Искусственный Интеллект

Лекция 3: Градиентный спуск

Мартынюк Полина Антоновна

telegram: @PAMartynyuk

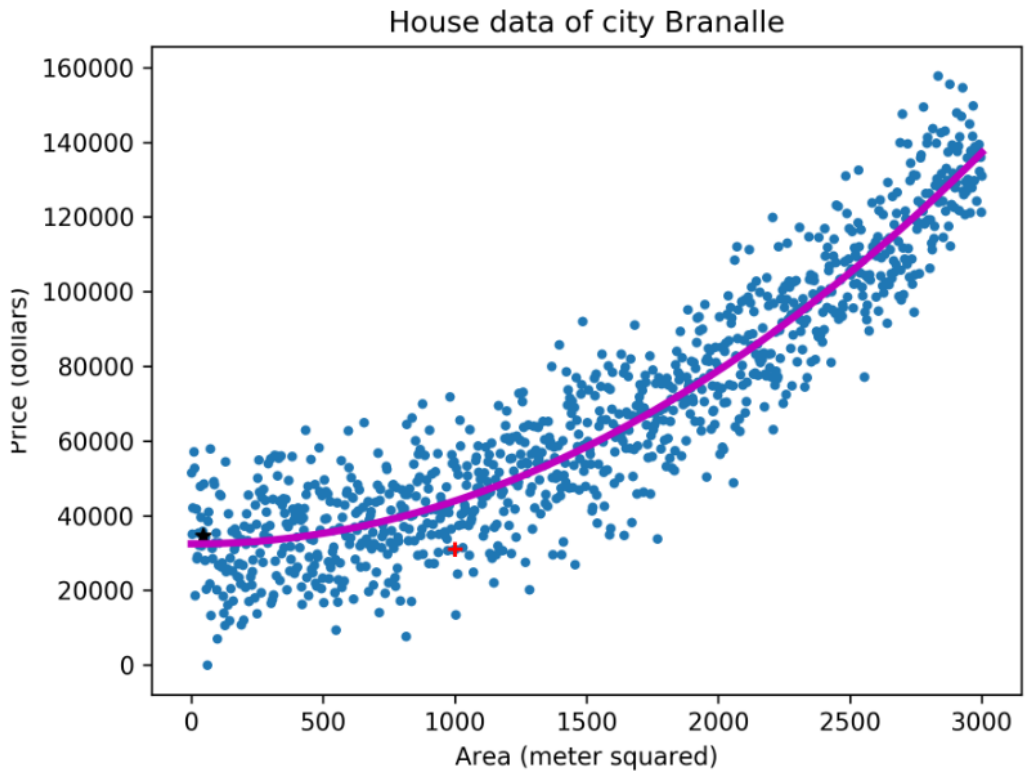
email: pa-martynyuk@yandex.ru

Регрессия

Регрессия (Regression)

- **Прогнозирование цены недвижимости** - определение стоимости домов или квартир на основе их характеристик, таких как площадь, количество комнат, район и другие факторы.

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9



Данные

Регрессия (Regression)

Формат данных

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9

Экземпляр размеченных данных

. Наблюдение (значения признаков)

. Метка/целевое значение

Данные

Регрессия (Regression)

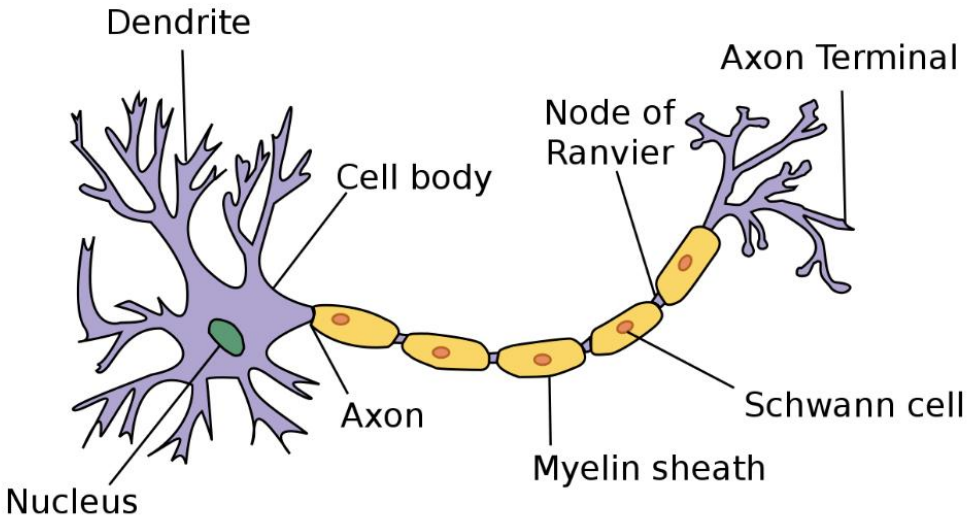
Нейрон - основная строительная единица нервной системы и основной элемент обработки информации в мозге

Дендриты: Нейрон имеет множество коротких ветвей, называемых дендритами, которые служат для приема входящих сигналов от других нейронов или рецепторов. Дендриты получают электрические сигналы и химические сигналы в виде нейротрансмиттеров от синапсов (мест контакта между нейронами).

Сома (клеточное тело): Дендриты передают сигналы в клеточное тело нейрона, где происходит интеграция входящих сигналов. Если сумма этих сигналов превышает определенный пороговый уровень, нейрон активируется.

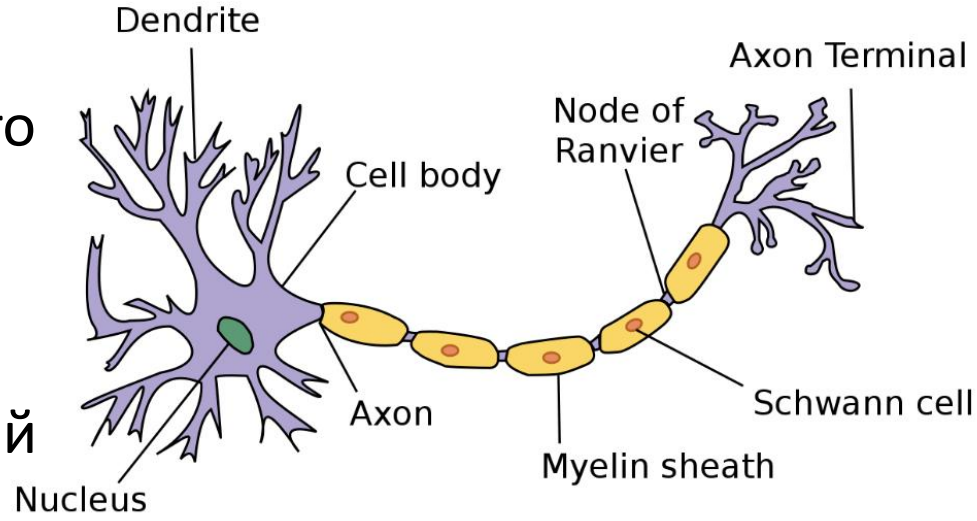
Аксон: Если нейрон активируется, сигнал передается по длинной нитевидной структуре, называемой аксоном.

Синапсы: Аксон нейрона встречается с дендритами других нейронов на местах, называемых синапсами. В синапсах сигнал передается от аксона к дендритам других нейронов с помощью химических нейротрансмиттеров. Это место передачи сигнала между нейронами.



Нейрон - основная строительная единица нервной системы и основной элемент обработки информации в мозге

Этот процесс передачи сигналов между нейронами и создает **основу для обработки информации в нервной системе.**
Нейроны работают в сети, обеспечивая обработку и передачу информации от одного участка мозга к другому и между разными частями организма.
Эта сложная сеть нейронов и синапсов формирует основу для всех функций нервной системы, включая мышление, чувствительность, движение и другие процессы.



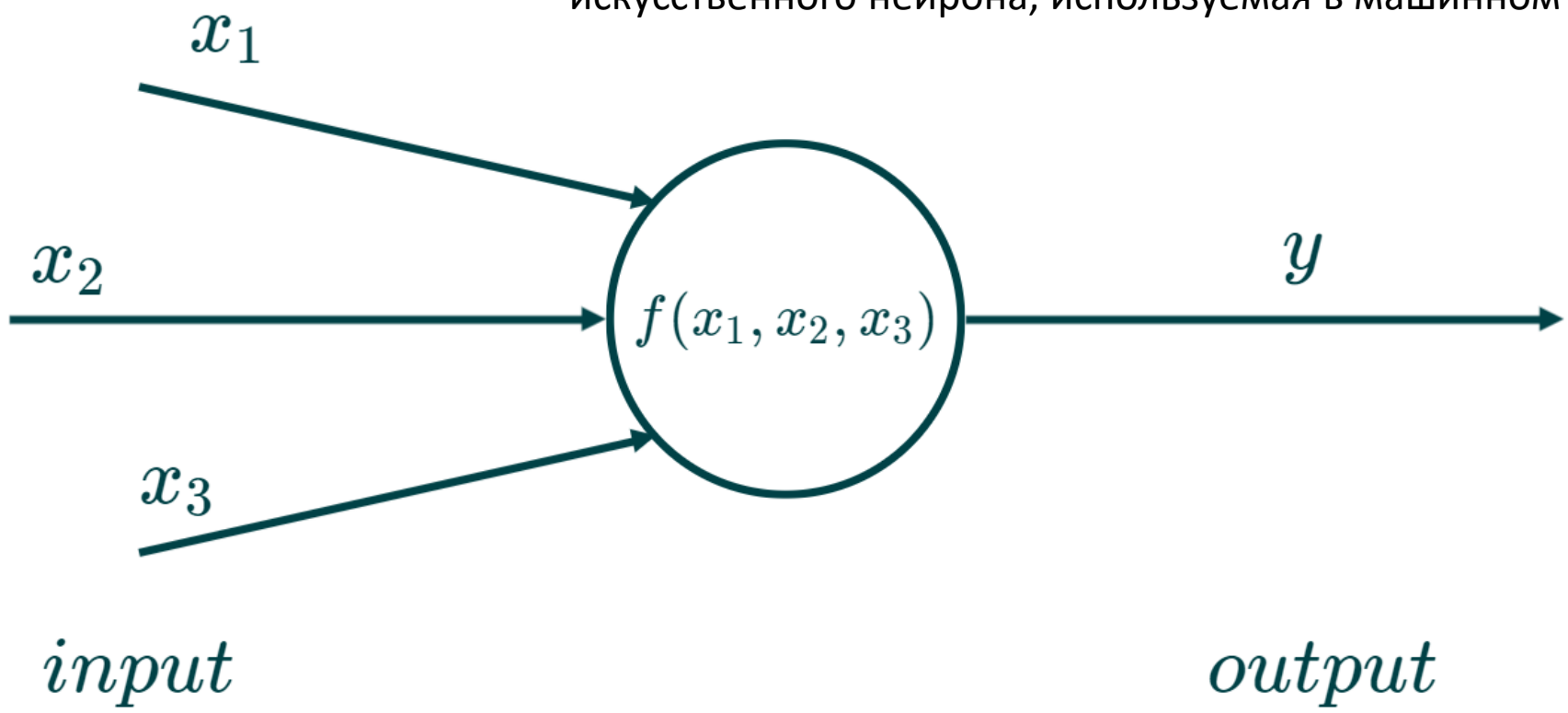
Перцептрон

Перцептро́н (или *персептрон*(англ. *perceptron* от лат. *perceptio* — восприятие) — математическая или компьютерная модель восприятия информации мозгом, предложенная Фрэнком Розенблаттом в 1958 году и впервые реализованная в виде электронной машины «Марк-1» в 1960 году.

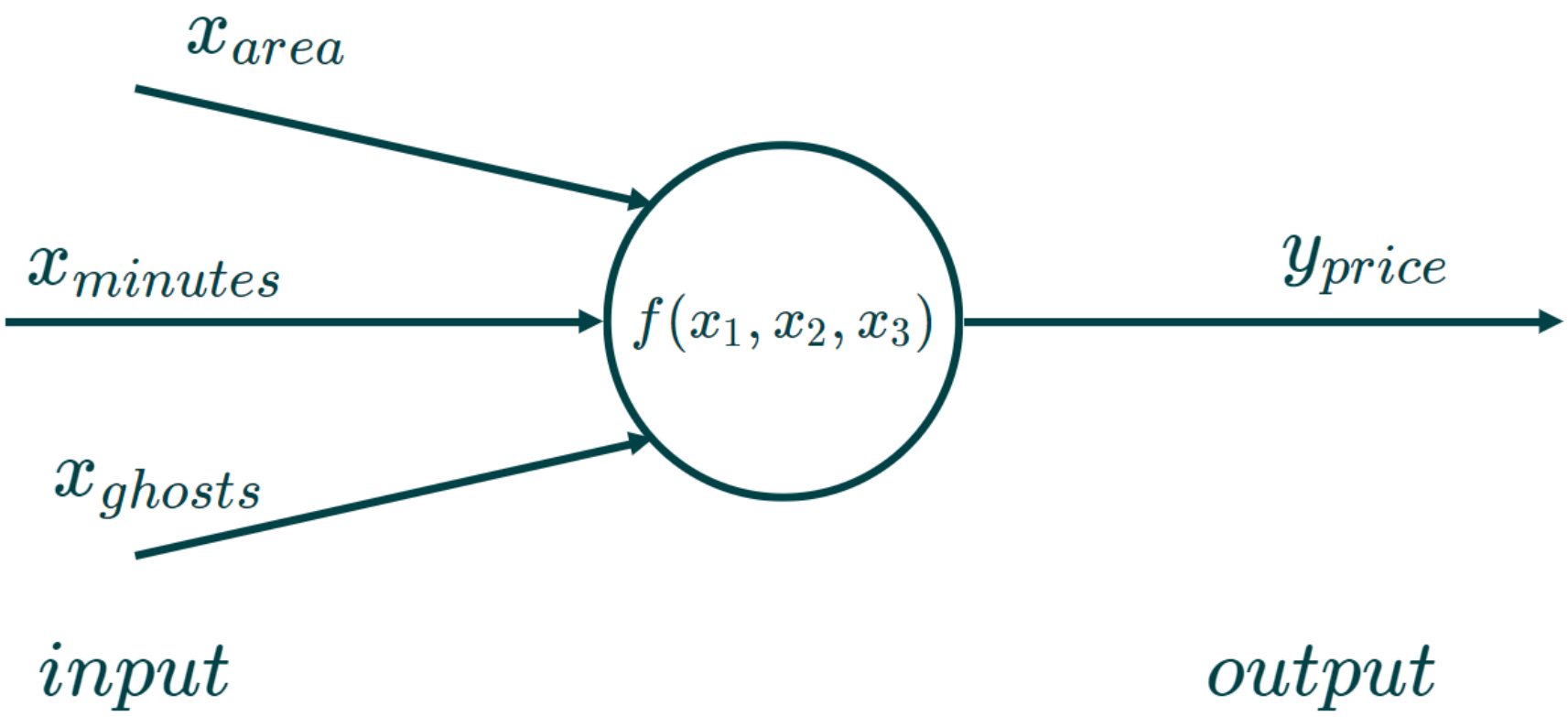
Перцептрон стал одной из первых моделей **нейросетей**, а «Марк-1» — первым в мире **нейрокомпьютером**, способным распознавать некоторые буквы английского алфавита.

Однослойный перцептрон

Однослойный перцептрон - это простая модель нейрона или искусственного нейрона, используемая в машинном обучении



Однослойный перцептрон



Однослойный перцептрон

Как смоделировать функцию f(x)?

Интуитивный подход – всё сложить!

$$x_1 + x_2 + x_3 = y$$

Однослойный перцептрон

НО: степень значимости признаков – разная!

Решение: **веса** признаков

$$w_1 * x_1 + w_2 * x_2 + w_3 * x_3 = y$$

Однослойный перцептрон

Каждому признаку – свой вес:

$$w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y$$

Однослойный перцептрон

Чем важнее признак, тем больше его вес:

$$\uparrow w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y \uparrow$$

Если признак негативно влияет на результат, его вес отрицателен:

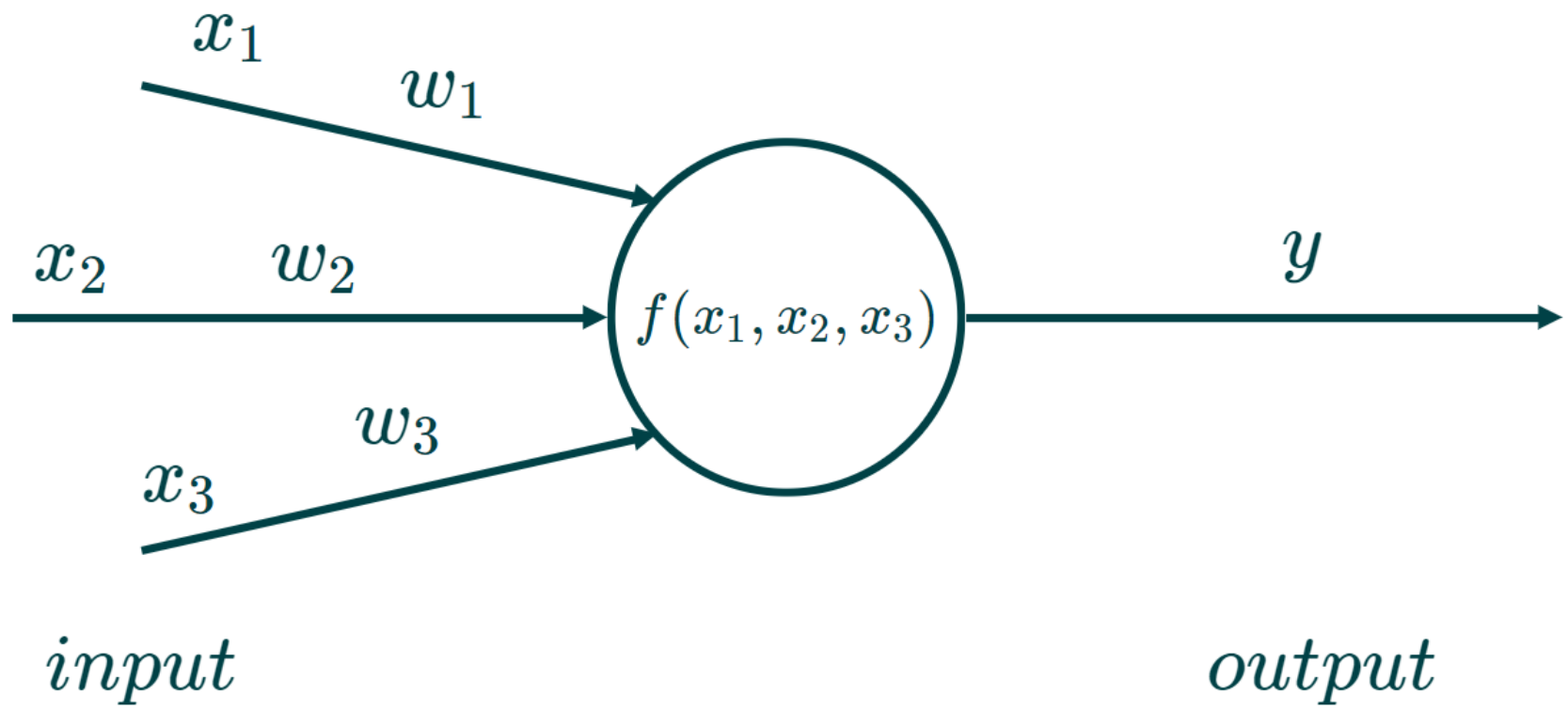
$$w_{area} * x_{area} + \downarrow w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y \downarrow$$

Если признак почти не влияет на результат, его вес близок к нулю:

$$w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y$$

Однослойный перцептрон

Скорректируем картинку, добавив веса



Однослойный перцептрон

$$y = \sum_{i=1}^{n} w_i \cdot x_i + b$$

- y - выход перцептрона.
- n - количество входных признаков (входных сигналов).
- w_i - вес (количество важности) для i-го входа (признака).
- x_i - значение i-го входа (признака).
- b - порог (bias) или смещение, которое добавляется к сумме взвешенных входов.

Однослойный перцептрон

$$y = \sum_{i=1}^{n} w_i \cdot x_i + b$$

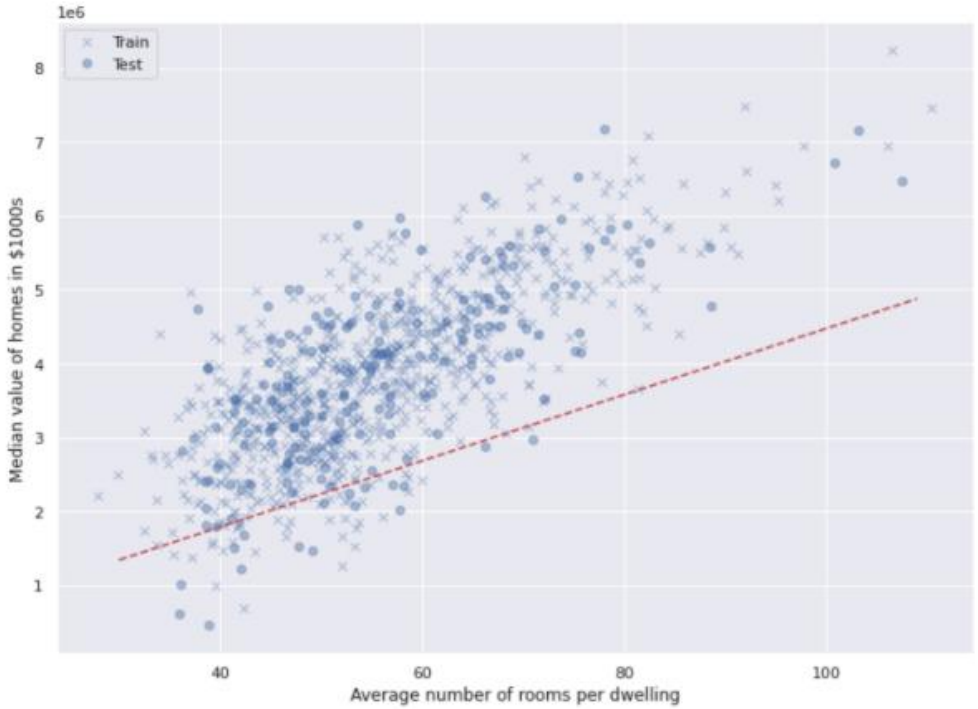
Смещение *b* позволяет контролировать, когда нейрон будет активирован.

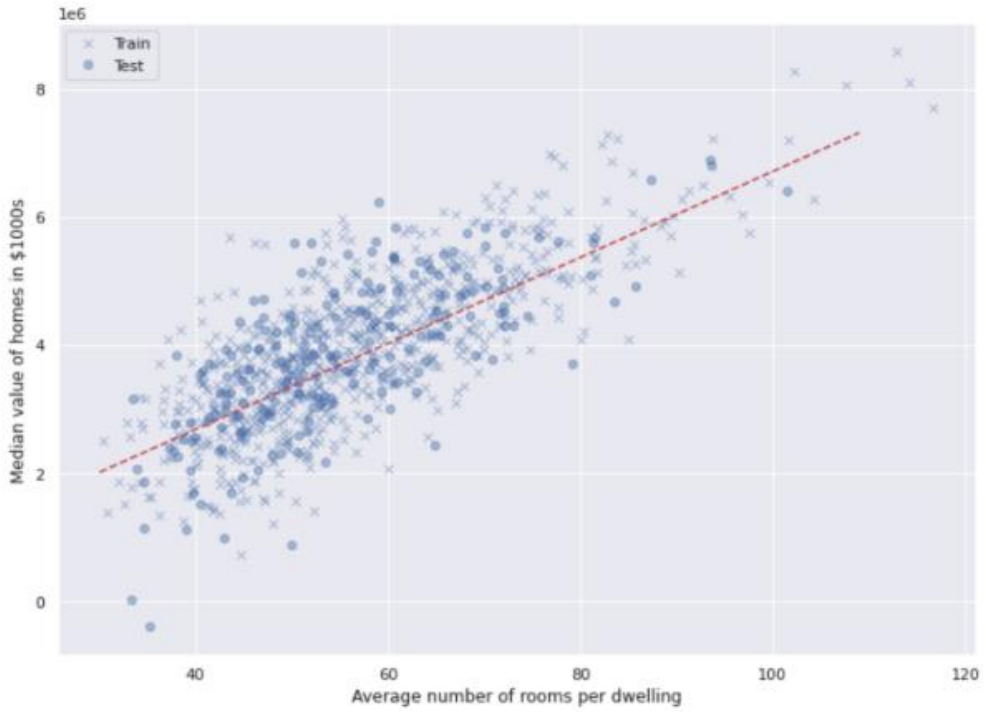
Оно действует как пороговое значение, и если сумма взвешенных входов превышает это смещение, то нейрон активируется, иначе он остается неактивным.

Смещение позволяет нейронам настраивать свою активацию и принимать решения на основе различных входных условий.

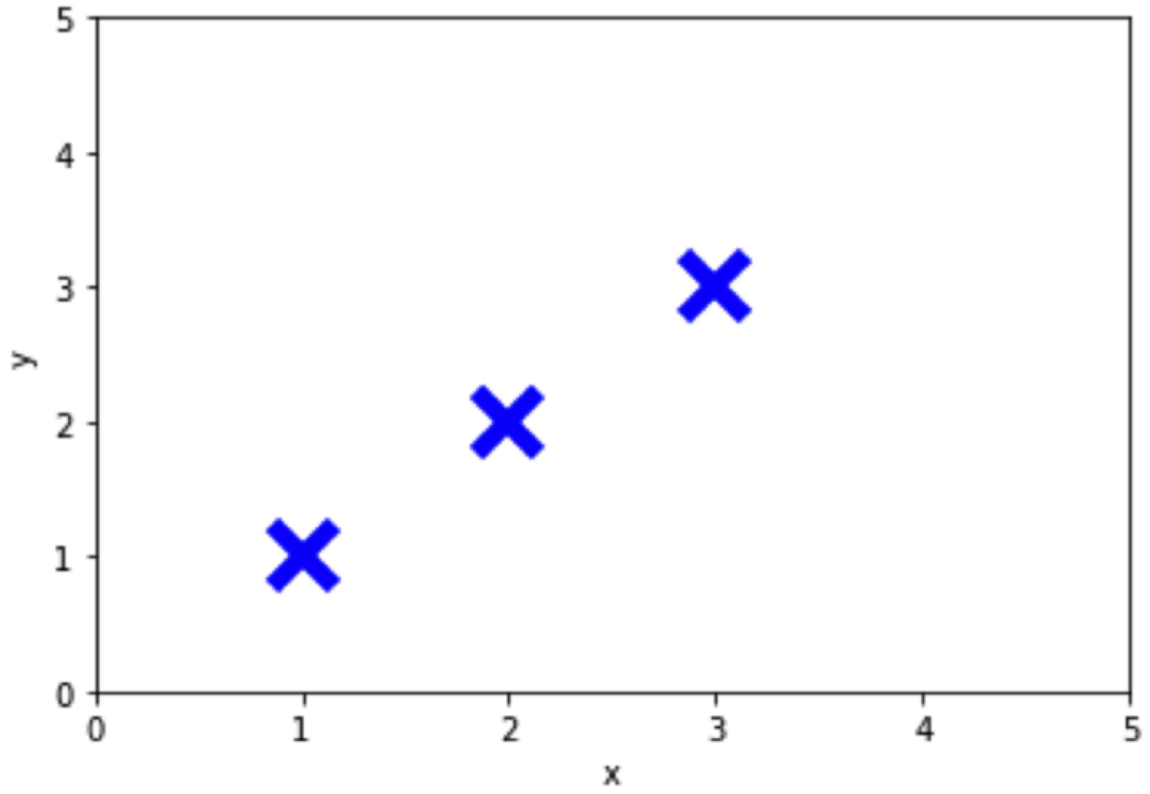
Функция ошибки

Модель нужно как-то оценить



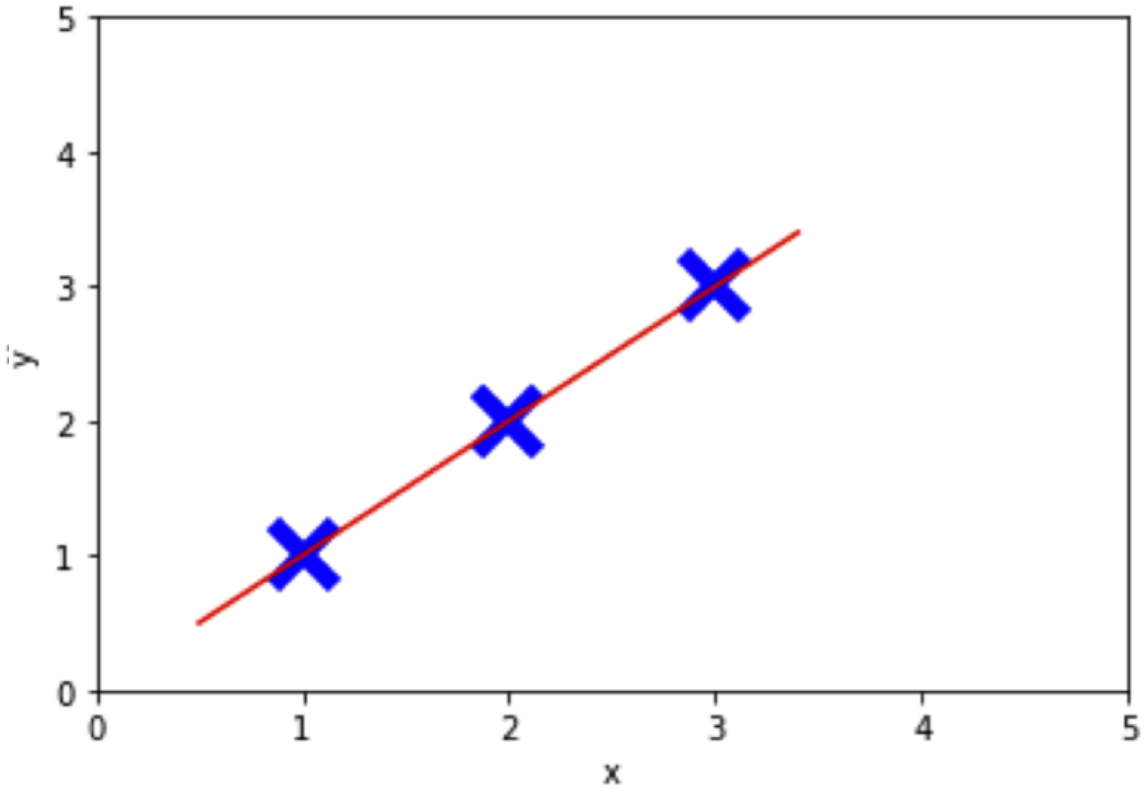


Функция ошибки



$$\hat{y} = wx + b$$

Функция ошибки



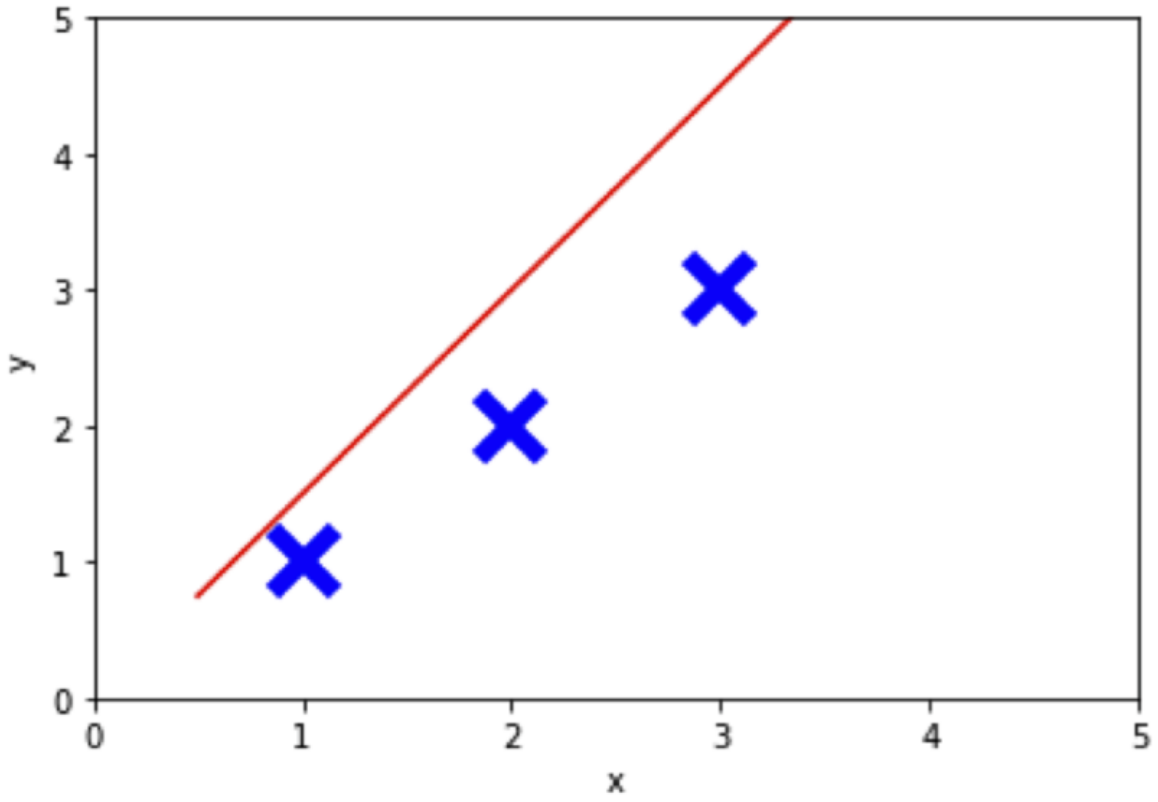
$$\hat{y} = wx + b$$

$$w = 1$$

$$b = 0$$

$$L(y, \hat{y}) = 0$$

Функция ошибки



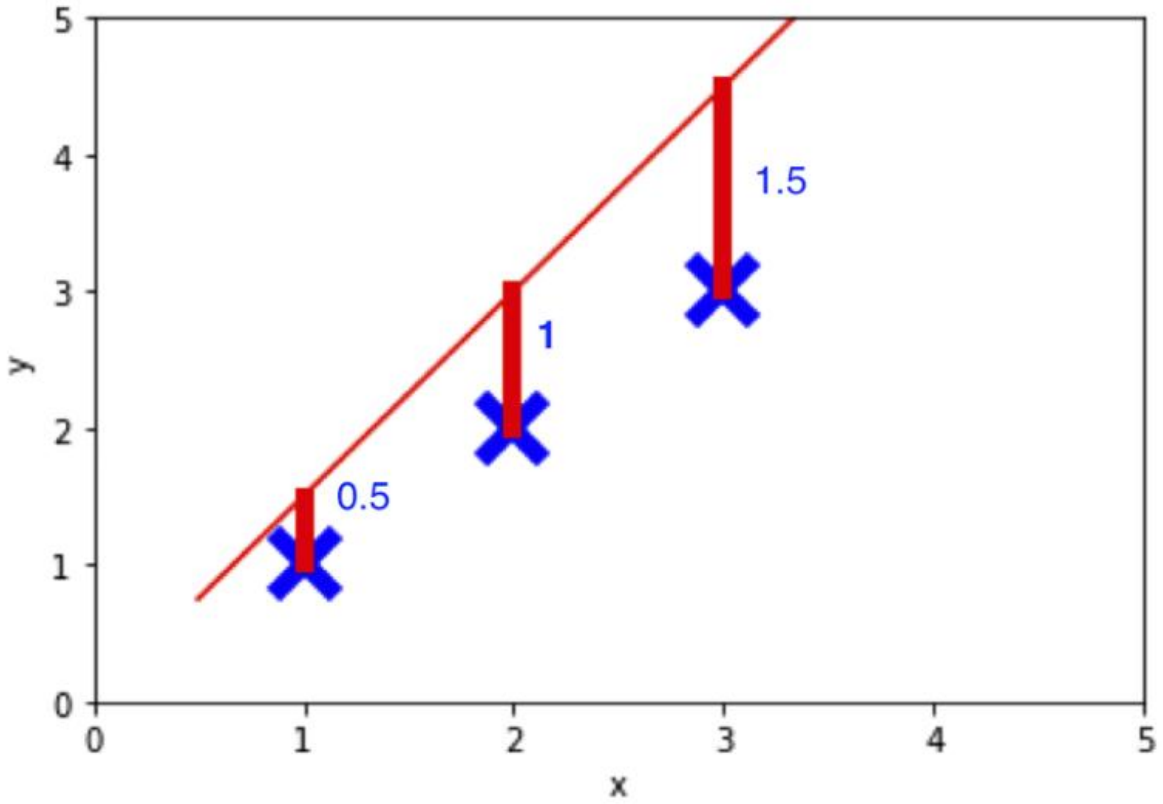
$$\hat{y} = wx + b$$

$$w = 1.5$$

$$b = 0$$

$$L(y, \hat{y}) = 0$$

Функция ошибки



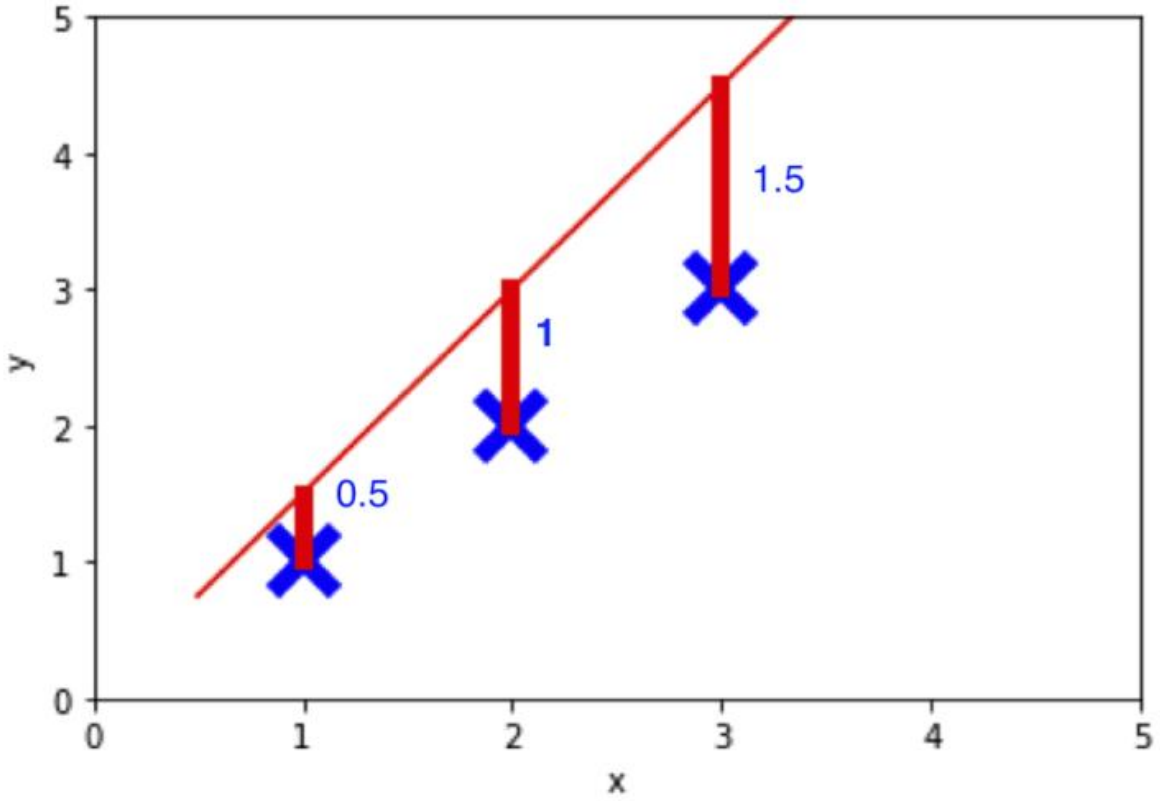
$$\hat{y} = wx + b$$

$$w = 1.5$$

$$b = 0$$

$$L(y, \hat{y}) = \hat{y} - y$$

Функция ошибки



$$\hat{y} = wx + b$$

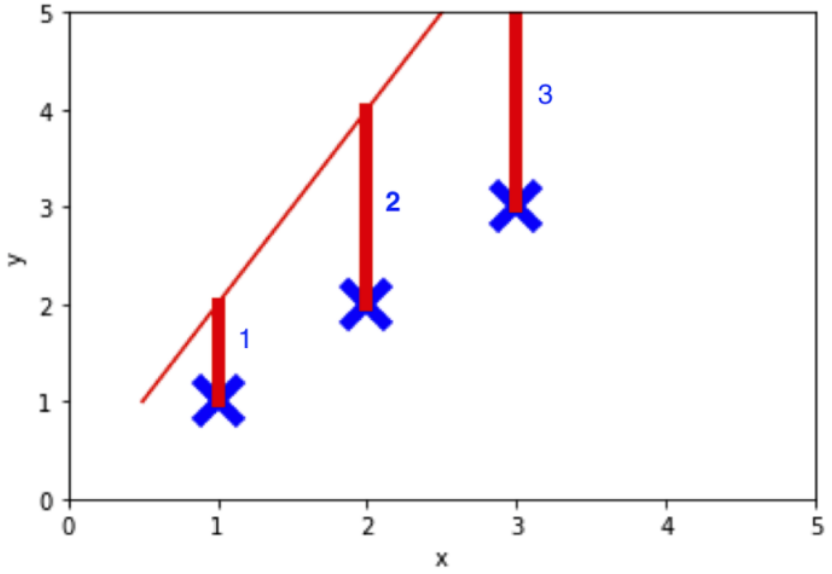
$$w = 1.5$$

$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

$$L(y, \hat{y}) = 3$$

Функция ошибки



$$\hat{y} = wx + b$$

$$w = 2$$

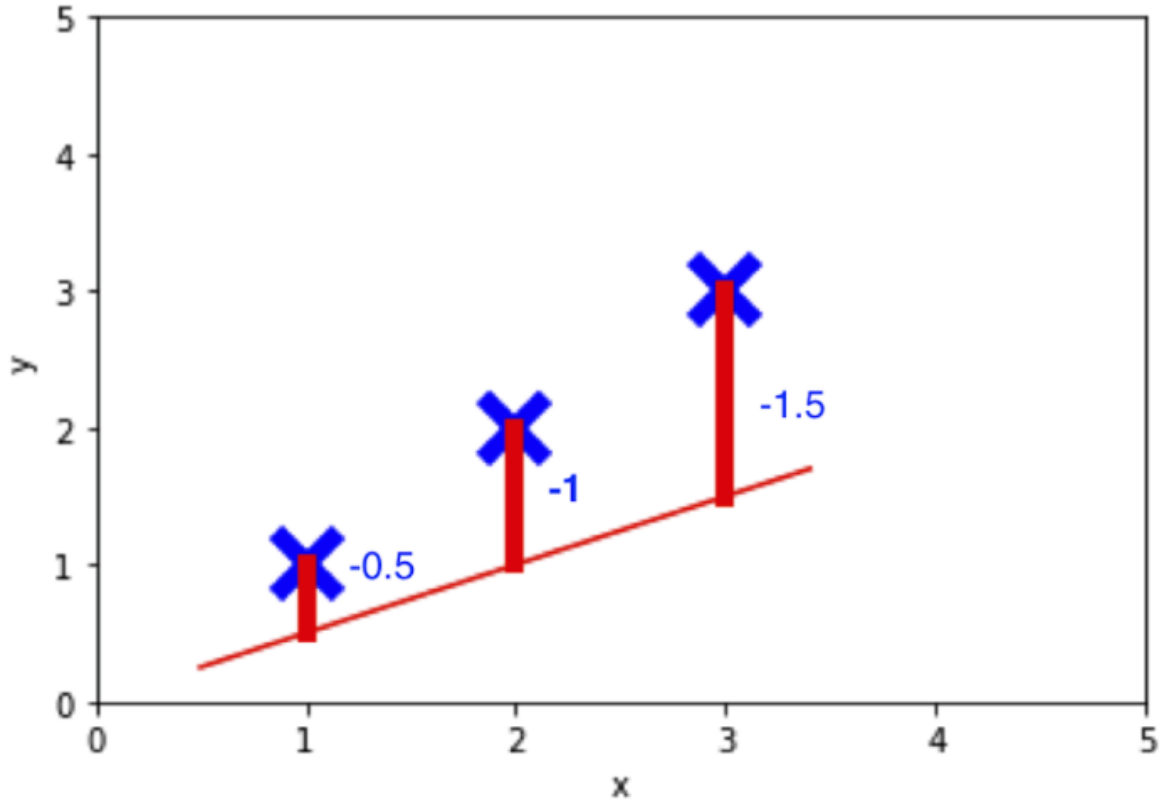
$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

$$L(y, \hat{y}) = 6$$

X_1	Y	Y_pred
1	1	2
2	2	4
3	3	6

Функция ошибки



$$\hat{y} = wx + b$$

$$w = 0.5$$

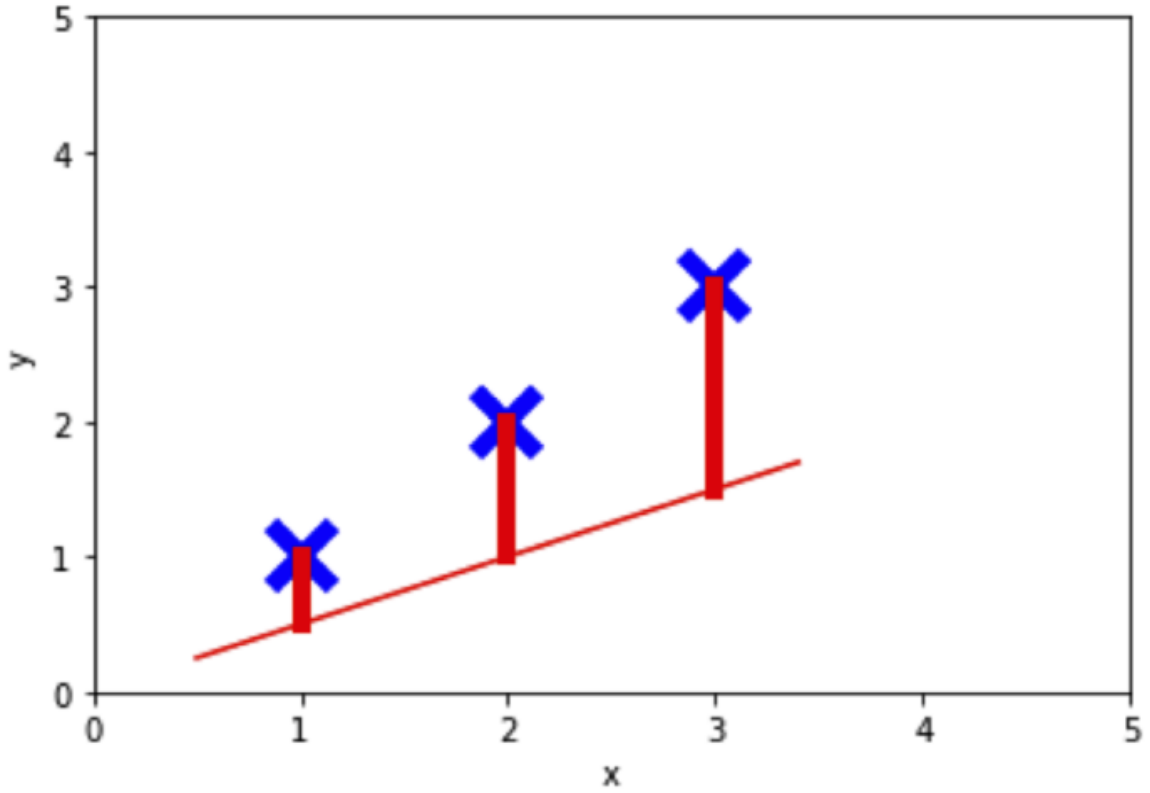
$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

$$L(y, \hat{y}) = -3$$

Нелогично

Функция ошибки



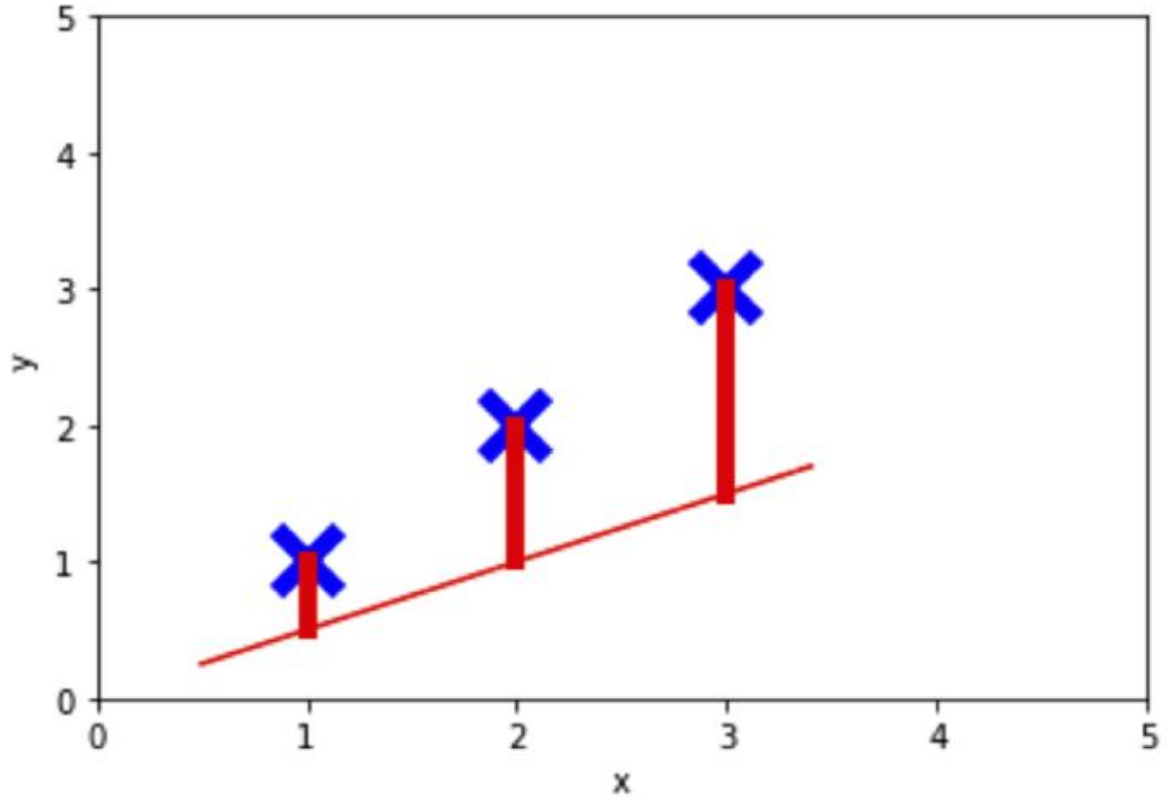
$$w = 2$$

$$b = 0$$

$$L(y, \hat{y}) = \sum (\hat{y} - y)^2$$

$$L(y, \hat{y}) = 3.5$$

Функция ошибки



$$w = 2$$

$$b = 0$$

$$L(y, \hat{y}) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$L(y, \hat{y}) = 1.17$$

Среднеквадратичная ошибка

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

- MSE - среднеквадратичная ошибка.
- n - количество наблюдений (или точек данных) в наборе данных.
- y_i - фактическое значение для i-го наблюдения.
- $\hat{y}_i$ - предсказанное значение для i-го наблюдения.

Формула для **среднеквадратичной ошибки** (Mean Squared Error, MSE) используется для измерения средней квадратичной разницы между фактическими и предсказанными значениями.
Она широко применяется в задачах регрессии для оценки качества модели.

Особенность функции ошибки

Функция ошибки / Функция потерь (Loss function)

Зачастую функция ошибки не имеет никакого смысла сама по себе;

Её предназначение - **сравнивать модели друг с другом**.

Поиск параметров - весов

Обучение - процесс автоматической настройки параметров функции.
Цель – добиться **минимизации** Функции ошибки.
Перебор всех весов - плохая идея (слишком большое количество комбинаций).

Минимизация функции потерь

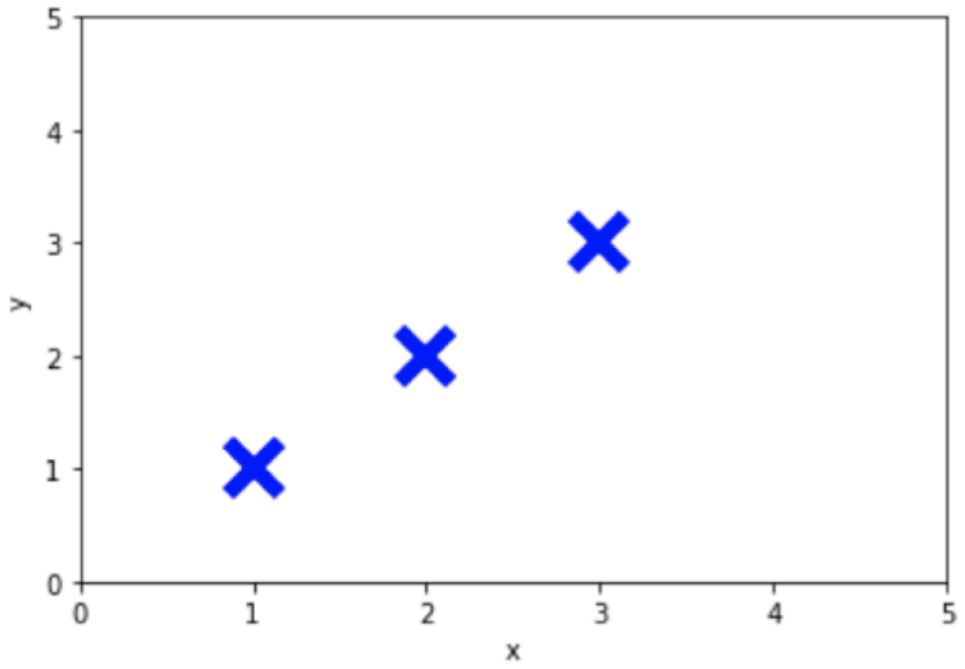
Задача оптимизации: $\arg\min_w L(f(x), y)$

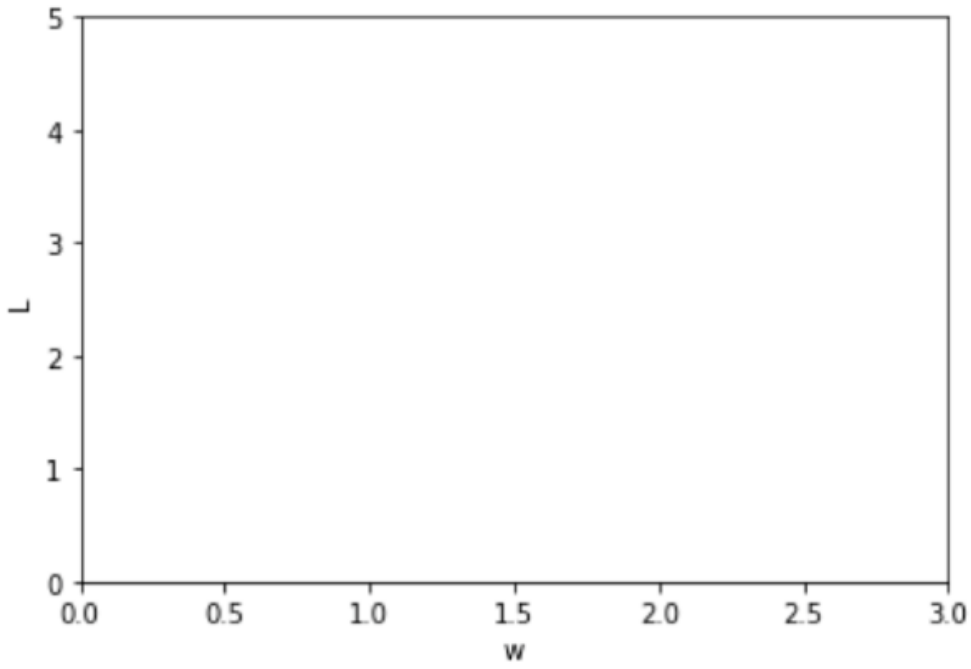
$$\min_w L(w_1, ..., w_n)$$

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=0}^{m} (y_i - \hat{y}_i)^2$$

$$L(w_1, ..., w_n) = \frac{1}{m} \sum_{i=0}^{m} (y_i - (x_1 w_1 + ... + x_n w_n)))^2$$

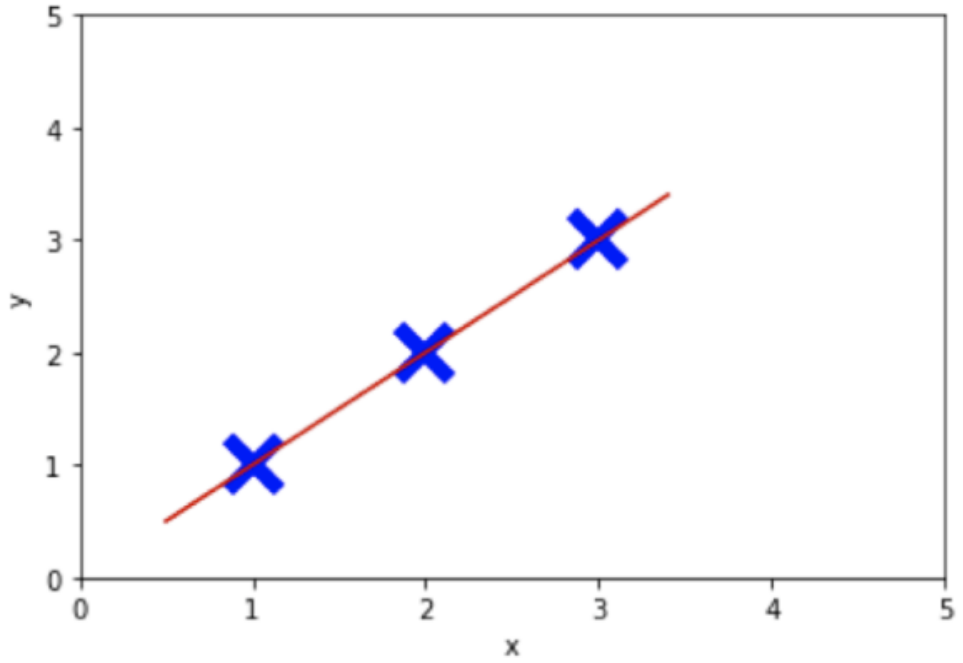
Визуализация ошибок при разных весах

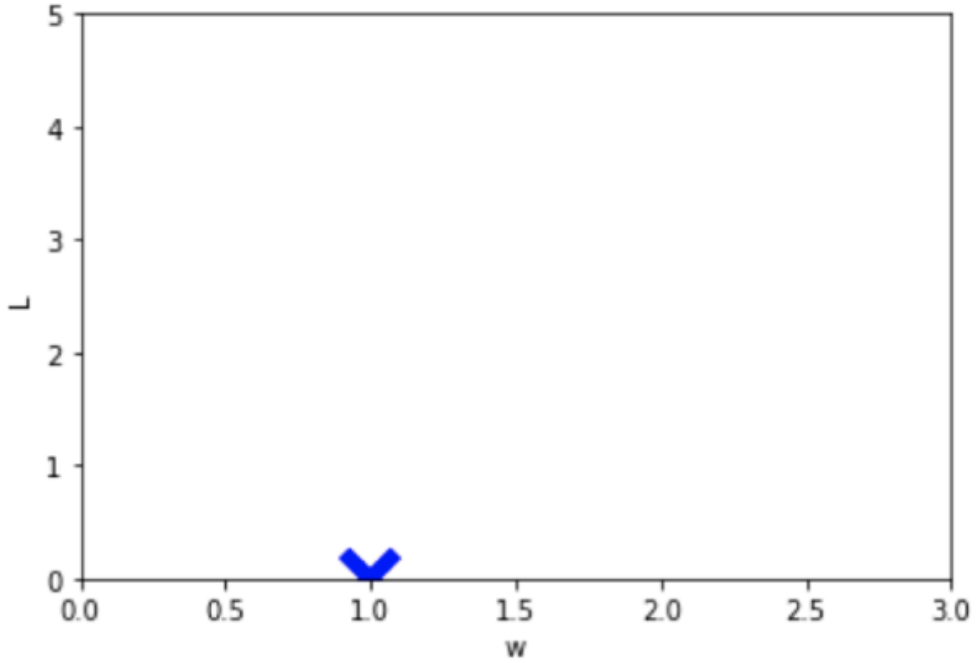




Визуализация ошибок при разных весах

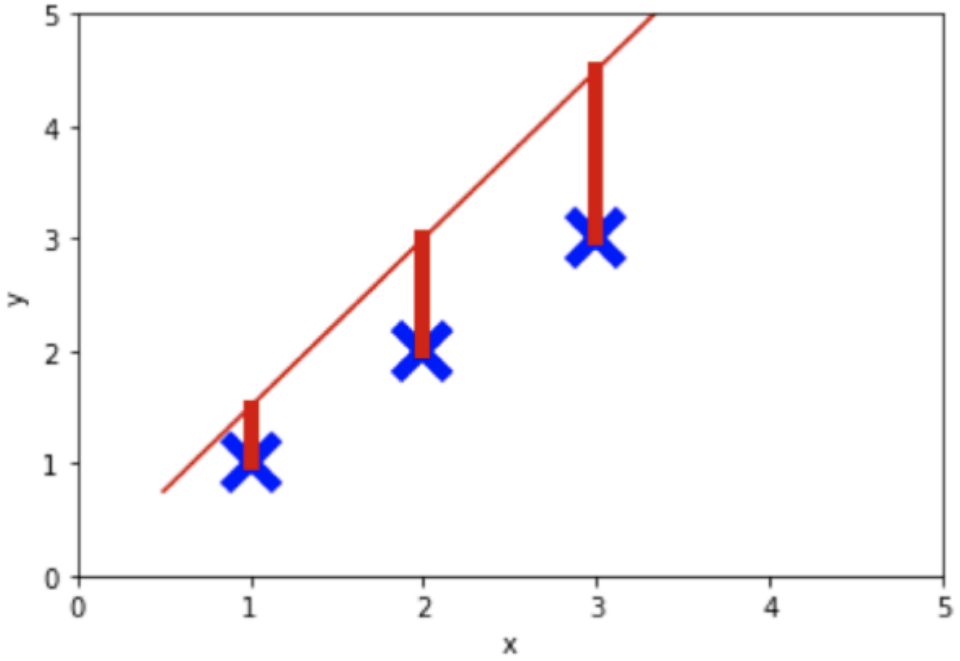
w = 1,0

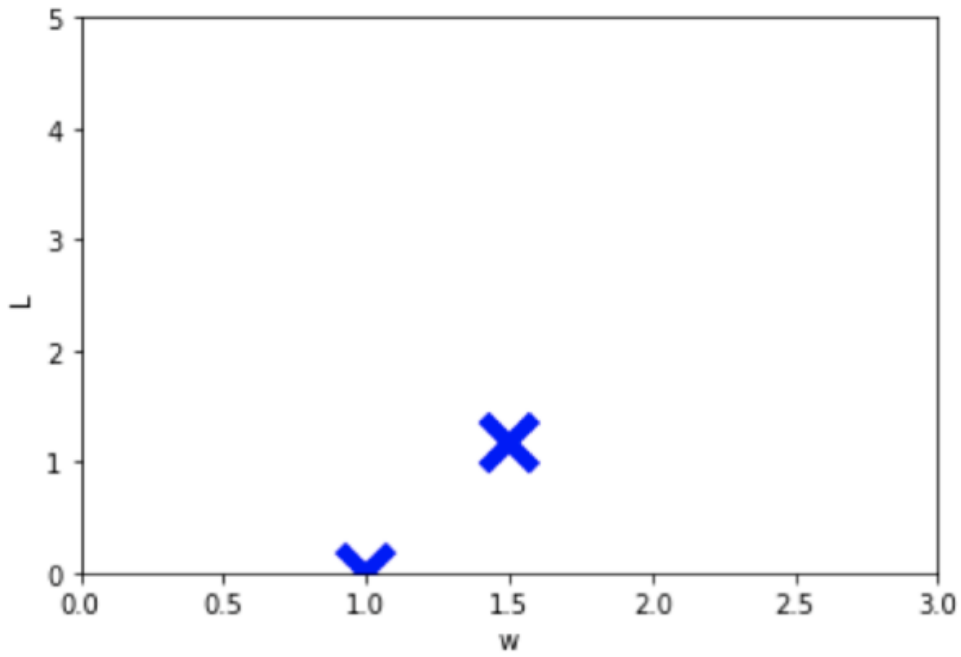




Визуализация ошибок при разных весах

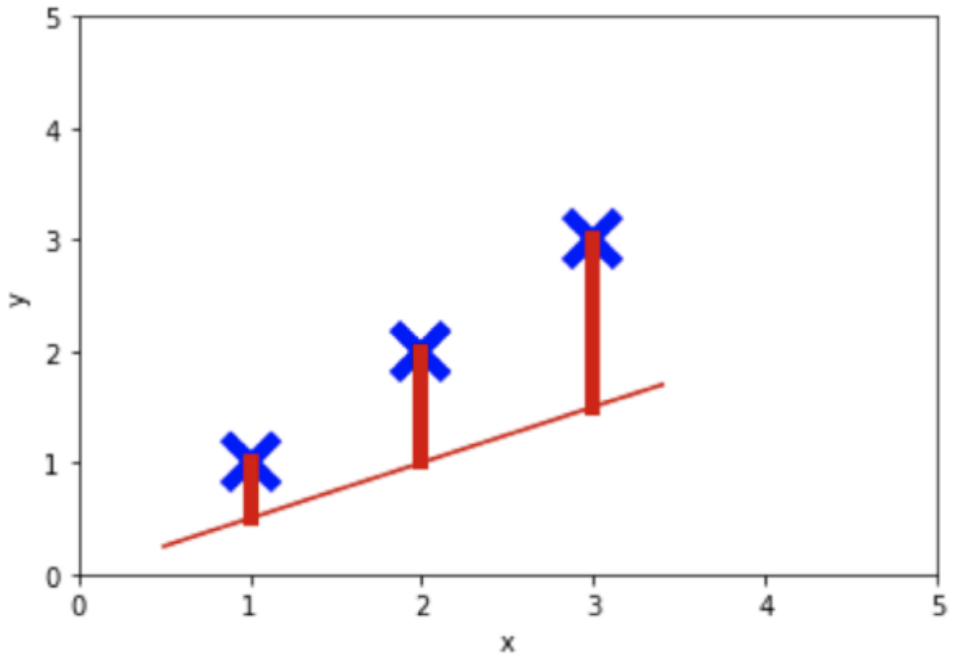
w = 1,5

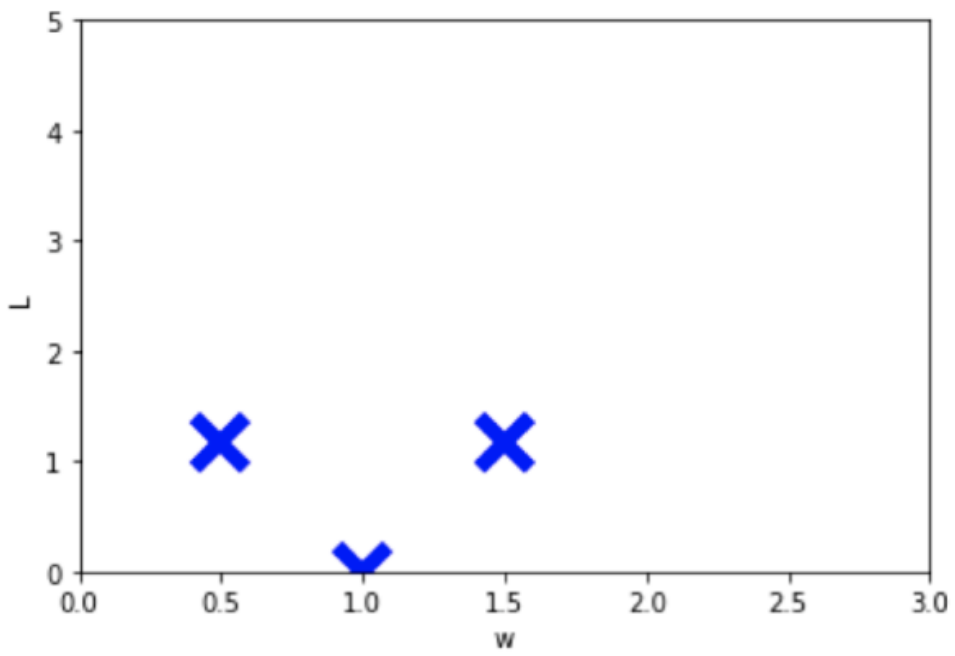




Визуализация ошибок при разных весах

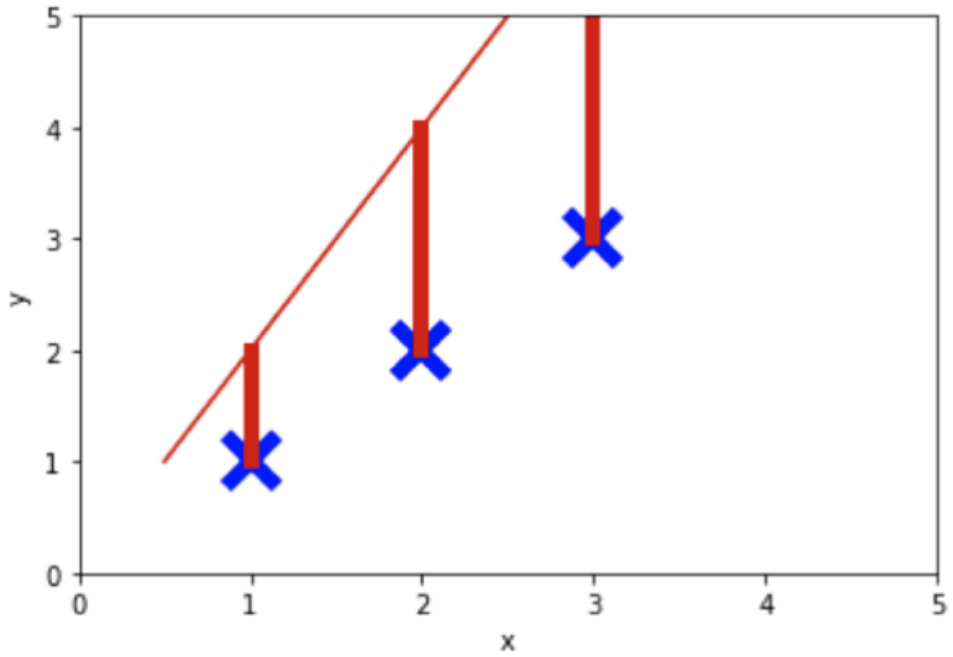
w = 0,5

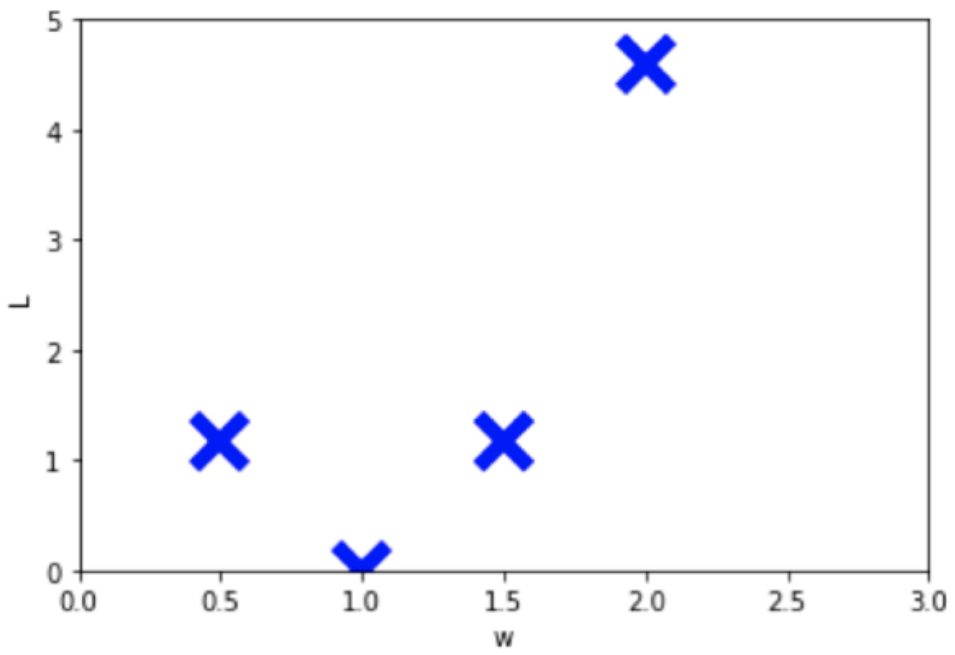




Визуализация ошибок при разных весах

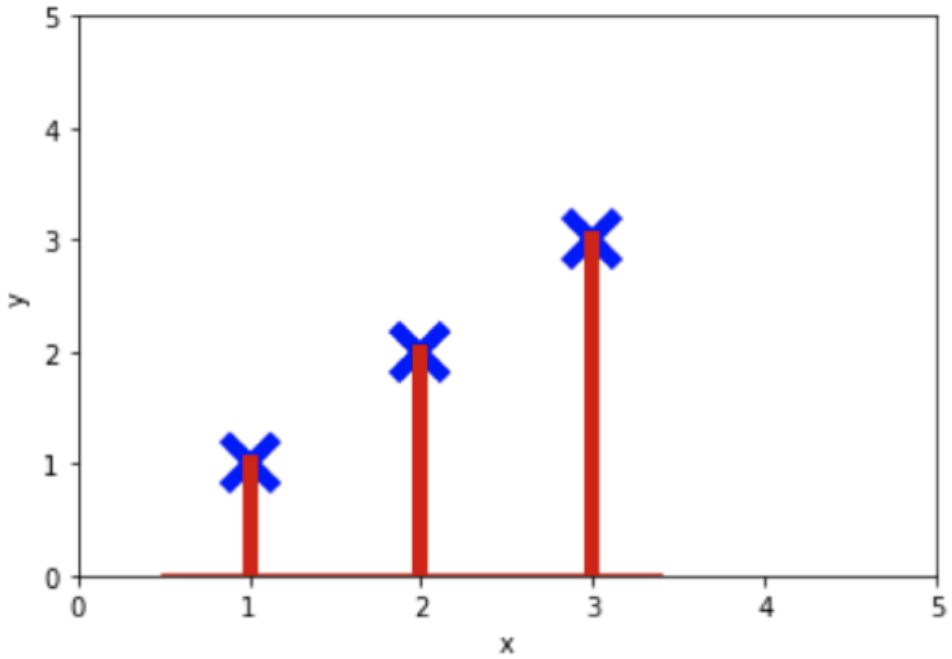
w = 2,0

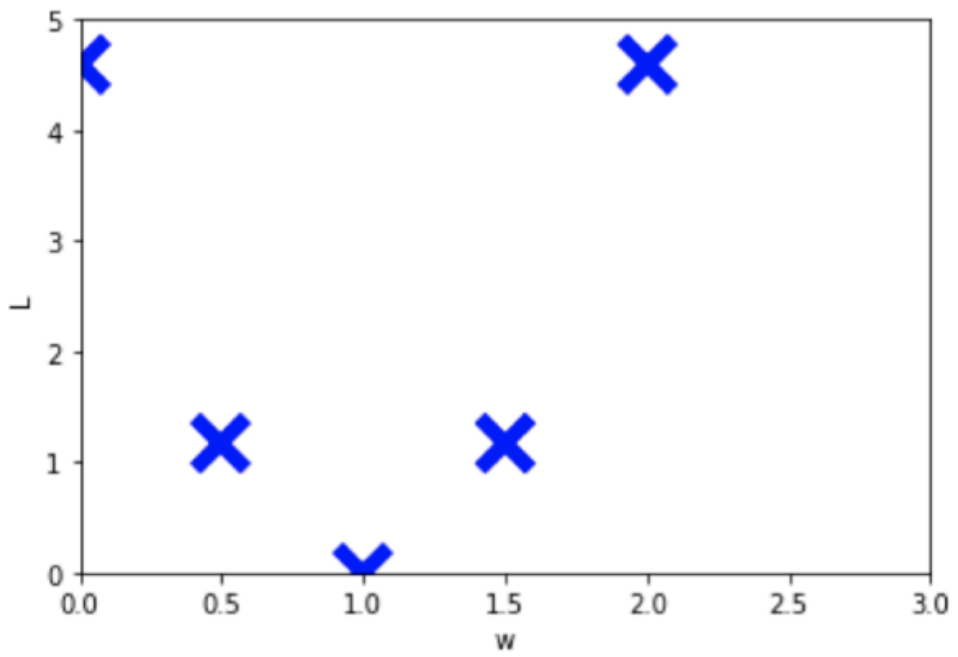




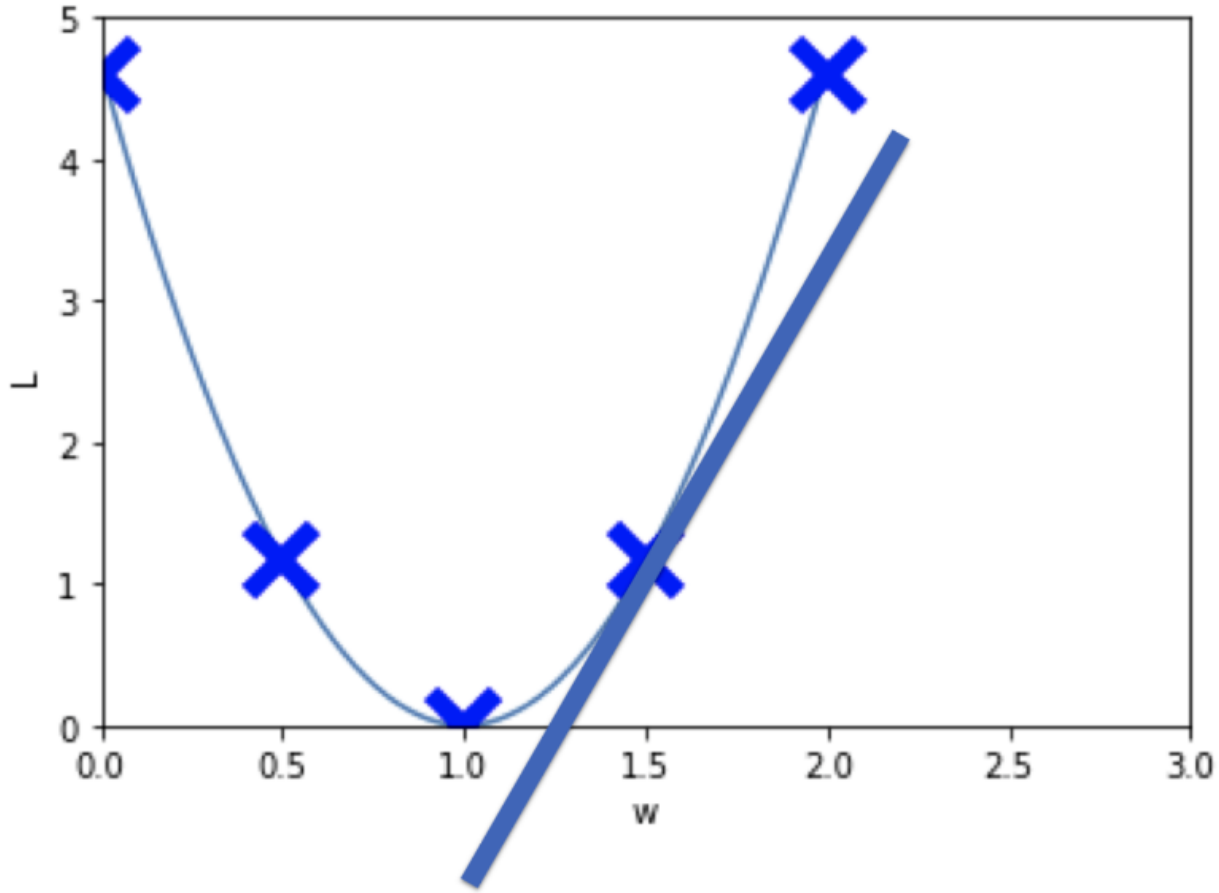
Визуализация ошибок при разных весах

w = 0,0





Градиент



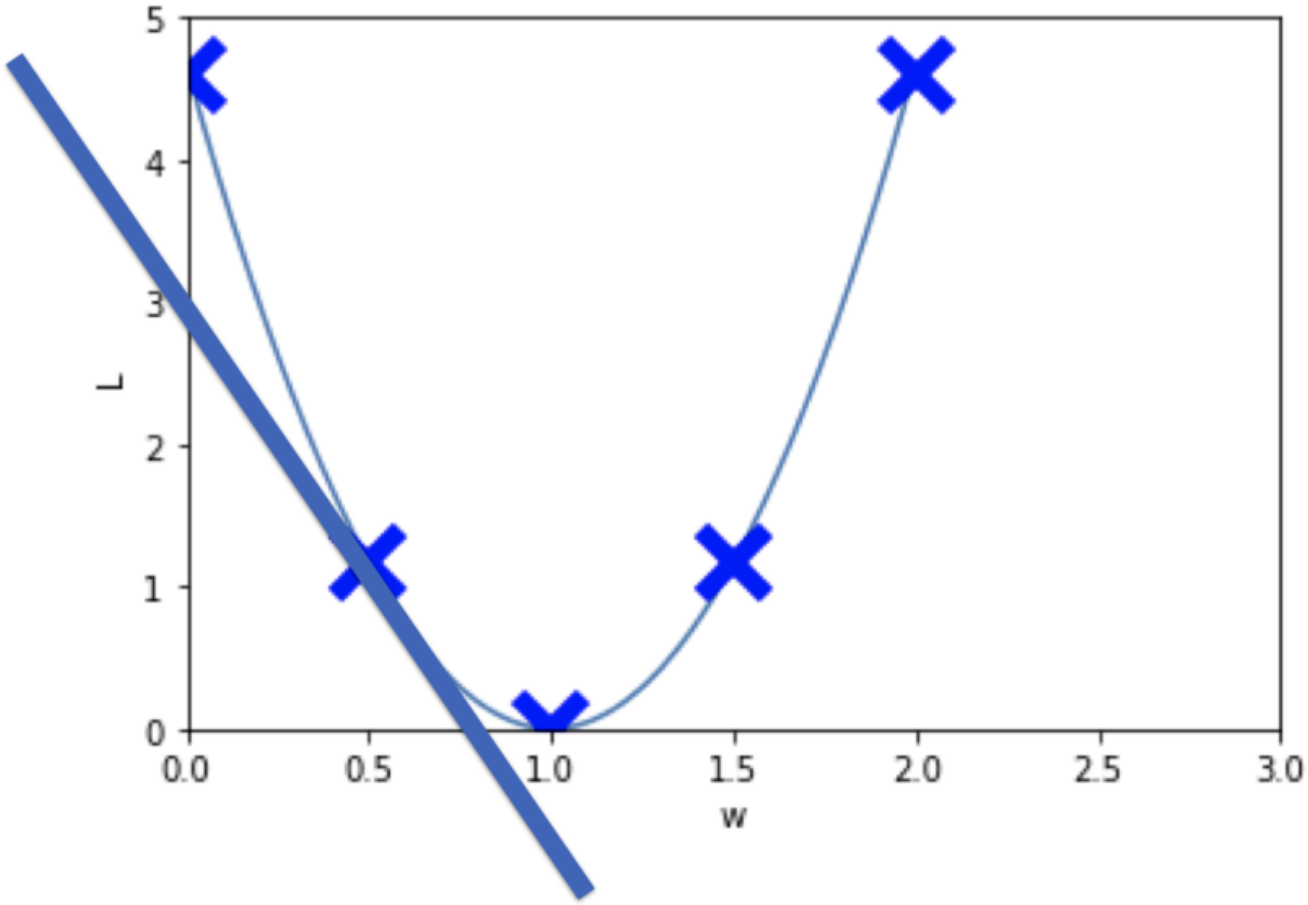
Если

$$f(x) = x^2$$

то

$$f'(x) = 2x$$

Градиент



Если

$$f(x) = x^2$$

то

$$f'(x) = 2x$$

Производная от MSE

$$L(y, \hat{y}) = \frac{1}{m}\sum_{i=0}^{m}(y_i - \hat{y}_i)^2$$

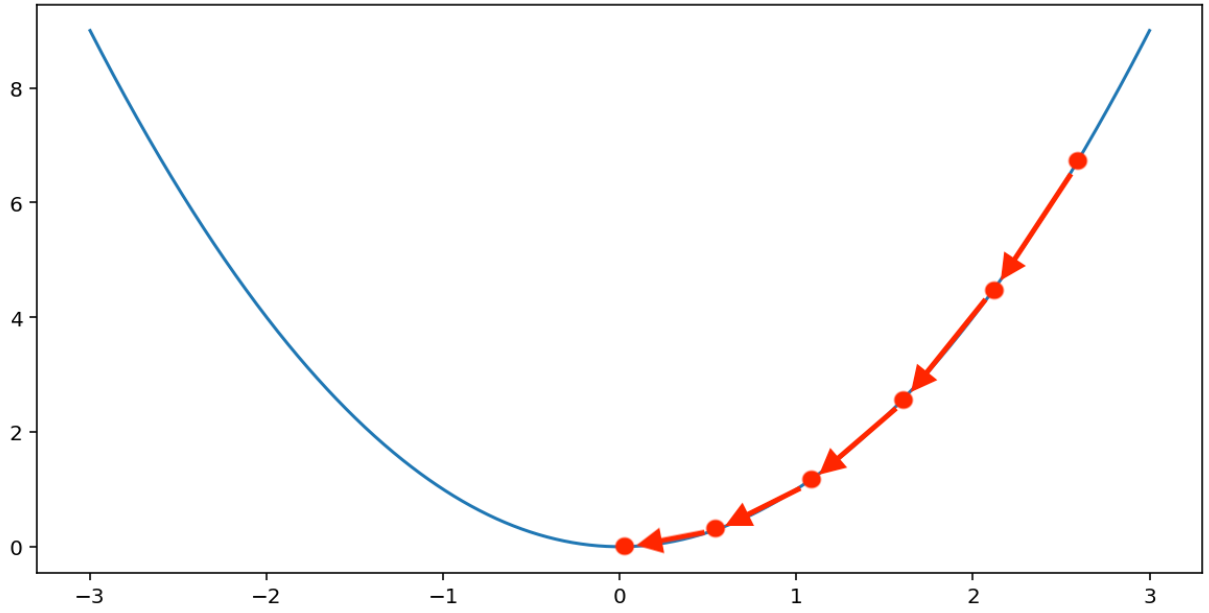
$$L(w_1, ..., w_n) = \frac{1}{m}\sum_{i=0}^{m}(y_i - (w_1x_1 + w_2x_2 + ... + w_nx_n))^2$$

$$\frac{\partial L(w_1)}{\partial w_1} = \frac{2}{m}\sum_{i=0}^{m}(wx + b - y_i)x_1$$

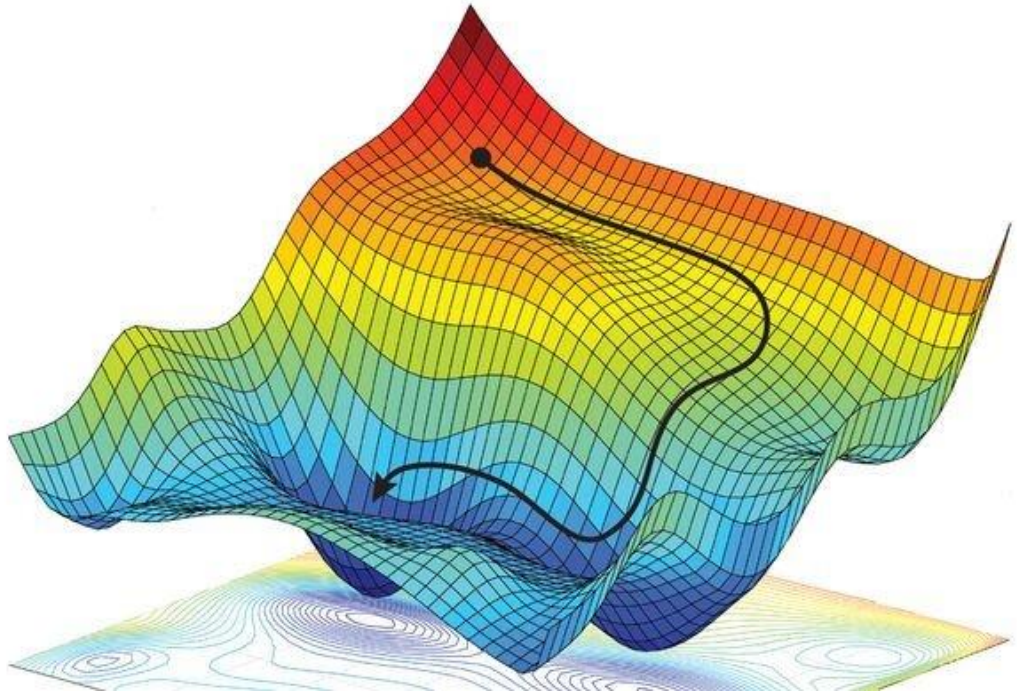
$$\frac{\partial L(b)}{\partial b} = \frac{2}{m}\sum_{i=0}^{m}(wx + b - y_i)$$

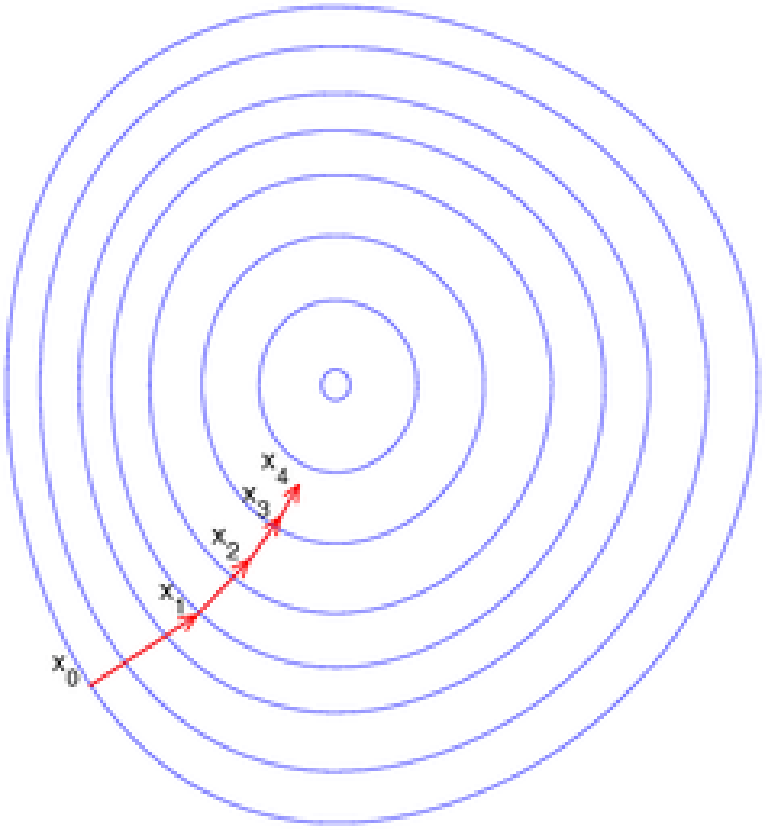
Градиентный спуск

Градиентный спуск - это оптимизационный алгоритм, который используется для минимизации функций, в частности, для настройки параметров модели в машинном обучении.
Главная идея заключается в поиске минимума (или максимума) функции, двигаясь по направлению, противоположному градиенту (первой производной) функции.
Это означает, что алгоритм "спускается" по функции, пока не достигнет локального минимума.

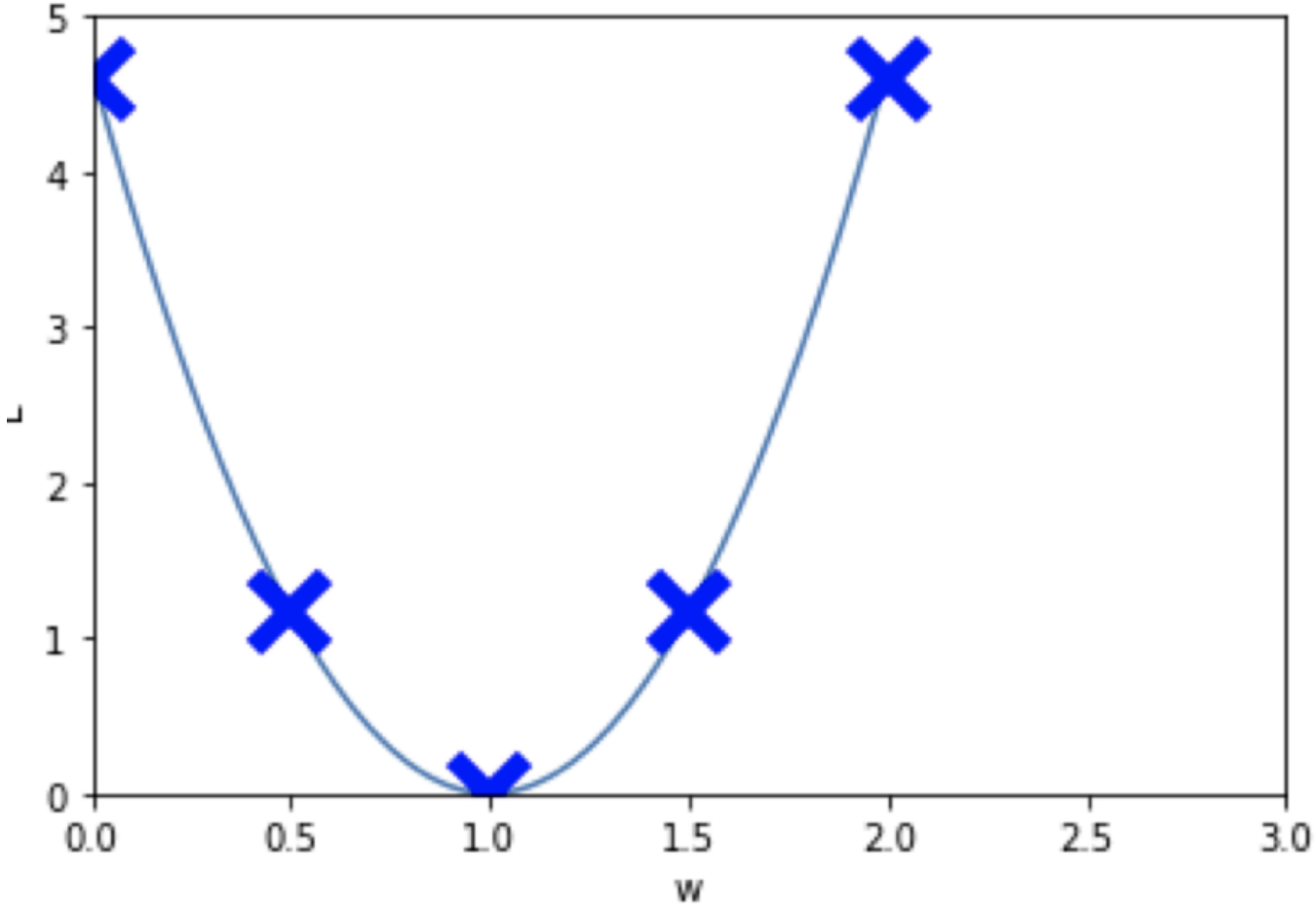


Градиентный спуск





Производная от MSE



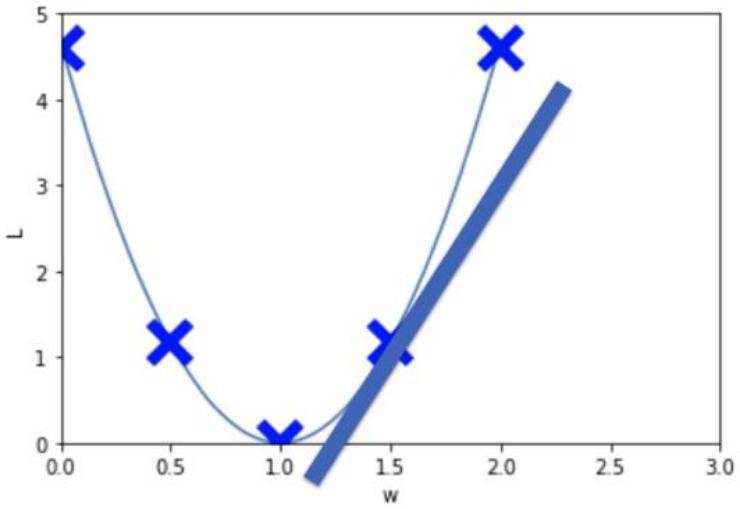
$$w_1 := w_1 - \frac{\partial L(w_1)}{\partial w_1}$$

$$\dots$$

$$w_n := w_n - \frac{\partial L(w_n)}{\partial w_n}$$

$$b := b - \frac{\partial L(b)}{\partial b}$$

Градиентный спуск

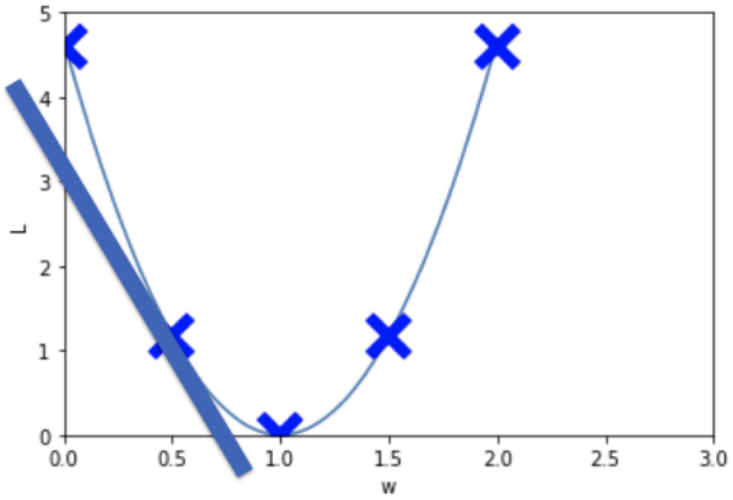


$$X = [1, 2, 3] \quad Y = [1, 2, 3] \quad w = 1.5 \quad b = 0$$

1. $$f(x) = \frac{1}{m} \sum (wx + b - y)^2$$
2. $$\frac{\partial f(x)}{\partial w} = \frac{2}{m} \sum (wx + b - y)x$$
3. $$\frac{\partial f(x)}{\partial w} = \frac{2}{3} \sum (1.5x - y)x$$

$$\frac{\partial f(x)}{\partial w} = \frac{2}{3}((1.5 \times 1 - 1) \times 1 + (1.5 \times 2 - 2) \times 2 + (1.5 \times 3 - 3) \times 3) = 4.6667$$

Градиентный спуск



$$X = [1, 2, 3] \quad Y = [1, 2, 3] \quad w = 0.5 \quad b = 0$$

1. $f(x) = \frac{1}{m} \sum (wx + b - y)^2$
2. $\frac{\partial f(x)}{\partial w} = \frac{2}{m} \sum (wx + b - y)x$
3. $\frac{\partial f(x)}{\partial w} = \frac{2}{3} \sum (1.5x - y)x$

$$\frac{\partial f(x)}{\partial w} = \frac{2}{3}((0.5 \times 1 - 1) \times 1 + (0.5 \times 2 - 2) \times 2 + (0.5 \times 3 - 3) \times 3) = -4.6667$$

Градиентный спуск

Градиент функции потерь

$$\nabla L(w) = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n} \right)$$

Обновление весов на каждом шаге

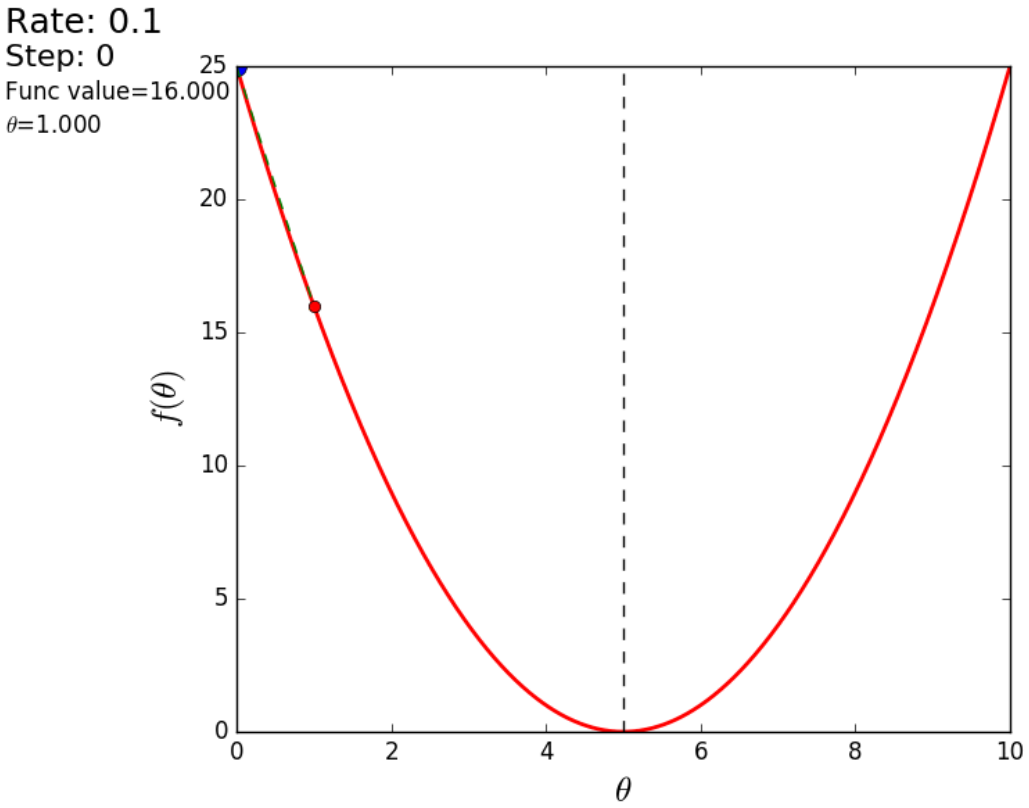
$$w = w - \alpha \cdot \nabla L(w)$$

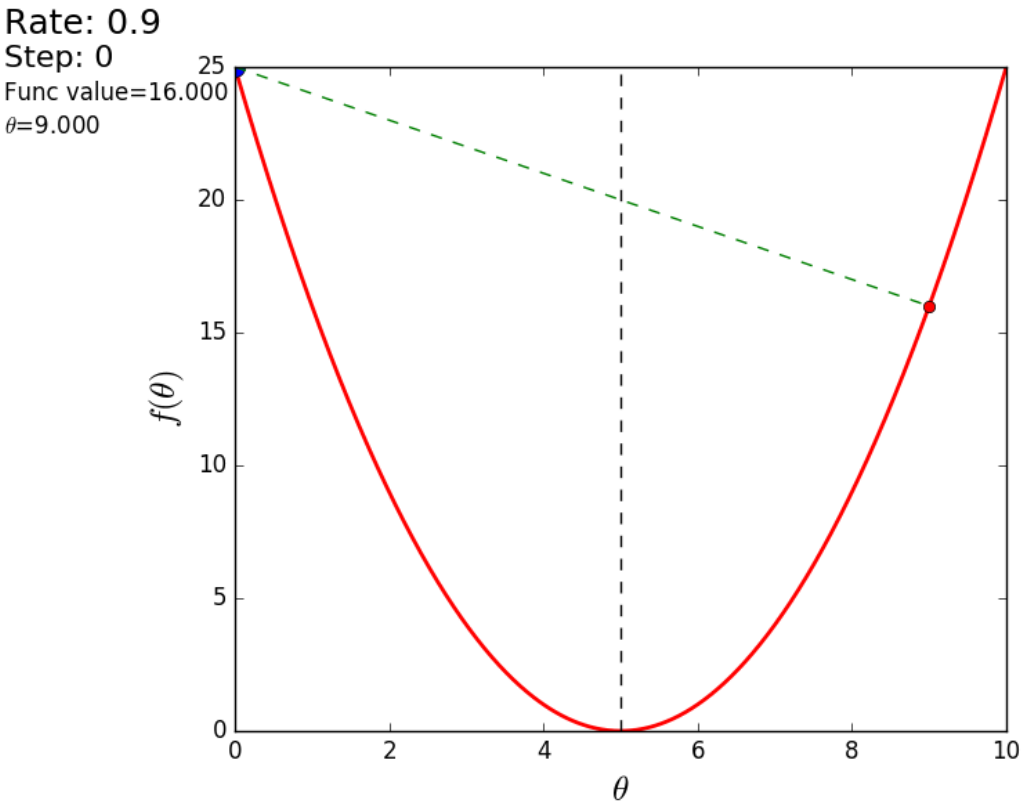
Градиентный спуск

$$\nabla L(w) = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n} \right)$$

$$w = w - \alpha \cdot \nabla L(w)$$

***α* - скорость обучения (learning rate)**





Шаги алгоритма градиентного спуска

Инициализация параметров: Начните с инициализации начальных значений параметров. Это может быть случайным образом или другими методами.

Определение шага (скорости обучения): Выберите скорость обучения (learning rate), которая определяет, насколько "большими шагами" будет двигаться алгоритм по функции. Этот параметр важен, и слишком большой или слишком маленький шаг может повлиять на сходимость алгоритма.

Вычисление градиента: На каждой итерации алгоритм вычисляет градиент функции, который представляет собой вектор, указывающий на направление наибольшего увеличения функции в данной точке. Градиент вычисляется как частные производные функции по каждому параметру.

Обновление параметров: Новые значения параметров вычисляются на основе градиента и скорости обучения. Это делается с целью двигаться в направлении, противоположном градиенту, чтобы уменьшить значение функции.

Гиперпараметры

Гиперпараметры - это параметры, которые **не настраиваются автоматически** алгоритмом машинного обучения в процессе обучения модели, а **задаются вручную** до начала обучения. Они представляют собой конфигурационные параметры модели, которые влияют на ее обучение и производительность, но не определяются непосредственно из данных.

learning rate - контролирует скорость обучения
n_iter - количество итераций

Эпоха

Эпоха - это одна полная итерация через всю тренировочную выборку данных в процессе обучения модели.
Во время одной эпохи модель проходит через все обучающие примеры один раз с целью обновления своих параметров (весов) на основе ошибки предсказания и улучшения своей способности делать более точные прогнозы.
Эпохи используются в алгоритмах обучения с учителем, таких как нейронные сети и алгоритмы градиентного спуска. Процесс обучения обычно включает в себя множество эпох, и количество эпох является одним из гиперпараметров, которые могут быть настроены при обучении модели.

Каждая эпоха может включать в себя несколько итераций, в которых модель делает предсказания для каждого обучающего примера, вычисляет ошибку (потери), а затем обновляет свои параметры (веса) на основе градиента функции потерь.

Эпоха

$$w_1, \ldots, w_n := 0$$

$$b := 0$$

$$for\ i\ in\ range(n\_iter):$$

$$w_1 := w_1 - \alpha \frac{\partial L(w_1)}{\partial w_1}$$

$$\ldots$$

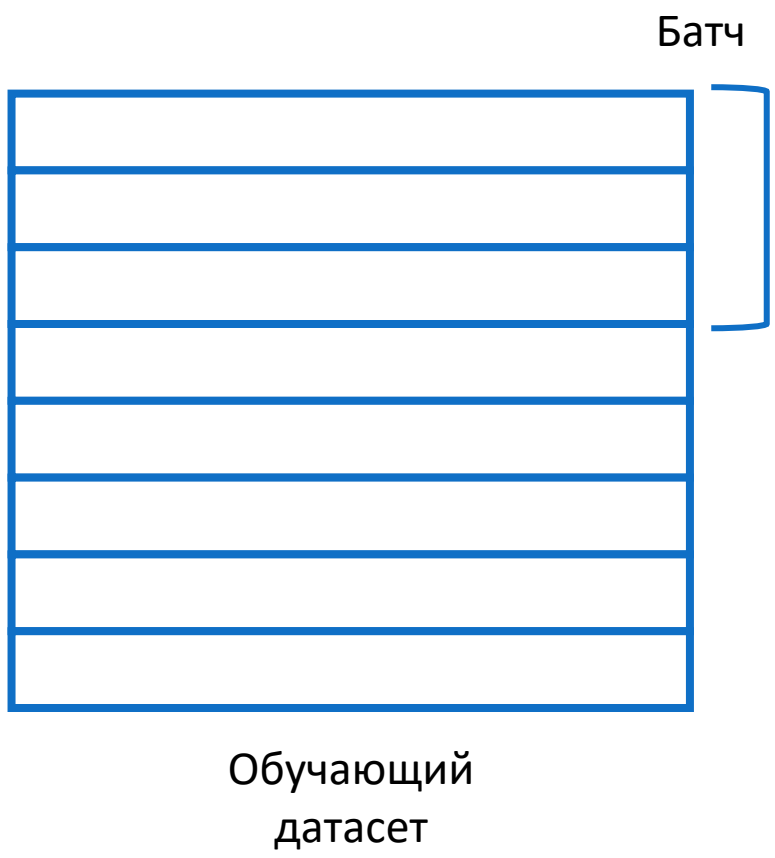
$$w_n := w_n - \alpha \frac{\partial L(w_n)}{\partial w_n}$$

$$b := b - \alpha \frac{\partial L(b)}{\partial b}$$

Батчи

В контексте машинного обучения, "батч" (batch) - это небольшой поднабор данных, который используется для одной итерации (одного шага) обучения модели. Батчи используются в алгоритмах градиентного спуска

Способ, при котором шаг градиентного спуска делается не один раз в эпоху, а несколько раз.



Батчи

Размер батча

Размер батча определяет, сколько обучающих примеров будет использоваться на каждой итерации обучения. Например, в задачах глубокого обучения типичные размеры батчей могут составлять 32, 64, 128 и так далее. Выбор размера батча может влиять на скорость обучения, использование памяти и стабильность сходимости модели.

Преимущества батчей:

- Ускорение обучения: Использование батчей позволяет распараллеливать вычисления, что может значительно ускорить обучение на многопроцессорных системах и графических процессорах.
- Управление памятью: Многие модели и наборы данных могут быть слишком большими для хранения в оперативной памяти целиком, поэтому батчи позволяют обрабатывать данные по частям.

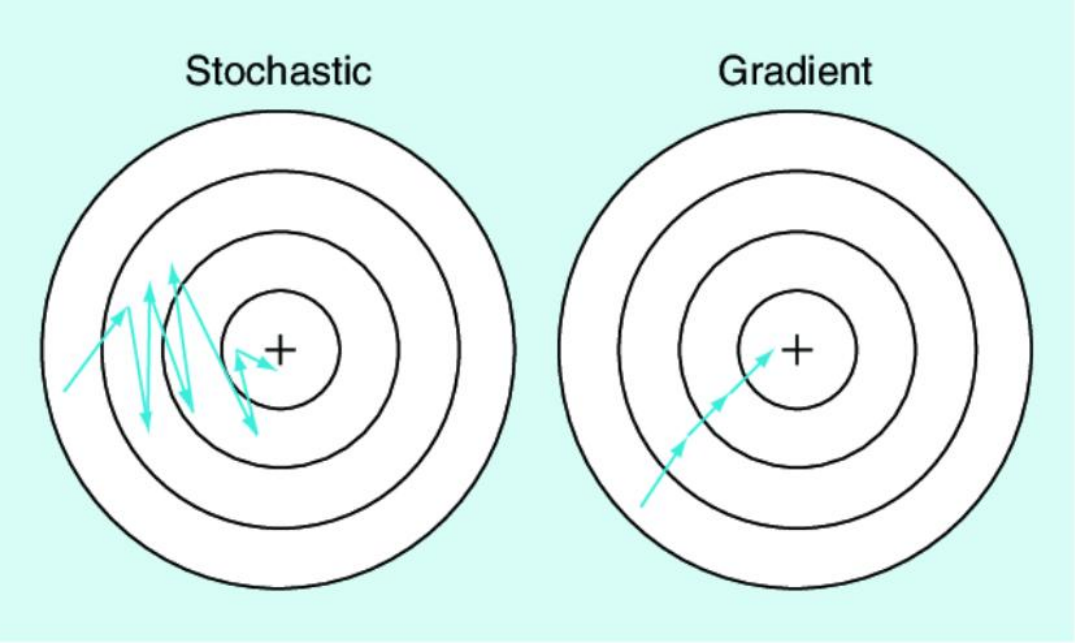
Батчи

Типы батчей:

- **Полный батч (Batch GD):** Все доступные данные используются на каждой итерации. Это обеспечивает стабильную и точную сходимость, но может быть вычислительно затратным для больших наборов данных.
- **Стохастический батч (SGD):** На каждой итерации случайно выбирается один обучающий пример. Это ускоряет сходимость, но может привести к шумным обновлениям параметров.
- **Мини-батч (Mini-batch GD):** Используется случайное подмножество данных заданного размера. Это компромисс между Batch GD и SGD и широко используется в глубоком обучении.

Стохастический градиентный спуск

Способ, при котором шаг градиентного спуска делается не один раз в эпоху, а на каждый экземпляр данных



Стохастический градиентный спуск

Обработка данных:

Обычный градиентный спуск (Batch GD): В этом методе на каждой итерации используется вся тренировочная выборка данных для вычисления градиента функции потерь. Это означает, что все обучающие примеры участвуют в одной итерации.

Стохастический градиентный спуск (SGD): В этом методе на каждой итерации случайно выбирается один обучающий пример (или небольшой мини-пакет) для вычисления градиента. Это означает, что данные обрабатываются по одному или нескольким примерам за итерацию.

Скорость сходимости:

Обычный градиентный спуск (Batch GD): Обычно сходится более стабильно и предсказуемо, так как каждая итерация использует полный набор данных. Однако, в зависимости от размера выборки, он может быть более медленным и требовательным к памяти, особенно для больших наборов данных.

Стохастический градиентный спуск (SGD): Сходится быстрее на начальных этапах обучения, так как обновления параметров происходят более часто. Однако, из-за случайного выбора обучающих примеров, SGD может иметь большие колебания в процессе обучения и требует тщательной настройки скорости обучения.

Шум и вариативность:

Обычный градиентный спуск (Batch GD): Использование всей тренировочной выборки сглаживает градиент и уменьшает шум, что делает его менее подверженным вариативности в обновлениях параметров.

Стохастический градиентный спуск (SGD): Из-за случайного выбора примеров SGD может создавать более шумные обновления параметров, что может помочь выходить из локальных минимумов, но также может привести к более непостоянной сходимости.

Спасибо за внимание!

Конец Лекции 3