ENEE 6582 – Neural Nets
Project 3 – Convolutional Neural Nets

Week 1: Due Mon 3/26

We will be experimenting with code on the following github:
Original Python 2.7 code:
https://github.com/mnielsen/neural-networks-and-deep-learning/tree/master/src
Modified to Python 3.5.2 :
https://github.com/MichalDanielDobrzanski/DeepLearningPython35
MNIST Data:
https://github.com/mnielsen/neural-networks-and-deep-learning/tree/master/data

1. You will need to install the Theano Library:
   http://deeplearning.net/software/theano/
2. The CNN training is intensive and time consuming and can be GREATLY (1000x) accelerated using GPUs. For details on GPU acceleration see:
   http://deeplearning.net/software/theano/tutorial/using_gpu.html
3. Alternatively, you may create a free AWS amazon account and get $70+ worth of credit to run your code on an EC2 G2 instance:
   https://aws.amazon.com/ec2/instance-types/
4. Import the MNIST data and CNN library:
   ```
   Import the CNN code:
   import network3
   from network3 import Network
   from network3 import ConvPoolLayer, FullyConnectedLayer, SoftmaxLayer
   training_data, validation_data, test_data = network3.load_data_shared()
   ```
5. Create a fully connected nets with 784 inputs, 100 outputs, 10 softmax layer, and a mini-batch of 10:
   ```
   mini_batch_size = 10
   net = Network([
       FullyConnectedLayer(n_in=784, n_out=100),
       SoftmaxLayer(n_in=100, n_out=10)], mini_batch_size)
   ```
6. Train using SGD for 60+ epoch, using a learning rate = 0.1:
   ```
   net.SGD(training_data, 60, mini_batch_size, 0.1,
           validation_data, test_data)
   ```
   Note the testing accuracy. This will be a **baseline** accuracy to compare the CNN against.
7. Create a CNN that:
   takes in the 28x28 MNIST images,
   uses 20 convolutional layers
   each layer is created by 5x5 shared filter
   with a stride of 1.
   Applies a 2x2 max pooling.
   Pooling layers is fully connected to 100 sigmoid neurons,
   followed by 10 softmax neurons.

   ```
   net = Network([
        ConvPoolLayer(image_shape=(mini_batch_size, 1, 28, 28),
                      filter_shape=(20, 1, 5, 5),
                      poolsize=(2, 2)),
        FullyConnectedLayer(n_in=20*12*12, n_out=100),
   ```

```
     SoftmaxLayer(n_in=100, n_out=10)], mini_batch_size)
```

8. Train using SGD for 60+ epoch, using a learning rate = 0.1. Note the performance accuracy and the number of epoch to reach the baseline accuracy.
9. Modify 7 by adding convolutional layers following the pooling with the following specs:

   40 convolutional layers
   5x5 filters
   Stride of 20
   Pool 2x2

```
 net = Network([
     ConvPoolLayer(image_shape=(mini_batch_size, 1, 28, 28),
                   filter_shape=(20, 1, 5, 5),
                   poolsize=(2, 2)),
     ConvPoolLayer(image_shape=(mini_batch_size, 20, 12, 12),
                   filter_shape=(40, 20, 5, 5),
                   poolsize=(2, 2)),
     FullyConnectedLayer(n_in=40*4*4, n_out=100),
     SoftmaxLayer(n_in=100, n_out=10)], mini_batch_size)
```

Retrain using the specs in 6 (or 8). Note accuracy the number of epochs it takes to reach the baseline accuracy.

10. Modify the network in 9 to use RELU activation, use L2 regularization $\lambda = 0.1$, learning rate of 0.03. Keep the softmax output.

```
net = Network([
     ConvPoolLayer(image_shape=(mini_batch_size, 1, 28, 28),
                   filter_shape=(20, 1, 5, 5),
                   poolsize=(2, 2),
                   activation_fn=ReLU),
     ConvPoolLayer(image_shape=(mini_batch_size, 20, 12, 12),
                   filter_shape=(40, 20, 5, 5),
                   poolsize=(2, 2),
                   activation_fn=ReLU),
     FullyConnectedLayer(n_in=40*4*4, n_out=100, activation_fn=ReLU),
     SoftmaxLayer(n_in=100, n_out=10)], mini_batch_size)
```

Retrain the network using SGD:

```
net.SGD(training_data, 60, mini_batch_size, 0.03,
         validation_data, test_data, lmbda=0.1)
```

Note the improvement in accuracy.

11. Modify the network in 10 by using two 1000-neuron hidden layers before softmax. Retrain the network and note the improvement in accuracy and how many epochs does it take to reach the baseline accuracy.
12. Modify the network in 11 so that it implements 50% dropout and retrain.

```
net = Network([
     ConvPoolLayer(image_shape=(mini_batch_size, 1, 28, 28),
                   filter_shape=(20, 1, 5, 5),
                   poolsize=(2, 2),
                   activation_fn=ReLU),
     ConvPoolLayer(image_shape=(mini_batch_size, 20, 12, 12),
                   filter_shape=(40, 20, 5, 5),
                   poolsize=(2, 2),
                   activation_fn=ReLU),
     FullyConnectedLayer(
         n_in=40*4*4, n_out=1000, activation_fn=ReLU, p_dropout=0.5),
     FullyConnectedLayer(
         n_in=1000, n_out=1000, activation_fn=ReLU, p_dropout=0.5),
     SoftmaxLayer(n_in=1000, n_out=10, p_dropout=0.5)],
     mini_batch_size)
```

Retrain the network and note the improvement in accuracy and how many epochs does it take to reach the baseline accuracy.

Week 2: Due Sun 4/1

Implement a CNN that can achieve 80+% accuracy on the notMNIST data. For ideas on how to get there checkout:
https://www.kaggle.com/sharmila5656/a-starter-lenet5-dropout-data-augmentati-8907d8
https://www.kaggle.com/jwjohnson314/a-starter-lenet5-dropout-data-augmentation