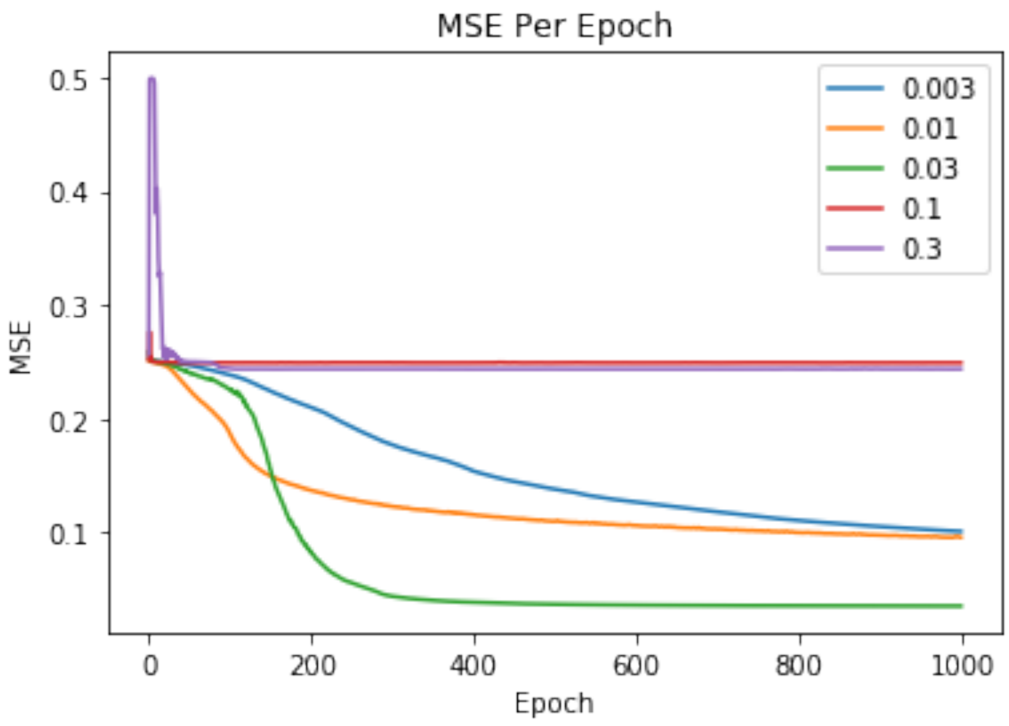
Math 189Z Homework 4: Automatic Differentiation Software + RNN's

Daphne Poon

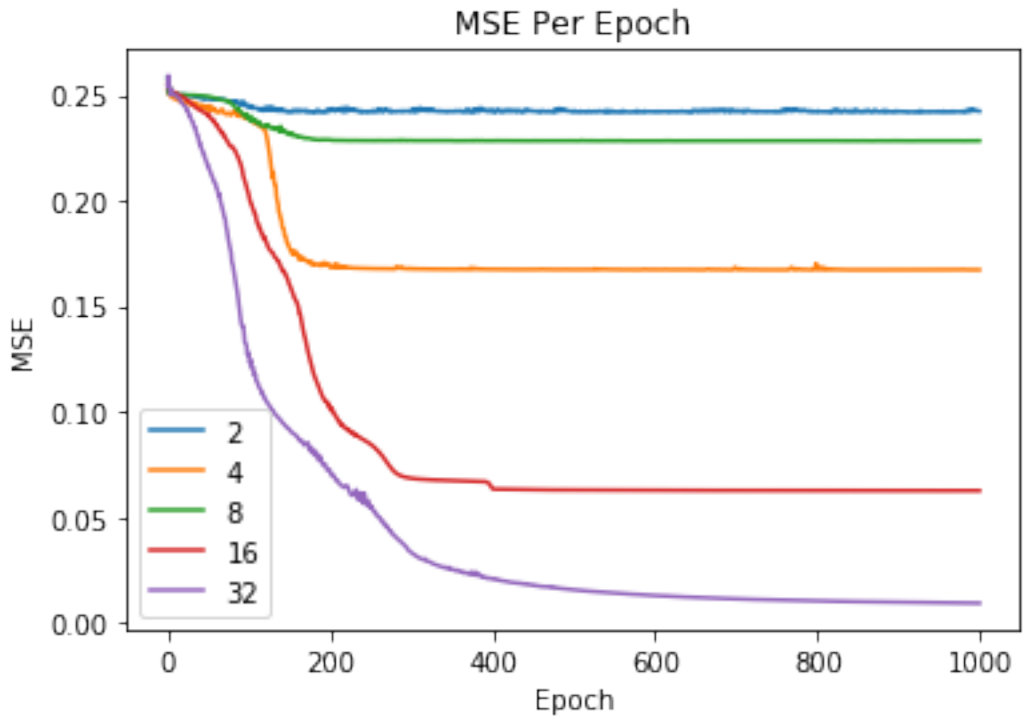
May 1 2020

1 Feed Forward Tasks

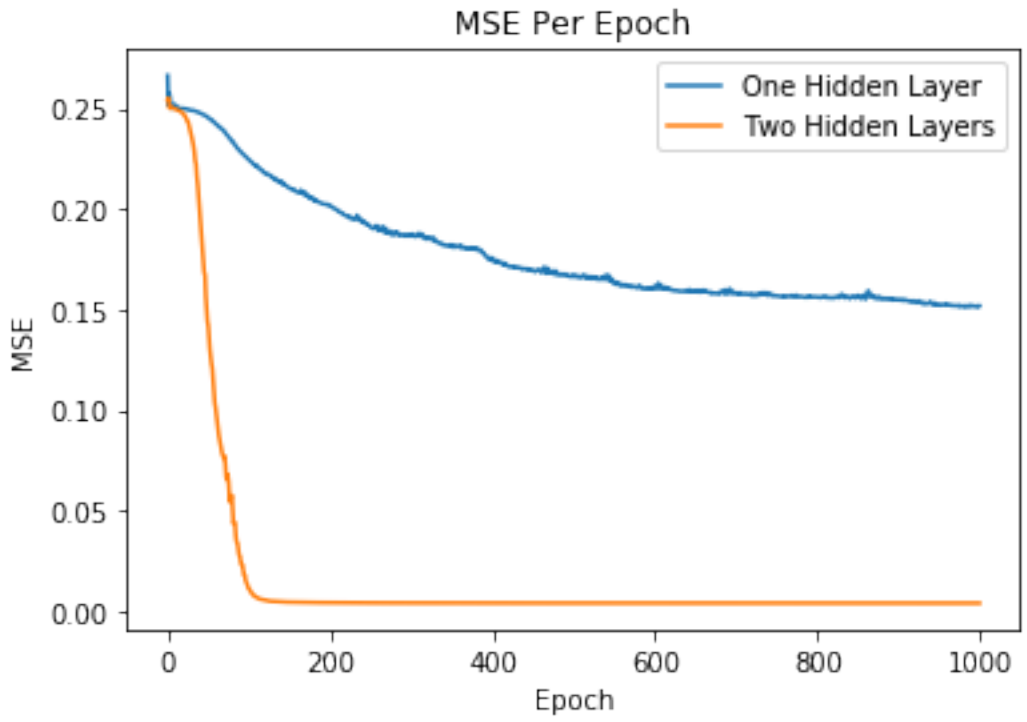
1.1 Graph of loss curve for 5 different learning rates



1.2 Graph of loss curve for 5 different numbers of hidden nodes

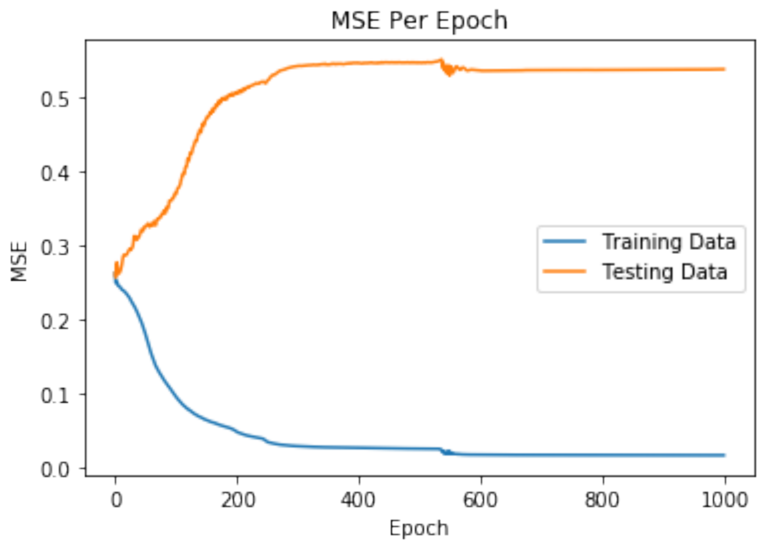


1.3 Graph of loss curve with an additional layer



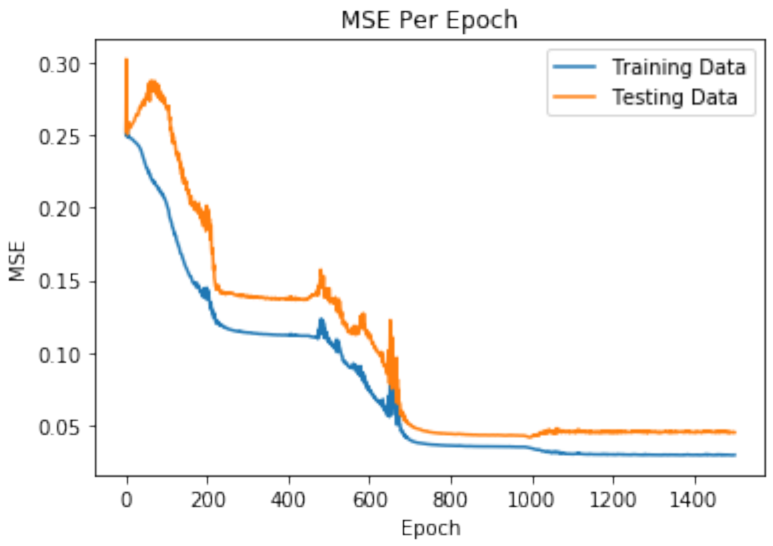
1.4 Graph of training and validation loss

I started by splitting the dataset halfway, and using the first half as the training set and the second half as the testing set, and I didn't tweak any parameters. I ended up with the following loss graph – it shows that the model was overfitting the data we had.



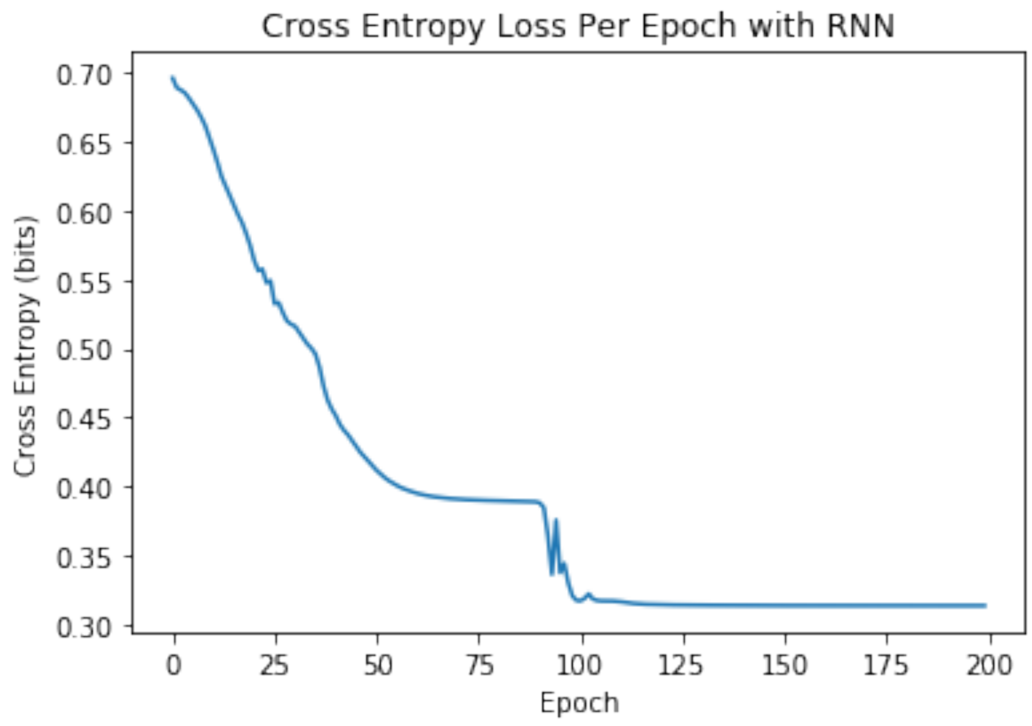
I tried to modify some of the parameters – changing the vector length (so we had more data to work with) and hidden size (to make the network more complicated and hopefully fit the test data better). I also added a weight decay to hopefully stop overfitting (an idea from Conner DiPaolo '19), which helped a little, but not much.

The thing that really improved the model significantly was shuffling the data that went into the two datasets. I think the issue was the data that went into each subset wasn't representative of the whole, and as a result it led to more overfitting. By shuffling the data, we ended up with the following graph in about 80% of runs.



2 RNN Task

2.1 Loss curve for RNN



2.2 RNN classifying at least 5 sequences of different lengths

```
# Now that we have trained the network, if all is good we should be able to classify
# a sequence of many lengths using forward predict. Try it out below and see for yourself!

# Odd
print(mynet.forward_predict(torch.FloatTensor([1,0,0,1,0,1,1,0,0,0,0,1])))
# Even
print(mynet.forward_predict(torch.FloatTensor([1,0,0,1,0,1,1])))
# Odd
print(mynet.forward_predict(torch.FloatTensor([0,0,0,0,0,0,1,0,0,1,0,1,1,0,0,0,0,0,1])))
# Even
print(mynet.forward_predict(torch.FloatTensor([1,0,0,1,0,1,1,0,0,0,1,1])))
# Odd
print(mynet.forward_predict(torch.FloatTensor([1])))
```

```
tensor([[9.6395e-06, 9.9999e-01]], grad_fn=<SoftmaxBackward>)
tensor([[9.9962e-01, 3.7636e-04]], grad_fn=<SoftmaxBackward>)
tensor([[8.8269e-06, 9.9999e-01]], grad_fn=<SoftmaxBackward>)
tensor([[9.9953e-01, 4.6727e-04]], grad_fn=<SoftmaxBackward>)
tensor([[2.7129e-04, 9.9973e-01]], grad_fn=<SoftmaxBackward>)
```