

[HW1_Prob2]_2-D_Perceptron_Training_with_Gradient_Descent.ipynb

In [19]: *### Training with manually updating W with "Backward" ###*

```
import torch
#from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F

import torch.optim as optim


data = [(1.0,2.1,3.0), (2.0, 3.5, 6.0), (3.0, 3.0, 9.0), (4.0, 2.1, 12.0), (5.0, 4.8, 11.0), (6.0, 3.0, 5.0), (7.0, 2.0, 3.0), (8.0, 1.0, 1.0), (9.0, 0.5, 0.5), (10.0, 0.0, 0.0)]
x = torch.Tensor([d[:2] for d in data])
y = torch.Tensor([d[2] for d in data])

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(2,5)
        self.fc2 = nn.Linear(5,10)
        self.fc3 = nn.Linear(10,1)

    def forward(self, x):
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.fc3(x)
        x = F.relu(x)
        return x


net = Net()

print(net)
print(list(net.parameters()))

#input = torch.randn(1)
#out = net(input)

#def criterion(out, label):
#    return (label - out)**2
criterion = nn.MSELoss()

optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.5)
```

```
#optimizer = optim.Adam(net.parameters(), lr=0.005)
for epoch in range(300):
    for i in range(0, len(x), len(x)//2):
        x_batch = x[i:i+len(x)//2]
        y_batch = y[i:i+len(x)//2]
        optimizer.zero_grad()
        out = net(x)
        loss = criterion(out, y)
        loss.backward()
        optimizer.step()
```

```
Net(
  (fc1): Linear(in_features=2, out_features=5, bias=True)
  (fc2): Linear(in_features=5, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=1, bias=True)
)
[Parameter containing:
tensor([[ 0.3118,  0.6851],
       [-0.3832,  0.1838],
       [-0.3732,  0.3696],
       [-0.0401, -0.0783],
       [ 0.1446,  0.1558]], requires_grad=True), Parameter containin
g:
tensor([-0.0506, -0.4546,  0.2436, -0.4431, -0.4523], requires_grad=T
rue), Parameter containing:
tensor([[ 0.4418, -0.2442, -0.4411, -0.0388, -0.1910],
       [ 0.2479,  0.2764,  0.4288, -0.4169,  0.1399],
       [-0.1059, -0.0392, -0.3316, -0.0323,  0.4375],
       [-0.1685, -0.1100,  0.1804, -0.2218, -0.3665],
       [ 0.3221,  0.1651, -0.3664, -0.0207, -0.0854],
       [ 0.3795, -0.4455,  0.1479, -0.1126, -0.2472],
       [ 0.1670, -0.1769,  0.4470, -0.2205, -0.3857],
       [-0.0336,  0.0752,  0.2979, -0.0482,  0.3269],
       [ 0.4138, -0.3692, -0.2594, -0.1371, -0.2753],
       [ 0.2758, -0.2787,  0.0206,  0.1032, -0.1971]], requires_grad
=True), Parameter containing:
tensor([-0.0317, -0.2902, -0.0321, -0.1589, -0.1082, -0.0515, -0.054
4, -0.2508,
       -0.3219,  0.1275], requires_grad=True), Parameter containing:
tensor([[[-0.1136, -0.1010, -0.2541, -0.1794, -0.0092, -0.1849, -0.201
8, -0.2264,
       0.3065, -0.0363]], requires_grad=True), Parameter containin
g:
tensor([-0.0636], requires_grad=True)]
```

In [18]: `list(net.parameters())`

Out[18]: [Parameter containing:
tensor([[0.5507, 0.6671],
 [0.0057, -0.0594],
 [0.1171, 0.3885],
 [-0.4017, 0.3217],
 [0.3566, 0.3903]], requires_grad=True),
Parameter containing:
tensor([-0.2040, -0.5926, 0.1808, 0.2001, -0.0822], requires_grad=True),
Parameter containing:
tensor([[-0.4307, 0.0538, -0.4318, -0.3961, -0.3446],
 [-0.1856, 0.0666, -0.2548, 0.0757, 0.3238],
 [-0.2463, 0.0719, 0.2261, -0.2641, -0.2735],
 [-0.3800, 0.1741, 0.1700, 0.3176, -0.2003],
 [0.2596, -0.0236, 0.2906, -0.0195, 0.1158],
 [-0.1210, -0.2117, 0.3186, -0.0444, -0.0286],
 [0.4398, 0.3341, 0.0049, -0.3943, -0.1575],
 [-0.1439, 0.1994, -0.0898, -0.0133, -0.4191],
 [0.1297, 0.2036, 0.2468, -0.2808, 0.2951],
 [0.3672, 0.1684, 0.4064, 0.2457, 0.3445]], requires_grad=True),
Parameter containing:
tensor([-0.0924, 0.0385, -0.0651, 0.1596, 0.0412, -0.1266, 0.0589, -0.4239,
 0.3144, 0.2214], requires_grad=True),
Parameter containing:
tensor([[0.0098, 0.0273, 0.1927, -0.3146, -0.1369, -0.2542, 0.1059, 0.1735,
 0.1179, -0.2736]], requires_grad=True),
Parameter containing:
tensor([-0.2404], requires_grad=True)]

In [13]:

Out[13]: tensor([[1.0000, 2.1000],
 [2.0000, 3.5000],
 [3.0000, 3.0000],
 [4.0000, 2.1000],
 [5.0000, 7.2000],
 [6.0000, 10.1000]])

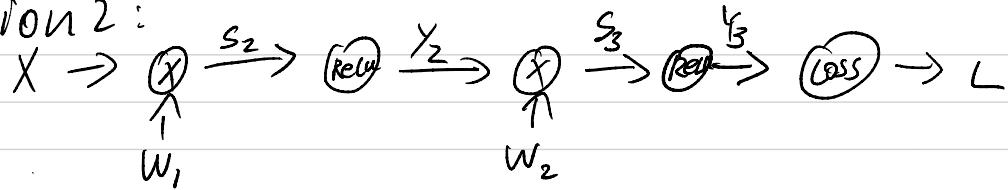
In [14]: `y`

Out[14]: tensor([3., 6., 9., 12., 15., 18.])

In []:

hand-written answers to questions 2, 3, and 4

Question 2:



$$\frac{\partial L}{\partial Y_3} = \begin{bmatrix} 5 \\ 10 \end{bmatrix} = \begin{bmatrix} Y_{31} \\ Y_{32} \end{bmatrix}$$

$$Y_2 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$Y_3 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

From the forward there is no negative value so I have assumed the relu layer is pass through

$$Y_3 = \begin{bmatrix} Y_{21} \cdot w_{211} + Y_{22} \cdot w_{221} \\ Y_{21} \cdot w_{212} + Y_{22} \cdot w_{222} \end{bmatrix}$$

$$Y_3 = \begin{bmatrix} 5 \\ 10 \end{bmatrix}$$

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial Y_{31}} \cdot Y_{21} & \frac{\partial L}{\partial Y_{32}} \cdot Y_{21} \\ \frac{\partial L}{\partial Y_{31}} \cdot Y_{22} & \frac{\partial L}{\partial Y_{32}} \cdot Y_{22} \end{bmatrix}$$

$$\frac{\partial L}{\partial W_2} = \begin{bmatrix} 5 \cdot 2 & 10 \cdot 2 \\ 5 \cdot 3 & 10 \cdot 3 \end{bmatrix}$$

$$\frac{\partial L}{\partial W_2} = \begin{bmatrix} 10 & 20 \\ 15 & 30 \end{bmatrix}$$

$$\frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial w_{21}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial w_{21}}$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial w_{11}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial w_{11}}$$

$$= \frac{\partial L}{\partial Y_{31}} \cdot Y_{21}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial w_{12}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial w_{12}}$$

$$= \frac{\partial L}{\partial Y_{32}} \cdot Y_{21}$$

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial w_{21}} = \frac{\partial L}{\partial Y_{31}} \cdot Y_{22}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial w_{22}} = \frac{\partial L}{\partial Y_{32}} \cdot Y_{22}$$

$$S_2 = X \cdot W_1 \quad S_3 = S_2 \cdot W_2 \quad W_1 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \quad W_2 = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

$$\frac{\partial L}{\partial Y_3} = \begin{bmatrix} 5 \\ 10 \end{bmatrix} \quad S_2 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$S_3 = \begin{bmatrix} S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} w_{211} & w_{212} \\ w_{221} & w_{222} \end{bmatrix} = \begin{bmatrix} S_{21} \cdot w_{211} + S_{22} \cdot w_{221} \\ S_{21} \cdot w_{212} + S_{22} \cdot w_{222} \end{bmatrix}$$

$$\frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial Y_{21}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial Y_{21}}$$

$$\frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial Y_{21}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial Y_{21}} = \frac{\partial L}{\partial Y_{31}} \cdot w_{211} + \frac{\partial L}{\partial Y_{32}} \cdot w_{212}$$

$$\frac{\partial L}{\partial Y_{31}} \cdot \frac{\partial Y_{31}}{\partial Y_{22}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial Y_{22}} = \frac{\partial L}{\partial Y_{31}} \cdot w_{221} + \frac{\partial L}{\partial Y_{32}} \cdot w_{222}$$

$$\frac{\partial L}{\partial Y_2} = \begin{bmatrix} 5 \cdot 1 & 10 \cdot 2 \\ 5 \cdot 1 & 10 \cdot 2 \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

$$\frac{\partial L}{\partial S_2} = \frac{\partial L}{\partial Y_2} \cdot \frac{\partial Y_2}{\partial S_2}, \quad S_2 = X^T W_1 = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} w_{111} & w_{112} \\ w_{121} & w_{122} \end{bmatrix}$$

$$S_2 = \begin{bmatrix} X_1 w_{111} + X_2 w_{121} \\ X_1 w_{112} + X_2 w_{122} \end{bmatrix}$$

$$\frac{\partial L}{\partial S_{21}} \frac{\partial S_{21}}{\partial w_{1,ij}} + \frac{\partial L}{\partial S_{22}} \frac{\partial S_{22}}{\partial w_{1,ij}}$$

$$w_{1,11} = \frac{\partial L}{\partial S_{21}} \cdot \frac{\partial S_{21}}{\partial w_{1,11}} - \frac{\partial L}{\partial S_{21}} \cdot X_1$$

$$w_{1,12} = \frac{\partial L}{\partial S_{21}} \cdot \frac{\partial S_{21}}{\partial w_{1,12}} + \frac{\partial L}{\partial S_{22}} \cdot \frac{\partial S_{22}}{\partial w_{1,12}} - \frac{\partial L}{\partial S_{22}} \cdot X_1$$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix} 25 \cdot 1 & 25 \cdot 1 \\ 25 \cdot 2 & 25 \cdot 2 \end{bmatrix} = \begin{bmatrix} 25 & 25 \\ 50 & 50 \end{bmatrix}$$

$$w_{121} = \frac{\partial L}{\partial S_{21}} \cdot \frac{\partial S_{21}}{\partial w_{121}} = \frac{\partial L}{\partial S_{21}} \cdot X_2$$

$$w_{122} = \frac{\partial L}{\partial S_{22}} \cdot \frac{\partial S_{22}}{\partial w_{122}} = \frac{\partial L}{\partial S_{22}} \cdot X_2$$

Problem 3.

$$S_2 = X W_1 \quad Y_2 = \text{relu}(S_2) \quad S_3 = Y_2 W_2 \quad Y_3 = \text{relu}(S_3) \quad \text{Loss} = \frac{1}{N} \sum (y - y)^2$$

$$X = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad W_1 = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \quad W_2 = \begin{bmatrix} -1 & 2 \\ 1 & 2 \end{bmatrix}$$

Forward: $S_2 = X W_1$

$$S_2 = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & -1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$Y_2 = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$S_3 = \begin{bmatrix} 2 & 0 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} -2 & 4 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$$

$$Y_3 = \begin{bmatrix} -2 \\ 4 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{31} \cdot r_1 \\ S_{32} \cdot r_2 \end{bmatrix} = \begin{bmatrix} -2 \cdot 0 \\ 4 \cdot 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$$

$$\text{Loss} = \frac{1}{2} (0)^2 + \frac{1}{2} (4)^2 = 0 + \frac{16}{2} = 8$$

$$\frac{\partial L}{\partial Y_3} = \frac{3}{2} (Y_3 - y) \cdot (1) = (Y_3 - y) \quad \frac{\partial L}{\partial Y_3} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$\frac{\partial L}{\partial S_3} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_{31}}{\partial S_{31}} + \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_{32}}{\partial S_{32}}$$

$$\text{for } S_{31} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_{31}}{\partial S_{31}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial S_{31}} = \frac{\partial L}{\partial Y_3} \cdot 0 + \frac{\partial L}{\partial Y_{32}} \cdot 0 = 0$$

$$\text{for } S_{32} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_{31}}{\partial S_{32}} + \frac{\partial L}{\partial Y_{32}} \cdot \frac{\partial Y_{32}}{\partial S_{32}} = \frac{\partial L}{\partial Y_{32}} \cdot 1 = \frac{\partial L}{\partial Y_{32}} = 4$$

$$\frac{\partial L}{\partial S_3} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial S_3} \cdot \frac{\partial S_3}{\partial W_2} \quad S_3 = Y_2 W_2$$

$$S_3 = \begin{bmatrix} Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} w_{211} & w_{212} \\ w_{221} & w_{222} \end{bmatrix}$$

$$S_3 = \begin{bmatrix} Y_{21} \cdot w_{211} + Y_{22} \cdot w_{221} \\ Y_{21} \cdot w_{212} + Y_{22} \cdot w_{222} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial S_{31}} \cdot \frac{\partial S_{31}}{\partial w_{211}} + \frac{\partial L}{\partial S_{32}} \cdot \frac{\partial S_{32}}{\partial w_{212}} = \begin{bmatrix} 0 & 4 \cdot 2 \\ 0 & 4 \cdot 0 \end{bmatrix} = \begin{bmatrix} 0 & 8 \\ 0 & 0 \end{bmatrix}$$

$$w_{211} = \frac{\partial L}{\partial S_{31}} \cdot \frac{\partial S_{31}}{\partial w_{211}} = \frac{\partial L}{\partial S_{31}} \cdot Y_{21}$$

$$w_{212} = \frac{\partial L}{\partial S_{32}} \cdot \frac{\partial S_{32}}{\partial w_{212}} = \frac{\partial L}{\partial S_{32}} \cdot Y_{21}$$

$$\frac{\partial L}{\partial S_3} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$$

$$w_{221} = \frac{\partial L}{\partial S_{31}} \cdot \frac{\partial S_{31}}{\partial w_{221}} = \frac{\partial L}{\partial S_{31}} \cdot Y_{22}$$

$$w_{222} = \frac{\partial L}{\partial S_{32}} \cdot \frac{\partial S_{32}}{\partial w_{222}} = \frac{\partial L}{\partial S_{32}} \cdot Y_{22}$$

$$w_2 = \begin{bmatrix} -1 & 2 \\ 1 & 2 \end{bmatrix}$$

$$\frac{\partial L}{\partial Y_2} = \frac{\partial L}{\partial S_3} \cdot \frac{\partial S_3}{\partial Y_2} = \frac{\partial L}{\partial S_{31}} \cdot \frac{\partial S_{31}}{\partial Y_{21}} + \frac{\partial L}{\partial S_{32}} \cdot \frac{\partial S_{32}}{\partial Y_{21}}, \quad \frac{\partial L}{\partial S_{31}} \cdot \frac{\partial S_{31}}{\partial Y_{22}} + \frac{\partial L}{\partial S_{32}} \cdot \frac{\partial S_{32}}{\partial Y_{22}}$$

$$Y_{21} = \frac{\partial L}{\partial S_{31}} \cdot w_{211} + \frac{\partial L}{\partial S_{32}} \cdot w_{212} \quad Y_{22} = \frac{\partial L}{\partial S_{31}} \cdot w_{221} + \frac{\partial L}{\partial S_{32}} \cdot w_{222}$$

$$\frac{\partial L}{\partial Y_2} = \begin{bmatrix} 0 + 4 \cdot 2 \\ 0 + 4 \cdot 2 \end{bmatrix} = \begin{bmatrix} 8 \\ 8 \end{bmatrix}$$

$$\begin{bmatrix} S_{21} \\ S_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ r_1 & r_2 \end{bmatrix} = \begin{bmatrix} S_{21}r_1 \\ S_{22}r_2 \end{bmatrix}$$

$$\frac{\partial L}{\partial S_2} = \frac{\partial L}{\partial Y_2} \frac{\partial Y_2}{\partial S_2} = \frac{\partial L}{\partial Y_{21}} \cdot \frac{\partial Y_{21}}{\partial S_{21}} + \frac{\partial L}{\partial Y_{22}} \cdot \frac{\partial Y_{22}}{\partial S_{21}} = \begin{bmatrix} 8 \\ 0 \end{bmatrix} = \frac{\partial L}{\partial S_2}$$

$$\text{for } S_{21} = \frac{\partial L}{\partial Y_{21}} \cdot r_1 + \frac{\partial L}{\partial Y_{22}} \cdot 0 = 8$$

$$S_{22} = \frac{\partial L}{\partial Y_{22}} \cdot 0 + \frac{\partial L}{\partial Y_{22}} \cdot r_2 = 0$$

$$S_2 = XW_1 \quad Y_2 = \text{relu}(S_2) \quad S_3 = Y_2 W_2 \quad Y_3 = \text{relu}(S_3)$$

$$X = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad w_1 = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} -1 & 2 \\ 1 & 2 \end{bmatrix} \quad \frac{\partial L}{\partial S_2} = \begin{bmatrix} -8 \\ 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial S_2} \cdot \frac{\partial S_2}{\partial w_1} = \frac{\partial L}{\partial S_{21}} \cdot \frac{\partial S_{21}}{\partial w_{11}} + \frac{\partial L}{\partial S_{22}} \cdot \frac{\partial S_{22}}{\partial w_{12}}$$

$$S_2 = XW_1$$

$$S_2 = X^T w_1 = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{12} & w_{22} \end{bmatrix}$$

$$w_{111} = \frac{\partial L}{\partial S_{21}} \cdot x_1$$

$$S_2 = \begin{bmatrix} x_1 w_{111} + x_2 w_{121} \\ x_1 w_{112} + x_2 w_{122} \end{bmatrix}$$

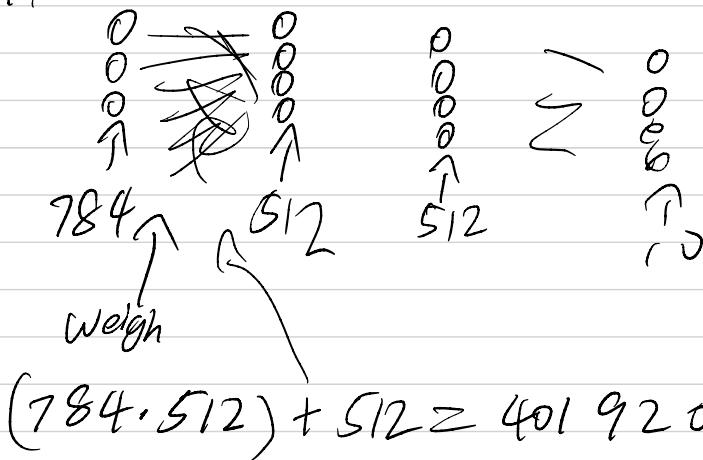
$$w_{121} = \frac{\partial L}{\partial S_{21}} \cdot x_2$$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix} 8 \cdot 1 & 0 \\ 8 \cdot 2 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 16 & 0 \end{bmatrix}$$

$$w_{112} = \frac{\partial L}{\partial S_{22}} \cdot x_1$$

$$w_{122} = \frac{\partial L}{\partial S_{22}} \cdot x_2$$

Question 4:



$$(512 \cdot 512) + 512 = 262,656 + = 669,706$$

total parameter

1 image = 784 input

$$y = xw + b$$

784node(512 multiply) + 512node(784 addition) + 512relu

512node(512 multiply) + 512node(512 addition) + 512relu

512node(10 multiply) + 10node(512 addition) + 10relu

= 1,338,378 total operation in 1 forward pass

[W2_hw]_Mult-layer_Perceptron_Non_linear_Training.ipynb

In [96]: *### Training with fancier version ###*

```
import torch
import torch.nn as nn
import torch.nn.functional as F

import torch.optim as optim
import matplotlib.pyplot as plt

class Net(nn.Module): ## nn.Module class is used
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(1,4096,bias=False) # in dim, out dim
        self.fc2 = nn.Linear(4096,2048,bias=False)
        self.fc3 = nn.Linear(2048,1,bias=False)

    def forward(self, x):
        x = self.fc1(x)
        x = F.sigmoid(x)
        x = self.fc2(x)
        x = F.sigmoid(x)
        x = self.fc3(x)
        return x

net = Net()

print(net)
print(list(net.parameters())) # parameters are randomized

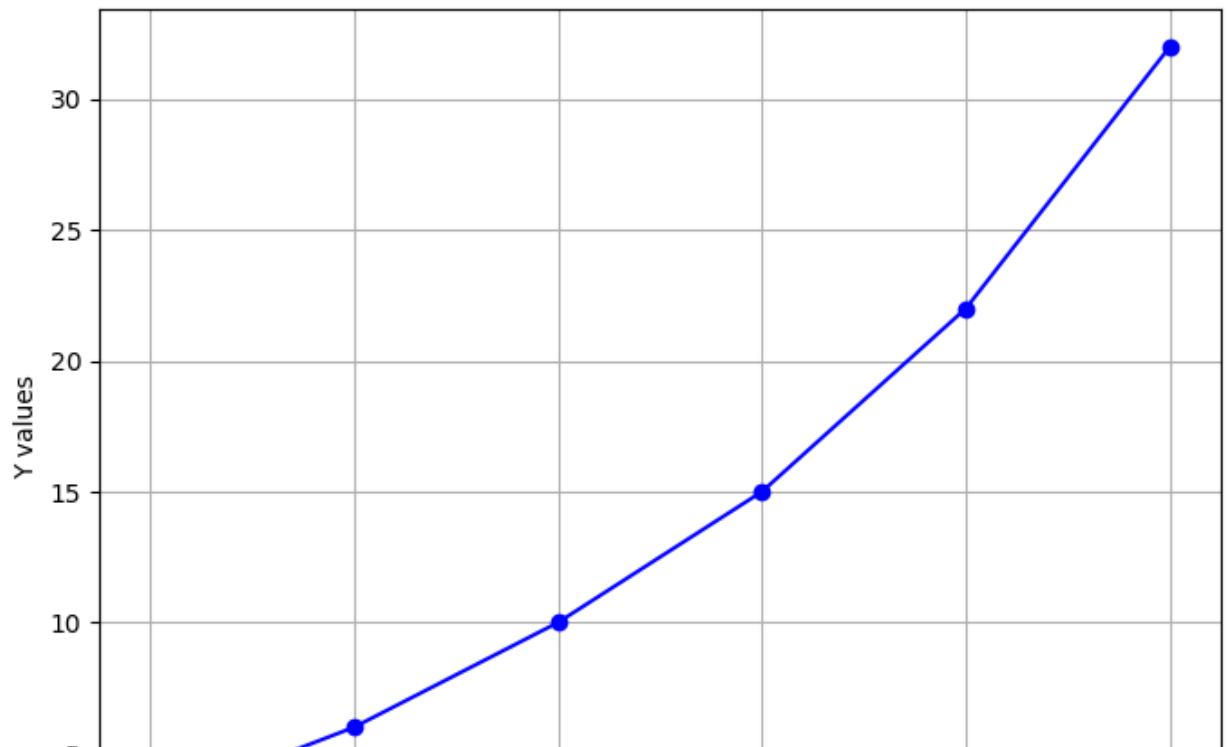
#def criterion(out, label):
#    return (label - out)**2
criterion = nn.MSELoss()

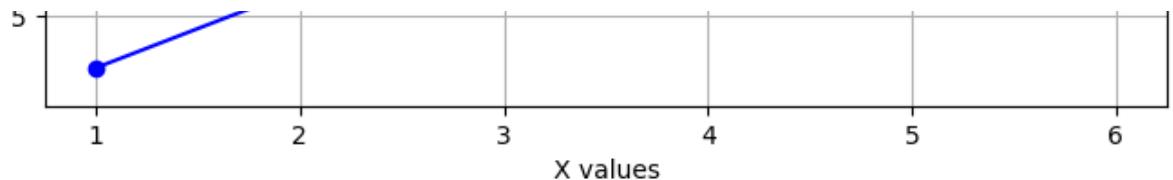
# optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.9)
optimizer = optim.Adam(net.parameters(), lr=0.00007,)

data = [(1.0,3.0), (2.0,6.0), (3.0,10.0), (4.0,15.0), (5.0,22.0), (6.0,30.0), (7.0,35.0), (8.0,40.0), (9.0,45.0), (10.0,50.0), (11.0,55.0), (12.0,60.0), (13.0,65.0), (14.0,70.0), (15.0,75.0), (16.0,80.0), (17.0,85.0), (18.0,90.0), (19.0,95.0), (20.0,100.0), (21.0,105.0), (22.0,110.0), (23.0,115.0), (24.0,120.0), (25.0,125.0), (26.0,130.0), (27.0,135.0), (28.0,140.0), (29.0,145.0), (30.0,150.0), (31.0,155.0), (32.0,160.0), (33.0,165.0), (34.0,170.0), (35.0,175.0), (36.0,180.0), (37.0,185.0), (38.0,190.0), (39.0,195.0), (40.0,200.0), (41.0,205.0), (42.0,210.0), (43.0,215.0), (44.0,220.0), (45.0,225.0), (46.0,230.0), (47.0,235.0), (48.0,240.0), (49.0,245.0), (50.0,250.0), (51.0,255.0), (52.0,260.0), (53.0,265.0), (54.0,270.0), (55.0,275.0), (56.0,280.0), (57.0,285.0), (58.0,290.0), (59.0,295.0), (60.0,300.0), (61.0,305.0), (62.0,310.0), (63.0,315.0), (64.0,320.0), (65.0,325.0), (66.0,330.0), (67.0,335.0), (68.0,340.0), (69.0,345.0), (70.0,350.0), (71.0,355.0), (72.0,360.0), (73.0,365.0), (74.0,370.0), (75.0,375.0), (76.0,380.0), (77.0,385.0), (78.0,390.0), (79.0,395.0), (80.0,400.0), (81.0,405.0), (82.0,410.0), (83.0,415.0), (84.0,420.0), (85.0,425.0), (86.0,430.0), (87.0,435.0), (88.0,440.0), (89.0,445.0), (90.0,450.0), (91.0,455.0), (92.0,460.0), (93.0,465.0), (94.0,470.0), (95.0,475.0), (96.0,480.0), (97.0,485.0), (98.0,490.0), (99.0,495.0), (100.0,500.0), (101.0,505.0), (102.0,510.0), (103.0,515.0), (104.0,520.0), (105.0,525.0), (106.0,530.0), (107.0,535.0), (108.0,540.0), (109.0,545.0), (110.0,550.0), (111.0,555.0), (112.0,560.0), (113.0,565.0), (114.0,570.0), (115.0,575.0), (116.0,580.0), (117.0,585.0), (118.0,590.0), (119.0,595.0), (120.0,600.0), (121.0,605.0), (122.0,610.0), (123.0,615.0), (124.0,620.0), (125.0,625.0), (126.0,630.0), (127.0,635.0), (128.0,640.0), (129.0,645.0), (130.0,650.0), (131.0,655.0), (132.0,660.0), (133.0,665.0), (134.0,670.0), (135.0,675.0), (136.0,680.0), (137.0,685.0), (138.0,690.0), (139.0,695.0), (140.0,700.0), (141.0,705.0), (142.0,710.0), (143.0,715.0), (144.0,720.0), (145.0,725.0), (146.0,730.0), (147.0,735.0), (148.0,740.0), (149.0,745.0), (150.0,750.0), (151.0,755.0), (152.0,760.0), (153.0,765.0), (154.0,770.0), (155.0,775.0), (156.0,780.0), (157.0,785.0), (158.0,790.0), (159.0,795.0), (160.0,800.0), (161.0,805.0), (162.0,810.0), (163.0,815.0), (164.0,820.0), (165.0,825.0), (166.0,830.0), (167.0,835.0), (168.0,840.0), (169.0,845.0), (170.0,850.0), (171.0,855.0), (172.0,860.0), (173.0,865.0), (174.0,870.0), (175.0,875.0), (176.0,880.0), (177.0,885.0), (178.0,890.0), (179.0,895.0), (180.0,900.0), (181.0,905.0), (182.0,910.0), (183.0,915.0), (184.0,920.0), (185.0,925.0), (186.0,930.0), (187.0,935.0), (188.0,940.0), (189.0,945.0), (190.0,950.0), (191.0,955.0), (192.0,960.0), (193.0,965.0), (194.0,970.0), (195.0,975.0), (196.0,980.0), (197.0,985.0), (198.0,990.0), (199.0,995.0), (200.0,1000.0), (201.0,1005.0), (202.0,1010.0), (203.0,1015.0), (204.0,1020.0), (205.0,1025.0), (206.0,1030.0), (207.0,1035.0), (208.0,1040.0), (209.0,1045.0), (210.0,1050.0), (211.0,1055.0), (212.0,1060.0), (213.0,1065.0), (214.0,1070.0), (215.0,1075.0), (216.0,1080.0), (217.0,1085.0), (218.0,1090.0), (219.0,1095.0), (220.0,1100.0), (221.0,1105.0), (222.0,1110.0), (223.0,1115.0), (224.0,1120.0), (225.0,1125.0), (226.0,1130.0), (227.0,1135.0), (228.0,1140.0), (229.0,1145.0), (230.0,1150.0), (231.0,1155.0), (232.0,1160.0), (233.0,1165.0), (234.0,1170.0), (235.0,1175.0), (236.0,1180.0), (237.0,1185.0), (238.0,1190.0), (239.0,1195.0), (240.0,1200.0), (241.0,1205.0), (242.0,1210.0), (243.0,1215.0), (244.0,1220.0), (245.0,1225.0), (246.0,1230.0), (247.0,1235.0), (248.0,1240.0), (249.0,1245.0), (250.0,1250.0), (251.0,1255.0), (252.0,1260.0), (253.0,1265.0), (254.0,1270.0), (255.0,1275.0), (256.0,1280.0), (257.0,1285.0), (258.0,1290.0), (259.0,1295.0), (260.0,1300.0), (261.0,1305.0), (262.0,1310.0), (263.0,1315.0), (264.0,1320.0), (265.0,1325.0), (266.0,1330.0), (267.0,1335.0), (268.0,1340.0), (269.0,1345.0), (270.0,1350.0), (271.0,1355.0), (272.0,1360.0), (273.0,1365.0), (274.0,1370.0), (275.0,1375.0), (276.0,1380.0), (277.0,1385.0), (278.0,1390.0), (279.0,1395.0), (280.0,1400.0), (281.0,1405.0), (282.0,1410.0), (283.0,1415.0), (284.0,1420.0), (285.0,1425.0), (286.0,1430.0), (287.0,1435.0), (288.0,1440.0), (289.0,1445.0), (290.0,1450.0), (291.0,1455.0), (292.0,1460.0), (293.0,1465.0), (294.0,1470.0), (295.0,1475.0), (296.0,1480.0), (297.0,1485.0), (298.0,1490.0), (299.0,1495.0), (300.0,1500.0), (301.0,1505.0), (302.0,1510.0), (303.0,1515.0), (304.0,1520.0), (305.0,1525.0), (306.0,1530.0), (307.0,1535.0), (308.0,1540.0), (309.0,1545.0), (310.0,1550.0), (311.0,1555.0), (312.0,1560.0), (313.0,1565.0), (314.0,1570.0), (315.0,1575.0), (316.0,1580.0), (317.0,1585.0), (318.0,1590.0), (319.0,1595.0), (320.0,1600.0), (321.0,1605.0), (322.0,1610.0), (323.0,1615.0), (324.0,1620.0), (325.0,1625.0), (326.0,1630.0), (327.0,1635.0), (328.0,1640.0), (329.0,1645.0), (330.0,1650.0), (331.0,1655.0), (332.0,1660.0), (333.0,1665.0), (334.0,1670.0), (335.0,1675.0), (336.0,1680.0), (337.0,1685.0), (338.0,1690.0), (339.0,1695.0), (340.0,1700.0), (341.0,1705.0), (342.0,1710.0), (343.0,1715.0), (344.0,1720.0), (345.0,1725.0), (346.0,1730.0), (347.0,1735.0), (348.0,1740.0), (349.0,1745.0), (350.0,1750.0), (351.0,1755.0), (352.0,1760.0), (353.0,1765.0), (354.0,1770.0), (355.0,1775.0), (356.0,1780.0), (357.0,1785.0), (358.0,1790.0), (359.0,1795.0), (360.0,1800.0), (361.0,1805.0), (362.0,1810.0), (363.0,1815.0), (364.0,1820.0), (365.0,1825.0), (366.0,1830.0), (367.0,1835.0), (368.0,1840.0), (369.0,1845.0), (370.0,1850.0), (371.0,1855.0), (372.0,1860.0), (373.0,1865.0), (374.0,1870.0), (375.0,1875.0), (376.0,1880.0), (377.0,1885.0), (378.0,1890.0), (379.0,1895.0), (380.0,1900.0), (381.0,1905.0), (382.0,1910.0), (383.0,1915.0), (384.0,1920.0), (385.0,1925.0), (386.0,1930.0), (387.0,1935.0), (388.0,1940.0), (389.0,1945.0), (390.0,1950.0), (391.0,1955.0), (392.0,1960.0), (393.0,1965.0), (394.0,1970.0), (395.0,1975.0), (396.0,1980.0), (397.0,1985.0), (398.0,1990.0), (399.0,1995.0), (400.0,2000.0), (401.0,2005.0), (402.0,2010.0), (403.0,2015.0), (404.0,2020.0), (405.0,2025.0), (406.0,2030.0), (407.0,2035.0), (408.0,2040.0), (409.0,2045.0), (410.0,2050.0), (411.0,2055.0), (412.0,2060.0), (413.0,2065.0), (414.0,2070.0), (415.0,2075.0), (416.0,2080.0), (417.0,2085.0), (418.0,2090.0), (419.0,2095.0), (420.0,2100.0), (421.0,2105.0), (422.0,2110.0), (423.0,2115.0), (424.0,2120.0), (425.0,2125.0), (426.0,2130.0), (427.0,2135.0), (428.0,2140.0), (429.0,2145.0), (430.0,2150.0), (431.0,2155.0), (432.0,2160.0), (433.0,2165.0), (434.0,2170.0), (435.0,2175.0), (436.0,2180.0), (437.0,2185.0), (438.0,2190.0), (439.0,2195.0), (440.0,2200.0), (441.0,2205.0), (442.0,2210.0), (443.0,2215.0), (444.0,2220.0), (445.0,2225.0), (446.0,2230.0), (447.0,2235.0), (448.0,2240.0), (449.0,2245.0), (450.0,2250.0), (451.0,2255.0), (452.0,2260.0), (453.0,2265.0), (454.0,2270.0), (455.0,2275.0), (456.0,2280.0), (457.0,2285.0), (458.0,2290.0), (459.0,2295.0), (460.0,2300.0), (461.0,2305.0), (462.0,2310.0), (463.0,2315.0), (464.0,2320.0), (465.0,2325.0), (466.0,2330.0), (467.0,2335.0), (468.0,2340.0), (469.0,2345.0), (470.0,2350.0), (471.0,2355.0), (472.0,2360.0), (473.0,2365.0), (474.0,2370.0), (475.0,2375.0), (476.0,2380.0), (477.0,2385.0), (478.0,2390.0), (479.0,2395.0), (480.0,2400.0), (481.0,2405.0), (482.0,2410.0), (483.0,2415.0), (484.0,2420.0), (485.0,2425.0), (486.0,2430.0), (487.0,2435.0), (488.0,2440.0), (489.0,2445.0), (490.0,2450.0), (491.0,2455.0), (492.0,2460.0), (493.0,2465.0), (494.0,2470.0), (495.0,2475.0), (496.0,2480.0), (497.0,2485.0), (498.0,2490.0), (499.0,2495.0), (500.0,2500.0), (501.0,2505.0), (502.0,2510.0), (503.0,2515.0), (504.0,2520.0), (505.0,2525.0), (506.0,2530.0), (507.0,2535.0), (508.0,2540.0), (509.0,2545.0), (510.0,2550.0), (511.0,2555.0), (512.0,2560.0), (513.0,2565.0), (514.0,2570.0), (515.0,2575.0), (516.0,2580.0), (517.0,2585.0), (518.0,2590.0), (519.0,2595.0), (520.0,2600.0), (521.0,2605.0), (522.0,2610.0), (523.0,2615.0), (524.0,2620.0), (525.0,2625.0), (526.0,2630.0), (527.0,2635.0), (528.0,2640.0), (529.0,2645.0), (530.0,2650.0), (531.0,2655.0), (532.0,2660.0), (533.0,2665.0), (534.0,2670.0), (535.0,2675.0), (536.0,2680.0), (537.0,2685.0), (538.0,2690.0), (539.0,2695.0), (540.0,2700.0), (541.0,2705.0), (542.0,2710.0), (543.0,2715.0), (544.0,2720.0), (545.0,2725.0), (546.0,2730.0), (547.0,2735.0), (548.0,2740.0), (549.0,2745.0), (550.0,2750.0), (551.0,2755.0), (552.0,2760.0), (553.0,2765.0), (554.0,2770.0), (555.0,2775.0), (556.0,2780.0), (557.0,2785.0), (558.0,2790.0), (559.0,2795.0), (560.0,2800.0), (561.0,2805.0), (562.0,2810.0), (563.0,2815.0), (564.0,2820.0), (565.0,2825.0), (566.0,2830.0), (567.0,2835.0), (568.0,2840.0), (569.0,2845.0), (570.0,2850.0), (571.0,2855.0), (572.0,2860.0), (573.0,2865.0), (574.0,2870.0), (575.0,2875.0), (576.0,2880.0), (577.0,2885.0), (578.0,2890.0), (579.0,2895.0), (580.0,2900.0), (581.0,2905.0), (582.0,2910.0), (583.0,2915.0), (584.0,2920.0), (585.0,2925.0), (586.0,2930.0), (587.0,2935.0), (588.0,2940.0), (589.0,2945.0), (590.0,2950.0), (591.0,2955.0), (592.0,2960.0), (593.0,2965.0), (594.0,2970.0), (595.0,2975.0), (596.0,2980.0), (597.0,2985.0), (598.0,2990.0), (599.0,2995.0), (600.0,3000.0), (601.0,3005.0), (602.0,3010.0), (603.0,3015.0), (604.0,3020.0), (605.0,3025.0), (606.0,3030.0), (607.0,3035.0), (608.0,3040.0), (609.0,3045.0), (610.0,3050.0), (611.0,3055.0), (612.0,3060.0), (613.0,3065.0), (614.0,3070.0), (615.0,3075.0), (616.0,3080.0), (617.0,3085.0), (618.0,3090.0), (619.0,3095.0), (620.0,3100.0), (621.0,3105.0), (622.0,3110.0), (623.0,3115.0), (624.0,3120.0), (625.0,3125.0), (626.0,3130.0), (627.0,3135.0), (628.0,3140.0), (629.0,3145.0), (630.0,3150.0), (631.0,3155.0), (632.0,3160.0), (633.0,3165.0), (634.0,3170.0), (635.0,3175.0), (636.0,3180.0), (637.0,3185.0), (638.0,3190.0), (639.0,3195.0), (640.0,3200.0), (641.0,3205.0), (642.0,3210.0), (643.0,3215.0), (644.0,3220.0), (645.0,3225.0), (646.0,3230.0), (647.0,3235.0), (648.0,3240.0), (649.0,3245.0), (650.0,3250.0), (651.0,3255.0), (652.0,3260.0), (653.0,3265.0), (654.0,3270.0), (655.0,3275.0), (656.0,3280.0), (657.0,3285.0), (658.0,3290.0), (659.0,3295.0), (660.0,3300.0), (661.0,3305.0), (662.0,3310.0), (663.0,3315.0), (664.0,3320.0), (665.0,3325.0), (666.0,3330.0), (667.0,3335.0), (668.0,3340.0), (669.0,3345.0), (670.0,3350.0), (671.0,3355.0), (672.0,3360.0), (673.0,3365.0), (674.0,3370.0), (675.0,3375.0), (676.0,3380.0), (677.0,3385.0), (678.0,3390.0), (679.0,3395.0), (680.0,3400.0), (681.0,3405.0), (682.0,3410.0), (683.0,3415.0), (684.0,3420.0), (685.0,3425.0), (686.0,3430.0), (687.0,3435.0), (688.0,3440.0), (689.0,3445.0), (690.0,3450.0), (691.0,3455.0), (692.0,3460.0), (693.0,3465.0), (694.0,3470.0), (695.0,3475.0), (696.0,3480.0), (697.0,3485.0), (698.0,3490.0), (699.0,3495.0), (700.0,3500.0), (701.0,3505.0), (702.0,3510.0), (703.0,3515.0), (704.0,3520.0), (705.0,3525.0), (706.0,3530.0), (707.0,3535.0), (708.0,3540.0), (709.0,3545.0), (710.0,3550.0), (711.0,3555.0), (712.0,3560.0), (713.0,3565.0), (714.0,3570.0), (715.0,3575.0), (716.0,3580.0), (717.0,3585.0), (718.0,3590.0), (719.0,3595.0), (720.0,3600.0), (721.0,3605.0), (722.0,3610.0), (723.0,3615.0), (724.0,3620.0), (725.0,3625.0), (726.0,3630.0), (727.0,3635.0), (728.0,3640.0), (729.0,3645.0), (730.0,3650.0), (731.0,3655.0), (732.0,3660.0), (733.0,3665.0), (734.0,3670.0), (735.0,3675.0), (736.0,3680.0), (737.0,3685.0), (738.0,3690.0), (739.0,3695.0), (740.0,3700.0), (741.0,3705.0), (742.0,3710.0), (743.0,3715.0), (744.0,3720.0), (745.0,3725.0), (746.0,3730.0), (747.0,3735.0), (748.0,3740.0), (749.0,3745.0), (750.0,3750.0), (751.0,3755.0), (752.0,3760.0), (753.0,3765.0), (754.0,3770.0), (755.0,3775.0), (756.0,3780.0), (757.0,3785.0), (758.0,3790.0), (759.0,3795.0), (760.0,3800.0), (761.0,3805.0), (762.0,3810.0), (763.0,3815.0), (764.0,3820.0), (765.0,3825.0), (766.0,3830.0), (767.0,3835.0), (768.0,3840.0), (769.0,3845.0), (770.0,3850.0), (771.0,3855.0), (772.0,3860.0), (773.0,3865.0), (774.0,3870.0), (775.0,3875.0), (776.0,3880.0), (777.0,3885.0), (778.0,3890.0), (779.0,3895.0), (780.0,3900.0), (781.0,3905.0), (782.0,3910.0), (783.0,3915.0), (784.0,3920.0), (785.0,3925.0), (786.0,3930.0), (787.0,3935.0), (788.0,3940.0), (789.0,3945.0), (790.0,3950.0), (791.0,3955.0), (792.0,3960.0), (793.0,3965.0), (794.0,3970.0), (795.0,3975.0), (796.0,3980.0), (797.0,3985.0), (798.0,3990.0), (799.0,3995.0), (800.0,4000.0), (801.0,4005.0), (802.0,4010.0), (803.0,4015.0), (804.0,4020.0), (805.0,4025.0), (806.0,4030.0), (807.0,4035.0), (
```

```
plt.ylabel('Y values')
plt.grid(True)
plt.show()
```

```
Net(
    (fc1): Linear(in_features=1, out_features=4096, bias=False)
    (fc2): Linear(in_features=4096, out_features=2048, bias=False)
    (fc3): Linear(in_features=2048, out_features=1, bias=False)
)
[Parameter containing:
tensor([[ 0.2647],
       [ 0.6125],
       [-0.9204],
       ...,
       [ 0.0500],
       [-0.9578],
       [ 0.2392]], requires_grad=True), Parameter containing:
tensor([[-0.0024, -0.0130, -0.0089, ..., 0.0104, -0.0093, 0.0004],
       [-0.0113, 0.0020, -0.0090, ..., -0.0022, 0.0090, 0.0082],
       [ 0.0095, -0.0065, 0.0018, ..., -0.0002, -0.0011, -0.0053],
       ...,
       [ 0.0075, -0.0020, -0.0113, ..., 0.0152, -0.0037, -0.0074],
       [ 0.0107, 0.0123, 0.0095, ..., 0.0040, -0.0150, 0.0105],
       [ 0.0084, 0.0105, 0.0039, ..., 0.0059, -0.0142, -0.001
0]], requires_grad=True), Parameter containing:
tensor([[ 0.0178, 0.0086, 0.0049, ..., -0.0132, -0.0218, -0.017
1]], requires_grad=True)]
```





```
In [97]: for epoch in range(1500): # 0 - 19
    for i, current_data in enumerate(data):
        X, Y = current_data
        X, Y = torch.FloatTensor([X]), torch.FloatTensor([Y])
        optimizer.zero_grad()
        outputs = net(X)
        loss = criterion(outputs, Y)
        loss.backward()
        optimizer.step()      ## This line is equivalent to "W = W - lr*
    print("Epoch {} - loss: {}".format(epoch, loss))

### Test the trained network ###
for i, current_data in enumerate(data):
    X, Y = current_data
    X, Y = torch.FloatTensor([X]), torch.FloatTensor([Y])
    out = net(torch.FloatTensor(X))
    print("when x = {}, y = {}".format(X, out))
```

```
Epoch 1486 - loss: 1.4406759873963892e-07
Epoch 1487 - loss: 1.0762596502900124e-07
Epoch 1488 - loss: 1.0637813829816878e-07
Epoch 1489 - loss: 7.754715625196695e-08
Epoch 1490 - loss: 7.968628779053688e-08
Epoch 1491 - loss: 5.6843418860808015e-08
Epoch 1492 - loss: 5.6843418860808015e-08
Epoch 1493 - loss: 4.165121936239302e-08
Epoch 1494 - loss: 4.165121936239302e-08
Epoch 1495 - loss: 2.8816430130973458e-08
Epoch 1496 - loss: 3.0126102501526475e-08
Epoch 1497 - loss: 2.0463630789890885e-08
Epoch 1498 - loss: 2.386877895332873e-08
Epoch 1499 - loss: 1.1819793144240975e-08
when x = tensor([1.]), y = tensor([3.0000], grad_fn=<SqueezeBackward4>)
when x = tensor([2.]), y = tensor([6.0000], grad_fn=<SqueezeBackward4>)
when x = tensor([3.]), y = tensor([10.0000], grad_fn=<SqueezeBackward4>)
```

```
In [ ]: for epoch in range(20):
    print(epoch)
```

```
In [ ]: W = torch.tensor([1.0], requires_grad=True)
W = W*2
label = 1.0
loss = W*5 - label
loss.backward()
W.grad
```

```
In [ ]:
```