**Практическое задание № 2**

# ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

**Цель работы:** получить практику анализа статистических данных с использованием логистической регрессии.

## Содержание задания

## 1. Общие сведения

1. Ознакомиться с материалами лекции № 2.

2. Установить необходимое программное обеспечение. При выполнении задания наверняка понадобятся **Python 3**, **NumPy** и **Matplotlib**.

3. Ознакомиться с содержимым папки с заданием, которая включает в себя файлы, представленные ниже.

**main.py** – «основной» модуль, необходимый для выполнения задания, который поможет выполнить его поэтапно. Настоящий программный код не требует какой-либо коррекции!

**data.txt** – база данных для выполнения задания.

**plotData.py** – модуль, содержащий функцию `plotData`, которая необходима для визуализации данных.

**plotDecisionBoundary.py** – модуль, содержащий функцию `plotDecisionBoundary`, которая необходима для визуализации данных с границей решения для заданного множества параметров модели логистической регрессии. Данный модуль не требует коррекции!

**computeCost.py** – модуль, содержащий функцию `computeCost`, которая необходима для вычисления значения стоимостной функции логистической регрессии.

**gradientDescent.py** – модуль, содержащий функцию `gradientDescent`, которая необходима для выполнения градиентного спуска с целью поиска параметров модели логистической регрессии.

**featureNormalize.py** – модуль, содержащий функцию `featureNormalize`, которая необходима для нормализации признаков. Данный модуль не требует коррекции!

**sigmoid.py** – модуль, содержащий функцию `sigmoid`, которая позволяет вычислить значение сигмоидной функции.

**predict.py** – модуль, содержащий функцию `predict`, которая необходима для предсказания метки класса (0 или 1).

4. Поэтапно выполнить задание, связанное с реализацией и исследованием логистической регрессии.

5. Ответить на вопросы, необходимые для составления отчета по данному практическому заданию. Отчет сдается на проверку в печатной или письменной форме в указанные сроки.

## 2. Логистическая регрессия

При выполнении задания требуется заполнить пустые места программного кода в блоках с комментарием «Ваш код здесь». Данную процедуру необходимо выполнить для следующих функций: `plotData`, `computeCost`, `gradientDescent`, `sigmoid`, `predict`.

1. При решении любой задачи с использованием инструментов машинного обучения важным является понимание структуры анализируемых данных и их визуализация в случае возможности. В настоящем задании предлагается использовать базу данных из файла **`data.txt`**. Данные представляют собой множество объектов, описываемых двумя признаками (оценка студента за первый экзамен и оценка студента за второй экзамен) и меткой (аттестован или не аттестован студент по итогам двух экзаменов). Необходимо обратить внимание на то, что база данных в настоящем задании размечена, а метка принимает дискретный набор из двух значений (0 – не аттестован, 1 – аттестован). Поэтому в рамках настоящего задания рассматривается решение задачи бинарной классификации, а не регрессии, как это было в практическом задании № 1. Завершите программный код в модуле **`plotData.py`**, который позволит выполнять визуализацию данных. Завершение модуля подразумевает под собой написание строчек программного кода, которые позволят вызвать функцию из соответствующего модуля в файле **`main.py`**, позволяя решить определенный кусок настоящего задания. Например, в данном случае завершенный программный код будет выглядеть так, как представлено на рис. 1.

После завершения каждого блока кода интерпретируйте файл **`main.py`** с целью проверки правильности работы соответствующей части задания. Результат визуализации данных с использованием функции `plotData` представлен на рис. 2. В случае успешной интерпретации программного кода разрешается перейти к следующему пункту задания.

2. Завершите программный код в модуле **`computeCost.py`**, который позволит вычислить значение стоимостной функции для логистической регрессии. Формулы, описывающие ее вычисление, представлены в лекции № 2. При выполнении данной части задания могут понадобиться функции из библиотеки **NumPy**, представленные ниже.

`dot` – позволяет вычислить матричное произведение для двумерных массивов и скалярное произведение для одномерных массивов (без комплексного сопряжения).

`sum` – позволяет вычислить сумму элементов вдоль определенной размерности двумерного массива и сумму всех элементов для одномерного массива.

`log` – позволяет вычислить натуральный логарифм от элементов массива.

```python
import matplotlib.pyplot as plt
import numpy as np

def plotData(X, y):
    """
        Функция позволяет выполнить визуализацию данных с маркером
        + для положительных примеров и маркером o для отрицательных
        примеров. X - матрица объекты-признаки размера mx2,
        а y - вектор меток размера mx1, где m - размер базы данных
    """

    # ==================== Ваш код здесь ====================
    # Инструкция: визуализируйте положительные и отрицатедьные
    # примеры на двумерной плоскости, используя маркер + для
    # обозначения положительных примеров и маркер o для обозначения
    # отрицательных примеров

    plt.figure()                                                                    # создание нового окна
    pos = np.where(y == 1)[0]                                                       # извлечение индексов положительных примеров
    neg = np.where(y == 0)[0]                                                       # извлечение индексов отрицательных примеров

    plt.plot(X[pos, 0], X[pos, 1], '+', markersize = 7, markeredgecolor = 'black', markeredgewidth = 2)       # построение положительных примеров
    plt.plot(X[neg, 0], X[neg, 1], 'o', markersize = 7, markeredgecolor = 'black', markerfacecolor = 'yellow') # построение отрицательных примеров
    plt.grid()                                                                      # создание координатной сетки

    # =========================================================
```

Рис. 1. Завершенный программный код для функции `plotData`

При заполнении программного кода в модуле **`computeCost.py`** потребуется вычисление значений сигмоидной функции. Реализуйте ее вычисление в модуле **`sigmoid.py`**. При реализации сигмоидной функции может понадобиться функция `exp` из библиотеки **NumPy**, которая позволяет вычислить значения экспоненциальной функции от элементов массива.

3. Завершите программный код в модуле **`gradientDescent.py`**, который позволит выполнить алгоритм градиентного спуска с целью обучения параметров модели логистической регрессии. Формулы, описывающие реализацию градиентного спуска для логистической регрессии, представлены в лекции № 2. При выполнении данной части задания могут понадобиться следующие функции из библиотеки **NumPy**: `dot` и `transpose`.

`transpose` – позволяет выполнить транспонирование массива. Для одномерного массива данная функция не оказывает никакого действия, а для двумерного массива использование функции соответствует обычному матричному транспонированию.

Так же совершенно будет необходима функция `sigmoid`, реализованная в модуле **`sigmoid.py`**.

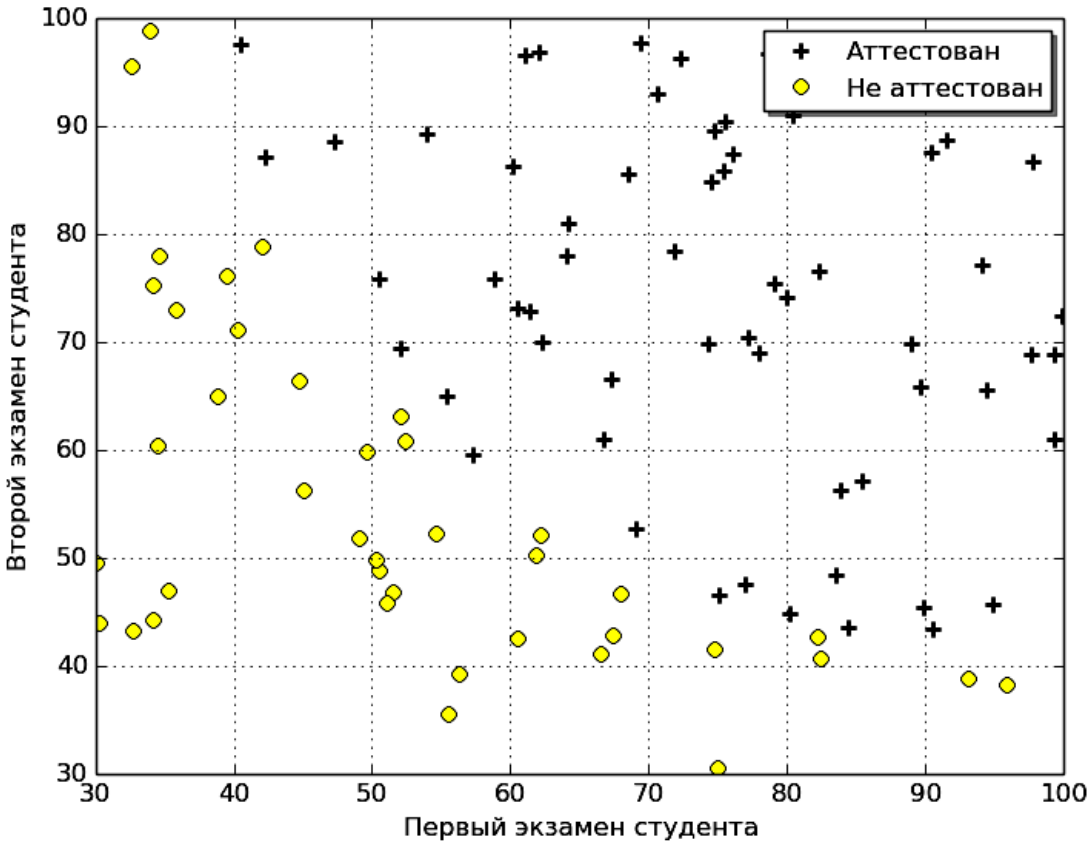


Рис. 2. Результат визуализации тренировочных данных

Обратите внимание на то, что при обучении параметров модели логистической регрессии в файле **`main.py`** используется нормализация признаков, позволяющая выполнить качественную сходимость градиентного спуска к единственному в данном случае минимуму стоимостной функции. Нормализация признаков полностью реализована в модуле **`featureNormalize.py`**, который завершать не требуется.

После обучения параметров модели логистической регрессии с настройками градиентного спуска, заданными по умолчанию в файле **`main.py`**, должен получиться результат, представленный на рис. 3, на котором помимо тренировочных данных изображена найденная граница решения для модели на основе логистической регрессии. Построение границы решения полностью реализовано в функции `plotDecisionBoundary` модуля **`plotDecisionBoundary.py`**. Объекты, которые описываются точками на рис. 3, лежащими выше границы решения, будут отнесены алгоритмом к классу 1 (аттестован), иначе к классу 0 (не аттестован). Необходимо обратить внимание на то, что в процессе принятия решения алгоритмом на тренировочной базе

данных возникают ошибки. Иногда точки, которые относятся к классу 1, классифицируются, как точки класса 0 и наоборот.

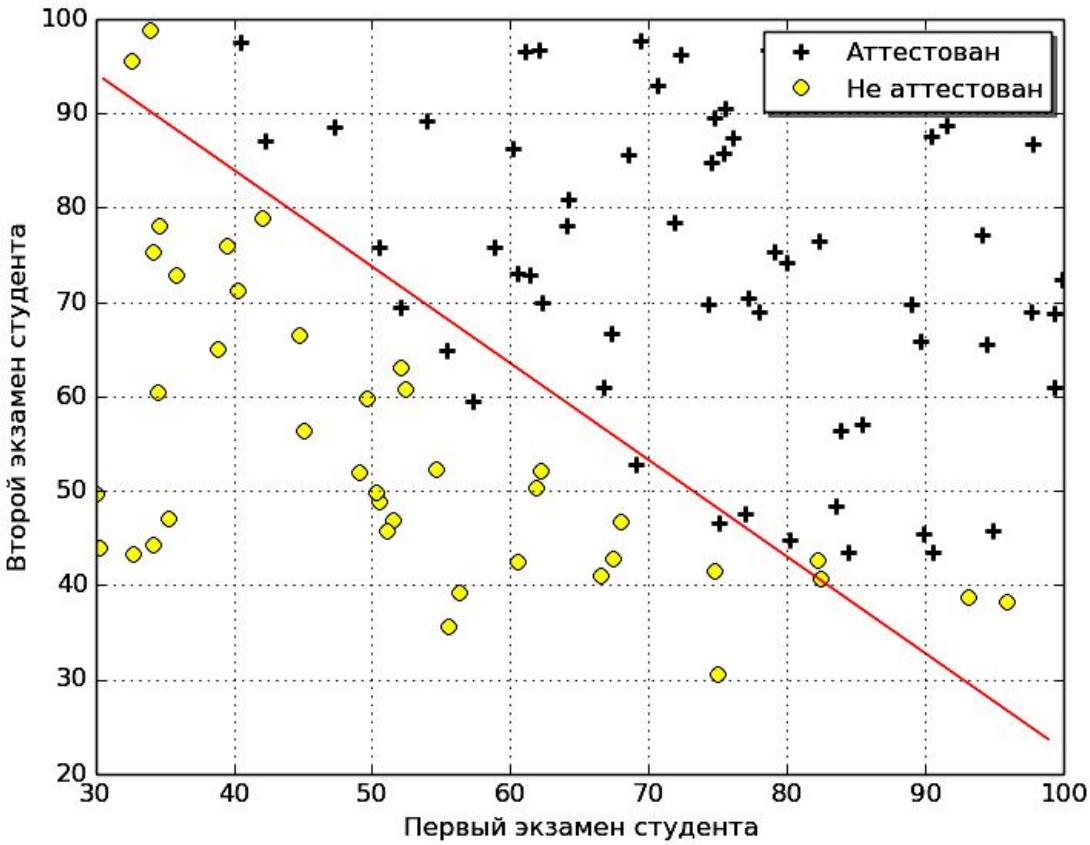


Рис. 3. Результат визуализации тренировочных данных с границей решения для логистической регрессии

Как и в практическом задании № 1, в данном случае возможно провести исследование сходимости градиентного спуска при различных настройках с использованием зависимости представленной на рис. 4.

4. Завершите программный код в модуле **`predict.py`**, который позволит выполнить предсказание метки класса для обученной модели логистической регрессии. В ходе предсказания порог классификатора необходимо выставить равным значению 0.5. При выполнении данной части задания могут понадобиться следующие функции из библиотеки **NumPy**: `dot` и `astype`.

`astype` – позволяет выполнить приведение элементов массива к определенному типу данных.

Так же совершенно будет необходима функция `sigmoid`, реализованная в модуле **`sigmoid.py`**.

5. После завершения предыдущих пунктов вычислите значение вероятности, с которой студент будет аттестован в случае, если его оценка за первый экзамен равна 45, а оценка за второй экзамен равна 85.

Обратите внимание на то, что перед выполнением процедуры предсказания требуется провести нормализацию признаков на соответствующие им математическое ожидание и среднеквадратическое отклонение.

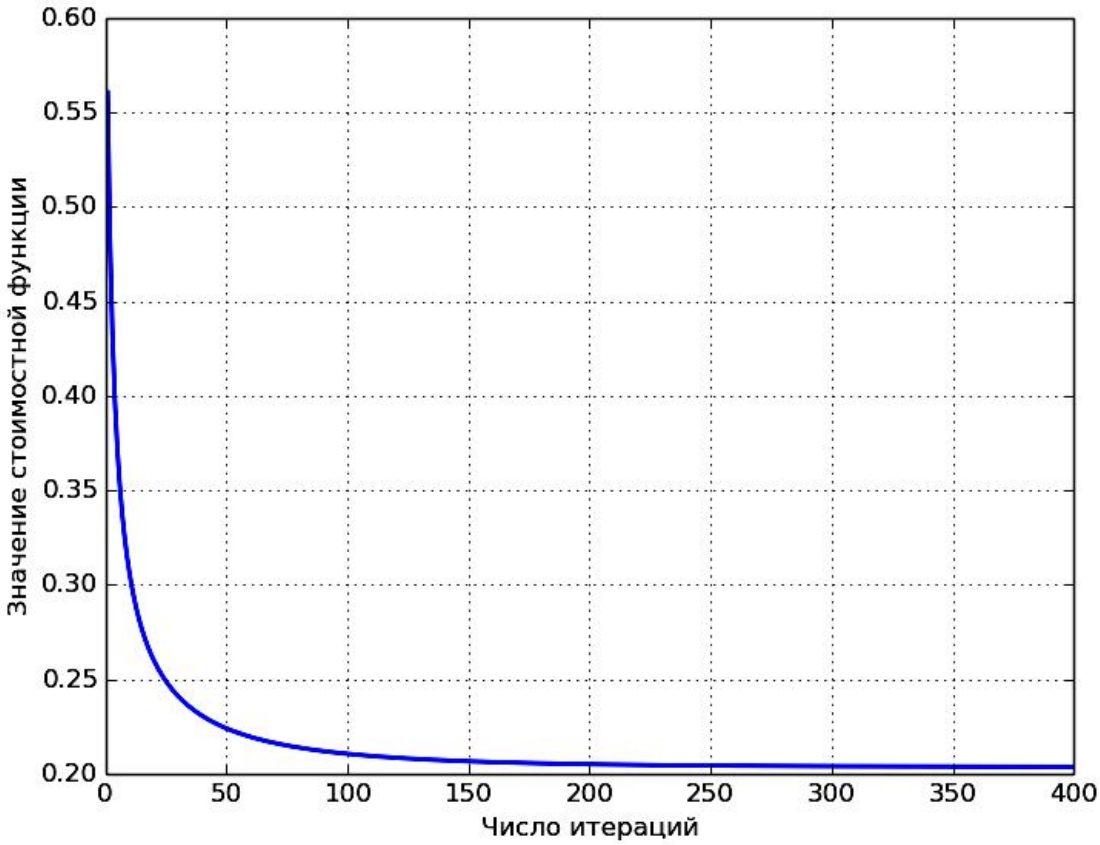


Рис. 4. Пример сходимости градиентного спуска для удачно подобранной скорости сходимости

6. Оцените долю правильных ответов обученной модели логистической регрессии. Необходимо обратить внимание на то, что в русскоязычной литературе доля правильных ответов иногда обозначается словом точность (от англ. accuracy). Последний перевод не всегда является удачным, так как существует другая метрика оценки качества работы классификатора – precision, которая на русский язык так же переводится, как точность. Однако смысл accuracy и precision является абсолютно разным. **Accuracy** равна отношению числа ответов, для которых алгоритм выполнил предсказание метки класса верно, к общему числу объектов в базе данных. **Precision** равна отношению числа объектов, для которых алгоритм верно предсказал метку класса 1, к числу объектов, для которых алгоритм предсказал метку класса 1.

## 3. Вопросы для составления отчета

Используя файл **`main.py`** ответьте на следующие вопросы.

1. Чему равно значение стоимостной функции для случая, когда все параметры модели равны нулю (**35 баллов**)?
2. Чему равны значения параметров обученной модели логистической регрессии для случая, когда параметр сходимости равен 1, а число итераций градиентного спуска равно 400 (**40 баллов**)?
3. Чему равна вероятность аттестации студента в случае, если его оценка за первый экзамен равна 45, а оценка за второй экзамен равна 85 (**15 баллов**) для обученной в вопросе 2 модели?
4. Чему равна доля правильных ответов обученной в вопросе 2 модели логистической регрессии (**5 баллов**)?
5. Опираясь на материал лекции № 2, опишите возможное решение (решения), которые позволят увеличить долю правильных ответов классификатора на основе логистической регрессии применительно к рассматриваемой в настоящем задании базе данных (**20 баллов**). Настоящий вопрос является необязательным, но позволяет заработать дополнительные баллы!