Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Кафедра Информатики

ЛАБОРАТОРНАЯ РАБОТА №1
«Линейная регрессия»

БГУИР 1-40 81 04

Магистрант:
гр. 858642
Кукареко А.В.

Проверил:
Стержанов М. В.

Минск, 2019

ХОД РАБОТЫ

Задание.

Набор данных ex1data1.txt представляет собой текстовый файл, содержащий информацию о населении городов (первое число в строке) и прибыли ресторана, достигнутой в этом городе (второе число в строке). Отрицательное значение прибыли означает, что в данном городе ресторан терпит убытки.

Набор данных ex1data2.txt представляет собой текстовый файл, содержащий информацию о площади дома в квадратных футах (первое число в строке), количестве комнат в доме (второе число в строке) и стоимости дома (третье число).

1. Загрузите набор данных ex1data1.txt из текстового файла.
2. Постройте график зависимости прибыли ресторана от населения города, в котором он расположен.
3. Реализуйте функцию потерь J(θ) для набора данных ex1data1.txt.
4. Реализуйте функцию градиентного спуска для выбора параметров модели. Постройте полученную модель (функцию) совместно с графиком из пункта 2.
5. Постройте трехмерный график зависимости функции потерь от параметров модели (θ0 и θ1) как в виде поверхности, так и в виде изолиний (contour plot).
6. Загрузите набор данных ex1data2.txt из текстового файла.
7. Произведите нормализацию признаков. Повлияло ли это на скорость сходимости градиентного спуска? Ответ дайте в виде графика.
8. Реализуйте функции потерь J(θ) и градиентного спуска для случая многомерной линейной регрессии с использованием векторизации.
9. Покажите, что векторизация дает прирост производительности.
10. Попробуйте изменить параметр α (коэффициент обучения). Как при этом изменяется график функции потерь в зависимости от числа итераций градиентного спуск? Результат изобразите в качестве графика.
11. Постройте модель, используя аналитическое решение, которое может быть получено методом наименьших квадратов. Сравните результаты данной модели с моделью, полученной с помощью градиентного спуска.

Результат выполнения:

1. Загрузите набор данных ex1data1.txt из текстового файла.

```
df1 = pandas.read_csv('ex1data1.txt', header=None, names=['population', 'income'])
```

```
df1.head()
```

	population	income
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

2. Постройте график зависимости прибыли ресторана от населения города, в котором он расположен.

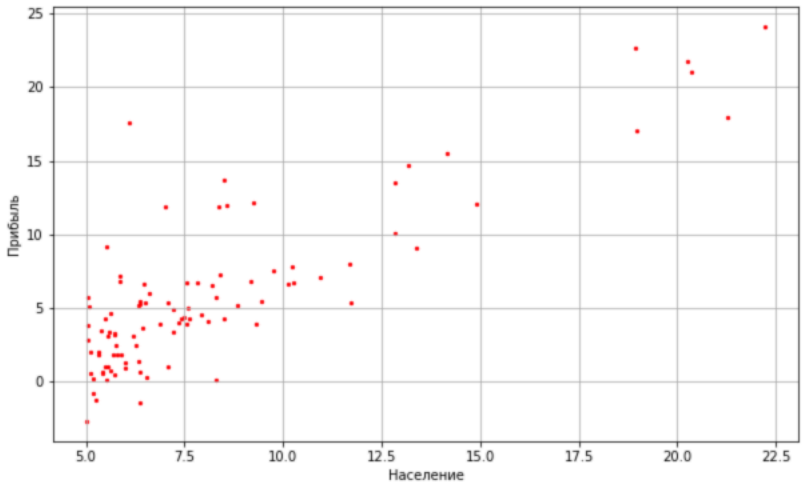

Рисунок 1 - график зависимости прибыли ресторана от населения города.

3. Реализуйте функцию потерь J(θ) для набора данных ex1data1.txt.

Функция гипотеза:

```
def h1(theta, x):
    return theta[0] + theta[1] * x
```

Функция потерь (или стоимости):

```
def cost_func1(theta, X, Y):
    m = len(Y)
    sum = 0
    for i in range(m):
        sum += (h1(theta, X[i]) - Y[i])**2

    return sum/(2*m)
```

4. Реализуйте функцию градиентного спуска для выбора параметров модели. Постройте полученную модель (функцию) совместно с графиком из пункта 2.

Функция градиентного спуска:

```
def der_theta1(theta, X, Y, alpha):
    m = len(Y)
    sum = 0
    for i in range(m):
        sum += (h1(theta, X[i]) - Y[i]) * X[i]

    return (alpha / m) * sum
```

```
def gradient_descent(X, Y, iterations = 200, alpha = 0.01, theta = [0, 0]):
    i = 0
    history = []

    cost = cost_func1(theta, X, Y)
    history.append(np.array([cost, np.array(theta)]))

    for it_number in range(iterations):
        tmptheta = theta
        tmptheta[0] = theta[0] - der_theta0(theta, X, Y, alpha)
        tmptheta[1] = theta[1] - der_theta1(theta, X, Y, alpha)

        cost = cost_func1(theta, X, Y)
        history.append(np.array([cost, np.array(theta)]))

    return np.array(theta), cost, np.array(history)
```

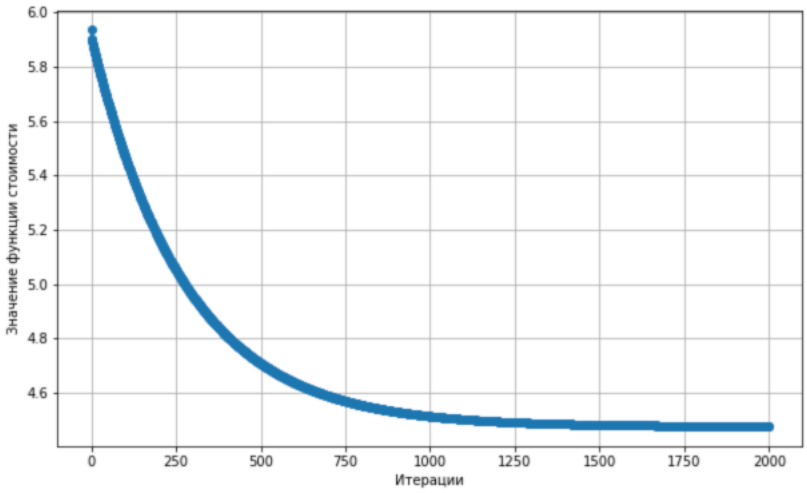

Рисунок 2 – график функции стоимости.

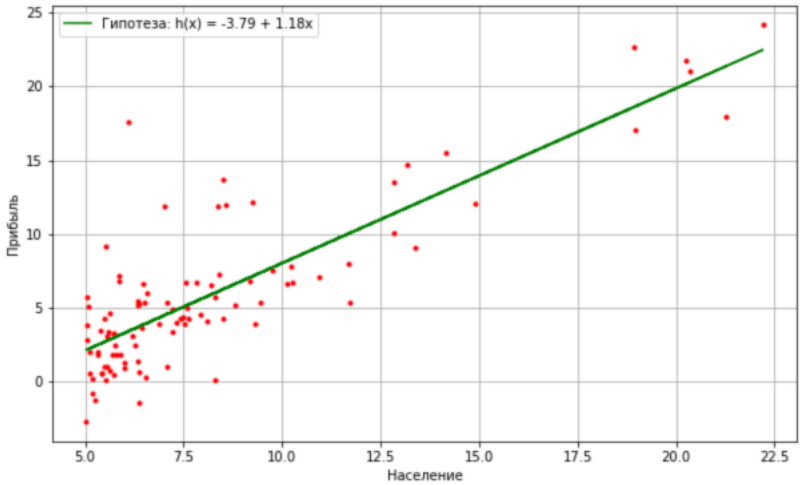

Рисунок 3 – график полученной модели.

5. Постройте трехмерный график зависимости функции потерь от параметров модели (θ0 и θ1) как в виде поверхности, так и в виде изолиний (contour plot).

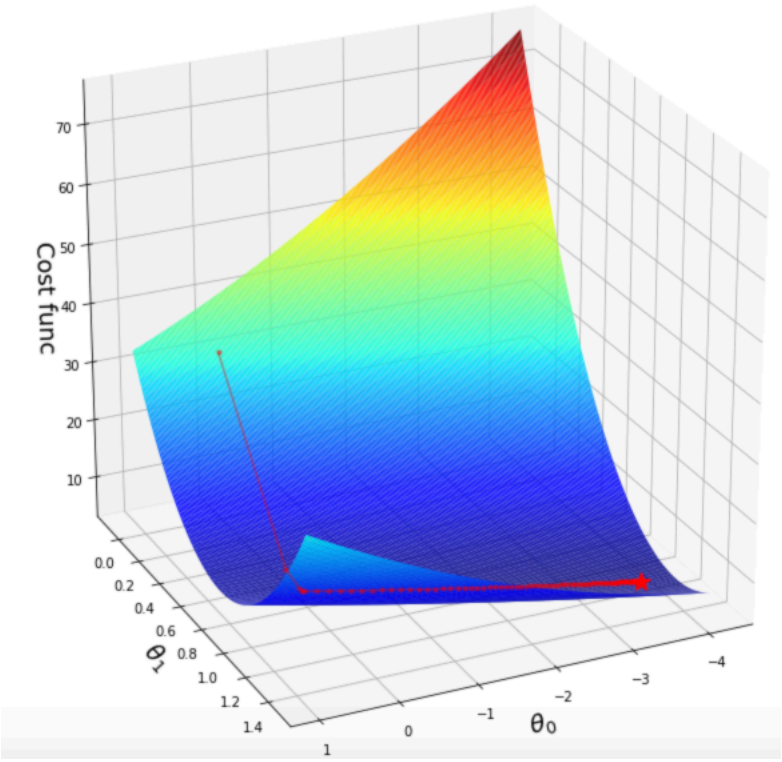

Рисунок 4 - трехмерный график зависимости функции потерь от параметров модели в виде поверхности.

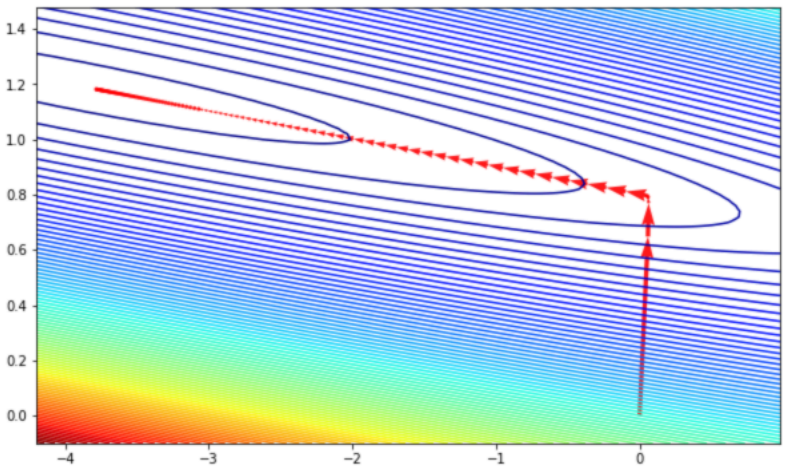

Рисунок 5 - график зависимости функции потерь от параметров модели в виде изолиний.

6. Загрузите набор данных ex1data2.txt из текстового файла.

```
original_df = pandas.read_csv('ex1data2.txt', header=None, names=['area', 'rooms', 'price'])
```

```
original_df.head()
```

	area	rooms	price
0	2104	3	399900
1	1600	3	329900
2	2400	3	369000
3	1416	2	232000
4	3000	4	539900

7. Произведите нормализацию признаков. Повлияло ли это на скорость сходимости градиентного спуска? Ответ дайте в виде графика.

При попытку запустить градиентный спуск на денормализованном датасете, Python выдает ошибки вида: «RuntimeWarning: overflow encountered in double_scalars». Проблема заключается в том, что во время работы алгоритма, значения числе выходят за допустимые пределы определенные стандартом «IEEE 754 - Floating-Point Arithmetic».

Решением проблемы является «нормализация».

8. Реализуйте функции потерь J(θ) и градиентного спуска для случая многомерной линейной регрессии с использованием векторизации.

Векторизация функций гипотезы и стоимости:

```
def h(theta, x):
    return np.dot(x, theta.T)

def cost_f(theta, X, Y):
    m = len(Y)
    results = h(theta, X) - Y
    return (np.dot(results.T, results) / (2*m)).item()
```

Векторизация функции градиентного спуска:

```
def gd_vec(X, Y_un_resh, iterations = 200, alpha = 0.01):
    Y = Y_un_resh.reshape(-1, 1)

    features_count = X.shape[1]
    m = Y.size

    theta = np.zeros([1, features_count])

    theta_history = np.zeros([iterations + 1, features_count])
    cost_history = np.zeros(iterations + 1)

    cost = cost_f(theta, X, Y)

    cost_history[0] = cost
    theta_history[0] = theta

    for it_idx in range(iterations):
        dt = np.dot((h(theta, X) - Y).T, X)
        theta = theta - (alpha / m) * dt

        cost = cost_f(theta, X, Y)

        cost_history[it_idx + 1] = cost
        theta_history[it_idx + 1] = theta

    return theta, cost, theta_history, cost_history
```

9. Покажите, что векторизация дает прирост производительности.

Сравнение производительности проводилось со следующими параметрами:

Alpha = 0.1

Итераций=9000

Таблица 1 – Результаты производительности

Метод	Время в миллисекундах
Без векторизации	5800
С векторизацией	98

Как видно из таблицы 1 – векторизация дает значительный прирост производительности.

10. Попробуйте изменить параметр α (коэффициент обучения). Как при этом изменяется график функции потерь в зависимости от числа итераций градиентного спуск? Результат изобразите в качестве графика.

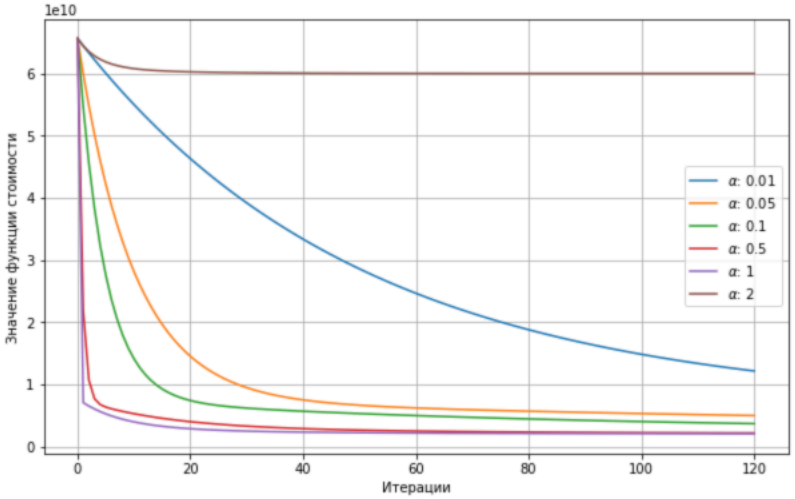

Рисунок 6 - график зависимости функции стоимости от количества итераций.

11. Постройте модель, используя аналитическое решение, которое может быть получено методом наименьших квадратов. Сравните результаты данной модели с моделью, полученной с помощью градиентного спуска.

Функция наименьших квадратов:

```
def normal_equation(X, Y):
    return np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)), X.T), Y)
```

Метод наименьших квадратов находит оптимальное решение дня 47 наборов данных за 5 миллисекунд, в то время как векторизованному

градиентному для того, чтобы достичь похожей точности нужно 9000 итераций, и время работы занимает 98 секунд, что в 20 раз медленнее.

Вывод.

В ходе выполнения лабораторной работы я изучил линейную регрессию для одной и нескольких переменных, изучил способы её реализации на языке python и основы numpy и pandas. Так же в результате экспериментов выяснил, что благодаря современным библиотекам применение векторизации дает значительный прирост производительности. Возникшие проблемы с подсчетами в пункте 7 получилось решить применив нормализацию признаков.