|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| ЛАБОРАТОРНАЯ РАБОТА №1  «Линейная регрессия» | | | |
| Магистрант:  гр. 858341  Жишкевич С.А. | |  | 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.

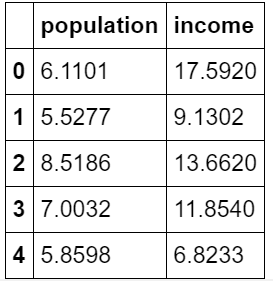
Набор данных ex1data2.txt представляет собой текстовый файл, содержащий информацию о площади дома в квадратных футах (первое число в строке), количестве комнат в доме (второе число в строке) и стоимости дома (третье число).

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 ɑ (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.

1. dataFrame = pandas.read\_csv("ex1data1.txt", header=None, names=["population", "incomes"])
2. **print**(dataFrame.head())



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:

1. **def** calcHypotesis(theta,  x):
2. **return** theta[0] + theta[1] \* x

Функция потерь (или стоимости):

1. **def** costFunction(theta, x, y):
2. **return** sum((calcHypotesis(theta, x\_i) - y\_i) \*\* 2 **for** x\_i, y\_i **in** zip(x, y)) / len(x) \* 2

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

Функция градиентного спуска:

1. **def** derivative\_theta0(theta, x, y, alpha):
2. **return** alpha \* sum(calcHypotesis(theta, x\_i) - y\_i **for** x\_i, y\_i **in** zip(x, y)) / len(x)

5. **def** derivative\_theta1(theta, x, y, alpha):
6. **return** alpha \* sum((calcHypotesis(theta, x\_i) - y\_i) \* x\_i **for** x\_i, y\_i **in** zip(x, y)) / len(x)

9. **def** gradient\_descent (X, Y, iterations = 400, alpha = 0.01 , theta = [0, 0]):
10. i = 0
11. history = []
13. cost = costFunction(theta, X, Y)
14. history.append(np.array([cost, np.array(theta)]))
16. **for** it\_number **in** range(iterations):
17. tmptheta = theta
18. tmptheta[0] = theta[0] - derivative\_theta0(theta, X, Y, alpha)
19. tmptheta[1] = theta[1] - derivative\_theta1(theta, X, Y, alpha)
21. cost = costFunction(theta, X, Y)
22. history.append(np.array([cost, np.array(theta)]))
24. **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).



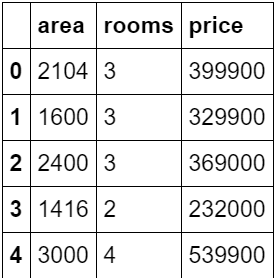
Рисунок 4 - трехмерный график зависимости функции потерь от параметров модели в виде поверхности.



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

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

1. original\_df = pandas.read\_csv('ex1data2.txt', header=None, names=['area', 'rooms', 'price'])
2. **print**(original\_df.head())



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

При попытке 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:

1. **def** h(theta, x):
2. **return** np.dot(x, theta.T)

5. **def** cost\_f(theta, X, Y):
6. m = len(Y)
7. results = h(theta, X) - Y
8. **return** (np.dot(results.T, results) / (2 \* m)).item()

Векторизация функции градиентного спуска:

1. **def** gd\_vec(X, Y\_un\_resh, iterations=200, alpha=0.01):
2. Y = Y\_un\_resh.reshape(-1, 1)
4. features\_count = X.shape[1]
5. m = Y.size
7. theta = np.zeros([1, features\_count])
9. theta\_history = np.zeros([iterations + 1, features\_count])
10. cost\_history = np.zeros(iterations + 1)
12. cost = cost\_f(theta, X, Y)
14. cost\_history[0] = cost
15. theta\_history[0] = theta
17. **for** it\_idx **in** range(iterations):
18. dt = np.dot((h(theta, X) - Y).T, X)
19. theta = theta - (alpha / m) \* dt
21. cost = cost\_f(theta, X, Y)
23. cost\_history[it\_idx + 1] = cost
24. theta\_history[it\_idx + 1] = theta
26. **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 | 6800 |
| С векторизацией | 13 |

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

10. Попробуйте изменить параметр ɑ (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:

1. **def** normal\_equation(X, Y):
2. **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.