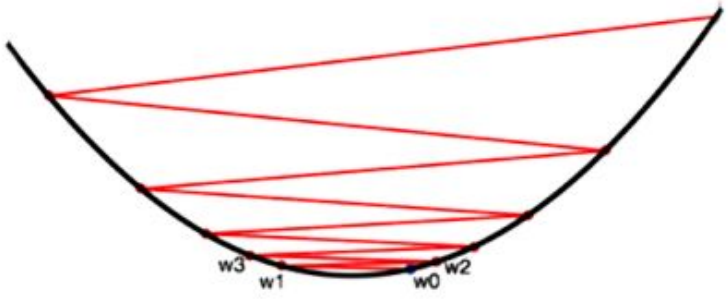


Выбор шага в градиентном спуске

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta_t \nabla Q(\mathbf{w}^{t-1}, X)$$



Маленький шаг



Большой шаг

Выбор шага в градиентном спуске

- В большинстве простейших случаев может неплохо будет следующая формула:

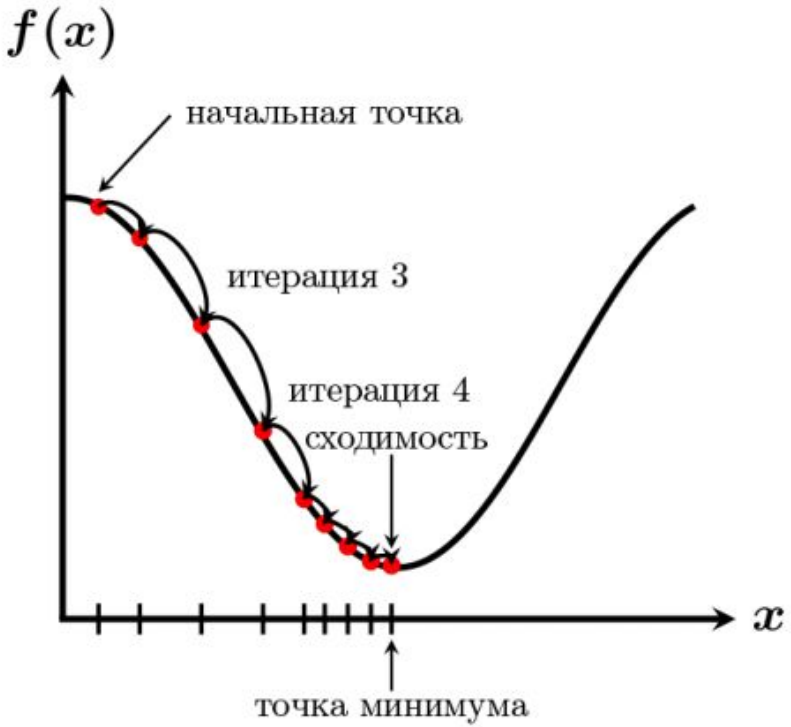
$$\eta_t = \frac{k}{t}$$

- **t** - номер итерации, а **k** - константа, которую необходимо подбирать

Выбор шага в градиентном спуске

- Гораздо лучше использовать эвристики
- Чем больше номер итерации, тем сильнее стоит уменьшать шаг

- Иногда имеет смысла наоборот повысить шаг, чтобы покинуть локальный максимум



Проблемы градиентного спуска

$$\nabla_{\mathbf{w}} Q(\mathbf{w}, X) = \frac{2}{l} X^T (X\,\mathbf{w} - y)$$

$$\frac{\partial Q}{\partial \mathbf{w}_j} = \frac{2}{\ell} \sum_{i=1}^{\ell} x_i^j (\langle \mathbf{w}, x_i \rangle - y_i)$$

Суммирование по всей выборке!

Как работает градиентный спуск

$$\nabla_{\mathbf{w}} Q(\mathbf{w}, X) = \frac{2}{l} X^T (X\,\mathbf{w} - y)$$

$$\frac{\partial Q}{\partial \mathbf{w}_j} = \frac{2}{\ell} \sum_{i=1}^{\ell} x_i^j (\langle \mathbf{w}, x_i \rangle - y_i)$$

Как поменять веса, чтобы улучшить качество на объекте x_i

Как работает градиентный спуск

$$\nabla_{\mathbf{w}} Q(\mathbf{w}, X) = \frac{2}{l} X^T (X\,\mathbf{w} - y)$$

$$\frac{\partial Q}{\partial \mathbf{w}_j} = \frac{2}{\ell} \sum_{i=1}^{\ell} x_i^j (\langle \mathbf{w}, x_i \rangle - y_i)$$

Как поменять веса, чтобы улучшить качество на всей выборке

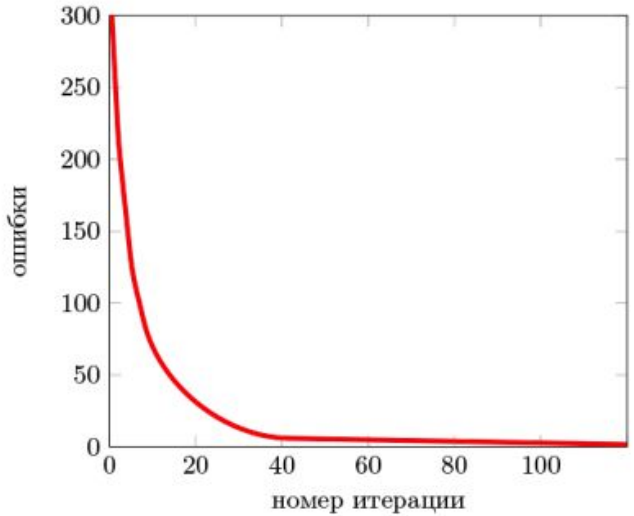
Стохастический градиентный спуск

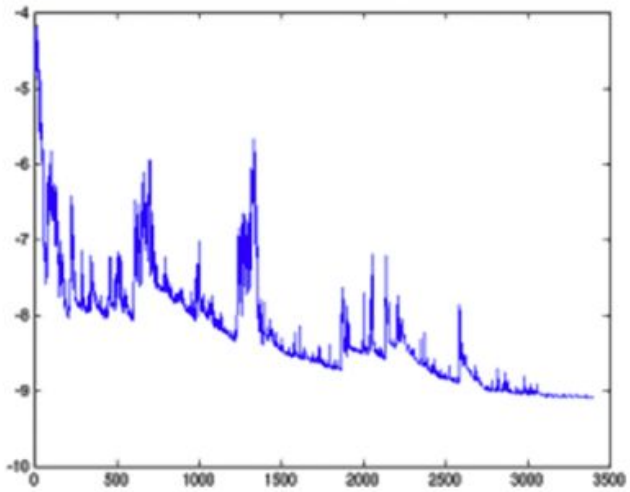
- Инициализация: $\mathbf{w}^0 = 0$
- Для итераций t=1,2,...
    - Выбрать случайный элемент x_i из X

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta_t \nabla Q(\mathbf{w}, \{x_i\})$$

- Условие остановки: $\| \mathbf{w}^t - \mathbf{w}^{t-1} \| < \varepsilon$

Стохастический градиентный спуск





Градиентный спуск

Стохастический градиентный спуск

Стохастический градиентный спуск. Преимущества

- Быстро выполняет один шаг
- Не требует хранения всего обучающего множества в памяти
- Подходит для онлайн-обучения

Mini-batch gradient descent

- Попробуем нечто среднее
- Будем каждый раз выбирать k случайных примеров из обучающей выборки и учить только на них
- Понятно, что это обобщение двух предыдущих техник:
  - Если k=1 получаем стохастический градиентный спуск
  - Если k=N, где N - размер всей обучающей выборки, то получаем классический градиентный спуск (или batch gradient descent)
- Таким образом, k - это один из гиперпараметров и его важно правильно подобрать

Mini-batch gradient descent

- Как можно строить мини-бачи?
  - Каждый раз выбирать случайное множество независимо от предыдущих выборок
    - Может использоваться, когда обучающая выборка имеет сложную структуру
  - На каждой эпохе (полному проходу по всей обучающей выборке) делать случайное разбиение на мини-бачи
    - Каждый пример будет подан на обучение ровно один раз

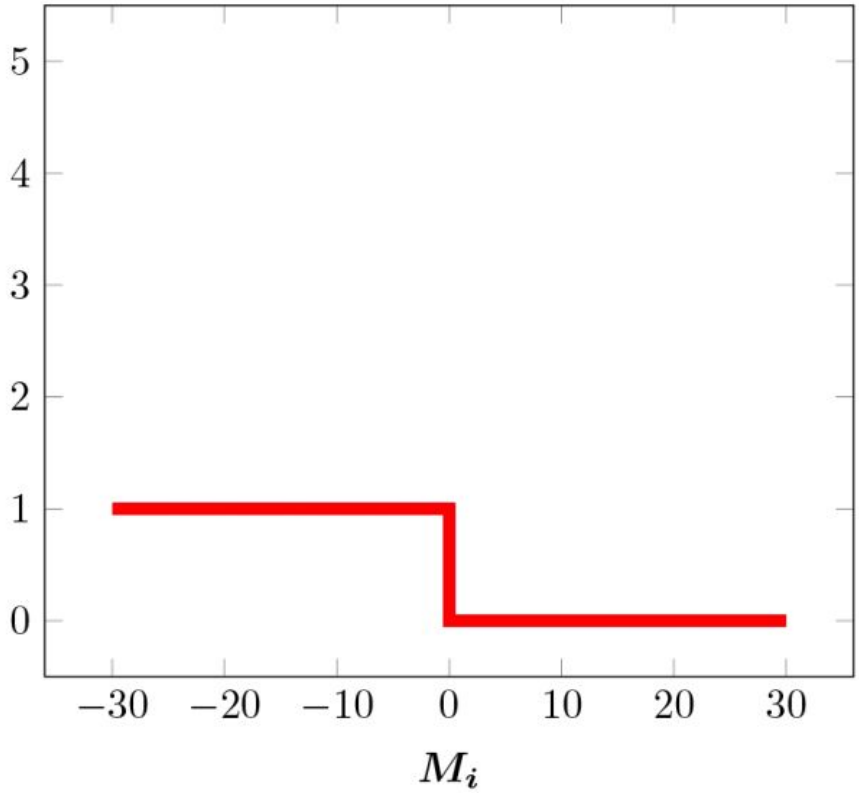
Линейная классификация

- Доля неправильных ответов в качестве метрики

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) \neq y_i]$$

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\underbrace{y_i \langle \mathbf{w}, x_i \rangle}_{M_i} < 0]$$

Линейная классификация



Линейная классификация

- Функция потерь разрывная
- Функция потерь негладкая
- Нельзя использовать методы гладкой оптимизации

Линейная классификация

- Возьмём некоторую гладкую верхнюю оценку функции потерь:

$$[M < 0] \leq \tilde{L}(M)$$

- Тогда получим новый функционал ошибки:

$$Q(a, X) \leq \tilde{Q}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \tilde{L}(M_i)$$

Линейная классификация

$$Q(a, X) \leq \tilde{Q}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \tilde{L}(M_i) \to \min_a$$

Минимизируем верхнюю оценку

Надеемся, что доля ошибок тоже уменьшится

Линейная классификация

- Логистическая

$$\tilde{L}(M) = \ln(1 + \exp(-M))$$

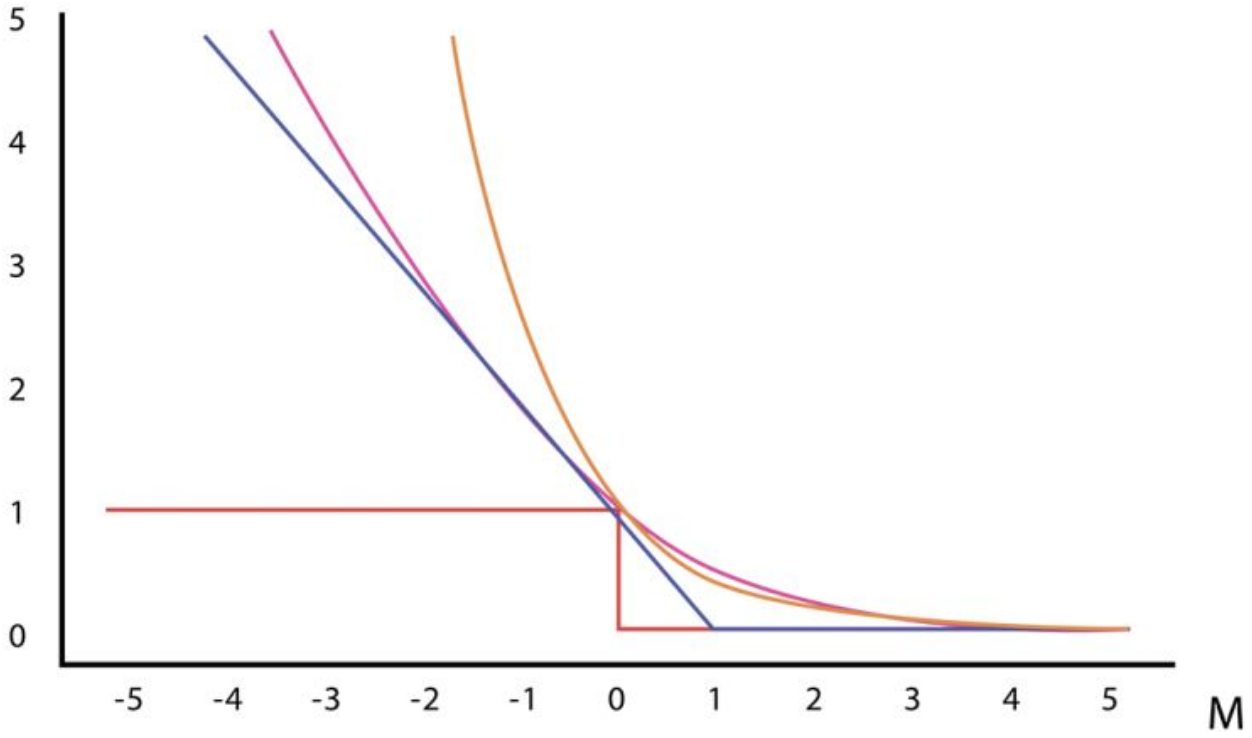
- Экспоненциальная:

$$\tilde{L}(M) = \exp(-M)$$

- Кусочно-линейная:

$$\tilde{L}(M) = \max(0, 1 - M)$$

Линейная классификация

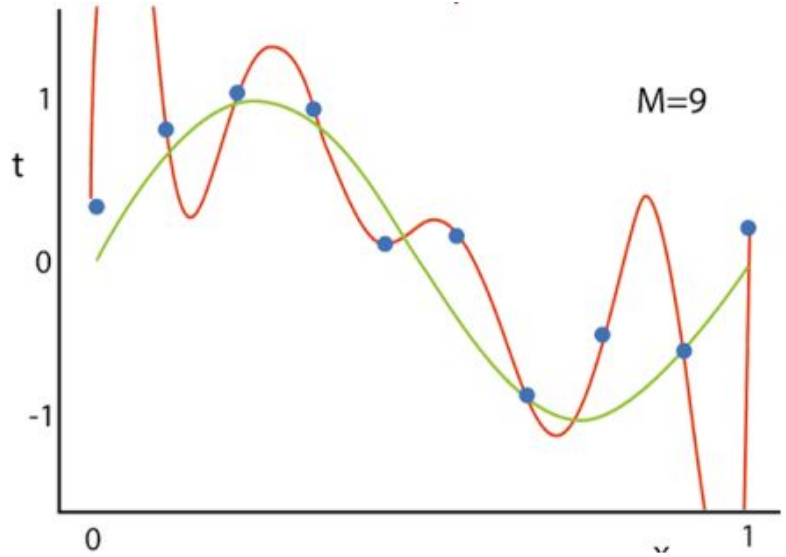


Логистическая регрессия

$$\tilde{Q}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \ln\left(1 + \exp\left(-M_i\right)\right)$$

$$\tilde{Q}(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} \ln\left(1 + \exp\left(-y_i \langle \mathbf{w}, x_i \rangle\right)\right)$$

Переобучение линейной регрессии



$$a(x) = 0.5 + 13458922x + \\ + 43983740x^2 + \ldots + 2740x^9$$

Мультиколлинеарность признаков

- Вектор признаков является мультиколлинеарным, если существует такой вектор что:

$$\alpha_1 x_i^1 + \ldots + \alpha_d x_i^d = 0$$

Мультиколлинеарность признаков

- Существует бесконечно много оптимальных ответов
- Многие из них имеют большие по модулю коэффициенты

Регуляризация

- Большие по модулю коэффициенты - зло
- Добавим в нашу модель штрафы за это:

$$\| \mathbf{w} \|^2 = \sum_{j=1}^{d} \mathbf{w}_j^2$$

- Получим новую функцию ошибки:

$$Q(\mathbf{w}, X) + \lambda \| \mathbf{w} \|^2 \to \min_{\mathbf{w}}$$

Регуляризация

$$Q(\mathbf{w}, X) + \lambda \| \mathbf{w} \|^2 \to \min_{\mathbf{w}}$$

- λ - коэффициент регуляризации
- Чем больше коэффициент регуляризации, тем ниже будет сложность модели
- Чем меньше коэффициент регуляризации, тем более сложная будет модель и тем выше риск переобучения
- Является одним из гипер-параметров модели

Почему регуляризация работает?

- По сути мы имеем следующую задачу:

$$\begin{cases} Q(\mathbf{w}, X) \to \min\limits_{\mathbf{w}} \\ \| \mathbf{w} \|^2 \leq C \end{cases}$$

Виды регуляризации

- L1-регуляризация
- L2-регуляризация

L2-регуляризация

- Гладкий и выпуклый
- “Жёстче” штрафует модель за сложность

L1-регуляризация

$$\| \mathbf{w} \|_1 = \sum_{j=1}^{d} | \mathbf{w}_j |$$

- Позволяет отбирать важные признаки
- Может делать некоторые веса нулевыми
- Негладкий

Линейные модели в Python

Доступны в sklearn.linear_model:

- RidgeClassifier
- SGDClassifier
- SGDRegressor
- LinearRegression
- LogisticRegression
- Lasso
- и многие другие

Полный список и описание доступны здесь:
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

K ближайших соседей (KNN)

Общая идея KNN

- Пусть у нас есть **n** объектов, каждый из которых определяется **d** вещественными признаками
- Представим каждый объект в виде точки с **d** координатами
- Получим **n** точек в пространстве **R^d**
- На этом пространстве можно использовать различные метрики для вычисления расстояния между точками (например, евклидову, косинусную или манхэттенскую)
- Таким образом, можно вычислять расстояние между объектами и, как следствие, находить ближайшие

Алгоритм ближайшего соседа

- Простейший алгоритм классификации
- Относит объект к классу **одного** ближайшего к нему объекта
- Можно выбрать метрику
- Можно построить аналогичный алгоритм регрессии (в большинстве случаев он будет крайне плох)
- Не содержит обучаемых параметров и не требует обучения
- Неустойчив к выбросам: если имеется объект находящийся в окружении объектов чужого класса, то он будет аффектить все точки близкие к нему

Алгоритм k ближайших соседей

- Для повышения устойчивости будем смотреть не на один ближайший, а сразу на k
- Будем относить наш объект к тому классу, который является классом наибольшего числа соседних объектов
- Выбор k крайне важен:
  - При k=1 получаем алгоритм ближайшего соседа
  - При k=n получаем константный алгоритм, относящий любую точку к наиболее часто встречающемуся в датасете классу
  - Обычно оптимальными значениями являются k=5..100 в зависимости от размера и качества данных

Выбор оптимального значения k

- Чаще всего на практике используется алгоритм LOO (leave-one-out, исключения объектов по одном)
- Будем последовательно удалять по одному объекту из нашего датасета и проверять правильность его классификации на основании оставшихся
- Чем больший процент объектов были корректно восстановлены, тем оптимальнее взятое значение k
- Обычно данная функция имеет четко выраженный минимум
- Если датасет очень большой, то не обязательно удалять все объекты, можно удалить только случайное подмножество. Полученные результаты должны быть репрезентативны

Борьба с неоднозначностями при выборе класса

- Представим ситуацию, когда два или более классов имеет одинаковое количество представителей среди **k** ближайших
- Если классов всего два, то решить проблему поможет использование нечётных **k**
- Гораздо более общее решение: задать веса **w_i** задающие вклад **i**-го по порядку объекта в итоговую сумму для класса. В результате выбираем класс с наибольшей суммой
- В таком случае вероятность неоднозначности гораздо ниже

Выбор весов w

- Можно взять линейную последовательность **w_i = (k + 1 - i) / k**
  - Неоднозначности всё ещё могут возникать
- Можно взять нелинейную последовательность **w_i = q ^ i**
  - **0 < q < 1**
  - Неоднозначности невозможны
  - Выбор **q** важен
  - Подбирать q можно, напрмер, с помощью LOO
- Можно взять функцию зависящую от расстояния

Метод парзеновского окна

- Будем фиксировать не число соседей, а расстояние до них
- Таким образом учитываем только соседей лежащих на расстоянии не более **h**
- **h** нужно подбирать так же как **k** ранее, например, с помощью LOO
- В качестве веса будем брать **h - r** или **(h - r) / r**, где **r** - расстояние до объекта

KNN в Python

- KNeighborsClassifier
- KNeighborsRegressor
- KDTree
- и многие другие

Полный список и описание доступны здесь:
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neighbors