CMSC 254 PSET 6

Xander Beberman

3/9/17

The network I made is in python 2.7. After loading the data, adding the bias neuron, and normalizing the data, the primary functions that comprise the network itself are

1. `init`: Given an input size, list of hidden layer sizes, and output size, creates a list of arrays that correspond to neuron weights
2. `think`: Given a list of weights and an input vector, computes the output of the network
3. `train`: Given the training (both normal set and holdout set) vectors and answers, list of weights, learning rate, and max number of epochs, train the network and output the final weights and accuracy of vector at each iteration

One nice feature about storing weights this way is that we can use `np.save` and `np.load` to store fully or partially trained weights. All of the training and setup is done with matrix operations. This implementation is generalized to have any number of hidden layers of any sizes, although I did not have time to study how different architectures affected accuracy.

After some very quick initial testing, a learning rate of 0.2 seemed to give decent results. Testing different layer sizes for $\ell \in \{28, 32, 64, 128, 256\}$ with learning rate 0.2 over a maximum of 200 iterations using 95% of the training data and the remaining 5% as a holdout set gave the results in figure 1. This test showed that a layer size of 32 worked the best. Then, using this as the hidden layer size, I tested the learning rates $\eta \in \{0.01, 0.1, 1, 10\}$ over a maximum of 200 iterations. The results are shown in figure 2. $\eta = 1$ slightly beat out $\eta = 0.1$, but after slightly more testing, it seems that this is because of $\eta = 1$ learns faster initially than $\eta = 0.1$. After about 200 epochs, $\eta = 1$ iterates too aggressively to increase in accuracy, and 0.1 works better. After about 250 epochs, even $\eta = 0.01$ does not help with layer size 32. Learning rate 0.25 was used to test the accuracy over a number of epochs with hidden layer size 32. A plot of this is shown in figure 3. I saved the weights of this testing and loaded them to then test $\eta = 0.01$ over another 100 epochs. A plot of this is given in figure 4 with an incorrect title.

I also tried to create a surface plot testing different learning rates over different layer sizes, which is shown in figure 5, but the method proved to be too time-intensive and not particularly helpful.

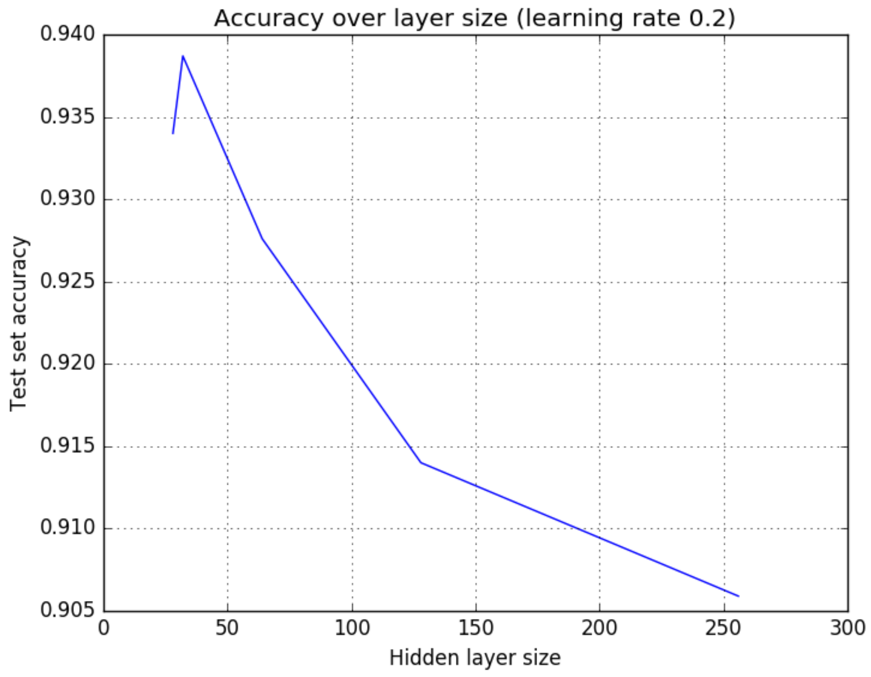

Figure 1: Accuracy for different layer sizes, $\eta = 0.2$, 200 epochs

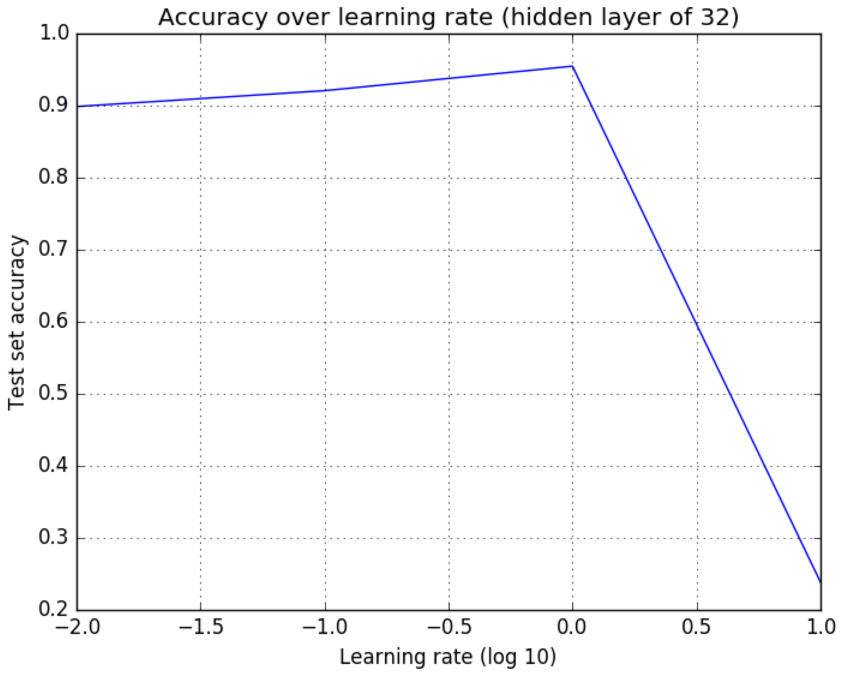

Figure 2: Accuracy for different learning rates, hidden layer size 32, 200 epochs

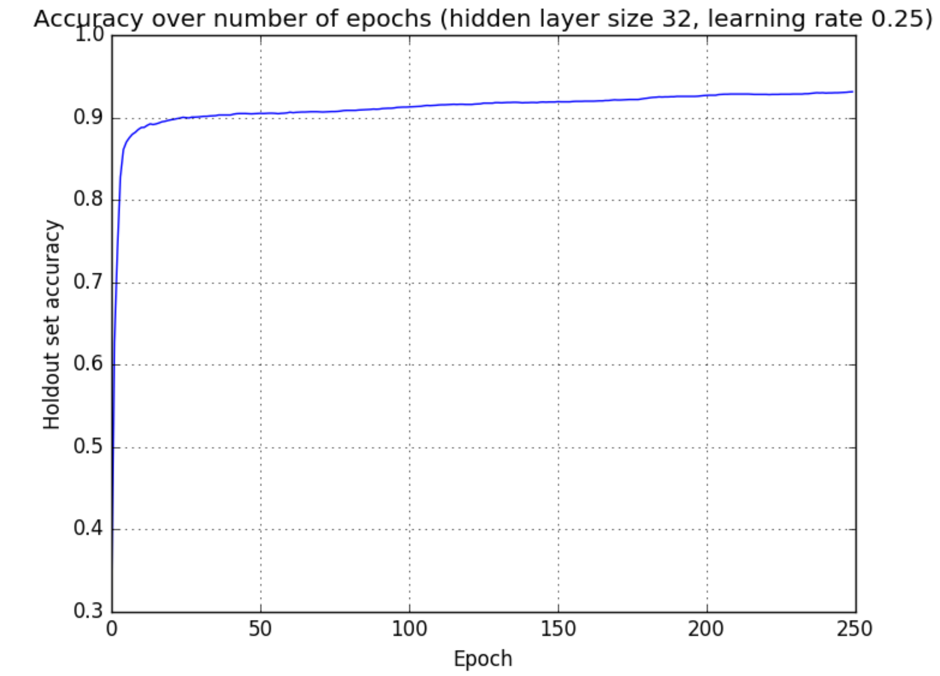

Figure 3: Accuracy over number of epochs, hidden layer size 32, $\eta = 0.25$

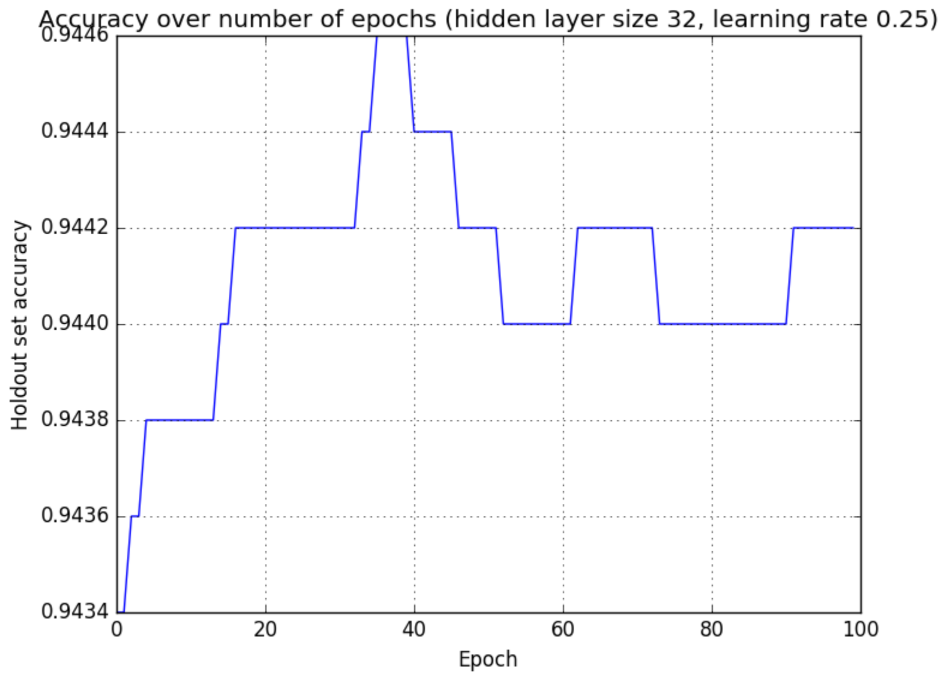

Figure 4: Accuracy over number of epochs after 250 epochs training with $\eta = 0.01$, hidden layer size 32, $\eta = 0.25$

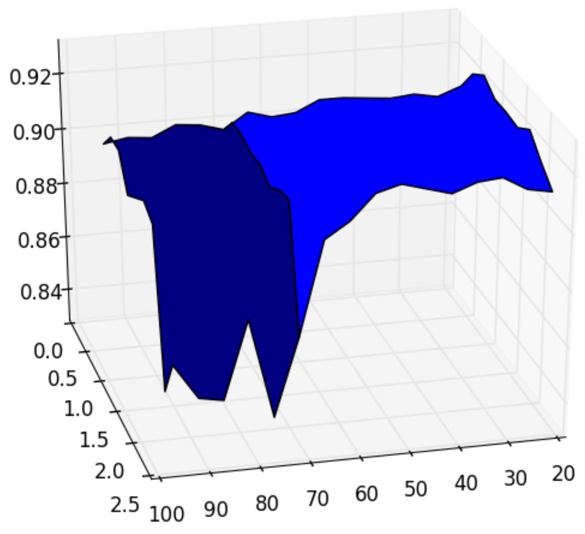

Figure 5: Accuracy over different hidden layer sizes, learning rates