# Pranav Submission

# 1a - Two parameters need to be learned, they are w & b

1b) $y = x_1 + wx_2 + b$.

$$L = \frac{1}{2n} \sum (\hat{y}^{[i]} - y^{[i]})^2$$

Considering it as scalar for the purpose of derivation.

$$L = \frac{1}{2n} (\hat{y} - y)^2$$

$$L = \frac{1}{2n} (x_1 + wx_2 + b - y)^2$$

$$\frac{\partial L}{\partial w} = \frac{\partial}{\partial w} \cdot \frac{1}{2n} \cdot (x_1 + wx_2 + b - y)^2$$

Applying chain rule: $\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \cdot g'(x)$.

Since $\frac{d}{dx}(x^2) = 2x$

$$= \frac{1}{2n} \cdot 2 \cdot (x_1 + wx_2 + b - y) \cdot \frac{\partial}{\partial w} \cdot (x_1 + wx_2 + b - y)$$

$$= \frac{1}{n} \cdot (x_1 + wx_2 + b - y) \cdot \frac{\partial}{\partial w} (x_1 + wx_2 + b - y)$$

Since $\frac{d}{dx} ax = a$ & $\frac{d}{dx}(\text{constant}) = 0$.

$$= \frac{1}{n} (x_1 + wx_2 + b - y) \cdot x_2 \quad \left\{ \text{Since } \frac{\partial}{\partial w}(wx_2) = x_2 \right\}$$

$$\frac{\partial L}{\partial w} = \frac{1}{n} (\hat{y} - y) \cdot x_2$$

1c) $y = x_1 + w x_2 + b$.

$$L = \frac{1}{2n} \sum (\hat{y}^{(i)} - y^{(i)})^2$$

considering it as scalar for the purpose of derivation

$$L = \frac{1}{2n} (\hat{y} - y)^2$$

$$= \frac{1}{2n} (x_1 + w x_2 + b - y)^2$$

~~App~~ $$\frac{\partial L}{\partial b} = \frac{\partial}{\partial b} \frac{1}{2n} (x_1 + w x_2 + b - y)^2$$

Applying chain rule: $\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \cdot g'(x)$

Since $\frac{d}{dx}(x^2) = 2x$

$$= \frac{1}{\cancel{2}n} \cdot \cancel{2} \cdot (x_1 + w x_2 + b - y) \cdot \frac{\partial}{\partial b}(x_1 + w x_2 + b - y)$$

Since $\frac{d}{dx}(x) = 1$ & $\frac{d}{dx}(\text{constant}) = 0$.

$$= \frac{1}{n} \cdot (x_1 + w x_2 + b - y) \cdot 1$$

$$\frac{\partial L}{\partial b} = \frac{1}{n} (\hat{y} - y) \cdot 1$$

$$\frac{\partial L}{\partial b} = \frac{1}{n} (\hat{y} - y)$$

1d) 1. Initialize $w := 0 \in \mathbb{R}^m, b := 0$.

2. For every training epoch:

A. For every $\langle x_1^{[i]}, x_2^{[i]}, y^{[i]} \rangle \in D$.

a) $\hat{y}^{[i]} = \sigma(x_1^{[i]} + w x_2^{[i]} + b)$ $\left\{ \begin{array}{l} \text{Activation function} \\ \sigma(x) = x \end{array} \right\}$

b) $\frac{\partial L}{\partial w} = \frac{1}{n}(\hat{y}^{[i]} - y^{[i]}) x_2^{[i]}$.

$\frac{\partial L}{\partial b} = \frac{1}{n}(\hat{y}^{[i]} - y^{[i]})$.

c) $w := w + \eta \times \left( +\frac{1}{n}(\hat{y}^{[i]} - y^{[i]}) \cdot x_2^{[i]} \right)$

$b := b + \eta \times \left( +\frac{1}{n}(\hat{y}^{[i]} - y^{[i]}) \right)$

learning rate

negative gradient

## 1e solution

```
import pandas as pd
import matplotlib.pyplot  as plt
import torch
%matplotlib inline
```

```
df = pd.read_csv('./linreg-data.csv', index_col=0)
```

```
# Assign features and target

X1 = torch.tensor(df[['x1']].values, dtype=torch.float)
X2 = torch.tensor(df[['x2']].values, dtype=torch.float)
```

```
y = torch.tensor(df['y'].values, dtype=torch.float)

# Shuffling & train/test split

torch.manual_seed(123)
shuffle_idx = torch.randperm(y.size(0), dtype=torch.long)

X1, X2, y = X1[shuffle_idx], X2[shuffle_idx], y[shuffle_idx]

percent70 = int(shuffle_idx.size(0)*0.7)

X1_train, X1_test = X1[shuffle_idx[:percent70]], X1[shuffle_idx[percent70:]]
X2_train, X2_test = X2[shuffle_idx[:percent70]], X2[shuffle_idx[percent70:]]
y_train, y_test = y[shuffle_idx[:percent70]], y[shuffle_idx[percent70:]]

# Normalize (mean zero, unit variance)

mu1, sigma1 = X1_train.mean(dim=0), X1_train.std(dim=0)
mu2, sigma2 = X2_train.mean(dim=0), X2_train.std(dim=0)
X1_train = (X1_train - mu1) / sigma1
X1_test = (X1_test - mu1) / sigma1

X2_train = (X2_train - mu2) / sigma2
X2_test = (X2_test - mu2) / sigma2
```

```
X1_train.shape
```

```
torch.Size([700, 1])
```

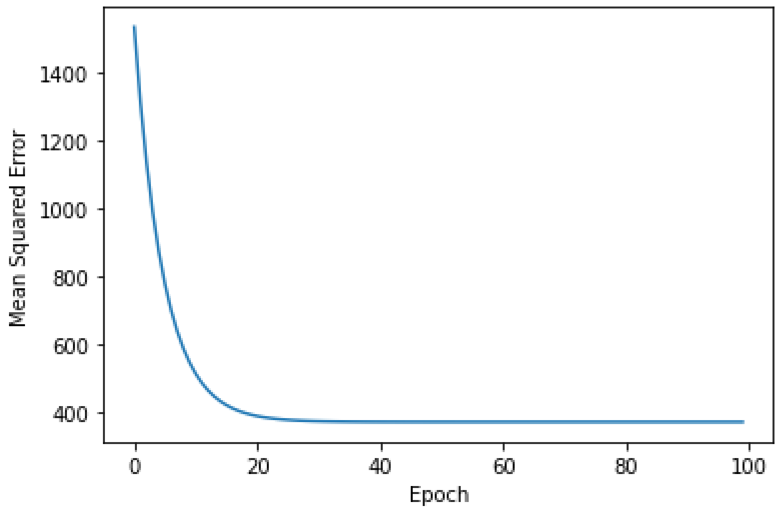
```
class LinearRegression():
    def __init__(self, num_features):
        self.num_features = num_features
        self.weights = torch.zeros(num_features, 1,
                                   dtype=torch.float)
        self.bias = torch.zeros(1, dtype=torch.float)

    def forward(self, x1, x2):
        netinputs = torch.add(torch.add(x1, torch.mm(x2, self.weights)), self.bias)
        activations = netinputs
        return activations.view(-1)

    def backward(self, x1, x2, yhat, y):

        grad_loss_yhat = 2*(yhat - y)

        grad_yhat_weights = x2
        grad_yhat_bias = 1.

        # Chain rule: inner times outer
        grad_loss_weights =  torch.mm(grad_yhat_weights.t(),
```

```
                                                    grad_loss_yhat.view(-1, 1)) / y.size(0)

        grad_loss_bias = torch.sum(grad_yhat_bias*grad_loss_yhat) / y.size(0)

        # return negative gradient
        return (-1)*grad_loss_weights, (-1)*grad_loss_bias


####################################################
##### Training and evaluation wrappers
###################################################

def loss(yhat, y):
    return torch.mean((yhat - y)**2)


def train(model, x1, x2, y, num_epochs, learning_rate=0.01):
    cost = []
    for e in range(num_epochs):

        #### Compute outputs ####
        yhat = model.forward(x1, x2)

        #### Compute gradients ####
        negative_grad_w, negative_grad_b = model.backward(x1, x2, yhat, y)

        #### Update weights ####
        model.weights += learning_rate * negative_grad_w
        model.bias += learning_rate * negative_grad_b

        #### Logging ####
        yhat = model.forward(x1, x2) # not that this is a bit wasteful here
        curr_loss = loss(yhat, y)
        print('Epoch: %03d' % (e+1), end="")
        print(' | MSE: %.5f' % curr_loss)
        cost.append(curr_loss)

    return cost


model = LinearRegression(num_features=X1_train.size(1))
cost = train(model,
             X1_train, X2_train, y_train,
             num_epochs=100,
             learning_rate=0.05)
```

```
Epoch: 049 | MSE: 372.06393
Epoch: 050 | MSE: 372.05490
Epoch: 051 | MSE: 372.04755
Epoch: 052 | MSE: 372.04160
Epoch: 053 | MSE: 372.03677
Epoch: 054 | MSE: 372.03287
Epoch: 055 | MSE: 372.02972
Epoch: 056 | MSE: 372.02713
Epoch: 057 | MSE: 372.02505
Epoch: 058 | MSE: 372.02341
Epoch: 059 | MSE: 372.02203
Epoch: 060 | MSE: 372.02090
Epoch: 061 | MSE: 372.02005
Epoch: 062 | MSE: 372.01929
Epoch: 063 | MSE: 372.01874
Epoch: 064 | MSE: 372.01825
Epoch: 065 | MSE: 372.01782
Epoch: 066 | MSE: 372.01749
Epoch: 067 | MSE: 372.01730
Epoch: 068 | MSE: 372.01709
Epoch: 069 | MSE: 372.01691
Epoch: 070 | MSE: 372.01678
Epoch: 071 | MSE: 372.01669
Epoch: 072 | MSE: 372.01657
Epoch: 073 | MSE: 372.01651
Epoch: 074 | MSE: 372.01642
Epoch: 075 | MSE: 372.01642
Epoch: 076 | MSE: 372.01633
Epoch: 077 | MSE: 372.01633
Epoch: 078 | MSE: 372.01630
Epoch: 079 | MSE: 372.01624
Epoch: 080 | MSE: 372.01630
Epoch: 081 | MSE: 372.01627
Epoch: 082 | MSE: 372.01624
Epoch: 083 | MSE: 372.01624
Epoch: 084 | MSE: 372.01624
Epoch: 085 | MSE: 372.01620
Epoch: 086 | MSE: 372.01624
Epoch: 087 | MSE: 372.01624
Epoch: 088 | MSE: 372.01624
Epoch: 089 | MSE: 372.01620
Epoch: 090 | MSE: 372.01620
Epoch: 091 | MSE: 372.01624
Epoch: 092 | MSE: 372.01624
Epoch: 093 | MSE: 372.01620
Epoch: 094 | MSE: 372.01620
Epoch: 095 | MSE: 372.01624
Epoch: 096 | MSE: 372.01617
Epoch: 097 | MSE: 372.01620
Epoch: 098 | MSE: 372.01624
Epoch: 099 | MSE: 372.01620
Epoch: 100 | MSE: 372.01620
```

## Evluating & Plotting

```
plt.plot(range(len(cost)), cost)
plt.ylabel('Mean Squared Error')
plt.xlabel('Epoch')
plt.show()
```




Start coding or generate with AI.

```
Train MSE: 372.01620
Test MSE: 409.19485
```

```
print('Weights', model.weights)
print('Bias', model.bias)
```

```
Weights tensor([[37.8872]])
Bias tensor([-0.5464])
```

1f The second weight and the bias terms are both similar to the model in the lecture video, however even in the first video the final weight of x1 was 0.36, the value we have for x1 is 1, so the accuracies were comparable to the model in lecture video

## Bonus Question

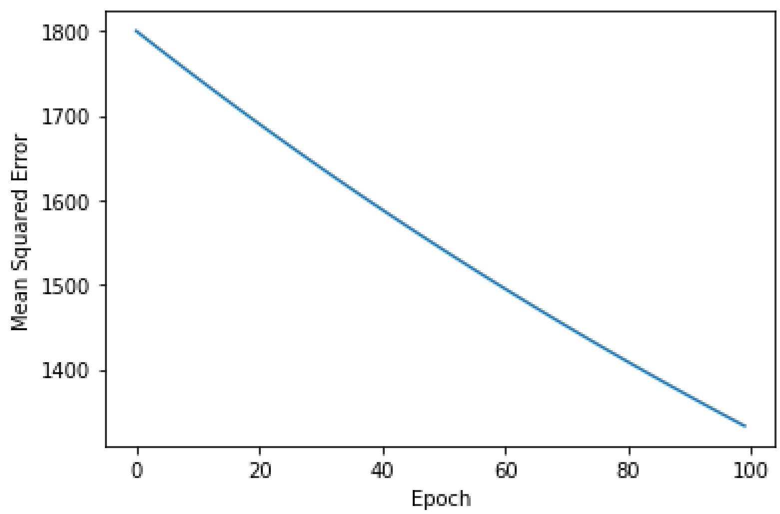
Learning rate - 0.001

```
model = LinearRegression(num_features=X1_train.size(1))
cost = train(model,
             X1_train, X2_train, y_train,
             num_epochs=100,
             learning_rate=0.001)
```

```
Epoch: 043 | MSE: 1579.30396
Epoch: 044 | MSE: 1574.48645
Epoch: 045 | MSE: 1569.68835
Epoch: 046 | MSE: 1564.90906
Epoch: 047 | MSE: 1560.14929
Epoch: 048 | MSE: 1555.40820
Epoch: 049 | MSE: 1550.68604
Epoch: 050 | MSE: 1545.98291
```

```
Epoch: 093 | MSE: 1360.54468
Epoch: 094 | MSE: 1356.60022
Epoch: 095 | MSE: 1352.67151
Epoch: 096 | MSE: 1348.75842
Epoch: 097 | MSE: 1344.86084
Epoch: 098 | MSE: 1340.97900
Epoch: 099 | MSE: 1337.11255
Epoch: 100 | MSE: 1333.26135
```

```
plt.plot(range(len(cost)), cost)
plt.ylabel('Mean Squared Error')
plt.xlabel('Epoch')
plt.show()
```



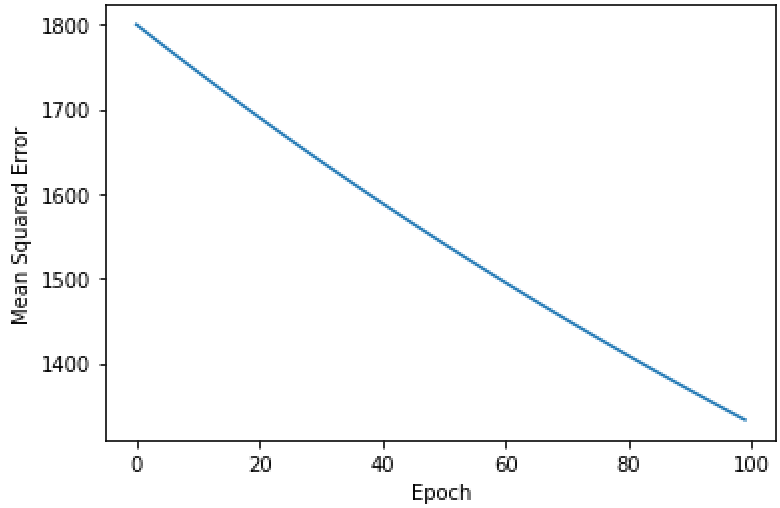

Learning rate - 0.01

```
model = LinearRegression(num_features=X1_train.size(1))
cost = train(model,
             X1_train, X2_train, y_train,
             num_epochs=100,
             learning_rate=0.001)
```

```
Epoch: 001 | MSE: 1800.05798
Epoch: 002 | MSE: 1794.35962
Epoch: 003 | MSE: 1788.68408
Epoch: 004 | MSE: 1783.03125
Epoch: 005 | MSE: 1777.40088
Epoch: 006 | MSE: 1771.79285
Epoch: 007 | MSE: 1766.20752
Epoch: 008 | MSE: 1760.64429
Epoch: 009 | MSE: 1755.10327
Epoch: 010 | MSE: 1749.58423
Epoch: 011 | MSE: 1744.08728
Epoch: 012 | MSE: 1738.61230
Epoch: 013 | MSE: 1733.15930
Epoch: 014 | MSE: 1727.72791
Epoch: 015 | MSE: 1722.31824
```

```
Epoch: 016 | MSE: 1716.93005
Epoch: 017 | MSE: 1711.56360
Epoch: 018 | MSE: 1706.21826
Epoch: 019 | MSE: 1700.89441
Epoch: 020 | MSE: 1695.59180
Epoch: 021 | MSE: 1690.31030
Epoch: 022 | MSE: 1685.05005
Epoch: 023 | MSE: 1679.81055
Epoch: 024 | MSE: 1674.59216
Epoch: 025 | MSE: 1669.39429
Epoch: 026 | MSE: 1664.21729
Epoch: 027 | MSE: 1659.06104
Epoch: 028 | MSE: 1653.92542
Epoch: 029 | MSE: 1648.81018
Epoch: 030 | MSE: 1643.71558
Epoch: 031 | MSE: 1638.64111
Epoch: 032 | MSE: 1633.58679
Epoch: 033 | MSE: 1628.55273
Epoch: 034 | MSE: 1623.53870
Epoch: 035 | MSE: 1618.54480
Epoch: 036 | MSE: 1613.57068
Epoch: 037 | MSE: 1608.61646
Epoch: 038 | MSE: 1603.68213
Epoch: 039 | MSE: 1598.76746
Epoch: 040 | MSE: 1593.87256
Epoch: 041 | MSE: 1588.99683
Epoch: 042 | MSE: 1584.14075
Epoch: 043 | MSE: 1579.30396
Epoch: 044 | MSE: 1574.48645
Epoch: 045 | MSE: 1569.68835
Epoch: 046 | MSE: 1564.90906
Epoch: 047 | MSE: 1560.14929
Epoch: 048 | MSE: 1555.40820
Epoch: 049 | MSE: 1550.68604
Epoch: 050 | MSE: 1545.98291
Epoch: 051 | MSE: 1541.29834
Epoch: 052 | MSE: 1536.63245
Epoch: 053 | MSE: 1531.98535
Epoch: 054 | MSE: 1527.35681
Epoch: 055 | MSE: 1522.74683
Epoch: 056 | MSE: 1518.15503
Epoch: 057 | MSE: 1513.58167
Epoch: 058 | MSE: 1509.02625
```

```python
plt.plot(range(len(cost)), cost)
plt.ylabel('Mean Squared Error')
plt.xlabel('Epoch')
plt.show()
```




Learning rate - 0.1

```
model = LinearRegression(num_features=X1_train.size(1))
cost = train(model,
             X1_train, X2_train, y_train,
             num_epochs=100,
             learning_rate=0.1)
```

```
Epoch: 001 | MSE: 1290.28015
Epoch: 002 | MSE: 960.12476
Epoch: 003 | MSE: 748.67462
Epoch: 004 | MSE: 613.24976
Epoch: 005 | MSE: 526.51587
Epoch: 006 | MSE: 470.96671
Epoch: 007 | MSE: 435.38974
Epoch: 008 | MSE: 412.60419
Epoch: 009 | MSE: 398.01108
Epoch: 010 | MSE: 388.66476
Epoch: 011 | MSE: 382.67889
Epoch: 012 | MSE: 378.84521
Epoch: 013 | MSE: 376.38986
Epoch: 014 | MSE: 374.81735
Epoch: 015 | MSE: 373.81021
Epoch: 016 | MSE: 373.16519
Epoch: 017 | MSE: 372.75204
Epoch: 018 | MSE: 372.48746
Epoch: 019 | MSE: 372.31802
Epoch: 020 | MSE: 372.20950
Epoch: 021 | MSE: 372.14001
Epoch: 022 | MSE: 372.09549
Epoch: 023 | MSE: 372.06702
Epoch: 024 | MSE: 372.04871
Epoch: 025 | MSE: 372.03702
Epoch: 026 | MSE: 372.02954
Epoch: 027 | MSE: 372.02478
Epoch: 028 | MSE: 372.02170
Epoch: 029 | MSE: 372.01971
```

```
Epoch: 030 | MSE: 372.01840
Epoch: 031 | MSE: 372.01764
Epoch: 032 | MSE: 372.01715
Epoch: 033 | MSE: 372.01678
Epoch: 034 | MSE: 372.01657
Epoch: 035 | MSE: 372.01642
Epoch: 036 | MSE: 372.01639
Epoch: 037 | MSE: 372.01630
Epoch: 038 | MSE: 372.01630
Epoch: 039 | MSE: 372.01624
Epoch: 040 | MSE: 372.01624
Epoch: 041 | MSE: 372.01624
Epoch: 042 | MSE: 372.01620
Epoch: 043 | MSE: 372.01617
Epoch: 044 | MSE: 372.01624
Epoch: 045 | MSE: 372.01617
Epoch: 046 | MSE: 372.01620
Epoch: 047 | MSE: 372.01620
Epoch: 048 | MSE: 372.01620
Epoch: 049 | MSE: 372.01620
Epoch: 050 | MSE: 372.01620
Epoch: 051 | MSE: 372.01624
Epoch: 052 | MSE: 372.01620
Epoch: 053 | MSE: 372.01620
Epoch: 054 | MSE: 372.01620
Epoch: 055 | MSE: 372.01624
Epoch: 056 | MSE: 372.01620
Epoch: 057 | MSE: 372.01624
Epoch: 058 | MSE: 372.01617
```

```
plt.plot(range(len(cost)), cost)
plt.ylabel('Mean Squared Error')
plt.xlabel('Epoch')
plt.show()
```