Amir Hossein Eyvazkhani

1747696

Exercise 2

Group 1

Task 1)

a) $\theta^{*} = \underset{\theta}{\arg\min} \left(L(\theta) \right) = \frac{1}{N} \sum_{n=1}^{N} \left(f_{\theta}(x_n) - y_n \right)^2$

$X = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}^T$

$Y = (3 \; 4 \; 6)^T$

$A = X^T X = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix}$

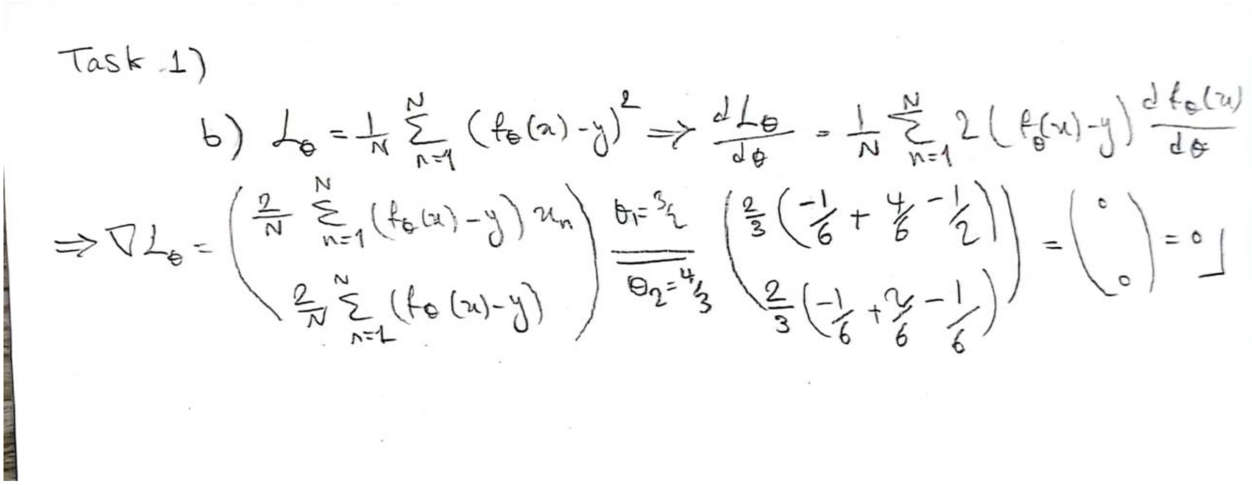
$B = X^T Y = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 29 \\ 13 \end{pmatrix}$

Solve $A\theta^{*} = B$ $\longrightarrow$ Guassian elimination

$\left(\begin{array}{cc|c} 14 & 6 & 29 \\ 6 & 3 & 13 \end{array} \right) = \theta^{*} \xrightarrow[\text{echelon form}]{\text{row}} \theta^{*} = \begin{pmatrix} 3/2 \\ 4/3 \end{pmatrix}$ Code in numpy

H:\Uni\WiSe 2024\ML\Exercises\2\Task1.py

```python
import numpy as np

x = np.array([[1,1] , [2,1], [3,1]])
y = np.array([3,4,6])
theta = np.zeros(2)

x_t = np.transpose(x)
a = x_t@x
b = x_t@y

solution = np.linalg.solve(a, b)
print(solution)


```



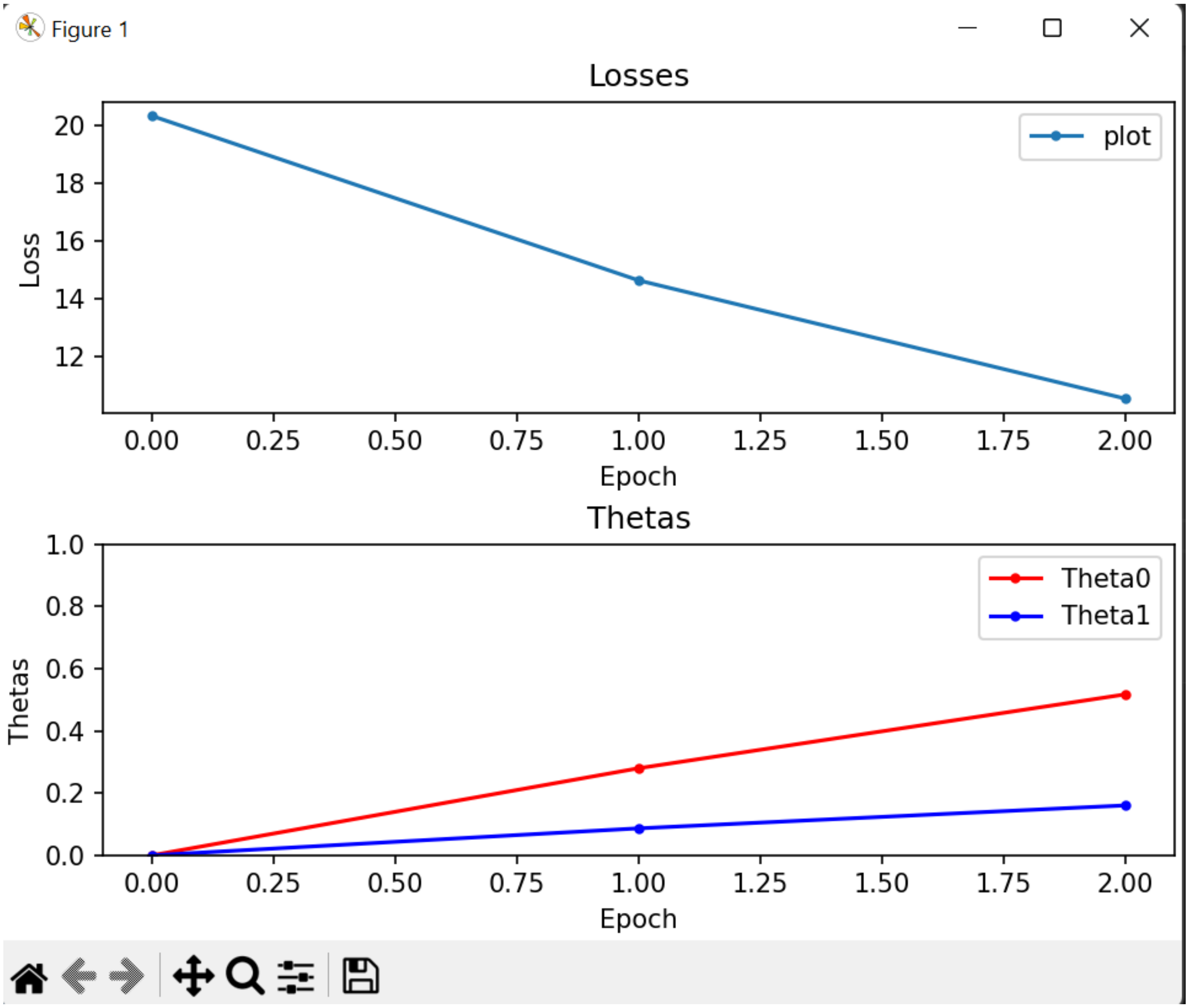
Task 1)

b) $L_\theta = \frac{1}{N}\sum_{n=1}^{N}\left(f_\theta(x)-y\right)^2 \Rightarrow \frac{dL_\theta}{d\theta} = \frac{1}{N}\sum_{n=1}^{N} 2\left(f_\theta(x)-y\right)\frac{df_\theta(x)}{d\theta}$

$$\Rightarrow \nabla L_\theta = \begin{pmatrix} \frac{2}{N}\sum_{n=1}^{N}\left(f_\theta(x)-y\right)x_n \\ \frac{2}{N}\sum_{n=1}^{N}\left(f_\theta(x)-y\right) \end{pmatrix} \underset{\theta_2=4/3}{\overset{\theta_1=3/2}{=\!=}} \begin{pmatrix} \frac{2}{3}\left(\frac{-1}{6}+\frac{4}{6}-\frac{1}{2}\right) \\ \frac{2}{3}\left(\frac{-1}{6}+\frac{2}{6}-\frac{1}{6}\right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0 \;\lrcorner$$

H:\Uni\WiSe 2024\ML\Exercises\2\Task2.py

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

epochs = 3
lr = 0.01 # learning rate
x = np.array([[1,1] , [2,1], [3,1]])
y = np.array([3,4,6]).reshape((3,1))
theta = np.zeros(2).reshape((2,1))

losses = []
thetas = []
for i in range(epochs):
    y_predict = x@theta
    loss = np.mean(
        (y_predict - y)**2
    )
    losses.append(loss)
    thetas.append(theta.copy())
    if i == 2:
        break
    grad_theta_0 = (
        2 * np.sum(x.T @ (y_predict - y))
        ) / len(x)
    grad_theta_1 = (
        2 * np.sum((y_predict - y))
        ) / len(x)
    theta[0] = theta[0] - lr*grad_theta_0
    theta[1] = theta[1] - lr*grad_theta_1

thetas = np.array(thetas).reshape(3, 2)
epoch_range = list(range(epochs))

fig, ax = plt.subplots(nrows=2, ncols=1, layout='constrained')
# plot Losses
ax[0].set_xlabel('Epoch')
ax[0].set_ylabel('Loss')
ax[0].plot(epoch_range, losses, ".-", label='plot')
ax[0].legend()
ax[0].set_title("Losses")

# plot Thetas
ax[1].set_xlabel('Epoch')
ax[1].set_ylabel('Thetas')
ax[1].plot(epoch_range, thetas[:,0], ".-r", label='Theta0')
ax[1].plot(epoch_range, thetas[:,1], ".-b", label='Theta1')
ax[1].legend()
ax[1].set_ylim(0, 1)
ax[1].set_title("Thetas")

plt.show()
```



Task 3) $\forall A \in R^{N \times M}, \theta \in R^{M}$:

$$\theta^T A^T A \theta = (A\theta)^T (A\theta) = \| A\theta \|_2^2 \geq 0 \quad \lrcorner$$