Recurrent Neural Networks

Bilal Ahmad

October 15, 2019

1 Problem

1.1 Applying Neural Networks to Sequential Datasets

In the exercise, we used back propagation to train a network with a dataset. The dataset I used was non-sequential in that each input was independent of one another. For this exercise, we are going to use a sequential dataset where the order of input values has an effect on the output. The back propagation algorithm from the previous exercise won't do well on these types of datasets because the algorithm doesn't remember the values of previous inputs. Instead, we can implement recurrent neural networks (RNNs) that exhibit temporal dynamic behavior that will better process our dataset. Specifically, we look at three algorithms for this exercise: back propagation through time (BPTT), long short-term memory (LSTM) and gated recurrent units (GRU).

1.2 Dataset

For this exercise I am using a database of cryptocurrency prices [1]. The dataset contains the close and volume values for 4 different cryptocurrencies: Bitcoin, Bitcoin Cash, Ethereum, and Litecoin in one minute intervals. The close value corresponds to the US dollar amount the cryptocurrency is was worth during the one minute interval, and the volume value corresponds to the amount of that cryptocurrency that was being traded during that one minute interval.

The data is split into training and testing sections. The networks work by taking the values from the previous sixty minutes worth of data for the 4 cryptocurrencies and making a prediction about the price of Litecoin 3 minutes into the future. Based on the current and predicted value for Litecoin, a classification is made to either "buy" or "not buy" Litecoin. A comparison is made between the actual value of Litecoin 3 minutes into the future and the predicted value and then each network does its own error propagation. This example was originally worked out on the pythonprogramming.net website [1]. I am modifying the example to fit the objectives of this exercise.

1.3 Implementing The Neural Networks

For the networks, I used built-in libraries available in Keras and TensorFlow. The BPTT, LSTM, and GRU networks are using sigmoid activation functions in RNN layer and soft maximum activations in the output layer. Additionally, I am using Adam optimization instead of stochastic gradient descent because it increases accuracy without adding much complexity. I hadn't used Keras previous to this class; to learn how to use the models I primarily relied on the pythonprogramming.net website [1]. The implementation of each network is primarily based on their examples. The full implementation can be seen in the Appendix.

2 Results

2.1 Test Accuracy

I ran each of the BPTT, LSTM, and GRU networks for 10 epochs. It would have been ideal if I kept the initial weights the same for each network but because there are over 77,000 examples in each epoch, the final error should not be significantly impacted by differences in initial weights. After each training epoch, I tested the data against the 3,800 test examples. For the original back propagation network, I trained the network on 5000 of the training examples and then ran the network against the test examples. Table 1, shows the final accuracy rate of each network as well as processing time information for the three new RNNs. In addition, Figure 1, Figure 2, and Figure 3, show the progression of the accuracy rate of the training and test examples over each epoch for the BPTT, LSTM, and GRU networks respectively.

	BackProp	**BPTT**	**LSTM**	**GRU**
Final Train Accuracy	.3610	.5606	.5831	.5693
Final Test Accuracy	.3768	.5615	.5619	.5707
Average Time/Sample	-	1 ms	6 ms	7 ms

Table 1: Accuracy and Computational Time of Each Network

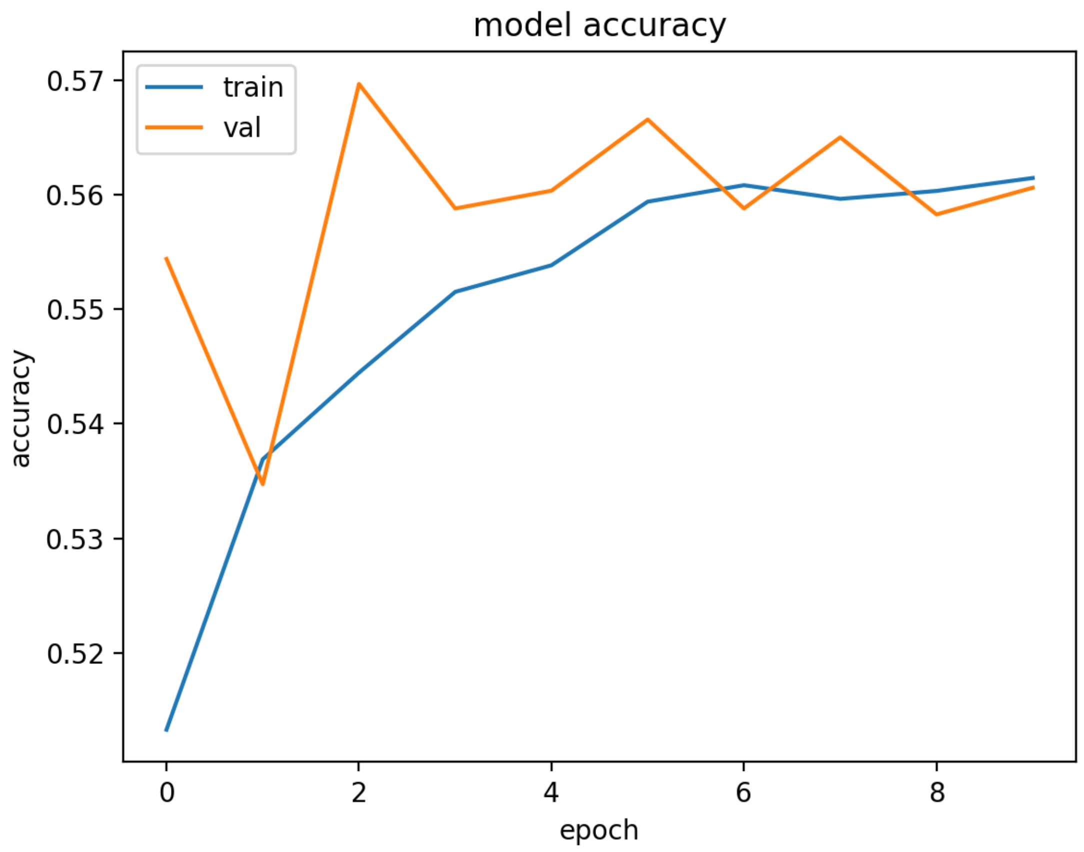

Figure 1: Accuracy vs Epoch - BPTT

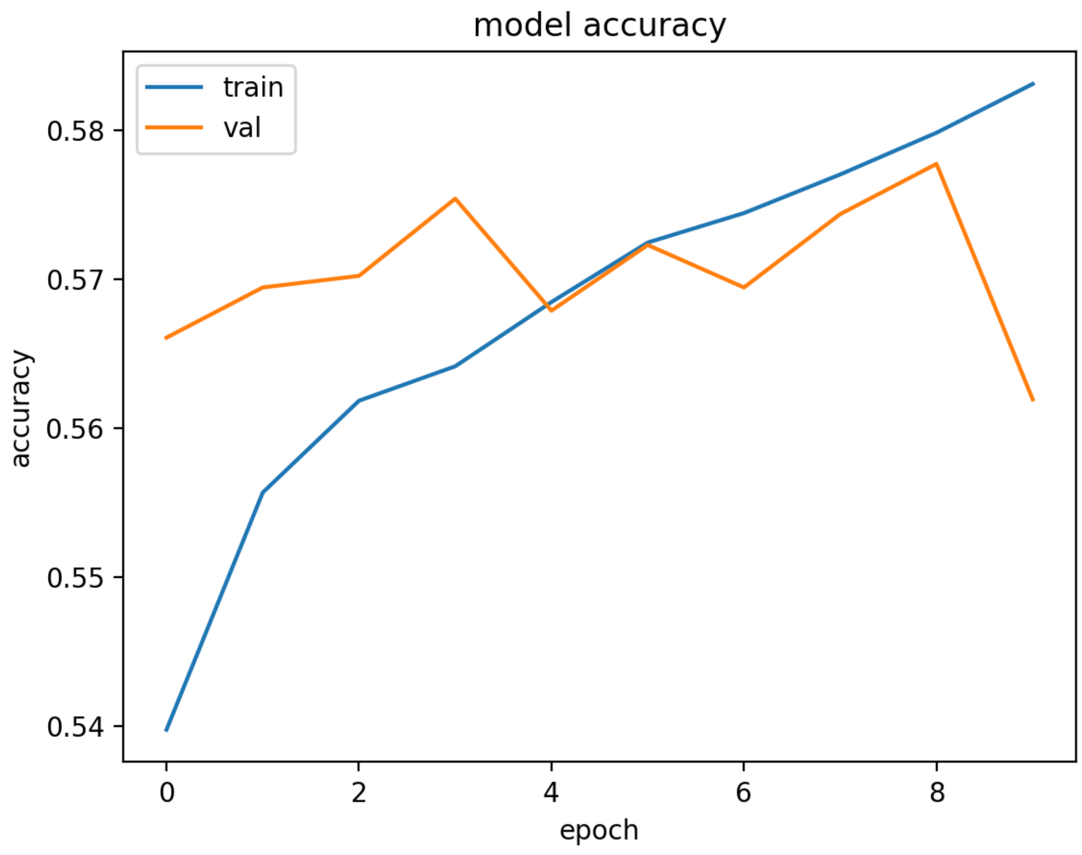

Figure 2: Accuracy vs Epoch - LSTM

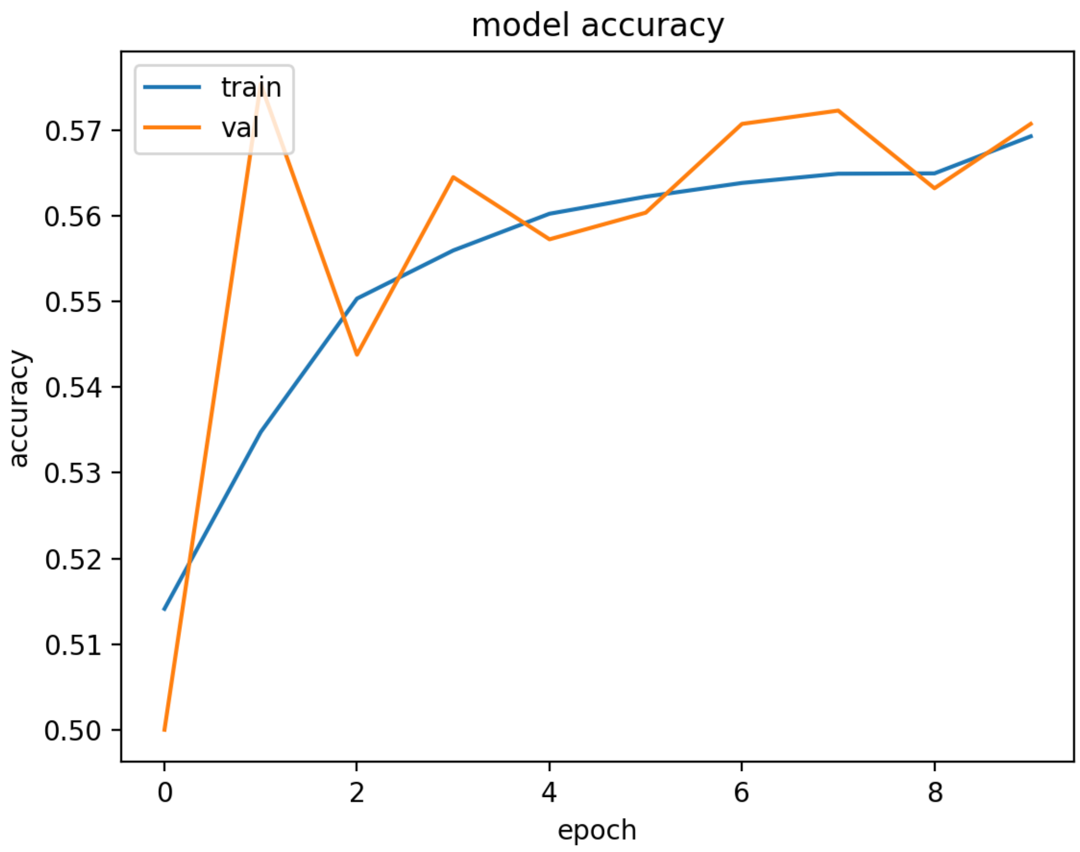

Figure 3: Accuracy vs Epoch - GRU

3 Conclusion

From Table 1, it is evident that the new RNNs performed better on the sequential dataset for this exercise than the back propagation style network from the previous exercise. The improvement in test accuracy was inconsistent during the epochs however from Figure 1, Figure 2, and Figure 3, it looks as though training data improvement for BPTT is tapering off whereas accuracy improvements for GRU and LSTM can still improve over more epochs. This is possibly because BPTT is susceptible to the vanishing gradient problem. It should be noted however in Table 1 that BPTT calculations took less time per sample so there may be instances when it is better to choose BPTT over LSTM and GRU if the BPTT network provides satisfactory accuracy results and computational time is a deciding factor.

Both the GRU and LSTM networks performed similarly. Though the GRU network had a slightly better final test accuracy, it also took slightly longer per sample. While it appears for this dataset that the networks both function about the same, it may not always be the case for other datasets.

For further study, I think I would choose a dataset that is more subjective in its evaluation criteria. For example, text prediction exercises where a network tries to produce work similar to a known author. For my course project, I want to use RNNs for music prediction so I will experiment more with text prediction examples and see if the techniques are applicable to music prediction as well. In addition, I went to presentations last week on Thursday and Friday about neuromorphic computing applications and the speakers mostly implemented spiked neural networks (SNNs). I would like to try and see if I could set one up for either a future exercise or perhaps for my course project. Finally, I would try to use virtual machines next time. It was taking upwards of forty minutes to do a single LSTM epoch so I had to reduce the amount of layers I was using. Access to more computing power would allow me to include more layers which may allow the benefits of differing network approaches to stand out.

4 References

[1] Balancing Recurrent Neural Network sequence data for our crypto predicting RNN - Deep Learning basics with Python, TensorFlow and Keras https://pythonprogramming.net/