Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

(http://cocl.us/pytorch_link_top)

Neural Networks More Hidden Neutrons

In [ ]:

```
#by Christopher Harrison
```

Objective

- How to create complex Neural Network in pytorch.

Table of Contents

Estimated Time Needed: **25 min**

Preparation

We'll need to import the following libraries for this lab.

In [1]:

```
import torch
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader
```

Define the plotting functions.

In [2]:

```
def get_hist(model,data_set):
    activations=model.activation(data_set.x)
    for i,activation in enumerate(activations):
        plt.hist(activation.numpy(),4,density=True)
        plt.title("Activation layer " + str(i+1))
        plt.xlabel("Activation")
        plt.xlabel("Activation")
        plt.legend()
        plt.show()
```

In [3]:

```
def PlotStuff(X,Y,model=None,leg=False):

    plt.plot(X[Y==0].numpy(),Y[Y==0].numpy(),'or',label='training points y=0 ' )
    plt.plot(X[Y==1].numpy(),Y[Y==1].numpy(),'ob',label='training points y=1 ' )

    if model!=None:
        plt.plot(X.numpy(),model(X).detach().numpy(),label='neral network ')

    plt.legend()
    plt.show()
```

Get Our Data

Define the class to get our dataset.

In [4]:

```
class Data(Dataset):
    def __init__(self):
        self.x=torch.linspace(-20, 20, 100).view(-1,1)

        self.y=torch.zeros(self.x.shape[0])
        self.y[(self.x[:,0]>-10)& (self.x[:,0]<-5)]=1
        self.y[(self.x[:,0]>5)& (self.x[:,0]<10)]=1
        self.y=self.y.view(-1,1)
        self.len=self.x.shape[0]
    def __getitem__(self,index):

        return self.x[index],self.y[index]
    def __len__(self):
        return self.len
```

Define the Neural Network, Optimizer and Train the Model

Define the class for creating our model.

In [5]:

```
class Net(nn.Module):
    def __init__(self,D_in,H,D_out):
        super(Net,self).__init__()
        self.linear1=nn.Linear(D_in,H)
        self.linear2=nn.Linear(H,D_out)


    def forward(self,x):
        x=torch.sigmoid(self.linear1(x))
        x=torch.sigmoid(self.linear2(x))
        return x
```

Create the function to train our model, which accumulate lost for each iteration to obtain the cost.

In [6]:

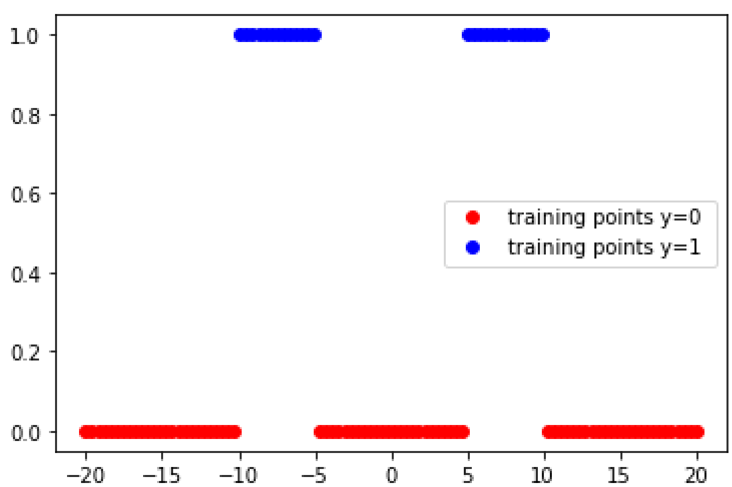
```
def train(data_set,model,criterion, train_loader, optimizer, epochs=5,plot_number=
10):
    cost=[]

    for epoch in range(epochs):
        total=0

        for x,y in train_loader:
            optimizer.zero_grad()

            yhat=model(x)
            loss=criterion(yhat,y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            total+=loss.item()

        if epoch%plot_number==0:
            PlotStuff(data_set.x,data_set.y,model)

        cost.append(total)
    plt.figure()
    plt.plot(cost)
    plt.xlabel('epoch')
    plt.ylabel('cost')
    plt.show()
    return cost
```

In [7]:

```
data_set=Data()
```

In [8]:

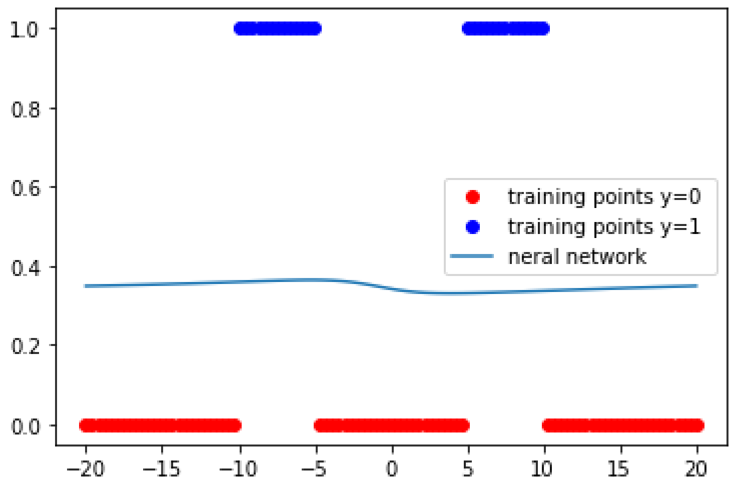
```
PlotStuff(data_set.x,data_set.y,leg=False)
```

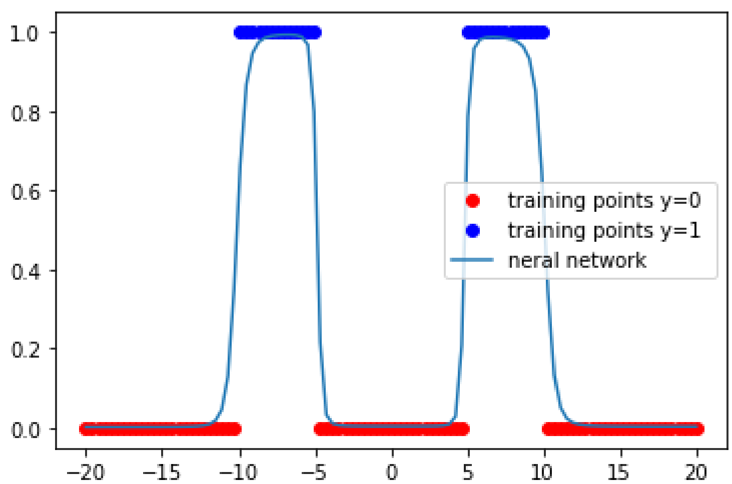


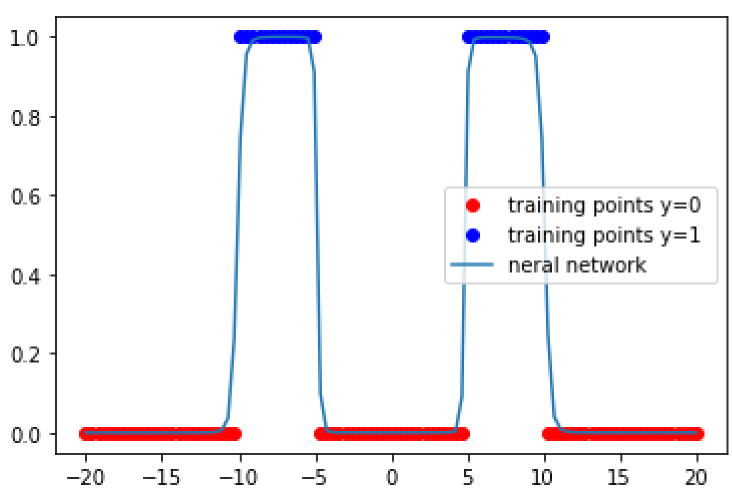
Create our model with 9 neurons in the hidden layer. And then create a BCE loss and an Adam optimizer.

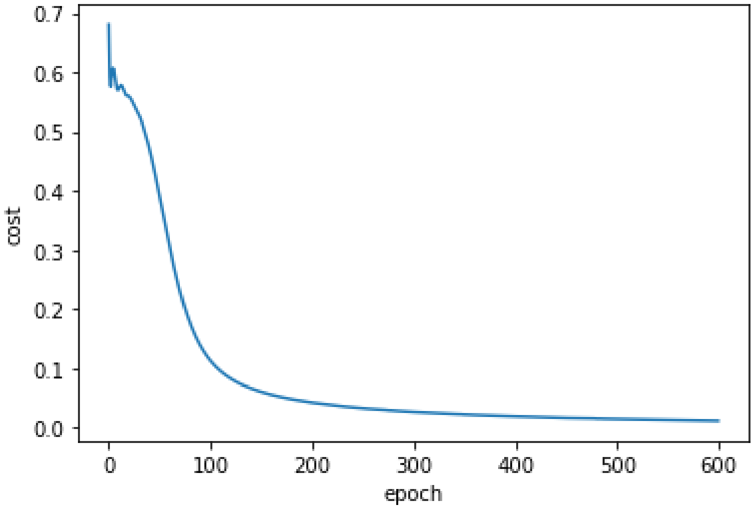
In [9]:

```
torch.manual_seed(0)
model=Net(1,9,1)
learning_rate=0.1
criterion=nn.BCELoss()
optimizer=torch.optim.Adam(model.parameters(), lr=learning_rate)
train_loader=DataLoader(dataset=data_set,batch_size=100)
COST=train(data_set,model,criterion, train_loader, optimizer, epochs=600,plot_numbe
r=200)
```


training points y=0
training points y=1
neral network


training points y=0
training points y=1
neral network


training points y=0
training points y=1
neral network



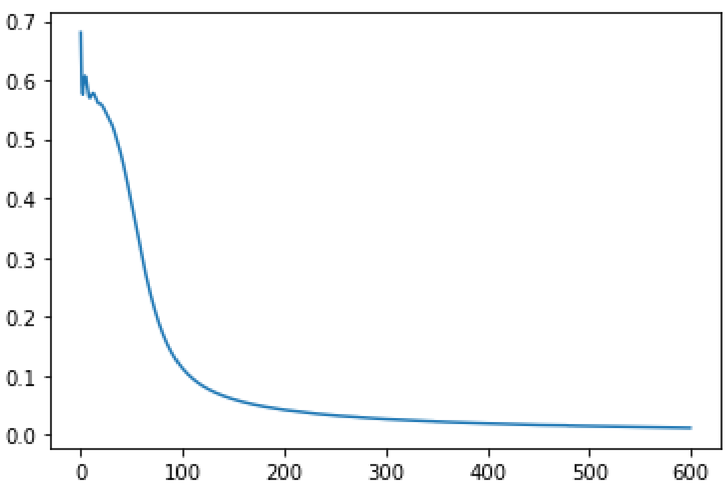
this is for exercises model= torch.nn.Sequential(torch.nn.Linear(1, 6), torch.nn.Sigmoid(), torch.nn.Linear(6,1), torch.nn.Sigmoid())

In [10]:

```
plt.plot(COST)
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7f01104b4f28>]
```



About the Authors:

Joseph Santarcangelo (https://www.linkedin.com/in/joseph-s-50398b136/) has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Michelle Carey (https://www.linkedin.com/in/michelleccarey/), Mavis Zhou (www.linkedin.com/in/jiahui-mavis-zhou-a4537814a), Fan Jiang (https://www.linkedin.com/in/fanjiang0619/), Yi Leng Yao (https://www.linkedin.com/in/yi-leng-yao-84451275/), Sacchit Chadha (https://www.linkedin.com/in/sacchitchadha/)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-23	2.0	Shubham	Migrated Lab to Markdown and added to course repo in GitLab