Assignment 3

Jacob Joy

Kent State University

Dr. CJ Wu

4/06/2025

Setup

The weather data was imported and split into header/data as well as creating a separate vector that contained the temperatures. The data portion had the date feature removed. Next, an amount of observations was determined to be included in the training, validation, and test data which corresponds to 50%, 25%, and 25% of total observations respectively. Next, the data was normalized. Finally, the data was split into training, validation, and testing starting with the oldest data used in training, then validation, and finally testing. This is a change from our normal use of random sampling, but it is required to maintain the time these observations were taken.

Procedure

1. timeseries_dataset_from_array was used to create training, validation, and test datasets. A sampling rate of 6 was used so that 1 10min time was used per hour to reduce the amount of redundant data as over an hour period the temperature will not change drastically. A sequence length of 120 was used to correspond with the past 5 days.

2. A common sense baseline was created based on the assumption that 24 hours in the future will be the same as it is currently. This comes to a MAE of 2.44 Validation and 2.62 on test data. We will need to beat 2.44 at a minimum in order to tell that our models are working.

3. Multiple models were created using different types of dense, conv1D, RNN, LSTM, and GRU layers

4. The models were compiled and trained using a callback to keep track of the best validation loss and epoch. To better see the results additional formatting was applied to the checkpoint file that allowed for the epoch number to be amended to the file.

5. The training and validation loss was recorded as well as a viewing of the training vs validation loss graphs.

6. Additionally, some models were reran using data that had feature engineering performed which consisted of removing many features taking the total used from 14 to 7. This was conducted by removing the features that contained the same information from other features and using a list of the most important conditions to determine temperature.

Summary

It was determined that the commonsense baseline had an MAE of 2.44. Using dense layers only this baseline was not able to be beaten, and the dense layers had one of the worst performances out of all models tried. Next, Conv1D layers with max pooling and global average pooling were used and while this did not beat our baseline because of its MAE being 2.94, this can be attributed to the pooling layers destroying the order and importance of the time series. A simpleRNN that used any size and returned full results was the worst performing model at 6.65

MAE and did not train. Bidirectional LSTM was used and while technically performed better then the common sense it was only by 0.001. This can be contributed to the type of question being ask about temperature where prediction in the future is what was being looked for and bidirectional not only goes forward but backwards as well. The last model that failed the commonsense baseline check used two GRU layers, dropout, slower learning rate, and adamax. While this model failed with a 2.51 MAE it may have been able to beat the baseline if more epochs were running as, it was still learning at 10 epochs.

Many models were tried first with all features and rsmprop as the optimizer. While testing it was determined that the Adamax optimizer had performed better. The same 2 layer GRU model was used and rsmprop had a validation MAE of 2.29 and the Adamax model had a validation MAE of 2.244. When researching different optimizers Adamax was chosen for its ability to learn time-variant processes. (Kingma et al, 2014)

The top performing models were reused with data that only had 7 features as described in procedure 6. The best performing models used all the features. The top 3 performing models based on validation MAE were GRU and Dropout using rsmprop at 2.2357, two layers of GRU and Adamax at 2.2444 and A GRU layer of size 64, two at 32 dropout and adamx at 2.2486. These models all performed almost equally on the testdata at a test MAE of 2.42, 2.4, and 2.43 respectively.

Conclusion

Time series analysis with ML is a time-consuming process and depends greatly on what question you are asking and the type of data you have. One reason for the increase in time is due to the need for executing on a CPU as most GPUs are not able to perform better. I did test this and on a GPU it took as much as 4x longer on some models. Another reason discovered from the experiments conducted is the need to run many epochs before overfitting or a bottleneck was hit in performance. An additional observation was made that the validation MAEs would perform better and would require additional tuning to overfit. This makes sense because of the way our data sets were created the training data is the oldest so it would fit that it would work better on validation than itself as we are trying to predict future temperatures. From the experiments I conducted it seems that the best validation MAE result was 2.24 and on Test data 2.4 which is better than the commonsense baseline.

Experiment	Training_MAE	Training_loss	Validation_MAE	Validation_Loss	Test_MAE	Test_loss
Common Sense Baseline			2.44		2.62	
two dense layers	7.0738	76.0871	6.6461	65.1983	6.64	65.6508
Conv nets	2.9033	13.3704	2.9403	13.8329	3.09	15.1478
LSTM 1 layer	2.3812	9.3127	2.377	9.3387	2.56	10.4773
SimpleRNN any size	2.452	9.9317	2.2984	8.7426	2.48	10.2191
SimpleRNN any size, returns last output	2.4191	9.6849	2.3198	8.9307	2.53	10.8055
SimpleRNN any size, returns full	7.0766	76.3676	6.6546	65.3431	6.64	66.48
three simpleRNN layers	2.4815	10.0801	2.3289	8.9223	2.5	10.3563
LSTM and Dropout	2.5763	10.9665	2.319	8.8532	2.5243	10.073
GRU and Dropout	2.6311	11.4084	2.2357	8.375	2.42	9.5302
Bidirectional LSTM	2.303	8.7006	2.4391	9.9168	2.56	10.4653
GRU 2 layers and Adamax optimization	2.7883	12.9615	2.2444	8.4067	2.4	9.454
GRU 2 layers and rmsprop optimization	2.597	11.1876	2.2951	8.7496	2.42	9.5247
Cov1D, 2 GRU, Dropout, adamax	2.7606	12.6376	2.3023	8.7707	2.5098	10.2644
Extra Dense layer, 2 GRU, Dropout, Adamax	2.5609	10.7427	2.2778	8.5822	2.3911	9.4281
64 layer 32, 32, dropout, adamax	2.7996	13.0259	2.2486	8.47	2.43	9.6761
3 GRU and dropout	2.7929	13.007	2.263	8.5181	2.41	9.5672
GRU 2 layers and Adamax optimization dropout lowered	2.5375	10.5833	2.2604	8.5043	2.4	9.4846
Feature reduction						
GRU and Dropout	2.7489	12.4654	2.2657	8.5582	2.44	9.6901
GRU 2 layers and Adamax optimization	2.8261	13.2464	2.2737	8.6114	2.44	9.7464
64 layer 32, 32, dropout, extra dense adamax	2.6831	11.9198	2.2693	8.5311	2.42	9.6146
Conv1D, 2 GRU, Droput, Learning rate, adamax	2.5382	8.7537	2.3034	8.7537	2.48	10.1737
GRU 2 layers, dropout, lower learning rate, adamax	3.1678	16.8416	2.51	10.8539	2.72	13.008

Red highlights failed to beat commonsense baseline. Yellow highlights are the top 3 performers.