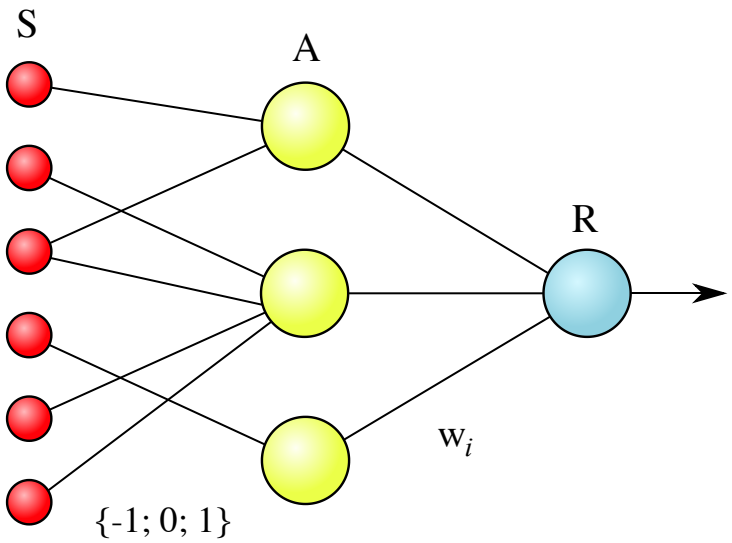
Лабораторная работа 5. Перцептрон

Определение

Перцептрон (англ. perceptron от лат. perceptio – восприятие) – математическая или компьютерная модель восприятия информации мозгом (кибернетическая модель мозга), предложенная Фрэнком Розенблаттом в 1957 году и впервые воплощённая в виде электронной машины «Марк-1» в 1960 году Wiki

- Перцептрон – одна из первых моделей нейросетей
- «Марк-1» – первым в мире нейрокомпьютером

Описание элементарного перцептрона



Обучение элементарного перцептрона состоит в изменении весовых коэффициентов w_i связей A–R.
Веса связей S–A (которые могут принимать значения {−1; 0; +1}) и значения порогов A-элементов выбираются случайным образом в самом начале и затем не изменяются.

После обучения перцептрон готов работать в режиме **распознавания**. В этом режиме перцептрону предъявляются ранее не известные ему объекты, и перцептрон должен установить, к какому классу они принадлежат.

Пример реализации элементарного перцептрона

Постановка задачи

Определить риск одобрения кредита на основе 10 критериев.

Дано: искусственный датасет, основанный на информации об одобрении или отказе 5000 заявок на кредиты. Каждая заявка содержит следующие критерии: возраст, доход, кредитная история, долг, стаж работы, количество кредитов, семейное положение, наличие недвижимости, образование, сбережения. **Все значения нормализованы**. Каждой заявке соответствует результат: 1 – кредит одобрен, 0 – отказ в кредите.

Задача: на основе данного датасета обучить модель (подобрать веса), которая может определять степень риска одобрения кредита.

Инициализация

- В качестве функции активации используем сигмоиду

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

- Введем количество заявок в датасете и количество критериев каждой заявки

```
samples = 5000
features = 10
```

- Зададим начальные веса и пороговое значение. Веса домножаем на 0.01, чтобы значения были близки к нулю

```
weights = np.random.rand(features) * 0.01
bias = 0.0
```

Чтение датасета

Ваша задача самостоятельно прочитать файл и:

- Сформировать двумерный numpy массив `x` признаков размера `(5000, 10)`
- Сформировать numpy массив `y` меток размера `(5000,)`
- Разбить датасет на training set и test_set в соотношении 80% к 20%

Обучение

Упростим модель и предположим, что S-слой и A-слой объединены в один. Таким образом, необходимо вычислить только взвешенную сумму и подставить результат в функцию активации.

Для каждой эпохи выполним следующие действия:

- Вычислим взвешенную сумму по всем объектам датасета x_i, где $i \in [1, 5000]$

$$a_i = \sum_{j=1}^{10} x_{ij} w_j - \theta \tag{2}$$

- Подставим результат в функцию активации и вычислим ошибки

$$\begin{aligned} r_i &= \sigma(x_i) \\ \varepsilon_i &= r_i - y_i \end{aligned} \tag{3}$$

- Вычислим градиенты:

$$\begin{aligned} dw_j &= \sum_{i=1}^{n} x_{ij} \varepsilon_i \\ d\theta &= \sum_{j=1}^{10} \varepsilon_j \end{aligned} \tag{4}$$

  где n – количество примеров в обучающем датасете (training set)
- Далее вычисляем новые веса и пороговое значение

Реализация обучения на python

```
for epoch in range(epochs):
    a = np.dot(x, weights) + bias
    r = sigmoid(a)

    # Ошибка
    error = r - y

    # Градиенты
    dw = np.dot(x.T, error) / samples
    db = np.sum(error) / samples

    # Градиентный спуск
    weights -= learning_rate * dW
    bias -= learning_rate * dB
```

Проверим работу обученной модели

- Из тестового набора возьмем элемент `x_test`
- Вычислим предсказание:

```
a_test = np.dot(a_test, weights) + bias
r_test = sigmoid(a_test)

prediction = 1 if r_test >= 0.5 else 0
```

Задания

1. Точность модели

После обучения вычислите точность модели на тестовом датасете по формуле:

$$accuracy = \frac{Количество\ правильный\ ответов}{Общее\ количество\ объектов} \tag{5}$$

2. Функция потерь (loss)

Функция потерь (loss function) в машинном обучении и математической статистике – математическая формула, которая измеряет ошибку между прогнозом модели и реальными данными.

Проведите обучение модели на 1000 эпохах. Для каждой эпохи вычислите функцию потерь. Для бинарного перцептрона с сигмоидой принято использовать **Binary Cross-Entropy** (Бинарная кросс-энтропия)
Для одного объекта:

$$loss_i = -(y_i\ ln(r_i) + (1 - y_i)\ ln(1 - r_i)) \tag{6}$$

Соответственно, для всего датасета:

$$loss = \frac{1}{n}\sum_{i=i}^{n} loss_i \tag{7}$$

где n – количество примеров в обучающем датасете (training set)

Постройте график изменения функции потерь по эпохам.

3. Скорость обучения

Обучите модель при разных значениях:

- `learning_rate = 0.001`
- `learning_rate = 0.01`
- `learning_rate = 0.1`
- `learning_rate = 1`

Сравните:

- Скорость сходимости
- Итоговую ошибку
- На одной axes постройте четыре графика loss-функции для различных learning rate

Дополнительное задание. Функция активации

Измените функцию активации на `tanh`. Что изменится в модели? Сравните результаты обучения.

Ожидаемые вопросы на защите проекта

- Что такое перцептрон?
- Что такое веса модели?
- Для чего нужен bias?
- Что показывает функция потерь?
- Что такое градиент?
- Почему используется градиентный спуск?
- Что делает сигмоида?
- Почему данные нормализуют?
- Что произойдет при слишком большом learning rate?