

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

Отчет по лабораторной работе № 1

Дисциплина: Компьютерные науки и технологии программирования

Студент: Каримов Тимур Ринатович

Группа: НММбд-02-24

Преподаватель: Бегишев В.О.

МОСКВА 2025 г.

Цель работы:

Теоретическое и практическое освоение фундаментальных основ машинного обучения на примере задачи линейной регрессии.

Выполнение работы

Задание 1:

Формулировка задачи: 1Подберите скорость обучения (eta) и количество итераций

В первую очередь мне необходимо было определить эталонные веса и MSE. Выше в теории было вычисление MSE для МНК, и я использовал функцию MSE для подбора вручную eta и кол-ва итераций. Так же еще проанализировал через график функцию ошибок.

```
n = X.shape[0]

eta = 0.099

n_iter = 10000

W = np.array([1, 0.5])

print(f'Number of objects = {n} \
```

```
\nLearning rate = {eta} \n\nInitial weights = {W} \n')\n\nfor i in range(n_iter):\n\n    y_pred = np.dot(X, W)\n\n    err = calc_mse(y, y_pred)\n\n    # for k in range(W.shape[0]):\n\n    #     W[k] -= eta * (1/n * 2 * X[:, k] @ (y_pred - y))\n\n    W -= eta * (1/n * 2 * np.dot(X.T, y_pred - y))\n\n    if i % 10 == 0:\n\n        eta /= 1.1\n\n        print(f'Iteration #{i}: W_new = {W}, MSE = {round(err, 2)}')
```

Вот результаты:

```

Iteration #40: W_new = [1.56655357e+10 8.63902269e+10], MSE = 7.178363996970921
Iteration #50: W_new = [6.72840520e+10 3.71049202e+11], MSE = 1.864013890660447
Iteration #60: W_new = [4.51163298e+10 2.48801576e+11], MSE = 1.215070417757818
Iteration #70: W_new = [3.88625981e+09 2.14314320e+10], MSE = 1.359071443527208
Iteration #80: W_new = [3.28880920e+07 1.81366663e+08], MSE = 1.548079178797589
Iteration #90: W_new = [ 18555.63128774 102084.47113512], MSE = 843313385532.35
Iteration #100: W_new = [45.31096125 5.89757323], MSE = 713.56
Iteration #110: W_new = [44.97150548 3.82900096], MSE = 43.97
Iteration #120: W_new = [44.99454145 3.82482324], MSE = 43.97
Iteration #130: W_new = [45.01036235 3.82195436], MSE = 43.97
Iteration #140: W_new = [45.02151276 3.8199324 ], MSE = 43.97
Iteration #150: W_new = [45.02955794 3.81847353], MSE = 43.97
Iteration #160: W_new = [45.03548742 3.81739831], MSE = 43.97
Iteration #170: W_new = [45.03994268 3.81659042], MSE = 43.97
Iteration #180: W_new = [45.04334935 3.81597268], MSE = 43.97
Iteration #190: W_new = [45.04599589 3.81549277], MSE = 43.97
Iteration #200: W_new = [45.04808175 3.81511453], MSE = 43.97
...
Iteration #9980: W_new = [45.05874864 3.81318025], MSE = 43.97
Iteration #9990: W_new = [45.05874864 3.81318025], MSE = 43.97
5.7875 43.96875000000001
Аналитическое решение (МНК): [45.0625 3.8125]

```

{#fig:01 width=70%} Заметим, что MSE для МНК (в предпоследней строке) почти совпадает с MSE для градиентного спуска.

**Задание 2:

Формулировка задачи: 2: В этом коде мы избавляемся от итераций по весам, но тут есть ошибка, исправьте ее.

Исходный код с ошибкой:

```

n = X.shape[0]

eta = 1e-2

n_iter = 100

W = np.array([1, 0.5])

print(f'Number of objects = {n} \
      \nLearning rate = {eta} \
      \nInitial weights = {W} \n')

```

```
for i in range(n_iter):  
  
    y_pred = np.dot(X, W)  
  
    err = calc_mse(y, y_pred)  
  
    W -= eta * (1/n * 2 * np.dot(X, y_pred - y))  
  
    if i % 10 == 0:  
  
        print(f'Iteration #{i}: W_new = {W}, MSE = {round(err,2)}')
```

Ошибка была в этой строке `W -= eta * (1/n * 2 * np.dot(X, y_pred - y))`. Ошибка в коде заключается в неправильном использовании матричного умножения при обновлении весов. Вместо транспонирования матрицы признаков `X` используется исходная матрица, что приводит к несоответствию размерностей и некорректному вычислению градиента.

Рабочий код

```
n = X.shape[0]  
  
eta = 1e-2  
n_iter = 100  
  
W = np.array([1, 0.5])  
print(f'Number of objects = {n} \\  
      \nLearning rate = {eta} \\  
      \nInitial weights = {W} \n')  
  
for i in range(n_iter):  
  
    y_pred = np.dot(X, W)  
  
    err = calc_mse(y, y_pred)
```

```
W -= eta * (1/n * 2 * np.dot(X.T, y_pred - y))

if i % 10 == 0:

    print(f'Iteration #{i}: W_new = {W}, MSE = {round(err,2)}')
```

Задание 3:

Формулировка задачи 3: Вместо того, чтобы задавать количество итераций, задайте другое условие остановки алгоритма - когда веса перестают изменяться меньше определенного порога ϵ

Реализация:

```
n = X.shape[0]

eta = 1e-2

epsilon = 1e-6

max_iter = 10000

W = np.array([1.0, 0.5])

print(f'Number of objects = {n} \nLearning rate = {eta} \nInitial weights = {W} \n')

for i in range(max_iter):

    W_old = W.copy()

    y_pred = np.dot(X, W)

    gradient = (2/n) * np.dot(X.T, y_pred - y)

    W -= eta * gradient

    weight_change = np.linalg.norm(W - W_old)

    if weight_change < epsilon:
```

```
        print(f'Iteration #{i}: W = {W}, изменение весов {weight_change:.6f} <
{epsilon}')

        break

    if i % 10 == 0:

        err = calc_mse(y, y_pred)

        print(f'Iteration #{i}: W = {W}, MSE = {err:.2f}, изменение весов =
{weight_change:.6f}')

    else:

        print(f'Достигнуто максимальное количество итераций {max_iter}')
    ...
```

Я добавил переменную *epsilon*, затем в функцию градиента и сохранял значения весов, чтобы у меня отслеживалось отклонение при каждой итерации. При достижении отклонения меньше epsilon цикл обрывался.

![Результат для задания 2](image/20250918233201.png){#fig:02 width=70%}