In [1]:

```
#how to use google colab: https://pytorch.org/tutorials/beginner/co
#pytorch tutorial: https://pytorch.org/tutorials/beginner/deep_lear
import torch
import torch.nn.functional as F
from torchvision import datasets, transforms
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
# define the network structure
class fc_net(torch.nn.Module):
 def __init__(self, num_in, num_out):
  super(fc_net, self).__init__()
  # Initialize two linear neural networks, one as the input layer a
  self.h1 = torch.nn.Linear(in_features=num_in, out_features=256)
  self.h2 = torch.nn.Linear(in_features=256, out_features=num_out)
 def forward(self, inputs):
  # We use relu as the activation function as the first layer
  a1 = F.relu(self.h1(inputs))
  # We use softmax as the activation function as the second layer
  a2 = F.softmax(self.h2(a1),dim=-1)
  return a2
```

In [3]:

```
# use data_loader to load_in data
train_data = datasets.MNIST('./', train=True, download=True, transf
# We use DataLoader to load the train data.
# We specify the batch_size and the DataLoader will return the spli
# We use shuffle = False so we won't shuffle the data.
train_loader = torch.utils.data.DataLoader(train_data, batch_size=1
batch_size = 10
# Model is a fully-connected net.
model = fc_net(num_in=28*28, num_out=10)
# We use Cross Entropy as our loss.
loss = torch.nn.CrossEntropyLoss()
# We use SGD as our optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
```

In [4]:

```
epoch = 10
epoch_accuracies = []
for j in range (epoch):
    loader = iter(train_loader)
    epoch_accuracy = 0.0
    print (j)
    for i in range (1, len(train_loader)):
        # cur_s, cur_y gets the data and label for the next iterati
        cur_x, cur_y = next(loader)
        # We unsqueeze the data for one batch.
        cur_x = torch.reshape(cur_x, (10, 28*28))
        # We get the predictions, which are probabilities, after th
        preds = model.forward(cur_x)
        # Use the predictions and labels to compute loss.
        cur_loss = loss(preds, cur_y)
        optimizer.zero_grad()
        # Backward
        cur_loss.backward()
        optimizer.step()
        preds_numpy = preds.detach().numpy()
        preds_label = np.argmax(preds_numpy, axis=1)
        cur_y_numpy = cur_y.detach().numpy()
        acc_iter = np.sum(1*(preds_label)==(cur_y_numpy))/batch_siz
        epoch_accuracy += acc_iter
    epoch_accuracy =  epoch_accuracy/len(train_loader)
    epoch_accuracies.append(epoch_accuracy)
    print (epoch_accuracy)
#         new_preds = model.forward(cur_x)
#         print(new_preds[0])
```

```
0
0.8604666666666702
1
0.9342666666666593
2
0.9492333333333249
3
0.9586333333333246
4
0.965449999999991
5
0.9702833333333245
6
0.9743333333333243
7
0.9771666666666581
8
0.9798166666666586
9
[5 0 4 1 9 2 1 3 1 4]
```

In [ ]:

In [ ]:

In [ ]: