Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №1

«Линейная регрессия»

Выполнил: Яловчук Валерий Валерьевич

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**Данные.**

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).

**Выполнение:**

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

file\_path = 'ex1data1.csv'

data\_frames = pd.read\_csv(file\_path)

x = data\_frames['population']

y = data\_frames['profit']

x = list(x) # np.array(x)

y = list(y)

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

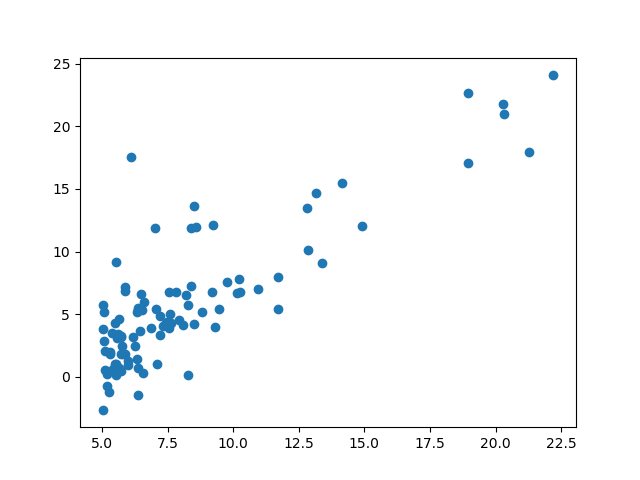
****

Рисунок 1 – график зависимости прибыли ресторана от населения (x – численность популяции, y – прибыль ресторана)

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

def compute\_cost(X, Y, theta):

m = len(X)

diff = []

for i in range(0, m):

val = pow(h0x(X[i], theta) - Y[i], 2)

diff.append(val)

cost = (1 / (2 \* m)) \* sum(diff)

return cost

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

def gradient\_descent(X, Y, theta, iterations, alpha):

"""

From Andrew Ng implementation: without ones vector in X

"""

m = len(X)

J = []

for i in range(iterations):

val = np.zeros(len(theta))

for j in range(0, m):

val[0] += h0x(X[j], theta) - Y[j]

for k in range(1, len(theta)):

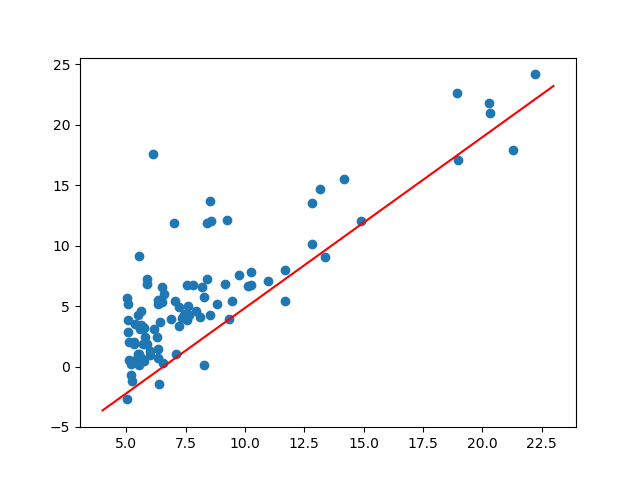
val[k] += (h0x(X[j], theta) - Y[j]) \* X[j]

for z in range(0, len(theta)):

theta[z] = theta[z] - (alpha / m) \* val[z]

J.append(compute\_cost(X, Y, theta))

return [theta, J]

Рисунок 2 – 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):

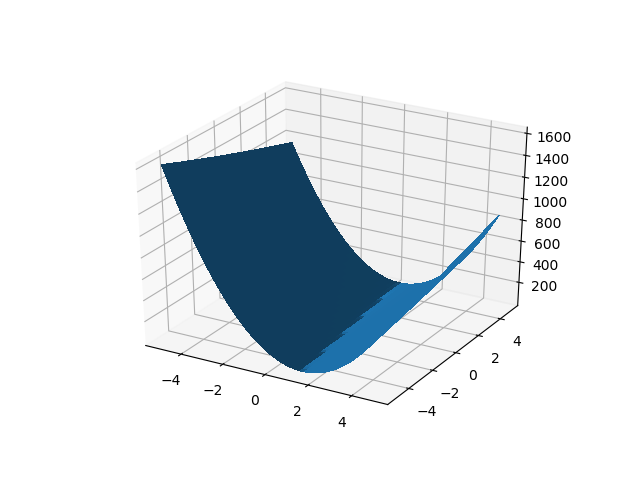


Рисунок 3 – трёхмерный график зависимости потерь от параметров модели в виде поверхности

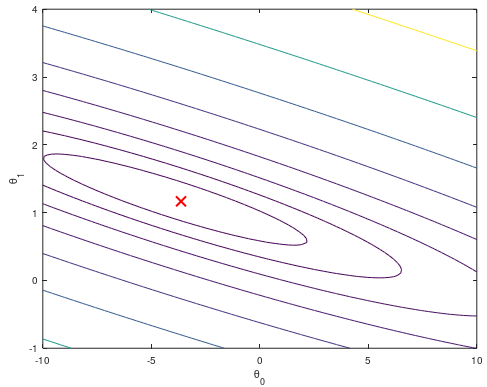


Рисунок 4 – график зависмости потерь от параметров модели в виде поверности

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

file\_path = 'ex1data2.csv'

data = pd.read\_csv(file\_path)

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

Python float types не хватает точности для представления. Возможный вариант решения проблемы - использовать decimal.Decimal класс. Для простоты решения, этот пункт был опущен. Нормализация features должна дать прирост в производительности.

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

def compute\_cost\_vectorized(X, Y, theta):

# J = (1 / (2 \* m)) \* (X \* theta - y)' \* (X \* theta - y); % equally (sum(power(X, 2)))

m = len(X)

temp = (h0x\_vectorized(X, theta) - Y)

return (1 / (2 \* m)) \* np.dot(temp.T, temp)[0][0]

def gradient\_descent\_vectorized(X, Y, theta, iterations, alpha):

m = len(Y)

J\_history = []

for i in range(iterations):

# theta = theta - alpha \* (1/m) \* (((X\*theta) - y)' \* X)'; % Vectorized

h0x = (h0x\_vectorized(X, theta) - Y).T

dt = np.dot(h0x, X).T

a = alpha \* (1 / m) \* dt

theta = theta - a

J\_history.append(compute\_cost\_vectorized(X, Y, theta))

return [theta, J\_history]

9. Покажите, что векторизация дает прирост производительности:

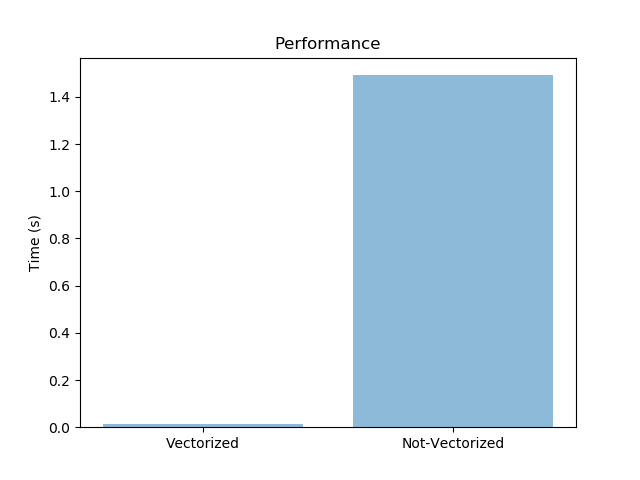


Рисунок 5 – разница между векторизованным и невекторизованным вычислением на одном и том же датасете

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:

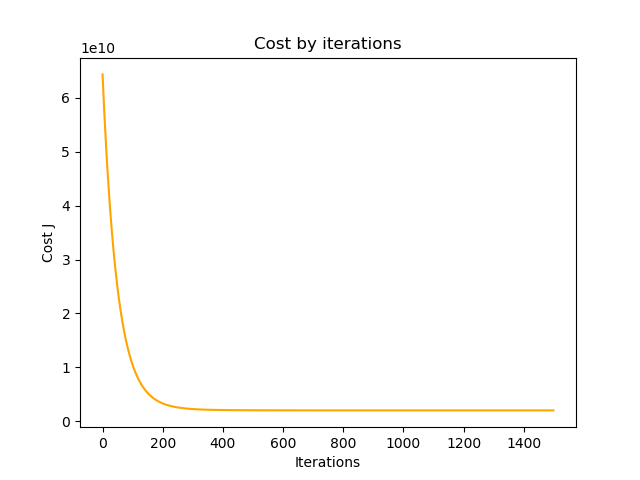


Рисунок 6 – график зависимости функции стоимости от количества итераций (alpha=0.01)

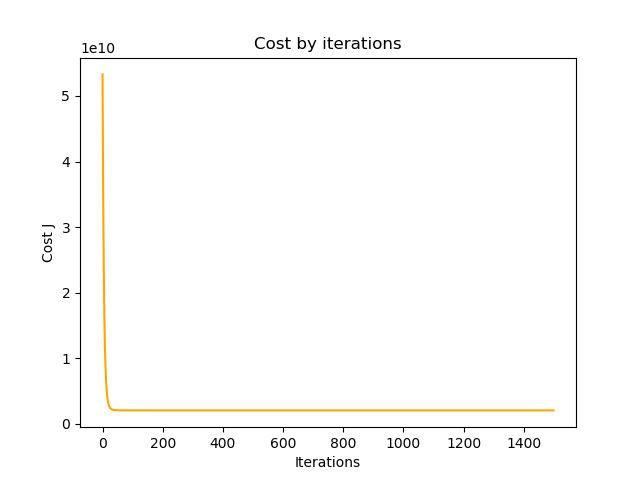


Рисунок 7 – график зависимости функции стоимости от количества итераций (alpha=0.1)

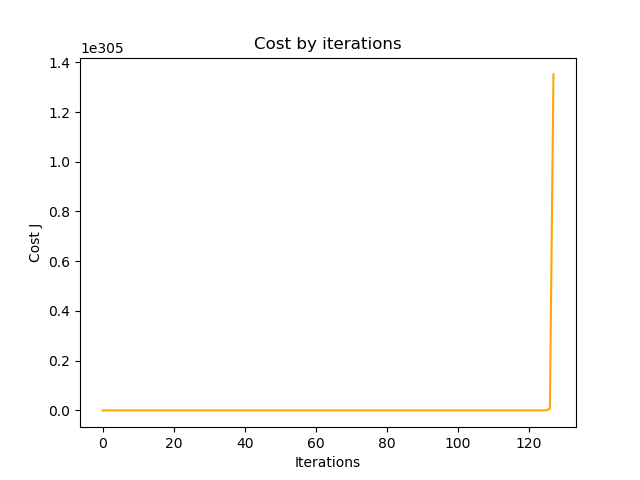


Рисунок 8 – график зависимости функции стоимости от количества итераций (alpha=10)

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

Solution:

[[340412.65957447]

[110631.05027885]

[ -6649.47427082]]

Normal equation:

[[340412.65957447]

[110631.05027885]

[ -6649.47427082]]