Times Series Assignment

Report

This Report works on a weather forecasting as a timeseries of hourly measurements of quantities such as atmospheric pressure and humidity, recorded over the recent past by the sensors on the roof of building and we are working with the records of timeseries dataset at the weather station at the Max Planck Institute for Biogeochemistry in Jena, Germany and it was recorded for every 10 minutes over several years and dealt with different kinds of machine learning techniques--- Recurrent Neural Networks(RNNs).

Additionally, it looks at how Recurrent Neural Networks (RNNs) can be used to analyze time-series data, with a particular emphasis on weather forecasting. The objective is to investigate various methods for enhancing RNN models' capacity to forecast weather patterns. These methods involve adjusting the structure of the RNN model, experimenting with different kinds of recurrent layers like GRU and LSTM, and adding 1D convolutions in addition to RNN layers. The study describes the implementation of these strategies, assesses their effectiveness with validation datasets, and displays the models that verified most successfully on the test set. The report's ultimate goals are to demonstrate RNNs' aptitude for managing time-series data and to pinpoint methods for raising their precision in weather prediction.

An overview of time-series data models:

Model	Dense Units	Dropout	Loss	Test MAE	Val MAE
Basic Machine Learning Model	8	No	11.6215	2.68	2.651
1D Convolution model	16	No	14.5634	3.03	3.0716

RNN Model: RNN Model	Dense Units	Dropout	Loss	Test MAE	Val MAE

Simple RNN Model	16	No	151.3478	9.93	9.8538
Stacked Simple RNN Model	16	No	151.1180	9.91	9.8365
Stacked Simple RNN Model	32	No	151.1021	9.90	9.8317
Stacked Simple RNN Model	48	No	151.1245	9.90	9.84

Long Short-Term Memory with different dropouts:

Long Short-Term Memory: LSTM	**Dense Units**	**Dropout**	**Loss**	**Test MAE**	**Val MAE**
LSTM- with 0.25 dropout	16	Yes	44.315	5.20	5.3625
LSTM – with 0.35 dropout	16	Yes	18.707	3.27	3.1378
LSTM – 0.45 dropout	32	Yes	25.6877	3.85	4.4855

Gated Recurrent Unit:

Gated Recurrent Unit: GRU	**Dense Units**	**Dropout**	**Loss**	**Test MAE**	**Val MAE**
Simple GRU	16	0.5	67.6212	6.69	6.71
Stacked GRU	16	0.5	9.7373	2.43	2.26
Stacked GRU	32	0.5	9.4651	2.43	2.48
Stacked GRU	48	0.5	8.2850	2.52	2.52

Long Short-Term Memory :

Long Short-Term	**Dense Units**	**Dropout**	**Loss**	**Test MAE**	**Val MAE**

Memory: LSTM					
LSTM-Simple	16	0.5	10.9701	6.68	6.70
LSTM - Stacked setup with 16 units	16	Yes	10.4957	2.5340	2.3777
LSTM - Stacked setup with 32 units	32	Yes	10.8244	2.56	2.5240
LSTM - dropout-regularized, stacked model with 48 units	48	Yes	10.4181	2.52	2.3433
Bidirectional LSTM	16	No	30.3490	4.39	4.2132

Combination:

Combination : Combination	Dense Units	Dropout	Loss	Test MAE	Val MAE
V1D Convnets and LSTM with dropout together	64	0.5	27.4856	4.07	4.05

From the above models, the 1D convnet model performed worse than the densely connected one, only by achieving the validation of 3.07, which is far from the commonsense baseline because it doesn't quite follow the properties of morning and evening reports in the data. The recent data is very informative for predicting future data, and the maxpooling and global average pooling largely destroy the information.

We need to use Recurrent Neural Networks(RNNs) is the family of neural networks and among them we will use Long Short Term memory(LSTM) which is very popular, as for the LSTM model, it is far better than and beats the common-sense baseline by achieving the validation MAE 2.34 and test MAE 2.58 because using the RNN reuses quantities computed during the previous iteration for the loop and we use simple RNN and stacked RNN which we seen that the performance was very low with test MAE 9.90 and 9.93. It showed much difference when compared to other models because of vanishing gradient problem due to training of deep neural networks which is like feedforward networks. There are more methods, and we are currently going with deep learning sequence models.

To get the best MAE, we altered the units in each recurrent layer for stacking units as 16, 32, and 48 for LSTM, and as we increased the number of units in each layer, we got the most suitable validation MAE and test MAE as 2.34 and 2.52, which give the best results so far. As for GRU, we got the validation and test MAE as 2.52 and 2.52 as we increased the number of units due to its architectural differences and how they handled the sequential data (as for long sequential data).

After we run different models of LSTM with dropout, there seems to be better results while increasing the dropout rate to prevent overfitting, not as of stacked layers due to various factors related to data and model performance.

As we replaced LSTM instead of GRU, we got the validation and test MAE almost the same when we changed the number of units in each of them, but it is slightly better for GRU, and both are very effective in real-world applications. As we did, the bidirectional LSTM did not do very well because of the continued increase in overfitting due to its data complexity.

Finally, we did a 1D convolutional model combined with RNN with regularized layers and got MAE 4.07, which is far more than the baseline model, most likely in terms of complexity, model capacity, hyperparameter tuning (units, kernel size, etc.), and data characteristics.

Summary:

- Based on capturing optimal performance, LSTM and GRU showed promising results. RNNs should not be used for lengthy sequencing data jobs like temperature prediction.
- Overfitting in RNNs can be avoided with the use of strategies like dropout, which enhance the generalization of the model.
- In stacked recurrent layers, sometimes it will get better results and sometimes it won't, and that could even cause overfitting and result in generalization with the complexity of the model.
- While bidirectional RNNs did not give the expected results due to their better performance due to their model complexity and data characteristics.
- We get similar, better outcomes when we use LSTM instead of GRU, and that will be useful for real-world applications.

To enhance the model's performance, we need to reduce overfitting, preprocess the data, normalize the data, and do feature engineering.