Методическое пособие

1 Матрицы. Работа с матрицами в python.

1.1 Действительные чисела

Для начала нужно вспомнить, что такое действительные числа и как мы можем с ними работать.

Аксиомы для действительных чисел:

- $\forall a, b \in \mathbb{R} \rightarrow\ a + b = b + a$ — аксиома коммутативности сложениия,
- $\forall a, b, c \in \mathbb{R} \rightarrow\ a + (b + c) = (a + b) + c$ — аксиома ассоциативности сложениия,
- $\exists 0 \in \mathbb{R} :\ \forall a \in \mathbb{R} \rightarrow\ a + 0 = 0 + a = a$ — аксиома существование нулевого элемента,
- $\forall a \in \mathbb{R}\ \exists(-a) \in \mathbb{R} :\ a+(-a) = 0$ — аксиома существования противоположного элемента,
- $\forall a, b \in \mathbb{R} \rightarrow\ a \cdot b = b \cdot a$ — аксиома коммутативности умножения,
- $\forall a, b, c \in \mathbb{R} \rightarrow\ a \cdot (b \cdot c) = (a \cdot b) \cdot c$ — аксиома ассоциативности умножения,
- $\exists 1 \in \mathbb{R} :\ \forall a \in \mathbb{R} \rightarrow\ a \cdot 1 = 1 \cdot a = a$ — аксиома существование единичного элемента,
- $\forall a \in \mathbb{R}, a \neq 0\ \exists \frac{1}{a} \in \mathbb{R} :\ a \cdot \frac{1}{a} = 1$ — аксиома существования обратного элемента,
- $\forall a, b, c \in \mathbb{R} \rightarrow\ (a + b) \cdot c = a \cdot c + b \cdot c$ — дистрибутивность умножения относительно сложения.

1.2 Матрицы

Рассмотрим следующую систему линейных уравнений:

$$\begin{cases} x_1 + 2x_2 + 3x_3 & = 6 \\ -x_1 - 2x_2 + 3x_3 & = 0\ , \\ -x_1 + 3x_2 + x_3 & = 3 \end{cases} \tag{1}$$

В системе (1) нужно найти $x_1,\ x_2,\ x_3$.

Обозначим $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ — назовем этот вектор как вектор решения системы (1), $\mathbf{b} = \begin{bmatrix} 6 \\ 0 \\ 3 \end{bmatrix}$ — вектор правой части уравнения (1).

Теперь нужно как-то обозначить коэффициенты при $x_1,\ x_2,\ x_3$ в системе (1). Занесем их в некоторую «таблицу» $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ -1 & -2 & 3 \\ -1 & 3 & 1 \end{bmatrix}$

Утверждается, что систему (1), можно записать в виде:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}. \tag{2}$$

Но, с чем у нас проблемы тогда? А с тем, как именно определяется операция умножения «таблицы» на вектор.

«Определение» «Таблица» с определенной на ней операциями сложения и умножения называется матрица.

Пока мы дали не очень строгое определение матрицы, но главное, что нужно понять с него, это то, что матрица это некоторая таблица размера $n \times m$, где $n, m \in \mathbb{N}$ (что значит, что в матрице n строк и m столбцов).

Размер матрицы будем обозначать индексами снизу, то есть запись $\mathbf{A}_{n\times m}$ означает, что в матрице $\mathbf{A}$ n строк и m столбцов.

Сложения двух матриц. Пусть заданы матрицы $\mathbf{A}_{n\times m}$ и $\mathbf{B}_{n\times m}$. Тогда матрицей $\mathbf{C} = \mathbf{A}+\mathbf{B}$ называется суммой матриц $\mathbf{A}$ и $\mathbf{B}$ задаваемая уравнением (3).

$$\mathbf{A}_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}, \quad \mathbf{B}_{n\times m} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

$$\mathbf{C}_{n\times m} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1m}+b_{1m} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2m}+b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \cdots & a_{nm}+b_{nm} \end{bmatrix}. \tag{3}$$

Как видно из уравнения (3) сумма двух матриц это просто поэлементная сумма. Важно заметить, что размеры матрицы $\mathbf{A}$ и $\mathbf{B}$ должны быть равны.

Умножения двух матриц. Пусть заданы матрицы $\mathbf{A}_{n\times k}$ и $\mathbf{B}_{k\times m}$. Тогда результатом умножения двух матриц является матрица $\mathbf{D}_{n\times m} = \mathbf{A}_{n\times k} \cdot \mathbf{B}_{k\times m}$, которая задается уравнением (4).

$$\mathbf{A}_{n\times k} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}, \quad \mathbf{B}_{k\times m} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{bmatrix}$$

$$\mathbf{D}_{n\times m} = \begin{bmatrix} \sum_{j=1}^{k} a_{1j}b_{j1} & \sum_{j=1}^{k} a_{1j}b_{j2} & \cdots & \sum_{j=1}^{k} a_{1j}b_{jm} \\ \sum_{j=1}^{k} a_{2j}b_{j1} & \sum_{j=1}^{k} a_{2j}b_{j2} & \cdots & \sum_{j=1}^{k} a_{2j}b_{jm} \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{j=1}^{k} a_{nj}b_{j1} & \sum_{j=1}^{k} a_{nj}b_{j2} & \cdots & \sum_{j=1}^{k} a_{nj}b_{jm} \end{bmatrix}. \tag{4}$$

Как видно из уравнения (4) каждый элемент матрицы $\mathbf{D}_{n\times m}$ это скалярное произведения соответсвующие скалярные произведения строки матрицы $\mathbf{A}_{n\times k}$ на столбец матрицы $\mathbf{B}_{k\times m}$.

Важно обратить внимания на размерности матриц. Количество столбцов в первой матрице должно соответствовать количеству строк второй матрицы.

Обозначение:

$$\mathbf{0}_{n\times n} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \qquad \mathbf{E}_n = \mathbf{1}_{n\times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Аксиомы для операций над матрицами:

- $\forall \mathbf{A}_{n\times m}, \mathbf{B}_{n\times m} \rightarrow \ \mathbf{A}_{n\times m} + \mathbf{B}_{n\times m} = \mathbf{B}_{n\times m} + \mathbf{A}_{n\times m}$ — аксиома коммутативности сложениия,
- $\forall \mathbf{A}_{n\times m}, \mathbf{B}_{n\times m}, \mathbf{C}_{n\times m} \rightarrow \ \mathbf{A}_{n\times m} + (\mathbf{B}_{n\times m} + \mathbf{C}_{n\times m}) = (\mathbf{A}_{n\times m} + \mathbf{B}_{n\times m}) + \mathbf{C}_{n\times m}$ — аксиома ассоциатиивности сложениия,
- $\exists \mathbf{0}_{n\times m} : \forall \mathbf{A}_{n\times m} \rightarrow \ \mathbf{0} + \mathbf{A} = \mathbf{A}$ — аксиома существование нулевого элемента,
- $\forall \mathbf{A}_{n\times m} \exists (-\mathbf{A}_{n\times m}) : \mathbf{A} + (-\mathbf{A}) = \mathbf{0}$ — аксиома существования протиивоположного элемента,
- $\forall \mathbf{A}_{n\times k}, \mathbf{B}_{k\times m}, \mathbf{C}_{m\times p} \rightarrow \ \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ — аксиома ассоциатиивности умножения
- $\exists \mathbf{E}_n : \forall \mathbf{A}_{n\times n} \rightarrow \ \mathbf{E} \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{E} = \mathbf{A}$ — аксиома существование единичного элемента,
- $\forall \mathbf{A}_{n\times k}, \mathbf{B}_{n\times k}, \mathbf{C}_{k\times m} \rightarrow \ (\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$ — дистрибутивность умножения относительно сложения.

Как можно заметить количество **аксиом** по сравнению с количеством **аксиом** для действительных чисел уменьшилось на **2**. Для матриц не выполняется аксиома коммутативности умножения и существования обратного элемента (обратный элемент, если он существует обозначается $\mathbf{A}^{-1}$). Обратный элемент существует, только для квадратных матриц.

1.3 Решение системы линейных уравнений

Рассматривается случай, когда $\mathbf{A}_{n\times m}$ — квадратная, тоесть случай, когда $n = m$. Будем предполагать, что для $\mathbf{A}$ существует обратная матрица. Вернемся к нашему уравнению (2):

$$\mathbf{A}_{n\times n} \cdot \mathbf{x}_{n\times 1} = \mathbf{b}_{n\times 1}. \tag{5}$$

Теперь зная, как умножаются матрицы мы можем попытаться решить это уравнение. Заметим, что вектор это частный случай матрицы при количестве столбцов равном 1.

Как бы вы решали это уравнение, если бы $\mathbf{A}$ было бы просто числом? Просто поделили на $\mathbf{A}$ слева и справа? Другими словами вы бы умножили на обратный элемент к $\mathbf{A}$ и по аксиоме об обратном получили бы результат.

Проделаем тоже самое для матриц:

$$\mathbf{A}_{n\times n} \cdot \mathbf{x}_{n\times 1} = \mathbf{b}_{n\times 1}$$

$$\mathbf{A}^{-1}_{n\times n}\mathbf{A}_{n\times n}\cdot\mathbf{x}_{n\times 1}=\mathbf{A}^{-1}_{n\times n}\mathbf{b}_{n\times 1}$$

$$\mathbf{E}_n\cdot\mathbf{x}_{n\times 1}=\mathbf{A}^{-1}_{n\times n}\mathbf{b}_{n\times 1}$$

$$\mathbf{x}_{n\times 1}=\mathbf{A}^{-1}_{n\times n}\mathbf{b}_{n\times 1}, \tag{6}$$

И так получаем, что решение уравнения (1) задается уравнением (6), если решение существует.

1.4 Еще раз важные замечания

При работе с матрицами всегда нужно смотреть за размерностями самих матриц. Нужно всегда контролировать которая матрица умножается справа, а какая слева, так-как для матричного умножения не выполняется аксиома коммутативности. Также стоит помнить, что обратный элемент матрицы существует не всегда.

2 Основные понятия в машином обучении. Простейший пример задачи классификации.

2.1 Постановка задачи

2.2 Что задано?

- задано множество объектов $\mathbf{X}$,
- задано множество ответов $\mathbf{Y}$,
- задана некоторая неизвестная функция $\mathbf{f}:\mathbf{X}\to\mathbf{Y}$.

Очевидно, что мы не знаем истиной $\mathbf{f}$. Тогда давайте будем играть в следующую игру, нам дают некоторые $x\in\mathbf{X}$ и дают ответ $y=\mathbf{f}(x)$, а мы хотим найти истинное $\mathbf{f}$.

2.3 Формальная постановка задачи

- дано множество объектов $\{x_1,\cdots x_l\}\subset\mathbf{X}$,
- дано множество ответов $y_i=\mathbf{f}(x_i), i=1,\cdots,l$,
- найти алгоритм $\mathbf{a}:\mathbf{X}\to\mathbf{Y}$ — приближающий неизвестную функцию $\mathbf{f}$.

Под $\mathbf{a}:\mathbf{X}\to\mathbf{Y}$ понимается некоторая функция, которая каждому объекту из множества $\mathbf{X}$ ставит в соответствие некоторый элемент из $\mathbf{Y}$.

2.4 В чем задача машинного обучения?

- понять как задаются объекты и какими могут быть ответы
- ответить на вопрос в каком смысле **a** приближает **f**
- понять как строить отображение **a**

На эти 3 вопроса мы и будем пытаться отвечать в течении нашего курса.

2.5 Задание объектов

Каждый объект выборки задается некоторым набором признаков (features). Что такое признаки? Признаком может быть все что угодно. Например если объект это человек, то в качестве признаков может быть рост, цвет глаз, вес, цвет волос, пол человека и т.д.

2.6 Как задать описание объктов?

$$f_j : X \to D_j, \quad j = 1, \cdots, n, \tag{1}$$

где n это количество признаков для каждого элемента в нашей выборке.

Что такое f_j? f_j это такая функция над объектом x, которая возвращает j-й признак объекта. Например у нас выборка $\mathbf{X}$ это люди, тогда f_j это например функция которая по заданному человеку x возвращает его вес.

Тогда рассмотрим матрицу «объект—признак»:

$$\mathbf{F} = ||f_j(x_i)||_{l\times n} = \begin{bmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \cdots & \cdots & \cdots \\ f_1(x_l) & \cdots & f_n(x_l) \end{bmatrix}. \tag{2}$$

Тоесть матрица $\mathbf{F}$ это матрица которая имеет количество строк равное количеству объектов, а количество столбцов равно количеству признаков.

В нас в курсе у нас всегда изначально будет задана матрица $\mathbf{F}$. Но в общем случаи придумать признаковое описание объектов является достаточно трудной задачей.

2.7 Задание ответов

Классификация :

- $\mathbf{Y} = \{0, 1\}$ — классификация на 2 класса (если в примере с людьми то м. или ж.),
- $\mathbf{Y} = \{0, 1, \cdots, M\}$ — классификация на M классов (к пример с людьми это цвет волос).

Регрессия :

- $\mathbf{Y} = \mathbb{R}^n$ — в качестве ответом у нас может быть вся числовая ось при $n = 1$.

2.8 Примеры смысла «a приближает f»

Задача **регрессии**.

Пусть у нас есть множество истинных ответов $\{\mathbf{y}_1, \cdots, \mathbf{y}_l\}$ и множество ответов которые нам дает алгоритм $\mathbf{a}$.

Определим $\delta(\mathbf{a})$ — как ошибку которую допускает алгоритм $\mathbf{a}$ на данном нам множестве $\{x_1, \cdots x_l\} \subset \mathbf{X}$.

$$\delta\mathbf{a} = \sum_{k=1}^{l} ||\mathbf{a}(x_k) - y_k||^2, \tag{3}$$

где под $||\mathbf{a}(x_k) - y_k||$ — подразумевается расстояние от предсказанного ответа до истинного. К примеру, если $Y = \mathbb{R}$, то в качестве расстояния можно взять модуль разности чисел.

Задача **классификации**.

Определим $\delta(\mathbf{a})$ — как ошибку которую допускает алгоритм $\mathbf{a}$ на данном нам множестве $\{x_1, \cdots x_l\} \subset \mathbf{X}$.

$$\delta\mathbf{a} = \sum_{k=1}^{l} [\mathbf{a}(x_k) \neq y_k], \tag{4}$$

где $[\mathbf{a}(x_k) \neq y_k]$ это 1, если условие в скобках истинное и 0 если ложное. Простыми словами $\delta\mathbf{a}$ определили как количество ошибок в классификации.

2.9 Примеры задач

Задача **регрессии**.

Рассмотрим постановочную задачу регрессии.

2.10 Что дано?

- $\mathbf{X} = \mathbb{R}$ — множество объектов,
- $\mathbf{Y} = \mathbb{R}$ — множество ответов,
- неизвестная функция $\mathbf{f}$ пусть будет просто квадратичная, то есть $y = \mathbf{f}(x) = x^2$

2.11 По каким данным мы восстанавливаем неизвестную функцию?

На рис. 1 показаны данные по которым мы должно построить отображение $\mathbf{a}$. По оси абсцисс отложены значения $\{x_1, \cdots, x_{10}\} \subset X$, а по оси ординат отложены значения $\{y_1, \cdots, y_{10}\} \subset Y$. Также на графике построен график функции $y = x^2$. Как видно из графика синие точке не ложатся идеально на красный график. Это все из-за того, что данные не бывают идеальные, об этом мы поговорим чуть позже.

Задача **классификации**.

Рассмотрим постановочную задачу классификации.

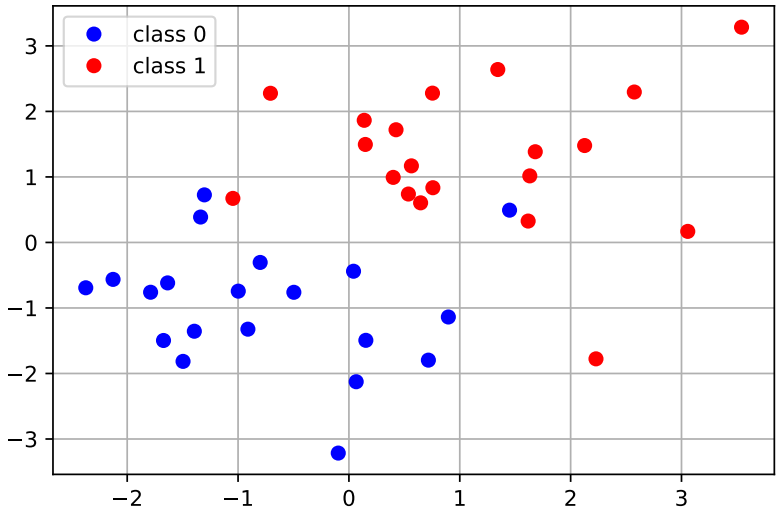

Рис. 1: Данные данные для нахождения **a**

2.12 Что дано?

- $\mathbf{X} = \mathbb{R}^2$ — множество объектов,
- $\mathbf{Y} = \{0, 1\}$ — множество ответов,
- неизвестная функция **f**

2.13 По каким данным мы восстанавливаем неизвестную функцию?

2.14 Метод k - ближайших соседей

Задача **классификации**.

Данный метод является очень простым. Пусть мы можем измерить расстояние между любыми объектами из множества **X**. Тогда алгоритм заключается в том, чтобы найти тот класс которого больше всего среди $k-$ближайших соседей.

2.15 Алгоритм

- k $= 9$ — сколько соседей будем учитывать
- M $= 2$ — количество классов
- $\{x_1, ..., x_l\}$ — объекты для которых мы знаем ответы
- $\{y_1, ..., y_l\}$ — ответы
- x — нужно классифицировать

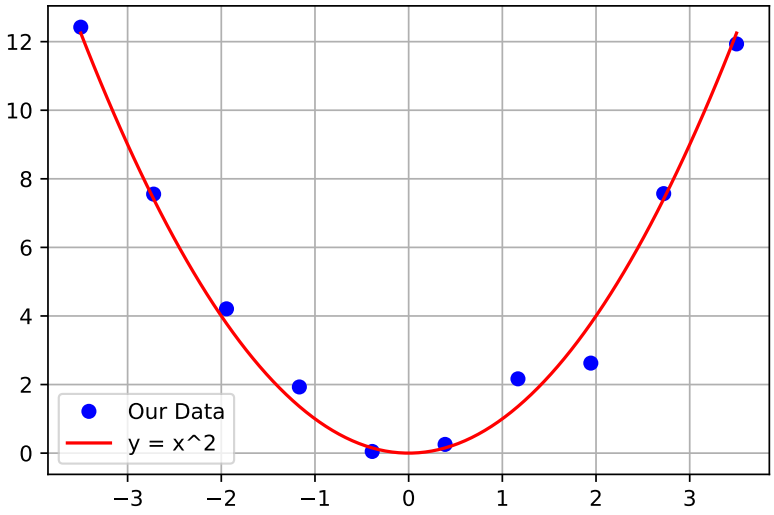

Рис. 2: Данные данные для нахождения **a**

- mesure
- mesure = mesure.sort — сортируем его по возрастанию (параллельно нужно сортировать и y)
- arr — массив счетчик каждого класса среди k ближайших
- for i in range(k)
- $arr[y_i] = arr[y_i] + 1$

3 Простейшие методы оптимизации. Градиентный спуск. Линейная модель.

3.1 Регрессия

$$\mathcal{D}^l = \{x_i, y_i\}_{i=1}^l, \quad x_i \in \mathbb{R}, \ y_i \in \mathbb{R}. \tag{1.1}$$

Пусть имеется некоторая обучающая выборка $\mathcal{D}^l$ размера l по которой мы хотим построить некоторую модель.

Определение 1.1: Линейной моделью регрессии назовем функцию $\mathbf{a}(x, \mathbf{w})$ из (1.2), которая зависит от некоторого неизвестного параметра $\mathbf{w} \in \mathbb{R}^n$.

$$\mathbf{a}(x, \mathbf{w}) = \sum_{j=1}^{n} w_j f_j(x), \tag{1.2}$$

где $f_j(x)$ — это функция которая по заданному объекту x выдает j-й признак этого объекта.

Заметим, что вектор параметров $\mathbf{w}$ является неизвестным и его нужно найти по заданной выборке $\mathcal{D}^l$

Определение 1.2: Введем понятия функции потерь модели $\mathbf{a}$ на некотором объекте $(x, y) \in \mathcal{D}^l$ следующим образом:

$$\mathcal{L}(\mathbf{w}, (x, y)) = (\mathbf{a}(x, \mathbf{w}) - y)^2. \tag{1.3}$$

Определение 1.3: Введем понятия функции потерь модели регрессии $\mathbf{a}$ на выборке $\mathcal{D}^l$ следующим образом:

$$\mathcal{Q}(\mathbf{w}) = \sum_{j=1}^{l} \mathcal{L}(\mathbf{w}, (x_j, y_j)). \tag{1.4}$$

Теперь, мы можем сформулировать задачу машинного обучения, как поиск $\mathbf{w}$, такого что $\mathcal{Q}(\mathbf{w})$ является минимальным. Формальная запись этого факта, это:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \mathcal{Q}(\mathbf{w}). \tag{1.5}$$

Тогда после нахождения такого $\hat{\mathbf{w}}$, мы получаем обученную линейную модель.

3.2 Классификация

$$\mathcal{D}^l = \{x_i, y_i\}_{i=1}^l, \quad x_i \in \mathbb{R},\ y_i \in \{-1, +1\}. \tag{2.1}$$

Пусть имеется некоторая обучающая выборка $\mathcal{D}^l$ размера l по которой мы хотим построить некоторую модель.

Определение 2.1: Линейной моделью классификации назовем функцию $\mathbf{a}(x, \mathbf{w})$ из (2.2), которая зависит от некоторого неизвестного параметра $\mathbf{w} \in \mathbb{R}^n$.

$$\mathbf{a}(x, \mathbf{w}) = \operatorname{sign} \sum_{j=1}^{n} w_j f_j(x), \tag{2.2}$$

где $f_j(x)$ — это функция которая по заданному объекту x выдает j-й признак этого объекта.

Определение 2.2: Введем понятия функции потерь модели классификации $\mathbf{a}$ на некотором объекте $(x, y) \in \mathcal{D}^l$ следующим образом:

$$\mathcal{L}(\mathbf{w}, (x, y)) = -\mathbf{a}(x, \mathbf{w}) \cdot y. \tag{2.3}$$

Определение 2.3: Введем понятия функции потерь модели регрессии **a** на выборке $\mathcal{D}^l$ следующим образом:

$$\mathcal{Q}(\mathbf{w}) = \sum_{j=1}^{l} \mathcal{L}(\mathbf{w}, (x_j, y_j)). \tag{2.4}$$

Теперь аналогично задачи регрессии сформулируем оптимизационную задачу:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \mathcal{Q}(\mathbf{w}). \tag{2.5}$$

3.3 Решение оптимизационной задачи

3.4 Производная

$$f'(x) = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}. \tag{3.1.1}$$

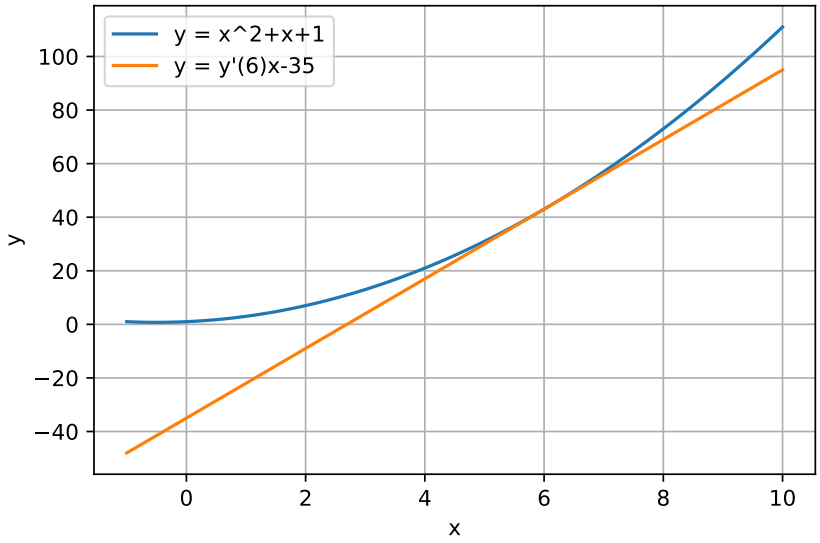

Рис. 3: График функции и касательная в точке**a**

Будем использовать свойство знака производной. Знак производной указывает на то растет функция в этой точке или убывает, это свойство прямо следует из определения производной. Покажем этот факт:

$$f'(x_0) = \lim_{\Delta x \to 0,\ \Delta x > 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}, \tag{3.1.2}$$

Из уравнения (3.2) видно, что знак производной в точке x_0 равен знаку $f(x_0+\Delta x)-f(x_0)$, что и требовалось показать.

Для примера из рис. 3 $y'(6) = 13$. Тогда с этого следует, что функция в этой точке растет при увеличения x, тогда для того, чтобы найти минимальное значение y, нужно уменьшить x.

Вот мы пришли к выводу, что если у нас есть некоторая функция одного переменного, то для того, чтобы найти ее минимум нужно менять x в противоположном направлении к знаку производной.

На этом базируется следующий итеративный подход к нахождению минимума функции.

Определение 3.1.1: Итеративный процесс обозначает, что мы делаем что-то шаг за шагом.

Рассмотрим следующий итеративный процесс, для нахождения минимума одномерной функции $f(x)$ с областью определения D_f.

1. Пусть имеется $x_0 \in D_f$ — некоторая точка из области определения функции. 2. Пересчитывать новую точку будем по формуле:

$$x_{n+1} = x_n - \alpha \cdot f'(x_n), \tag{3.1.3}$$

где α некоторое значение — шаг который мы делаем, он может быть как и постоянным так и переменным. Мы будем пока считать его постоянным числом, например 0.0001.

Мы научились находить минимум функции от скаляра. Но что же делать, для функции от вектора, которой является $\mathcal{Q}(\mathbf{w})$.

3.5 Градиент

Определение 3.2.1: Частной производной функции многих переменных $f(\mathbf{x})$ по x_j назовем производную функции $f'(x_j)$ считая все остальными переменные константой. Частная производная обозначается следующим образом:

$$\frac{\partial f}{\partial x_j} = f'_j(x_j), \tag{3.2.1}$$

где $f'_j(x)$ — это функция одной переменной, где все переменные кроме j-й фиксированы.

Определение 3.2.2: Градиентом функции $f(\mathbf{x})$ называется вектор $\nabla f(\mathbf{x})$ элементы которого, это частные производные функции $f(\mathbf{x})$.

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{x_1} \\ \frac{\partial f}{x_2} \\ \cdots \\ \frac{\partial f}{x_n} \end{bmatrix}, \tag{3.2.2}$$

где $x_1,\ x_2, \cdots,\ x_n$ — компоненты вектора $\mathbf{x}$.

По аналогии с функцией одного переменного можно определить итеративный процесс нахождения минимума функции многих переменных.

1. Пусть имеется $\mathbf{x}^0 \in D_f$ — некоторая точка из области определения функции $f(\mathbf{x})$. 2. Пересчитывать новую точку будем по формуле:

$$x^{n+1} = x^n - \alpha \cdot \nabla f(\mathbf{x}^n), \tag{3.2.3}$$

где α некоторое значение — шаг который мы делаем, он может быть как и постоянным так и переменным. Мы будем пока считать его постоянным числом, например 0.0001.

Как видно все изменения в итеративной формуле это производная на градиент.

3.6 Пример вычисления градиентов:

$$f(x_1, x_2) = x_1^2 + x_2^2 \Rightarrow \nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{x_1} \\ \frac{\partial f}{x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}. \tag{3.3.1}$$

$$f(x_1, x_2) = e^{x_1} + e^{x_2} + x_1 x_2 \Rightarrow \nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{x_1} \\ \frac{\partial f}{x_2} \end{bmatrix} = \begin{bmatrix} e^{x_1} + x_2 \\ e^{x_2} + x_1 \end{bmatrix}. \tag{3.3.2}$$

4 Дополнительные возможности в python.

5 Самостоятельная работа. Опиисание выборки.

5.1 Wine[1]

Эти данные являются результатом химического анализа вин, выращенных в одном и том же регионе в Италии, но полученных из трех разных сортов. Анализ определил количество 13 составляющих, найденных в каждом из трех типов вин.

Таблица 1: Описание выборки Wine

Алкоголь	Малиновая кислота	Зола	Алкалиния золы
Магний	Всего фенолов	Флаваноиды	Нефлаваноидные фенолы
Проантоцианы	Интенсивность цвета	Оттенок	OD280 / OD315 разведенных вин
Пролин	**Тип винограда**		

5.2 Iris[2]

Это, пожалуй, самая известная база данных, которая может быть найдена в литературе по распознаванию образов. Статья Фишера является классикой в этой области и часто упоминается по сей день. (Например, Duda & Hart). Набор данных содержит 3 класса по 50 экземпляров каждый, где каждый класс относится к типу ириса.

Таблица 2: Описание выборки Iris

Длина чашелистника	Ширина чашелистник	Длина лепестка	Ширина лепестка
Тип ириса			

5.3 Self-Noise[3]

Набор данных NACA, полученный из серии аэродинамических и акустических испытаний двух и трехмерных участков лезвия аэродинамического профиля, выполненных в безэховой аэродинамической трубе. Набор данных NASA содержит аэродинамические профили NACA 0012 различного размера на различных скоростях и углах атаки в аэродинамической трубе. Пролет аэродинамического профиля и положение наблюдателя были одинаковыми во всех экспериментах.

Таблица 3: Описание выборки Self-Noise

Частота	Угол атаки	Длина волны	Скорость потока
Толщина смещения	**Уровень звукового давления**		

5.4 Yacht Hydrodynamics[4]

Прогнозирование остаточного сопротивления парусных яхт на начальном этапе проектирования имеет большое значение для оценки эффективности судна и оценки требуемой пропульсивной мощности. Основные входы включают основные размеры корпуса и скорость лодки. Набор данных Delft содержит 308 полномасштабных экспериментов, которые были выполнены в Лаборатории гидромеханики Делфта для этой цели.

- Продольное положение центра плавучести
- Призматический коэффициент
- Отношение длины и смещения
- Соотношение лучей
- Отношение длины к лучу
- Число Фруда
- **Сопротивление резистивности**

6 Структура описания модели в python.

7 ООП в python.

8 Повторение основных моделей. Автоматическое дифференцирование.

8.1 Многослойный перцептрон

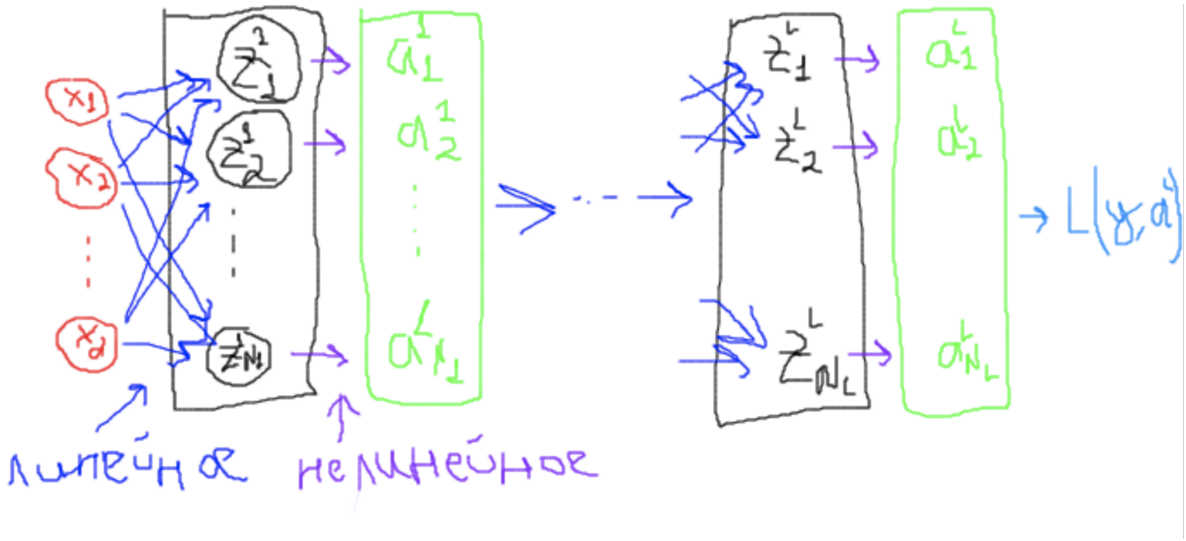

Рис. 4

Пусть у нас задана следующая система преобразований:

$$a_i^0 = x_i \quad z^{l+1} = w_{l+1}^{\mathsf{T}} a^l + b_{l+1}, \quad a_j^{l+1} = g(z_j^{l+1}), \tag{2.1}$$

где $i \in [1, d]$, $l \in [0, L-1]$, $j \in [1, N_{l+1}]$, $g(t)$ — некоторая функция активации.

8.2 функции активации

- $\sigma(z) = \frac{1}{1+\exp(-z)} \in (0, 1)$
- $\tanh = 2(\sigma(z) - \frac{1}{2}) \in (-1, 1)$
- $\mathrm{ReLu}(z) = \max(0, z)$
- $\mathrm{LeakyRelu}(z) = \max(\alpha z, z), \alpha \in (0, 1)$

Существует теорема, которая говорит, что любую непрерывную функцию можно сколь угодно близко приблизить нейросетью с функцией активации $\sigma(x)$.

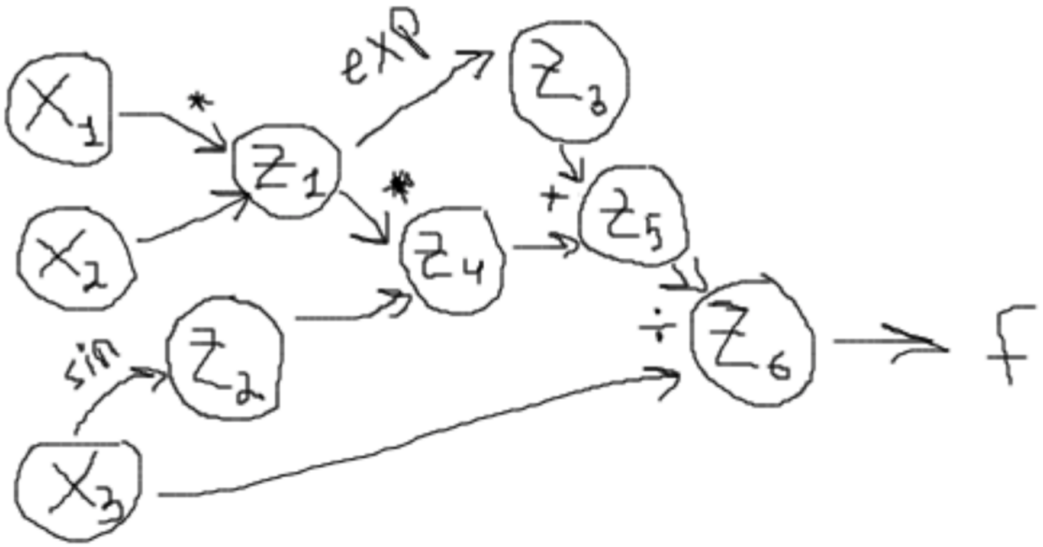

Рис. 5

8.3 Автоматическое дифференцирование

Рассмотрим на примере:

$$f(\mathbf{x}) = \frac{x_1 x_2 \sin(x_3) + \exp(x_1 x_2)}{x_3}. \tag{2.2}$$

Граф вычислений функции (2.2) изображен на рис. 5.

Для данной функции посчитаем $\frac{\partial f}{\partial x_1}$ через дифференцирование вперед и дифференцированием назад:

Дифференцирование вперед:

- $\frac{\partial z_1}{\partial x_1} = x_2$
- $\frac{\partial z_2}{\partial x_1} = 0$
- $\frac{\partial z_3}{\partial x_1} = \frac{\partial z_3}{\partial z_1}\frac{\partial z_1}{\partial x_1} = \exp(z_1)x_2$
- $\frac{\partial z_4}{\partial x_1} = \frac{\partial z_4}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial z_4}{\partial z_2}\frac{\partial z_2}{\partial x_1} = z_2 x_2$
- $\frac{\partial z_5}{\partial x_1} = \frac{\partial z_5}{\partial z_3}\frac{\partial z_3}{\partial x_1} + \frac{\partial z_5}{\partial z_4}\frac{\partial z_4}{\partial x_1} = \exp(z_1)x_2 + z_2 x_2$
- $\frac{\partial z_6}{\partial x_1} = \frac{\partial z_6}{\partial x_3}\frac{\partial x_3}{\partial x_1} + \frac{\partial z_6}{\partial z_5}\frac{\partial z_5}{\partial x_1} = \frac{1}{x_3}\left(\exp(z_1)x_2 + z_2 x_2\right)$
- $\frac{\partial f}{x_1} = \frac{\partial f}{\partial z_6}\frac{\partial z_6}{\partial x_1} = \frac{1}{x_3}\left(\exp(z_1)x_2 + z_2 x_2\right)$

Дифференцирование назад:

- $\frac{\partial f}{\partial z_6} = 1$
- $\frac{\partial f}{\partial z_5} = \frac{\partial f}{\partial z_6}\frac{\partial z_6}{\partial z_5} = \frac{1}{x_3}$
- $\frac{\partial f}{\partial z_4} = \frac{\partial f}{\partial z_5}\frac{\partial z_5}{\partial z_4} = \frac{1}{x_3}$
- $\frac{\partial f}{\partial z_3} = \frac{\partial f}{\partial z_5}\frac{\partial z_5}{\partial z_3} = \frac{1}{x_3}$
- $\frac{\partial f}{\partial z_2} = \frac{\partial f}{\partial z_4}\frac{\partial z_4}{\partial z_2} = \frac{1}{x_3}z_1$
- $\frac{\partial f}{\partial z_1} = \frac{\partial f}{\partial z_3}\frac{\partial z_3}{\partial z_1} + \frac{\partial f}{\partial z_4}\frac{\partial z_4}{\partial z_1} = \frac{1}{x_3}\exp(z_1) + \frac{1}{x_3}z_2$
- $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial z_1}\frac{\partial z_1}{\partial x_1} = \frac{1}{x_3}\left(\exp(z_1) + z_2\right)x_2$

Как видно результат работы алгоритмов дифференцирования вперед и назад совпадает.

9 Написание модели нейронной сети при помощи python.

10 Работа с классами. Реализации матриц в python.

11 Работа с изображениями. Точечные изменения изображений в python.

12 Работа с изображениямии. Свертки. Реализация свертки Собеля.

13 Введение в torch. Построение линейной модели.

14 Использование torch для обработки изображений.

15 Использование torch для обработки текстов. seq2seq модель.

16 Использование torch для описание изображений. Image to caption task.

Список литературы

[1] *Wine dataset.* http://archive.ics.uci.edu/ml/datasets/Wine

[2] *Iris dataset.* http://archive.ics.uci.edu/ml/datasets/Iris

[3] *Self-Noise.* http://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

[4] *Yacht Hydrodynamics.* http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics