Machine Learning	**Homework 3**
CSCI 5622 Fall 2017	**Due Time Oct 13, 2017**
Name: Luke Meszar	**CU identitykey: lume0018**

1 Back Propagation (35 Points)

Solution.

1. *What is the structure of your neural network (for both tinyTOY and tinyMNIST dataset)? Show the dimensions of the input layer, hidden layer and output layer.*

   In both cases, there are three layers: an input layer, a hidden layer, and an output layer.

   For the tinyTOY data, the size of the input layer is 2, the size of the hidden layer is 30, and the size of the output layer is 2.

   For the tinyMNIST data, the size of the input layer is 196, the size of the hidden layer is 64, and the size of the output layer is 10.

   In both cases, the size of the input layer is as reflection of the form of the data; the size of the output layer is the number of categories in the classification; and the size of the hidden layer is a variable that can be changed to affect training accuracy.

   In both

2. *What the role of the size of the hidden layer on train and test accuracy (plot accuracy vs. size of hidden layer using tinyMNIST dataset)?*

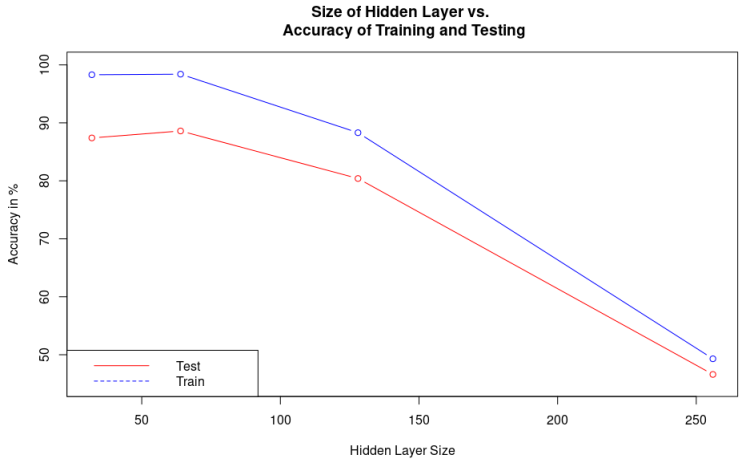

Figure 1: Size of Hidden Layer vs. Accuracy Trained with 195 Epochs

From the graph, we can see that accuracy improves slightly when the size of the hidden layer goes from 32 to 64. However, from then on, increasing the hidden layer size dramatically decreases the accuracy with the test accuracy going to 49.3% with a hidden layer size of 256.

3. *How does the number of epochs affect train and test accuracy (plot accuracy vs. epochs using tinyMINST dataset)?*

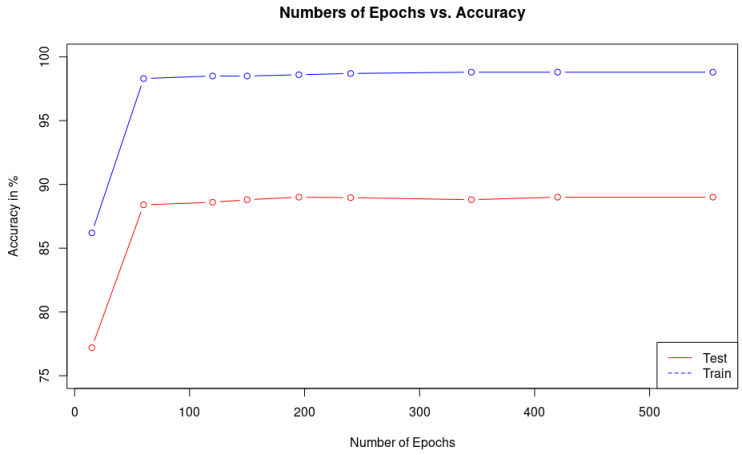

Figure 2: Number of Epochs vs. Accuracy with Hidden Layer Size of 64

We can see that there is a large increase in accuracy between 15 epochs and 60 epochs. After, the accuracy does not improve very much as the number of epochs increases. The maximum accuracy on the test data was after 345 epochs. Using more epochs than this didn't really change the accuracy for better or worse.

2 Keras CNN (35pts)

Solution.

1. *Point out at least three layer types you used in your model. Explain what are they used for.*

   Some of the layers I used in this model were a 2-dimensional convolution layer, a max pooling layer, a dropout layer, and a fully connected classification layer. The convolution layer applies a number of filters to the input image. Each filter can be thought of as a small lens which is slid over the image. Its use is for detecting features in an image. The neural network learns which filters to use to detect the features it needs for classification.

   The max pooling layer finds the maximum elements in a local neighborhood after applying the convolution layer. It can in some sense be thought of as finding the most important point in a local neighborhood. It also serves to reduce the dimensionality of the parameter space.

   The dropout layer chooses some percentage of the neurons to deactivate on each training instance. This helps to reduce overfitting since it forces the network to use all of the neuron more equally.

Finally, the fully connected (or dense) layer is the layer that actually does the classification. Every neuron in the previous layer is connected to every node in this layer. This layer uses softmax activation with appropriate weights and biases to classify the image.

2. *How did you improve your model for higher accuracy?*

   I improved my model in the following ways. I used one-hot encoding, increased the size of the filter in the convolution layer to 6×6 and added a fully connected layer between the dropout layer and the final dense classification layer. This intermediate fully connected layer was more successful when it was 256 neurons wide than when it was 128. I also determined that the adam optimizer performed the best of the ones I tried.

3. *Try different activation functions and batch sizes. Show the corresponding accuracy.*

   I changed the activation functions in the convulutional layer and in the intermediate dense layer. Here are the results.

Activation	Accuracy
ReLU	99.06%
eLU	98.97%
tanh	98.56%
selu	98.40%
sigmoid	98.35%
linear	97.53%

   The only significant variation between the activation function was that linear activation performed the worst which is not surprising. It is clear that ReLU performed the best.

   When I varied the batch size, I uses the ReLU activation function. The results for different batch sizes is as follows:

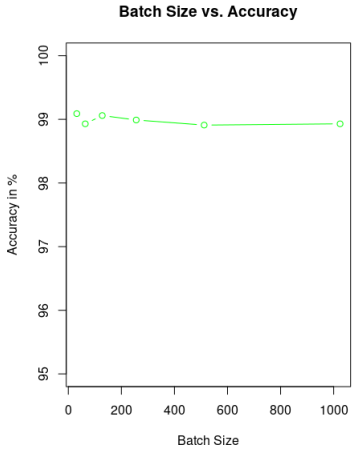

Figure 3:

Changing the batch size didn't have much affect on the accuracy though a batch size of 32 was the most accurate.

3 Keras RNN (30pts)

Solution.

1. *What is the purpose of the embedding layer? (Hint: think about the input and the output).*

   The raw data in the IMDB data set is an array of integers where each number refers to a word in the data set. This data lives in a very large vector space where each word is its own dimension. This is not particular useful since words do not occur in isolation. The embedding layer's job is to do word embedding. The idea is to map the words into a lower dimensional vector space where words that are contextual similar are mapped close to each other. This creates more meaningful features.

2. *What is the effect of the hidden dimension size in LSTM?*

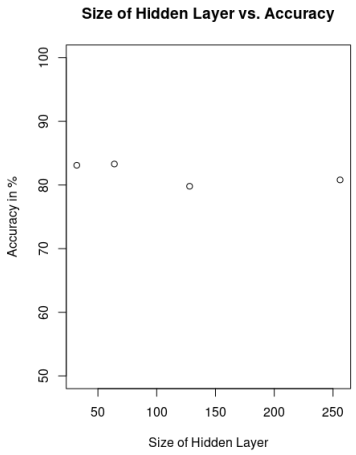

Figure 4: Hidden Layer size with sizes of 32,64,128, and 256 vs. Accuracy

   I collected data for this only using 5000 samples. The results show that changing the hidden layer size didn't significantly affect the accuracy. However, a hidden layer size of 64 was the most accurate.

3. *Replace LSTM with GRU and compare their performance.*

   With the model I have, changing LSTM to GRU did not significantly affect the performance. Both performed around in the range of 87% to 88%.