[HW_Code1]_2-D_Perceptron_Training_with_Gradient_Descent

October 14, 2025

[1]:
```python
### 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, 7.2, 15.0), (6.0, 10.1, 18.0)]

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(2,1,bias=False) # in dim=2, out dim=1

    def forward(self, x):
        x = self.fc1(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)
```

```
Net(
```

```
  (fc1): Linear(in_features=2, out_features=1, bias=False)
)
[Parameter containing:
tensor([[ 0.0031, -0.5307]], requires_grad=True)]
```

[2]:
```python
for epoch in range(20):
    for i, curr in enumerate(data):
        x1, x2, y = curr
        print(x1, x2, y)
        x = torch.FloatTensor([x1, x2])
        y = torch.FloatTensor([y])
        optimizer.zero_grad()
        y_pred = net(x)
        loss = criterion(y, y_pred)
        loss.backward()
        optimizer.step()
        print(f'Epoch {epoch}: loss {loss}')
```

```
1.0 2.1 3.0
Epoch 0: loss 16.90233612060547
2.0 3.5 6.0
Epoch 0: loss 50.1579704284668
3.0 3.0 9.0
Epoch 0: loss 50.38731002807617
4.0 2.1 12.0
Epoch 0: loss 36.389007568359375
5.0 7.2 15.0
Epoch 0: loss 25.5361385345459
6.0 10.1 18.0
Epoch 0: loss 11.352394104003906
1.0 2.1 3.0
Epoch 1: loss 1.6954344511032104
2.0 3.5 6.0
Epoch 1: loss 20.714921951293945
3.0 3.0 9.0
Epoch 1: loss 34.5633544921875
4.0 2.1 12.0
Epoch 1: loss 28.571147918701172
5.0 7.2 15.0
Epoch 1: loss 1.7189432382583618
6.0 10.1 18.0
Epoch 1: loss 17.0133113861084
1.0 2.1 3.0
Epoch 2: loss 1.1805447340011597
2.0 3.5 6.0
Epoch 2: loss 14.64073371887207
3.0 3.0 9.0
Epoch 2: loss 20.525575637817383
```

```
4.0 2.1 12.0
Epoch 2: loss 13.580549240112305
5.0 7.2 15.0
Epoch 2: loss 0.02331654541194439
6.0 10.1 18.0
Epoch 2: loss 11.089090347290039
1.0 2.1 3.0
Epoch 3: loss 0.6416836977005005
2.0 3.5 6.0
Epoch 3: loss 7.502073287963867
3.0 3.0 9.0
Epoch 3: loss 8.876242637634277
4.0 2.1 12.0
Epoch 3: loss 4.601325511932373
5.0 7.2 15.0
Epoch 3: loss 0.15831956267356873
6.0 10.1 18.0
Epoch 3: loss 5.137836933135986
1.0 2.1 3.0
Epoch 4: loss 0.2539424002170563
2.0 3.5 6.0
Epoch 4: loss 2.785703182220459
3.0 3.0 9.0
Epoch 4: loss 2.7002298831939697
4.0 2.1 12.0
Epoch 4: loss 0.9718313217163086
5.0 7.2 15.0
Epoch 4: loss 0.32675376534461975
6.0 10.1 18.0
Epoch 4: loss 1.701069951057434
1.0 2.1 3.0
Epoch 5: loss 0.06851554661989212
2.0 3.5 6.0
Epoch 5: loss 0.6838605999946594
3.0 3.0 9.0
Epoch 5: loss 0.465992271900177
4.0 2.1 12.0
Epoch 5: loss 0.0590851716697216
5.0 7.2 15.0
Epoch 5: loss 0.2857484817504883
6.0 10.1 18.0
Epoch 5: loss 0.34505495429039
1.0 2.1 3.0
Epoch 6: loss 0.009040413424372673
2.0 3.5 6.0
Epoch 6: loss 0.06999438256025314
3.0 3.0 9.0
Epoch 6: loss 0.008999157696962357
```

```
4.0 2.1 12.0
Epoch 6: loss 0.023306062445044518
5.0 7.2 15.0
Epoch 6: loss 0.16568826138973236
6.0 10.1 18.0
Epoch 6: loss 0.01761181280016899
1.0 2.1 3.0
Epoch 7: loss 4.33617242379114e-06
2.0 3.5 6.0
Epoch 7: loss 0.0026713244151324034
3.0 3.0 9.0
Epoch 7: loss 0.03928030654788017
4.0 2.1 12.0
Epoch 7: loss 0.09321609884500504
5.0 7.2 15.0
Epoch 7: loss 0.06957114487886429
6.0 10.1 18.0
Epoch 7: loss 0.011286793276667595
1.0 2.1 3.0
Epoch 8: loss 0.002196304500102997
2.0 3.5 6.0
Epoch 8: loss 0.03541134297847748
3.0 3.0 9.0
Epoch 8: loss 0.0862458124756813
4.0 2.1 12.0
Epoch 8: loss 0.0982000008225441
5.0 7.2 15.0
Epoch 8: loss 0.020359806716442108
6.0 10.1 18.0
Epoch 8: loss 0.03822524473071098
1.0 2.1 3.0
Epoch 9: loss 0.0033681599888950586
2.0 3.5 6.0
Epoch 9: loss 0.04531693831086159
3.0 3.0 9.0
Epoch 9: loss 0.07702486217021942
4.0 2.1 12.0
Epoch 9: loss 0.0637483298778534
5.0 7.2 15.0
Epoch 9: loss 0.0032400009222328663
6.0 10.1 18.0
Epoch 9: loss 0.03870105743408203
1.0 2.1 3.0
Epoch 10: loss 0.002627953654155135
2.0 3.5 6.0
Epoch 10: loss 0.03256683424115181
3.0 3.0 9.0
Epoch 10: loss 0.045274920761585236
```

```
4.0 2.1 12.0
Epoch 10: loss 0.02961055561900139
5.0 7.2 15.0
Epoch 10: loss 2.779305214062333e-05
6.0 10.1 18.0
Epoch 10: loss 0.024559713900089264
1.0 2.1 3.0
Epoch 11: loss 0.0014091769699007273
2.0 3.5 6.0
Epoch 11: loss 0.016424618661403656
3.0 3.0 9.0
Epoch 11: loss 0.019254494458436966
4.0 2.1 12.0
Epoch 11: loss 0.009834825061261654
5.0 7.2 15.0
Epoch 11: loss 0.00039397578802891076
6.0 10.1 18.0
Epoch 11: loss 0.011188123375177383
1.0 2.1 3.0
Epoch 12: loss 0.0005480207619257271
2.0 3.5 6.0
Epoch 12: loss 0.005988651886582375
3.0 3.0 9.0
Epoch 12: loss 0.0057303160429000854
4.0 2.1 12.0
Epoch 12: loss 0.0020062534604221582
5.0 7.2 15.0
Epoch 12: loss 0.0007531145238317549
6.0 10.1 18.0
Epoch 12: loss 0.0036306711845099926
1.0 2.1 3.0
Epoch 13: loss 0.00014419663057196885
2.0 3.5 6.0
Epoch 13: loss 0.0014290438266471028
3.0 3.0 9.0
Epoch 13: loss 0.0009451688965782523
4.0 2.1 12.0
Epoch 13: loss 0.00010506437683943659
5.0 7.2 15.0
Epoch 13: loss 0.0006409119232557714
6.0 10.1 18.0
Epoch 13: loss 0.0007098895730450749
1.0 2.1 3.0
Epoch 14: loss 1.7844713511294685e-05
2.0 3.5 6.0
Epoch 14: loss 0.00013416244473773986
3.0 3.0 9.0
Epoch 14: loss 1.2116743164369836e-05
```

```
4.0 2.1 12.0
Epoch 14: loss 6.300281529547647e-05
5.0 7.2 15.0
Epoch 14: loss 0.0003649994032457471
6.0 10.1 18.0
Epoch 14: loss 3.0342722311615944e-05
1.0 2.1 3.0
Epoch 15: loss 7.219847475425922e-08
2.0 3.5 6.0
Epoch 15: loss 9.038791176863015e-06
3.0 3.0 9.0
Epoch 15: loss 9.844663145486265e-05
4.0 2.1 12.0
Epoch 15: loss 0.0002179707371396944
5.0 7.2 15.0
Epoch 15: loss 0.00015066929336171597
6.0 10.1 18.0
Epoch 15: loss 2.9819671908626333e-05
1.0 2.1 3.0
Epoch 16: loss 5.304377737047616e-06
2.0 3.5 6.0
Epoch 16: loss 8.413364412263036e-05
3.0 3.0 9.0
Epoch 16: loss 0.00019910804985556751
4.0 2.1 12.0
Epoch 16: loss 0.0002218172885477543
5.0 7.2 15.0
Epoch 16: loss 4.3087937228847295e-05
6.0 10.1 18.0
Epoch 16: loss 8.906755829229951e-05
1.0 2.1 3.0
Epoch 17: loss 7.697723049204797e-06
2.0 3.5 6.0
Epoch 17: loss 0.00010301192378392443
3.0 3.0 9.0
Epoch 17: loss 0.00017287800437770784
4.0 2.1 12.0
Epoch 17: loss 0.00014117785030975938
5.0 7.2 15.0
Epoch 17: loss 6.527480763907079e-06
6.0 10.1 18.0
Epoch 17: loss 8.724094368517399e-05
1.0 2.1 3.0
Epoch 18: loss 5.86423129789182e-06
2.0 3.5 6.0
Epoch 18: loss 7.242202264023945e-05
3.0 3.0 9.0
Epoch 18: loss 9.977581794373691e-05
```

```
4.0 2.1 12.0
Epoch 18: loss 6.447990017477423e-05
5.0 7.2 15.0
Epoch 18: loss 2.5976078177336603e-08
6.0 10.1 18.0
Epoch 18: loss 5.431682802736759e-05
1.0 2.1 3.0
Epoch 19: loss 3.090910695391358e-06
2.0 3.5 6.0
Epoch 19: loss 3.591472341213375e-05
3.0 3.0 9.0
Epoch 19: loss 4.172173066763207e-05
4.0 2.1 12.0
Epoch 19: loss 2.099843550240621e-05
5.0 7.2 15.0
Epoch 19: loss 9.705117918201722e-07
6.0 10.1 18.0
Epoch 19: loss 2.4328604922629893e-05
```

[3]:
```python
for i, curr in enumerate(data):
    x1, x2, y = curr
    x = torch.FloatTensor([x1, x2])
    y = torch.FloatTensor([y])
    out = net(x)
    print(f'x={x}, y={out}')
```

```
x=tensor([1.0000, 2.1000]), y=tensor([2.9989], grad_fn=<SqueezeBackward4>)
x=tensor([2.0000, 3.5000]), y=tensor([5.9980], grad_fn=<SqueezeBackward4>)
x=tensor([3., 3.]), y=tensor([8.9978], grad_fn=<SqueezeBackward4>)
x=tensor([4.0000, 2.1000]), y=tensor([11.9976], grad_fn=<SqueezeBackward4>)
x=tensor([5.0000, 7.2000]), y=tensor([14.9956], grad_fn=<SqueezeBackward4>)
x=tensor([ 6.0000, 10.1000]), y=tensor([17.9943], grad_fn=<SqueezeBackward4>)
```

[4]:
```python
data
```

[4]:
```
[(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, 7.2, 15.0),
 (6.0, 10.1, 18.0)]
```

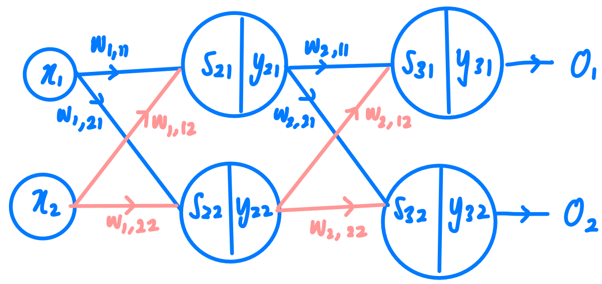
Homework 2 : Q2

$$W_1 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} w_{1,11} & w_{1,12} \\ w_{1,21} & w_{1,22} \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} w_{2,11} & w_{2,12} \\ w_{2,21} & w_{2,22} \end{bmatrix}$$

$$X = [1 \quad 2] = [x_1 \quad x_2]$$

$$\text{ReLU}: \phi(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$$



$$\frac{\partial L}{\partial y_{Li}} = \sum_{j=1}^{N} \frac{\partial L}{\partial y_{L+1,j}} w_{L,ij}$$

$$\frac{\partial L}{\partial w_{L,ij}} = y_{Li} \frac{\partial L}{\partial y_{L+1,j}}$$

Find $\frac{\partial L}{\partial W_1}$ & $\frac{\partial L}{\partial W_2}$

Forward :

$$S_2 = XW_1 = [1 \quad 2]\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} = [2 \quad 3]$$

$$Y_2 = \phi(S_2) = [2 \quad 3]$$

$$S_3 = Y_2 W_2 = [2 \quad 3]\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} = [5 \quad 10]$$

$$Y_3 = \phi(S_3) = [5 \quad 10]$$

Backpropagation :

$$L = \frac{1}{2}\sum_{i=1}^{2} (T_i - Y_{3i})^2 = \frac{1}{2}\left[(0-5)^2 + (0-10)^2\right] = 62.5$$

$$\frac{\partial L}{\partial w_{2,ij}} = y_{2i} \frac{\partial L}{\partial y_{3j}}$$

$$\frac{\partial L}{\partial y_{31}} = -(0-5) = 5 \quad , \quad \frac{\partial L}{\partial y_{32}} = -(0-10) = 10$$

$$\frac{\partial L}{\partial w_{2,11}} = y_{21} \frac{\partial L}{\partial y_{31}} = 2 \cdot 5 = 10$$

$$\frac{\partial L}{\partial w_{2,12}} = y_{21} \frac{\partial L}{\partial y_{32}} = 2 \cdot 10 = 20$$

$$\frac{\partial L}{\partial w_{2,21}} = y_{22} \frac{\partial L}{\partial y_{31}} = 3 \cdot 5 = 15$$

$$\frac{\partial L}{\partial w_{2,22}} = y_{22} \frac{\partial L}{\partial y_{32}} = 3 \cdot 10 = 30$$

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

$$L = \frac{1}{2}\sum_{i=1}^{2} (T_i - Y_{2i})^2 = \frac{1}{2}\left[(0-2)^2 + (0-3)^2\right] = 6.5$$

$$\frac{\partial L}{\partial w_{1,ij}} = x_i \frac{\partial L}{\partial y_{2j}}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial L}{\partial y_{31}} \cdot w_{2,11} + \frac{\partial L}{\partial y_{32}} \cdot w_{2,12} = 5 \cdot 1 + 10 \cdot 2 = 25$$

$$\frac{\partial L}{\partial y_{22}} = \frac{\partial L}{\partial y_{31}} \cdot w_{2,21} + \frac{\partial L}{\partial y_{32}} \cdot w_{2,22} = 5 \cdot 1 + 10 \cdot 2 = 25$$

$$\frac{\partial L}{\partial w_{1,11}} = x_1 \frac{\partial L}{\partial y_{21}} = 1 \cdot 25 = 25$$

$$\frac{\partial L}{\partial w_{1,12}} = x_1 \frac{\partial L}{\partial y_{22}} = 1 \cdot 25 = 25$$

$$\frac{\partial L}{\partial w_{1,21}} = x_2 \frac{\partial L}{\partial y_{21}} = 2 \cdot 25 = 50$$

$$\frac{\partial L}{\partial w_{1,22}} = x_2 \frac{\partial L}{\partial y_{22}} = 2 \cdot 25 = 50$$

$$\frac{\partial L}{\partial W_1} = \begin{bmatrix} 25 & 25 \\ 50 & 50 \end{bmatrix}$$

Homework 2 : Q4

How many parameters are needed and how many operations are needed to classify each input image (assume each MAC as two operations)?

	# of nodes
input	28 * 28 = 784
hidden layer 1 (H1)	512
hidden layer 2 (H2)	512
output	10

\# of parameters

= [(# input nodes × # H1 nodes) + # H1 bias)] + [(# H1 nodes × # H2 nodes) + # H2 bias] + [(# H2 nodes × # output nodes) + # output bias]

$= [(784 \times 512) + 512] + [(512 \times 512) + 512] + [(512 \times 10) + 10]$

$= 401920 + 262656 + 5130$

$= 669706$

MAC operations counted as 2

\# of operations

= [MAC between input & H1 nodes] + [MAC between H1 nodes & H2 nodes] + [MAC between H2 & output nodes] + [ReLU for H1 nodes, H2 nodes & output nodes]

$= [(784 \times 512) \times 2] + [(512 \times 512) \times 2] + [(512 \times 10) \times 2] + [512 + 512 + 10]$

$= 802816 + 524288 + 10240 + 1034$

$= 1338378$

∴ 1338378 operations needed and 669706 parameters needed.

[HW_Code3]_Mult-layer_Perceptron_Non_linear_Training

October 14, 2025

[1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import matplotlib.pyplot as plt
```

[2]:
```
class Net(nn.Module): ## nn.Module class is used
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(1,2,bias=True)
        self.fc2 = nn.Linear(2,8,bias=True)
        self.fc3 = nn.Linear(8,1,bias=True)
        self.act = nn.Tanh()
    def forward(self, x):
        x = self.act(self.fc1(x))
        x = self.act(self.fc2(x))
        y = self.fc3(x)
        return y
```

[3]:
```
net = Net()
net.fc1.weight = torch.nn.Parameter(torch.tensor([[0.0]] * 2,␣
 ↪requires_grad=True))
optimizer = optim.Adam(net.parameters(), lr=0.005, eps=1e-5)
criterion = nn.MSELoss()

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

```
Net(
  (fc1): Linear(in_features=1, out_features=2, bias=True)
  (fc2): Linear(in_features=2, out_features=8, bias=True)
  (fc3): Linear(in_features=8, out_features=1, bias=True)
  (act): Tanh()
)
[Parameter containing:
tensor([[0.],
        [0.]], requires_grad=True), Parameter containing:
```

```
tensor([0.1108, 0.3154], requires_grad=True), Parameter containing:
tensor([[-0.3113,  0.6426],
        [ 0.2954, -0.4443],
        [ 0.6001, -0.1274],
        [ 0.5943,  0.5343],
        [-0.7065, -0.0204],
        [-0.3096,  0.3166],
        [ 0.5900,  0.1377],
        [-0.5271, -0.4785]], requires_grad=True), Parameter containing:
tensor([-0.7037, -0.0238,  0.2514,  0.1095,  0.4476, -0.4862, -0.0645,  0.3112],
       requires_grad=True), Parameter containing:
tensor([[ 0.2427,  0.1975,  0.0613,  0.1042, -0.2229, -0.1197, -0.1986,
0.0387]],
       requires_grad=True), Parameter containing:
tensor([-0.3173], requires_grad=True)]
```

[4]:
```
data = [(1.0,3.0), (2.0,6.0), (3.0,10.0), (4.0,15.0), (5.0,22.0), (6.0,32.0)]
lowest_loss = float('inf')
for epoch in range(int(1e5)):
    total_loss = 0
    for d in data:
        X, Y = torch.FloatTensor([[d[0]]]), torch.FloatTensor([[d[1]]])
        optimizer.zero_grad()
        outputs = net(X)
        loss = criterion(outputs, Y)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    if total_loss < 1e-5:
        print(f'Epoch {epoch} - Loss: {total_loss:.6}')
        break
```

```
Epoch 2852 - Loss: 9.97796e-06
```

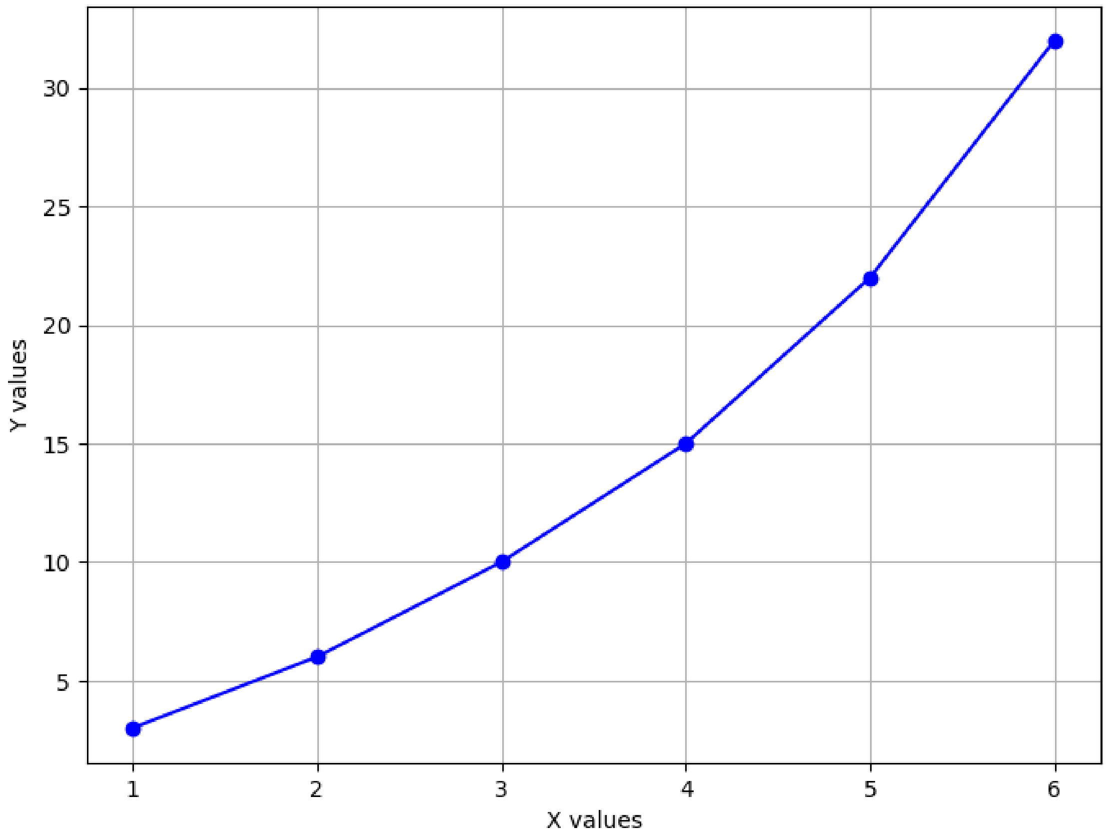
[5]:
```
### Test the trained network ###
for i, current_data in enumerate(data):
    X, Y = current_data
    X, Y = torch.FloatTensor([X]), torch.FloatTensor([Y])
    out = net(torch.FloatTensor(X))
    print("when x = {}, y = {}".format(X, out))
```

```
when x = tensor([1.]), y = tensor([2.9995], grad_fn=<ViewBackward0>)
when x = tensor([2.]), y = tensor([6.0015], grad_fn=<ViewBackward0>)
when x = tensor([3.]), y = tensor([9.9978], grad_fn=<ViewBackward0>)
when x = tensor([4.]), y = tensor([15.0011], grad_fn=<ViewBackward0>)
when x = tensor([5.]), y = tensor([21.9998], grad_fn=<ViewBackward0>)
when x = tensor([6.]), y = tensor([32.0001], grad_fn=<ViewBackward0>)
```

[6]:
```
data
```

[6]:
```
[(1.0, 3.0), (2.0, 6.0), (3.0, 10.0), (4.0, 15.0), (5.0, 22.0), (6.0, 32.0)]
```

[7]:
```
# Plotting
x,y = zip(*data)
plt.figure(figsize=(8, 6))
plt.plot(x, y, marker='o', linestyle='-', color='b')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.grid(True)
plt.show()
```



[1]:
```
# Code used to find the best model hyperparameter
# from itertools import product

# def train_model(h1, h2, act, adam_params):
#     class Net(nn.Module):
#         def __init__(self):
#             super(Net, self).__init__()
```

```
#             self.fc1 = nn.Linear(1, h1, bias=True)
#             self.fc2 = nn.Linear(h1, h2, bias=True)
#             self.fc3 = nn.Linear(h2, 1, bias=True)
#             self.act = act

#         def forward(self, x):
#             x = self.act(self.fc1(x))
#             x = self.act(self.fc2(x))
#             return self.fc3(x)

#     net = Net()
#     criterion = nn.MSELoss()
#     optimizer = optim.Adam(
#         net.parameters(),
#         lr = adam_params['lr'],
#         betas = adam_params['betas'],
#         eps = adam_params['eps'],
#         weight_decay = adam_params['wd']
#     )

#     for epoch in range(3000):
#         total_loss = 0.0
#         for x, y in data:
#             X = torch.FloatTensor([[x]])
#             Y = torch.FloatTensor([[y]])
#             optimizer.zero_grad()
#             loss = criterion(net(X), Y)
#             loss.backward()
#             optimizer.step()
#             total_loss += loss.item()

#         if total_loss < 1e-5:
#             break
#     return total_loss, epoch

# # Options
# hs1 = list(range(2, 11))      # hidden layer 1 size
# hs2 = list(range(2, 11))      # hidden layer 2 size
# acts = [nn.ReLU(), nn.Sigmoid(), nn.Tanh()] # activation function
# adam_configs = [
#     {'lr': 0.001, 'betas': (0.9, 0.999), 'eps': 1e-8, 'wd': 0.0}, # default
#     {'lr': 0.005, 'betas': (0.9, 0.999), 'eps': 1e-8, 'wd': 0.0}, # increase␣
 ↪learning rate
#     {'lr': 0.005, 'betas': (0.8, 0.999), 'eps': 1e-8, 'wd': 0.0}, # decrease␣
 ↪betas
#     {'lr': 0.005, 'betas': (0.8, 0.999), 'eps': 1e-6, 'wd': 0.0}, # increase␣
 ↪epsilon
```

```
#     {'lr': 0.005, 'betas': (0.8, 0.999), 'eps': 1e-6, 'wd': 1e-5},# increase
 ↪weight decay
# ]

# best_config = None
# best_loss = float('inf')
# for h1, h2, act, adam_params in product(hs1, hs2, acts, adam_configs):
#     loss, ep = train_model(h1, h2, act, adam_params)
#     print(f"h1={h1}, h2={h2}, act={act.__class__.__name__}, "
#           f"lr={adam_params['lr']}, betas={adam_params['betas']}, "
#           f"eps={adam_params['eps']}, wd={adam_params['wd']} "
#           f" -- loss={loss:.6e}, epoch={ep}")

#     if loss < best_loss:
#         best_loss = loss
#         best_config = (h1, h2, act.__class__.__name__, adam_params, ep)

# # Best configuration
# print(f"Hidden sizes: ({best_config[0]}, {best_config[1]})")
# print(f"Activation: {best_config[2]}")
# print(f"Adam params: {best_config[3]}")
# print(f"Epoch: {best_config[4]}")
# print(f"Final loss: {best_loss:.6}")
```

[HW_Code4]_Perceptron_batch_vs_SGD

October 16, 2025

[1]:
```
import torch
from torch.autograd import Variable
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):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(2,1,bias=False)
    def forward(self, x):
        x = self.fc1(x)
        return x

net = Net()
print(net)
net.fc1.weight = torch.nn.Parameter(torch.tensor([[1., -1.]],␣
↪requires_grad=True))
print(list(net.parameters()))
criterion = nn.MSELoss()

optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.5)
#optimizer = optim.Adam(net.parameters(), lr=0.005)

data = torch.tensor([[1.,3.], [2.,6.], [3.,9.]], dtype=torch.float)
target = torch.tensor([[1.],[5.],[13.]], dtype=torch.float)

hist = []

############# Batch GD based update #############

for epoch in range(300):
    optimizer.zero_grad()
    outputs = net(data)
    loss = criterion(outputs, target)
```

```
    loss.backward()
    hist.append(loss.detach())
    optimizer.step()
    print("Epoch {} - loss: {}".format(epoch, loss))
####################################################

### Test the trained network ###
for i, current_data in enumerate(data):
    out = net(current_data)
    print("when x = {}, y = {}".format(current_data, out))

plt.plot(hist, label = "training curve")
```

```
Net(
  (fc1): Linear(in_features=2, out_features=1, bias=False)
)
[Parameter containing:
tensor([[ 1., -1.]], requires_grad=True)]
Epoch 0 - loss: 150.3333282470703
Epoch 1 - loss: 6.120000839233398
Epoch 2 - loss: 36.42483139038086
Epoch 3 - loss: 18.104551315307617
Epoch 4 - loss: 6.065778732299805
Epoch 5 - loss: 10.36883544921875
Epoch 6 - loss: 6.2322306632995605
Epoch 7 - loss: 5.852260589599609
Epoch 8 - loss: 6.088119983673096
Epoch 9 - loss: 5.494865417480469
Epoch 10 - loss: 5.574592590332031
Epoch 11 - loss: 5.536750793457031
Epoch 12 - loss: 5.476492404937744
Epoch 13 - loss: 5.4938530921936035
Epoch 14 - loss: 5.480627536773682
Epoch 15 - loss: 5.477013111114502
Epoch 16 - loss: 5.478647232055664
Epoch 17 - loss: 5.476379871368408
Epoch 18 - loss: 5.476480007171631
Epoch 19 - loss: 5.4764628410339355
Epoch 20 - loss: 5.47619104385376
Epoch 21 - loss: 5.476251602172852
Epoch 22 - loss: 5.476212978363037
Epoch 23 - loss: 5.476190567016602
Epoch 24 - loss: 5.476199626922607
Epoch 25 - loss: 5.476192474365234
Epoch 26 - loss: 5.476190567016602
Epoch 27 - loss: 5.476191997528076
Epoch 28 - loss: 5.476190090179443
```

```
Epoch 29 - loss: 5.476190567016602
Epoch 30 - loss: 5.476190090179443
Epoch 31 - loss: 5.476190567016602
Epoch 32 - loss: 5.47619104385376
Epoch 33 - loss: 5.476190090179443
Epoch 34 - loss: 5.476190090179443
Epoch 35 - loss: 5.476190567016602
Epoch 36 - loss: 5.476190090179443
Epoch 37 - loss: 5.47619104385376
Epoch 38 - loss: 5.47619104385376
Epoch 39 - loss: 5.476190567016602
Epoch 40 - loss: 5.47619104385376
Epoch 41 - loss: 5.476188659667969
Epoch 42 - loss: 5.476190090179443
Epoch 43 - loss: 5.476189136505127
Epoch 44 - loss: 5.476190090179443
Epoch 45 - loss: 5.476191997528076
Epoch 46 - loss: 5.476191997528076
Epoch 47 - loss: 5.476190090179443
Epoch 48 - loss: 5.476190567016602
Epoch 49 - loss: 5.476190567016602
Epoch 50 - loss: 5.476190090179443
Epoch 51 - loss: 5.47619104385376
Epoch 52 - loss: 5.476190567016602
Epoch 53 - loss: 5.47619104385376
Epoch 54 - loss: 5.47619104385376
Epoch 55 - loss: 5.47619104385376
Epoch 56 - loss: 5.476190567016602
Epoch 57 - loss: 5.47619104385376
Epoch 58 - loss: 5.47619104385376
Epoch 59 - loss: 5.47619104385376
Epoch 60 - loss: 5.476190567016602
Epoch 61 - loss: 5.47619104385376
Epoch 62 - loss: 5.47619104385376
Epoch 63 - loss: 5.47619104385376
Epoch 64 - loss: 5.476190567016602
Epoch 65 - loss: 5.47619104385376
Epoch 66 - loss: 5.47619104385376
Epoch 67 - loss: 5.47619104385376
Epoch 68 - loss: 5.476190567016602
Epoch 69 - loss: 5.47619104385376
Epoch 70 - loss: 5.47619104385376
Epoch 71 - loss: 5.47619104385376
Epoch 72 - loss: 5.476190567016602
Epoch 73 - loss: 5.47619104385376
Epoch 74 - loss: 5.47619104385376
Epoch 75 - loss: 5.47619104385376
Epoch 76 - loss: 5.476190567016602
```

```
Epoch 77 - loss: 5.47619104385376
Epoch 78 - loss: 5.47619104385376
Epoch 79 - loss: 5.47619104385376
Epoch 80 - loss: 5.476190567016602
Epoch 81 - loss: 5.47619104385376
Epoch 82 - loss: 5.47619104385376
Epoch 83 - loss: 5.47619104385376
Epoch 84 - loss: 5.476190567016602
Epoch 85 - loss: 5.47619104385376
Epoch 86 - loss: 5.47619104385376
Epoch 87 - loss: 5.47619104385376
Epoch 88 - loss: 5.476190567016602
Epoch 89 - loss: 5.47619104385376
Epoch 90 - loss: 5.47619104385376
Epoch 91 - loss: 5.47619104385376
Epoch 92 - loss: 5.476190567016602
Epoch 93 - loss: 5.47619104385376
Epoch 94 - loss: 5.47619104385376
Epoch 95 - loss: 5.47619104385376
Epoch 96 - loss: 5.476190567016602
Epoch 97 - loss: 5.47619104385376
Epoch 98 - loss: 5.47619104385376
Epoch 99 - loss: 5.47619104385376
Epoch 100 - loss: 5.476190567016602
Epoch 101 - loss: 5.47619104385376
Epoch 102 - loss: 5.47619104385376
Epoch 103 - loss: 5.47619104385376
Epoch 104 - loss: 5.476190567016602
Epoch 105 - loss: 5.47619104385376
Epoch 106 - loss: 5.47619104385376
Epoch 107 - loss: 5.47619104385376
Epoch 108 - loss: 5.476190567016602
Epoch 109 - loss: 5.47619104385376
Epoch 110 - loss: 5.47619104385376
Epoch 111 - loss: 5.47619104385376
Epoch 112 - loss: 5.476190567016602
Epoch 113 - loss: 5.47619104385376
Epoch 114 - loss: 5.47619104385376
Epoch 115 - loss: 5.47619104385376
Epoch 116 - loss: 5.476190567016602
Epoch 117 - loss: 5.47619104385376
Epoch 118 - loss: 5.47619104385376
Epoch 119 - loss: 5.47619104385376
Epoch 120 - loss: 5.476190567016602
Epoch 121 - loss: 5.47619104385376
Epoch 122 - loss: 5.47619104385376
Epoch 123 - loss: 5.47619104385376
Epoch 124 - loss: 5.476190567016602
```

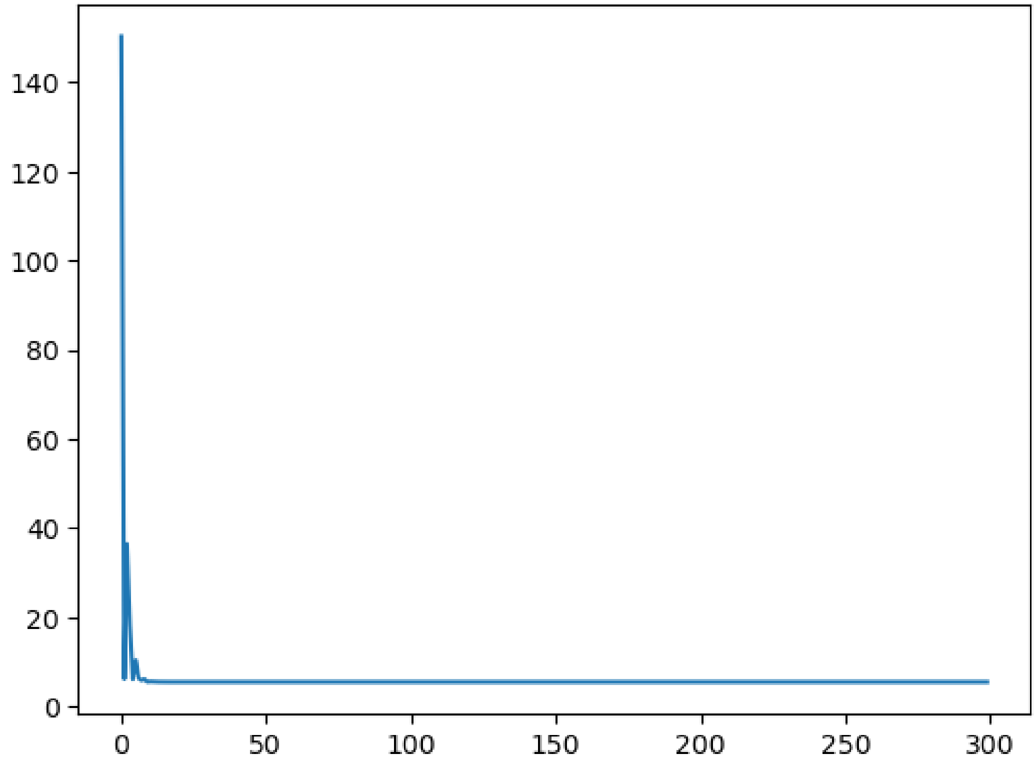
```
Epoch 125 - loss: 5.47619104385376
Epoch 126 - loss: 5.47619104385376
Epoch 127 - loss: 5.47619104385376
Epoch 128 - loss: 5.476190567016602
Epoch 129 - loss: 5.47619104385376
Epoch 130 - loss: 5.47619104385376
Epoch 131 - loss: 5.47619104385376
Epoch 132 - loss: 5.476190567016602
Epoch 133 - loss: 5.47619104385376
Epoch 134 - loss: 5.47619104385376
Epoch 135 - loss: 5.47619104385376
Epoch 136 - loss: 5.476190567016602
Epoch 137 - loss: 5.47619104385376
Epoch 138 - loss: 5.47619104385376
Epoch 139 - loss: 5.47619104385376
Epoch 140 - loss: 5.476190567016602
Epoch 141 - loss: 5.47619104385376
Epoch 142 - loss: 5.47619104385376
Epoch 143 - loss: 5.47619104385376
Epoch 144 - loss: 5.476190567016602
Epoch 145 - loss: 5.47619104385376
Epoch 146 - loss: 5.47619104385376
Epoch 147 - loss: 5.47619104385376
Epoch 148 - loss: 5.476190567016602
Epoch 149 - loss: 5.47619104385376
Epoch 150 - loss: 5.47619104385376
Epoch 151 - loss: 5.47619104385376
Epoch 152 - loss: 5.476190567016602
Epoch 153 - loss: 5.47619104385376
Epoch 154 - loss: 5.47619104385376
Epoch 155 - loss: 5.47619104385376
Epoch 156 - loss: 5.476190567016602
Epoch 157 - loss: 5.47619104385376
Epoch 158 - loss: 5.47619104385376
Epoch 159 - loss: 5.47619104385376
Epoch 160 - loss: 5.476190567016602
Epoch 161 - loss: 5.47619104385376
Epoch 162 - loss: 5.47619104385376
Epoch 163 - loss: 5.47619104385376
Epoch 164 - loss: 5.476190567016602
Epoch 165 - loss: 5.47619104385376
Epoch 166 - loss: 5.47619104385376
Epoch 167 - loss: 5.47619104385376
Epoch 168 - loss: 5.476190567016602
Epoch 169 - loss: 5.47619104385376
Epoch 170 - loss: 5.47619104385376
Epoch 171 - loss: 5.47619104385376
Epoch 172 - loss: 5.476190567016602
```

```
Epoch 173 - loss: 5.47619104385376
Epoch 174 - loss: 5.47619104385376
Epoch 175 - loss: 5.47619104385376
Epoch 176 - loss: 5.476190567016602
Epoch 177 - loss: 5.47619104385376
Epoch 178 - loss: 5.47619104385376
Epoch 179 - loss: 5.47619104385376
Epoch 180 - loss: 5.476190567016602
Epoch 181 - loss: 5.47619104385376
Epoch 182 - loss: 5.47619104385376
Epoch 183 - loss: 5.47619104385376
Epoch 184 - loss: 5.476190567016602
Epoch 185 - loss: 5.47619104385376
Epoch 186 - loss: 5.47619104385376
Epoch 187 - loss: 5.47619104385376
Epoch 188 - loss: 5.476190567016602
Epoch 189 - loss: 5.47619104385376
Epoch 190 - loss: 5.47619104385376
Epoch 191 - loss: 5.47619104385376
Epoch 192 - loss: 5.476190567016602
Epoch 193 - loss: 5.47619104385376
Epoch 194 - loss: 5.47619104385376
Epoch 195 - loss: 5.47619104385376
Epoch 196 - loss: 5.476190567016602
Epoch 197 - loss: 5.47619104385376
Epoch 198 - loss: 5.47619104385376
Epoch 199 - loss: 5.47619104385376
Epoch 200 - loss: 5.476190567016602
Epoch 201 - loss: 5.47619104385376
Epoch 202 - loss: 5.47619104385376
Epoch 203 - loss: 5.47619104385376
Epoch 204 - loss: 5.476190567016602
Epoch 205 - loss: 5.47619104385376
Epoch 206 - loss: 5.47619104385376
Epoch 207 - loss: 5.47619104385376
Epoch 208 - loss: 5.476190567016602
Epoch 209 - loss: 5.47619104385376
Epoch 210 - loss: 5.47619104385376
Epoch 211 - loss: 5.47619104385376
Epoch 212 - loss: 5.476190567016602
Epoch 213 - loss: 5.47619104385376
Epoch 214 - loss: 5.47619104385376
Epoch 215 - loss: 5.47619104385376
Epoch 216 - loss: 5.476190567016602
Epoch 217 - loss: 5.47619104385376
Epoch 218 - loss: 5.47619104385376
Epoch 219 - loss: 5.47619104385376
Epoch 220 - loss: 5.476190567016602
```

```
Epoch 221 - loss: 5.47619104385376
Epoch 222 - loss: 5.47619104385376
Epoch 223 - loss: 5.47619104385376
Epoch 224 - loss: 5.476190567016602
Epoch 225 - loss: 5.47619104385376
Epoch 226 - loss: 5.47619104385376
Epoch 227 - loss: 5.47619104385376
Epoch 228 - loss: 5.476190567016602
Epoch 229 - loss: 5.47619104385376
Epoch 230 - loss: 5.47619104385376
Epoch 231 - loss: 5.47619104385376
Epoch 232 - loss: 5.476190567016602
Epoch 233 - loss: 5.47619104385376
Epoch 234 - loss: 5.47619104385376
Epoch 235 - loss: 5.47619104385376
Epoch 236 - loss: 5.476190567016602
Epoch 237 - loss: 5.47619104385376
Epoch 238 - loss: 5.47619104385376
Epoch 239 - loss: 5.47619104385376
Epoch 240 - loss: 5.476190567016602
Epoch 241 - loss: 5.47619104385376
Epoch 242 - loss: 5.47619104385376
Epoch 243 - loss: 5.47619104385376
Epoch 244 - loss: 5.476190567016602
Epoch 245 - loss: 5.47619104385376
Epoch 246 - loss: 5.47619104385376
Epoch 247 - loss: 5.47619104385376
Epoch 248 - loss: 5.476190567016602
Epoch 249 - loss: 5.47619104385376
Epoch 250 - loss: 5.47619104385376
Epoch 251 - loss: 5.47619104385376
Epoch 252 - loss: 5.476190567016602
Epoch 253 - loss: 5.47619104385376
Epoch 254 - loss: 5.47619104385376
Epoch 255 - loss: 5.47619104385376
Epoch 256 - loss: 5.476190567016602
Epoch 257 - loss: 5.47619104385376
Epoch 258 - loss: 5.47619104385376
Epoch 259 - loss: 5.47619104385376
Epoch 260 - loss: 5.476190567016602
Epoch 261 - loss: 5.47619104385376
Epoch 262 - loss: 5.47619104385376
Epoch 263 - loss: 5.47619104385376
Epoch 264 - loss: 5.476190567016602
Epoch 265 - loss: 5.47619104385376
Epoch 266 - loss: 5.47619104385376
Epoch 267 - loss: 5.47619104385376
Epoch 268 - loss: 5.476190567016602
```

```
Epoch 269 - loss: 5.47619104385376
Epoch 270 - loss: 5.47619104385376
Epoch 271 - loss: 5.47619104385376
Epoch 272 - loss: 5.476190567016602
Epoch 273 - loss: 5.47619104385376
Epoch 274 - loss: 5.47619104385376
Epoch 275 - loss: 5.47619104385376
Epoch 276 - loss: 5.476190567016602
Epoch 277 - loss: 5.47619104385376
Epoch 278 - loss: 5.47619104385376
Epoch 279 - loss: 5.47619104385376
Epoch 280 - loss: 5.476190567016602
Epoch 281 - loss: 5.47619104385376
Epoch 282 - loss: 5.47619104385376
Epoch 283 - loss: 5.47619104385376
Epoch 284 - loss: 5.476190567016602
Epoch 285 - loss: 5.47619104385376
Epoch 286 - loss: 5.47619104385376
Epoch 287 - loss: 5.47619104385376
Epoch 288 - loss: 5.476190567016602
Epoch 289 - loss: 5.47619104385376
Epoch 290 - loss: 5.47619104385376
Epoch 291 - loss: 5.47619104385376
Epoch 292 - loss: 5.476190567016602
Epoch 293 - loss: 5.47619104385376
Epoch 294 - loss: 5.47619104385376
Epoch 295 - loss: 5.47619104385376
Epoch 296 - loss: 5.476190567016602
Epoch 297 - loss: 5.47619104385376
Epoch 298 - loss: 5.47619104385376
Epoch 299 - loss: 5.47619104385376
when x = tensor([1., 3.]), y = tensor([3.5714], grad_fn=<SqueezeBackward4>)
when x = tensor([2., 6.]), y = tensor([7.1429], grad_fn=<SqueezeBackward4>)
when x = tensor([3., 9.]), y = tensor([10.7143], grad_fn=<SqueezeBackward4>)
```

```
[1]: [<matplotlib.lines.Line2D at 0x7fc423960c90>]
```



[2]:
```
net = Net()
print(net)
net.fc1.weight = torch.nn.Parameter(torch.tensor([[1., -1.]],
    requires_grad=True))
print(list(net.parameters()))
criterion = nn.MSELoss()

optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.5)
#optimizer = optim.Adam(net.parameters(), lr=0.005)

data = torch.tensor([[1.,3.], [2.,6.], [3.,9.]], dtype=torch.float)
target = torch.tensor([[1.],[5.],[13.]], dtype=torch.float)

hist = []

############## SGD based update ##############

for epoch in range(300):
    for d, t in zip(data, target):
        optimizer.zero_grad()
        outputs = net(d)
        loss = criterion(outputs, t)
```

```python
        loss.backward()
        optimizer.step()
    hist.append(loss.detach())
    print("Epoch {} - loss: {}".format(epoch, loss))
####################################################

### Test the trained network ###
for i, current_data in enumerate(data):
    out = net(current_data)
    print("when x = {}, y = {}".format(current_data, out))

plt.plot(hist, label = "training curve")
```

```
Net(
  (fc1): Linear(in_features=2, out_features=1, bias=False)
)
[Parameter containing:
tensor([[ 1., -1.]], requires_grad=True)]
Epoch 0 - loss: 48.16359329223633
Epoch 1 - loss: 1.0173128843307495
Epoch 2 - loss: 88.83402252197266
Epoch 3 - loss: 1.685847282409668
Epoch 4 - loss: 89.11278533935547
Epoch 5 - loss: 1.635256290435791
Epoch 6 - loss: 88.66230010986328
Epoch 7 - loss: 1.5735151767730713
Epoch 8 - loss: 88.20365142822266
Epoch 9 - loss: 1.5130443572998047
Epoch 10 - loss: 87.74807739257812
Epoch 11 - loss: 1.4540135860443115
Epoch 12 - loss: 87.29572296142578
Epoch 13 - loss: 1.396427869796753
Epoch 14 - loss: 86.84661102294922
Epoch 15 - loss: 1.3402595520019531
Epoch 16 - loss: 86.40071105957031
Epoch 17 - loss: 1.2854949235916138
Epoch 18 - loss: 85.95800018310547
Epoch 19 - loss: 1.232109785079956
Epoch 20 - loss: 85.51839447021484
Epoch 21 - loss: 1.180097222328186
Epoch 22 - loss: 85.08194732666016
Epoch 23 - loss: 1.129433274269104
Epoch 24 - loss: 84.64859771728516
Epoch 25 - loss: 1.0801048278808594
Epoch 26 - loss: 84.21830749511719
Epoch 27 - loss: 1.032092809677124
Epoch 28 - loss: 83.79109191894531
```

```
Epoch 29 - loss: 0.9853919744491577
Epoch 30 - loss: 83.36691284179688
Epoch 31 - loss: 0.9399775266647339
Epoch 32 - loss: 82.94575500488281
Epoch 33 - loss: 0.8958330154418945
Epoch 34 - loss: 82.52757263183594
Epoch 35 - loss: 0.8529386520385742
Epoch 36 - loss: 82.11231231689453
Epoch 37 - loss: 0.8112924098968506
Epoch 38 - loss: 81.70002746582031
Epoch 39 - loss: 0.7708677053451538
Epoch 40 - loss: 81.29064178466797
Epoch 41 - loss: 0.7316538095474243
Epoch 42 - loss: 80.8841323852539
Epoch 43 - loss: 0.693635106086731
Epoch 44 - loss: 80.48049926757812
Epoch 45 - loss: 0.6567976474761963
Epoch 46 - loss: 80.0797119140625
Epoch 47 - loss: 0.6211260557174683
Epoch 48 - loss: 79.6817626953125
Epoch 49 - loss: 0.5866096019744873
Epoch 50 - loss: 79.28660583496094
Epoch 51 - loss: 0.5532287359237671
Epoch 52 - loss: 78.89422607421875
Epoch 53 - loss: 0.5209743976593018
Epoch 54 - loss: 78.50464630126953
Epoch 55 - loss: 0.48982885479927063
Epoch 56 - loss: 78.11776733398438
Epoch 57 - loss: 0.4597790241241455
Epoch 58 - loss: 77.73362731933594
Epoch 59 - loss: 0.4308067560195923
Epoch 60 - loss: 77.35214233398438
Epoch 61 - loss: 0.4029043912887573
Epoch 62 - loss: 76.97335815429688
Epoch 63 - loss: 0.3760589063167572
Epoch 64 - loss: 76.59722137451172
Epoch 65 - loss: 0.3502528965473175
Epoch 66 - loss: 76.22370147705078
Epoch 67 - loss: 0.32547610998153687
Epoch 68 - loss: 75.85279083251953
Epoch 69 - loss: 0.30171602964401245
Epoch 70 - loss: 75.48448944091797
Epoch 71 - loss: 0.27895811200141907
Epoch 72 - loss: 75.11873626708984
Epoch 73 - loss: 0.25719210505485535
Epoch 74 - loss: 74.75557708740234
Epoch 75 - loss: 0.23640374839305878
Epoch 76 - loss: 74.39496612548828
```

```
Epoch 77 - loss: 0.21657827496528625
Epoch 78 - loss: 74.03682708740234
Epoch 79 - loss: 0.1977057158946991
Epoch 80 - loss: 73.68118286132812
Epoch 81 - loss: 0.17977581918239594
Epoch 82 - loss: 73.32803344726562
Epoch 83 - loss: 0.16277191042900085
Epoch 84 - loss: 72.97732543945312
Epoch 85 - loss: 0.1466856300830841
Epoch 86 - loss: 72.62907409667969
Epoch 87 - loss: 0.13150320947170258
Epoch 88 - loss: 72.283203125
Epoch 89 - loss: 0.11721332371234894
Epoch 90 - loss: 71.93976593017578
Epoch 91 - loss: 0.10380657762289047
Epoch 92 - loss: 71.59872436523438
Epoch 93 - loss: 0.09126798063516617
Epoch 94 - loss: 71.260009765625
Epoch 95 - loss: 0.07958778738975525
Epoch 96 - loss: 70.92364501953125
Epoch 97 - loss: 0.06875503063201904
Epoch 98 - loss: 70.58960723876953
Epoch 99 - loss: 0.05875876545906067
Epoch 100 - loss: 70.25787353515625
Epoch 101 - loss: 0.04958769679069519
Epoch 102 - loss: 69.9284439086914
Epoch 103 - loss: 0.041231099516153336
Epoch 104 - loss: 69.6012954711914
Epoch 105 - loss: 0.0336776077747345
Epoch 106 - loss: 69.2763900756836
Epoch 107 - loss: 0.02691650390625
Epoch 108 - loss: 68.95370483398438
Epoch 109 - loss: 0.02093837969005108
Epoch 110 - loss: 68.63325500488281
Epoch 111 - loss: 0.015731994062662125
Epoch 112 - loss: 68.31502532958984
Epoch 113 - loss: 0.011287805624306202
Epoch 114 - loss: 67.99899291992188
Epoch 115 - loss: 0.007594224996864796
Epoch 116 - loss: 67.68510437011719
Epoch 117 - loss: 0.004642413929104805
Epoch 118 - loss: 67.37336730957031
Epoch 119 - loss: 0.0024216780439019203
Epoch 120 - loss: 67.06376647949219
Epoch 121 - loss: 0.0009226119145750999
Epoch 122 - loss: 66.75631713867188
Epoch 123 - loss: 0.00013510302233044058
Epoch 124 - loss: 66.45097351074219
```

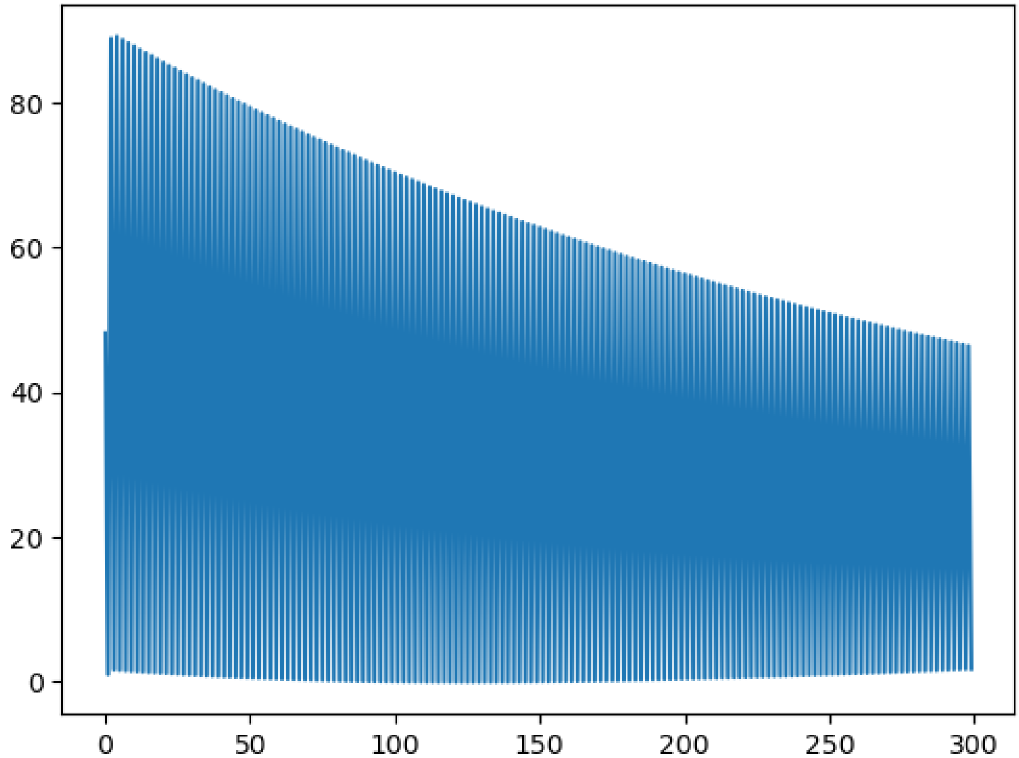
```
Epoch 125 - loss: 4.9588794354349375e-05
Epoch 126 - loss: 66.14769744873047
Epoch 127 - loss: 0.0006565545918419957
Epoch 128 - loss: 65.84648895263672
Epoch 129 - loss: 0.00194639153778553
Epoch 130 - loss: 65.54735565185547
Epoch 131 - loss: 0.0039095887914299965
Epoch 132 - loss: 65.25028228759766
Epoch 133 - loss: 0.00653728237375617
Epoch 134 - loss: 64.95523071289062
Epoch 135 - loss: 0.00982026569545269
Epoch 136 - loss: 64.66216278076172
Epoch 137 - loss: 0.013749008066952229
Epoch 138 - loss: 64.37110900878906
Epoch 139 - loss: 0.018314778804779053
Epoch 140 - loss: 64.0820541381836
Epoch 141 - loss: 0.023508287966251373
Epoch 142 - loss: 63.79496383666992
Epoch 143 - loss: 0.029322434216737747
Epoch 144 - loss: 63.50979995727539
Epoch 145 - loss: 0.03574701026082039
Epoch 146 - loss: 63.2265625
Epoch 147 - loss: 0.042773135006427765
Epoch 148 - loss: 62.94527053833008
Epoch 149 - loss: 0.05039283633232117
Epoch 150 - loss: 62.66588592529297
Epoch 151 - loss: 0.05859844386577606
Epoch 152 - loss: 62.388389587402344
Epoch 153 - loss: 0.06738011538982391
Epoch 154 - loss: 62.11277770996094
Epoch 155 - loss: 0.07672923803329468
Epoch 156 - loss: 61.83906555175781
Epoch 157 - loss: 0.0866406038403511
Epoch 158 - loss: 61.56716537475586
Epoch 159 - loss: 0.09710390120744705
Epoch 160 - loss: 61.29710006713867
Epoch 161 - loss: 0.10811145603656769
Epoch 162 - loss: 61.02886962890625
Epoch 163 - loss: 0.11965707689523697
Epoch 164 - loss: 60.762428283691406
Epoch 165 - loss: 0.13172948360443115
Epoch 166 - loss: 60.497802734375
Epoch 167 - loss: 0.14432398974895477
Epoch 168 - loss: 60.23495101928711
Epoch 169 - loss: 0.15743286907672882
Epoch 170 - loss: 59.97385025024414
Epoch 171 - loss: 0.17104679346084595
Epoch 172 - loss: 59.71451950073242
```

```
Epoch 173 - loss: 0.18515869975090027
Epoch 174 - loss: 59.456932067871094
Epoch 175 - loss: 0.19976170361042023
Epoch 176 - loss: 59.201087951660156
Epoch 177 - loss: 0.21484729647636414
Epoch 178 - loss: 58.94696044921875
Epoch 179 - loss: 0.23041146993637085
Epoch 180 - loss: 58.69451904296875
Epoch 181 - loss: 0.24644409120082855
Epoch 182 - loss: 58.44377899169922
Epoch 183 - loss: 0.26293960213661194
Epoch 184 - loss: 58.19470977783203
Epoch 185 - loss: 0.2798897325992584
Epoch 186 - loss: 57.94730758666992
Epoch 187 - loss: 0.29728710651397705
Epoch 188 - loss: 57.70157241821289
Epoch 189 - loss: 0.31512653827667236
Epoch 190 - loss: 57.45747756958008
Epoch 191 - loss: 0.33339980244636536
Epoch 192 - loss: 57.215023040771484
Epoch 193 - loss: 0.35210320353507996
Epoch 194 - loss: 56.97416687011719
Epoch 195 - loss: 0.3712274730205536
Epoch 196 - loss: 56.73491287231445
Epoch 197 - loss: 0.39076685905456543
Epoch 198 - loss: 56.49724197387695
Epoch 199 - loss: 0.4107156991958618
Epoch 200 - loss: 56.26115798950195
Epoch 201 - loss: 0.43106719851493835
Epoch 202 - loss: 56.026641845703125
Epoch 203 - loss: 0.45181208848953247
Epoch 204 - loss: 55.79368209838867
Epoch 205 - loss: 0.47294872999191284
Epoch 206 - loss: 55.562252044677734
Epoch 207 - loss: 0.4944652318954468
Epoch 208 - loss: 55.332393646240234
Epoch 209 - loss: 0.5163602828979492
Epoch 210 - loss: 55.104042053222656
Epoch 211 - loss: 0.5386289358139038
Epoch 212 - loss: 54.877193450927734
Epoch 213 - loss: 0.5612589716911316
Epoch 214 - loss: 54.651859283447266
Epoch 215 - loss: 0.5842483043670654
Epoch 216 - loss: 54.42802047729492
Epoch 217 - loss: 0.6075949668884277
Epoch 218 - loss: 54.20563507080078
Epoch 219 - loss: 0.6312853693962097
Epoch 220 - loss: 53.984745025634766
```

```
Epoch 221 - loss: 0.6553191542625427
Epoch 222 - loss: 53.765296936035156
Epoch 223 - loss: 0.6796887516975403
Epoch 224 - loss: 53.54730224609375
Epoch 225 - loss: 0.7043895125389099
Epoch 226 - loss: 53.33073806762695
Epoch 227 - loss: 0.7294138669967651
Epoch 228 - loss: 53.1156005859375
Epoch 229 - loss: 0.7547574043273926
Epoch 230 - loss: 52.90187454223633
Epoch 231 - loss: 0.7804157137870789
Epoch 232 - loss: 52.68955993652344
Epoch 233 - loss: 0.8063812851905823
Epoch 234 - loss: 52.47864532470703
Epoch 235 - loss: 0.8326517343521118
Epoch 236 - loss: 52.269100189208984
Epoch 237 - loss: 0.8592230081558228
Epoch 238 - loss: 52.06093215942383
Epoch 239 - loss: 0.8860823512077332
Epoch 240 - loss: 51.8541374206543
Epoch 241 - loss: 0.9132347702980042
Epoch 242 - loss: 51.6486701965332
Epoch 243 - loss: 0.9406692385673523
Epoch 244 - loss: 51.44455337524414
Epoch 245 - loss: 0.9683784246444702
Epoch 246 - loss: 51.241783142089844
Epoch 247 - loss: 0.9963603019714355
Epoch 248 - loss: 51.040348052978516
Epoch 249 - loss: 1.0246132612228394
Epoch 250 - loss: 50.840206146240234
Epoch 251 - loss: 1.0531281232833862
Epoch 252 - loss: 50.641380310058594
Epoch 253 - loss: 1.0819014310836792
Epoch 254 - loss: 50.44384002685547
Epoch 255 - loss: 1.1109298467636108
Epoch 256 - loss: 50.247589111328125
Epoch 257 - loss: 1.1402060985565186
Epoch 258 - loss: 50.0526123046875
Epoch 259 - loss: 1.1697227954864502
Epoch 260 - loss: 49.858917236328125
Epoch 261 - loss: 1.199483036994934
Epoch 262 - loss: 49.666473388671875
Epoch 263 - loss: 1.2294775247573853
Epoch 264 - loss: 49.47526931762695
Epoch 265 - loss: 1.2597053050994873
Epoch 266 - loss: 49.285308837890625
Epoch 267 - loss: 1.2901616096496582
Epoch 268 - loss: 49.09656524658203
```

```
Epoch 269 - loss: 1.3208370208740234
Epoch 270 - loss: 48.909053802490234
Epoch 271 - loss: 1.3517264127731323
Epoch 272 - loss: 48.72277069091797
Epoch 273 - loss: 1.382836103439331
Epoch 274 - loss: 48.53766632080078
Epoch 275 - loss: 1.4141522645950317
Epoch 276 - loss: 48.35376739501953
Epoch 277 - loss: 1.4456698894500732
Epoch 278 - loss: 48.17106628417969
Epoch 279 - loss: 1.477393627166748
Epoch 280 - loss: 47.989524841308594
Epoch 281 - loss: 1.509313941001892
Epoch 282 - loss: 47.80915832519531
Epoch 283 - loss: 1.541426181793213
Epoch 284 - loss: 47.629966735839844
Epoch 285 - loss: 1.5737258195877075
Epoch 286 - loss: 47.451927185058594
Epoch 287 - loss: 1.6062079668045044
Epoch 288 - loss: 47.2750358581543
Epoch 289 - loss: 1.6388754844665527
Epoch 290 - loss: 47.09926223754883
Epoch 291 - loss: 1.671721339225769
Epoch 292 - loss: 46.92462158203125
Epoch 293 - loss: 1.7047362327575684
Epoch 294 - loss: 46.75111389160156
Epoch 295 - loss: 1.73792564868927
Epoch 296 - loss: 46.57870864868164
Epoch 297 - loss: 1.7712750434875488
Epoch 298 - loss: 46.407413482666016
Epoch 299 - loss: 1.804795265197754
when x = tensor([1., 3.]), y = tensor([3.1382], grad_fn=<SqueezeBackward4>)
when x = tensor([2., 6.]), y = tensor([6.2764], grad_fn=<SqueezeBackward4>)
when x = tensor([3., 9.]), y = tensor([9.4146], grad_fn=<SqueezeBackward4>)
```

[2]:
```
[<matplotlib.lines.Line2D at 0x7fc41ac3e150>]
```



Observation:

In batch training, the loss decreases smoothly and then stabilizes around the value 5. This is because batch training computes the gradient using all data points before each update, giving a more stable descent direction. On the other hand, SGD training shows large fluctuations throughout the training while still having a descrease trend. This is because SGD training updates gradient after each data point, causing instability in descent direction and resulting in a noisier training curve.