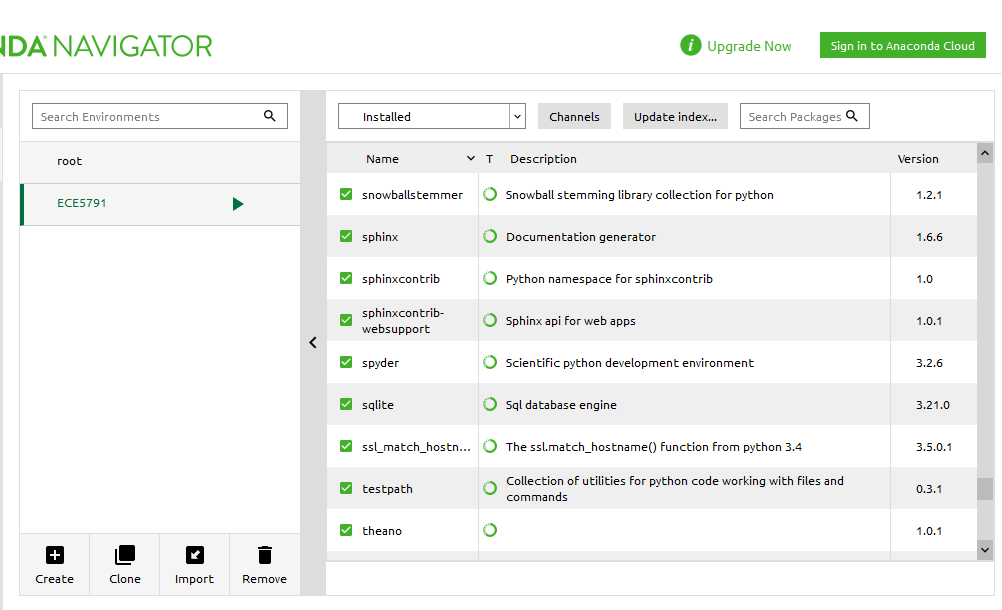
**ECE579 Assignment 1**

Xin Xu

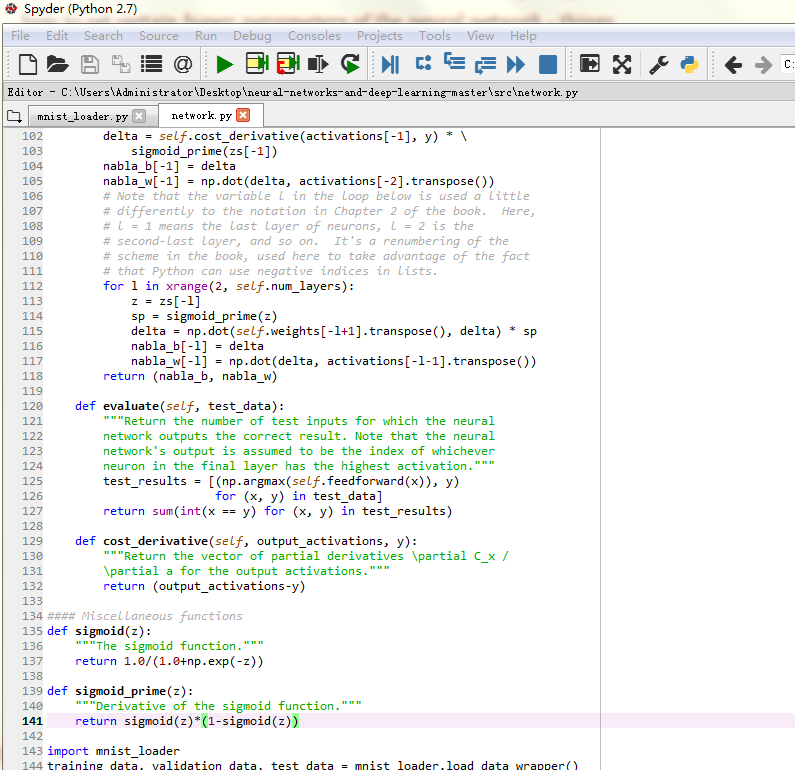
1. **Setting environment:**

Firstly, I need to install Numpy and Theano, I choose to use Anaconda to get all the packages that I need. In Anaconda Navigator, I created a environment - ’ECE5791’ and then apply Numpy and Theano into the environment. Then every application in Anaconda will run in ECE5791.

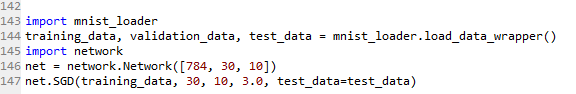


1. **Running codes:**

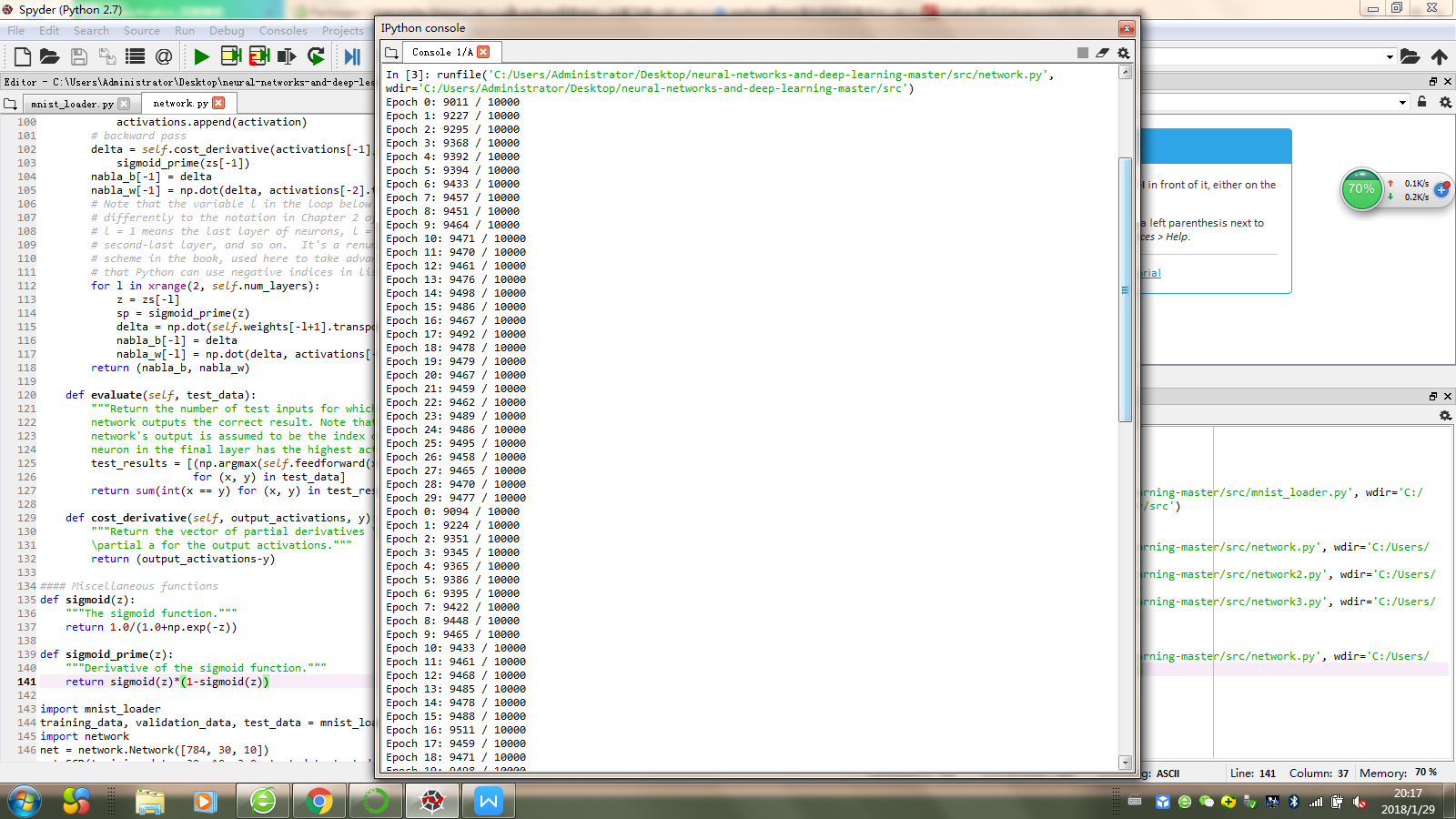
In order to run the codes, I need to install and launch an application - ‘spyder’ in Anaconda. It is a powerful Python IDE. Then open the ‘network.py’ first.



Add those codes as what the textbook says and then run the codes:



Part of the output is:

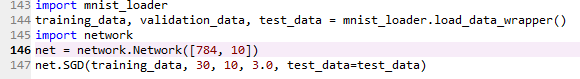


The trained network gives me a classification rate of about 95 percent, its peak is 95.11 percent at Epoch 16. It is different with the textbook. That’s because we initialize our network using different random weights and biases.

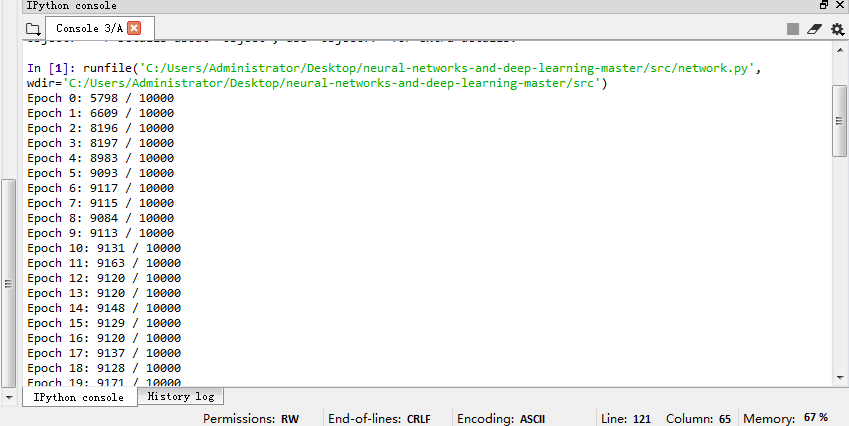
1. **Exercise:**

* ***Try creating a network with just two layers - an input and an output layer, no hidden layer - with 784 and 10 neurons, respectively. Train the network using stochastic gradient descent. What classification accuracy can you achieve?***

Changing code to:



The output is:



To be more specific:

Epoch 0: 5653 / 10000 Epoch 1: 5685 / 10000 Epoch 2: 5712 / 10000

Epoch 3: 5751 / 10000 Epoch 4: 5738 / 10000 Epoch 5: 5748 / 10000

Epoch 6: 5756 / 10000 Epoch 7: 6387 / 10000 Epoch 8: 6571 / 10000

Epoch 9: 6624 / 10000 Epoch 10: 6643 / 10000 Epoch 11: 6657 / 10000

Epoch 12: 6618 / 10000 Epoch 13: 6687 / 10000 Epoch 14: 7400 / 10000

Epoch 15: 7482 / 10000 Epoch 16: 7526 / 10000 Epoch 17: 7526 / 10000

Epoch 18: 7520 / 10000 Epoch 19: 7535 / 10000 Epoch 20: 7520 / 10000

Epoch 21: 7541 / 10000 Epoch 22: 7527 / 10000 Epoch 23: 7527 / 10000

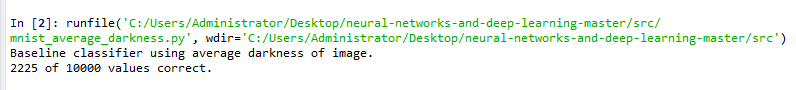
Epoch 24: 7531 / 10000 Epoch 25: 7530 / 10000 Epoch 26: 7535 / 10000

Epoch 27: 7534 / 10000 Epoch 28: 7539 / 10000 Epoch 29: 7543 / 10000

So its peak classification rate is 75.41% at Epoch 21.

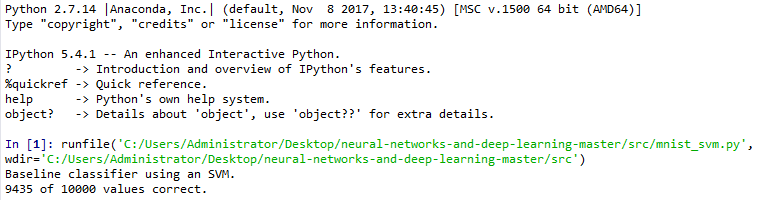
1. **Comparison with other algorithms**

At first, computing average darkness for each digit, the output is:



The classification rate is 22.25%.

Then, the SVMs, I need to add scikit-learn into environment first, and then run the code. The output is:



The classification rate is 94.35%. It's a big improvement over naive approach of classifying an image based on how dark it is.