Untitled

October 9, 2021

[4]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

import torch.optim as optim

#Training data
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)]

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

net = Net()

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

#Loss definition
criterion = nn.MSELoss()

#Optimizer definition
optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.5)

for epoch in range(30):
    for i, current_data in enumerate(data):
        X = current_data[:2]
        Y = current_data[2]
        X, Y = torch.FloatTensor([X]), torch.FloatTensor([Y])
        optimizer.zero_grad()
        outputs = net(X)
        loss = criterion(outputs, Y)
        loss.backward()
        optimizer.step()
        if (i % 10 == 0):
            print("Epoch {} - loss: {}".format(epoch, loss))

### Test the trained network ###
for i, current_data in enumerate(data):
    X, Y = current_data[:2], current_data[2]
    X, Y = torch.FloatTensor([X]), torch.FloatTensor([Y])
    out = net(torch.FloatTensor(X))
    print("when x = {}, y = {}".format(X, out))
```

```
Net(
  (fc1): Linear(in_features=2, out_features=1, bias=False)
)
[Parameter containing:
tensor([[ 0.1475, -0.2757]], requires_grad=True)]
Epoch 0 - loss: 11.775388717651367
Epoch 1 - loss: 1.4637712240219116
Epoch 2 - loss: 1.129019021987915
Epoch 3 - loss: 0.6518985629081726
Epoch 4 - loss: 0.27345195412635803
Epoch 5 - loss: 0.07992889732122421
Epoch 6 - loss: 0.012686351314187050
Epoch 7 - loss: 0.00010530889994697645
Epoch 8 - loss: 0.0015593082644045353
Epoch 9 - loss: 0.0029694295953959227
Epoch 10 - loss: 0.0025235824286937714
Epoch 11 - loss: 0.0014357934705913067
Epoch 12 - loss: 0.0005921877454966307
Epoch 13 - loss: 0.000169142906088382
Epoch 14 - loss: 2.552602018113248e-05
Epoch 15 - loss: 9.562251079842099e-08
Epoch 16 - loss: 3.845492756227031e-06
Epoch 17 - loss: 6.828122423030436e-06
Epoch 18 - loss: 5.653687821904896e-06
Epoch 19 - loss: 3.1617312288290123e-06
Epoch 20 - loss: 1.2825296380469808e-06
Epoch 21 - loss: 3.558400294423336e-07
Epoch 22 - loss: 5.119323986946256e-08
Epoch 23 - loss: 4.4565240386873484e-11
Epoch 24 - loss: 9.416055490873987e-09
Epoch 25 - loss: 1.548892214486841e-08
Epoch 26 - loss: 1.2343889466137625e-08
Epoch 27 - loss: 7.003166047070408e-09
Epoch 28 - loss: 2.7762894205807243e-09
Epoch 29 - loss: 7.914877642178908e-10
when x = tensor([[1.0000, 2.1000]]), y = tensor([[3.0000]],
grad_fn=<MmBackward>)
when x = tensor([[2.0000, 3.5000]]), y = tensor([[6.0000]],
grad_fn=<MmBackward>)
when x = tensor([[3., 3.]]), y = tensor([[9.]], grad_fn=<MmBackward>)
when x = tensor([[4.0000, 2.1000]]), y = tensor([[12.0000]],
grad_fn=<MmBackward>)
when x = tensor([[5.0000, 7.2000]]), y = tensor([[15.0000]],
grad_fn=<MmBackward>)
when x = tensor([[ 6.0000, 10.1000]]), y = tensor([[18.0000]],
grad_fn=<MmBackward>)
```

[ ]: