USING A LONG SHORT-TERM MEMORY (LSTM) MACHINE LEARNING MODEL TO PREDICT FORECAST BIAS

***Konstantin Mladenov*[1], *Boryana Tsenova*[2]**

[1]University of Mining and Geology “St. Ivan Rilski”, 1700 Sofia; E-mail konstantin.mladenov@meteo.bg
[2]National Institute of Meteorology and Hydrology, Tsarigradsko shose 66, 1784 Sofia, Bulgarial; E-mail: boryana.tsenova@meteo.bg

ABSTRACT. Deep learning is a part of machine learning that refers to neural networks with multiple hidden layers which can learn from increasingly abstract representations of input data. Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.
Using Python’s deep learning library Keras can allow us to predict future bias in forecast, giving us an idea of what the bias is going to be and letting us compensate for it beforehand.

Key words: Keras, Python, bias, machine learning, data manipulation, time series

ИЗПОЛЗВАНЕ НА МОДЕЛ ЗА МАШИННО ОБУЧЕНИЕ С ДЪЛГОТРАЙНА ПАМЕТ (LTSM) ЗА ПРОГНОЗИРАНЕ НА ОТКЛОНЕНИЯ В ПРОГНОЗИТЕ

***Константин Младенов*[1], *Боряна Ценова*[2]**
[1]Минно-геоложки университет „Св. Иван Рилски“, 1700 София
[2]Национален институт по метеорология и хидрология, „Цариградско шосе” № 66, 1784 София, България

РЕЗЮМЕ. Задълбоченото учене е част от машинното обучение, която се отнася до невронни мрежи с множество скрити слоеве, които могат да се учат от все по-абстрактни представяния на входните данни. Python е ясен и мощен обектно-ориентиран език за програмиране, сравним с Perl, Ruby, Scheme или Java.
Използването на библиотеката за дълбоко обучение Keras на Python може да ни позволи да се борим с бъдещо отклонение в прогнозата, като ни даде представа какво ще бъде то и ни позволи да го компенсираме предварително.

Ключови думи: Keras, Python, отклонение, машинно обучение, работа с данни, потоци от данни

Introduction

Numerical weather prediction is a type of weather forecasting that uses a set of equations describing the state of the atmosphere. These equations are translated into computer code and use numerical methods, parameterisations, and other equations to provide a forecast for a given domain. In order to create a perfect short-term weather forecast, a computer model would need a perfect representation of the initial state of the atmosphere, which has not been achieved yet.

As such, there are inaccuracies in observations, initial and boundary conditions that lead to forecast bias. The objective of bias correction is to minimise the systematic error in the next forecast using bias from past errors. The magnitude of bias at a grid point depends upon geographical location and season. In this paper, bias correction using deep learning is examined.

Deep learning has led to major advances in computer vision. Images can now be classified, objects can be found in them, and they can be labelled with captions.

Deep neural networks do that using many hidden layers that sequentially learn more complex features from the raw data. The first hidden layers might only learn local patterns, while every subsequent layer learns more complex patterns and representations. Using this method, the final layers can classify images, find disease in plants, or analyse data to be able to give a sufficient output as to what certain data might look like in two months. Using Keras for Python makes it easy to do so, not only due to Python’s easy-to-read approach to code, but also due to Keras’ minimalistic, modular approach, which allows using deep learning algorithms with little coding background. In the paper, the usage of a Long Short-Term Memory (LTSM) model to forecast the mean monthly temperature bias of the model forecast of Bulgaria’s capital, Sofia, is discussed. LTSMs are well-suited to classifying, processing, and making predictions based on time series data. They are a type of Recurrent Neural Networks (RNN) that have internal memory. RNNs perform the same function for every input of data, while the output of current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input. Sofia was chosen for the experiment not only for its importance in forecasting, but due to a tendency noticed within the past couple of years of increased mean monthly temperature bias relative to the rest of the domain. This is caused by a lot of factors, mainly by its geographical position and the inversions that happen around November increasing the bias of those months even more.

Installing and using Python and Python with Keras

Most of the times, using Python does not require installing it manually, as Python3.xx now comes pre-installed on most Linux distributions. Manual installation is required if a different version is needed than the one that comes pre-installed, or if running a Windows OS. Running Python scripts, like our model, is done using the "python" command, the usage of which might involve a set-up of the $PATH environment variable. Of the deep learning libraries we have used, Keras for Python is one of the most popular libraries for deep learning. In order to use it, several packages for Python need to be installed. Installation of Python packages usually happens in two ways: either using pip or conda, or most often using a virtual environment in order to avoid any conflicts between different package versions. The packages needed for Keras are:

- SciPy with NumPy
- Matplotlib
- Theano.

All of those are available for installation using a conda environment, except for Theano, which cannot be installed using conda and needs to be installed with pip.

After installing and entering the environment, we can check if everything has been set-up correctly.

```
>>> import numpy
>>> import theano
>>> print numpy.__version__
1.11.0
>>> print theano.__version__
0.8.2
>>> quit()
```

If everything is set-up correctly, we can move on to installing the Keras library itself using **pip install keras.** Upgrading Keras can be done with the same command with the –upgrade flag.

Here is how to check if everything is installed correctly:

```
$ python -c "import keras; print keras.__version__'
Using Theano backend.
1.0.4
```

Predicting temperature bias using Python for Keras

For the purposes of this experiment, the packages "scikit-learn" and "Pandas" for csv file manipulation were also used. The bias was retrieved from data for 2018, 2019, and nine months from 2020, all of which were written to .csv files. The values from those files were read using Pandas.

Creating a single forecast

For the needs of data assimilation, an experiment with an autoregressive method was run. Autoregressive models are the next step in the data as a linear function of the observations of the previous time steps.

The values of all of the previous months were used to do a prediction of a single next month's BIAS (Fig. 1)

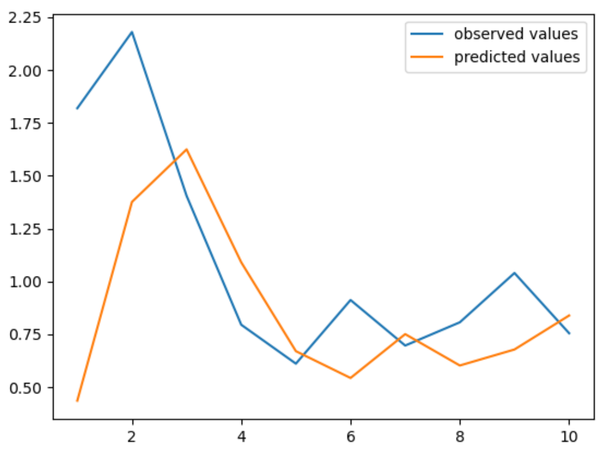

Fig. 1 Autoregressive model BIAS forecast

The model deviates from the observations, with a Root-Mean Square Deviation (RMSD) of 0.54929.

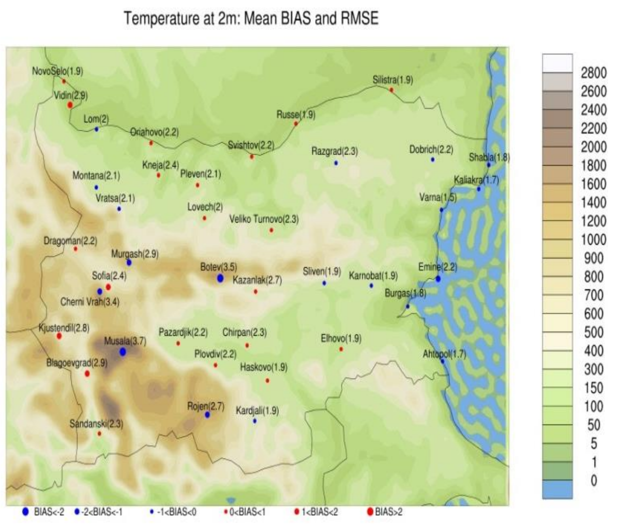

Fig. 2 Mean BIAS and RMSE of the temperature at 2 m (in °C) forecast by ALADIN during 2020 (from January to August) for each station (coloured circles show the BIAS, while RMSE is indicated next to each station name)

The first 19 months of data (Fig.2 and Fig. 3) are then used to train the model, and the 14 months left are used to test it.

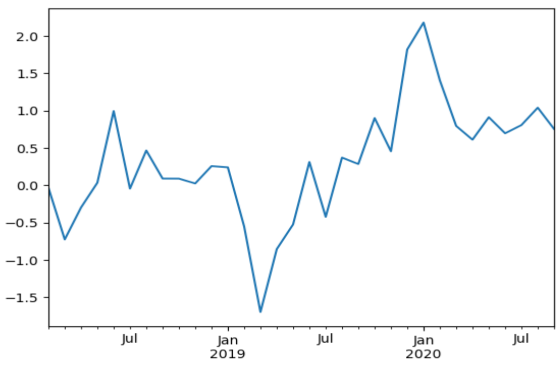

Fig. 3 Mean monthly bias of the temperature at 2 m (in °C) forecasted by ALADIN for 2018, 2019, and nine months from 2020

One month of data from the last 14 months was used to forecast three subsequent months of mean BIAS, so that we can see how the model does against actual observations. Each time step of the dataset is walked one at a time. The model is used to make a forecast for the time step, and then, for our test dataset, an actual expected value is taken and made available to the model for the forecast on the next time step. This is what happens when we have a new BIAS calculation every month.

After preparing the data, doing model evaluation and persistence modelling, forecasts can be made and evaluated. We started by making the model analyse the data with 1500 epochs and 1 neuron (Fig. 4).

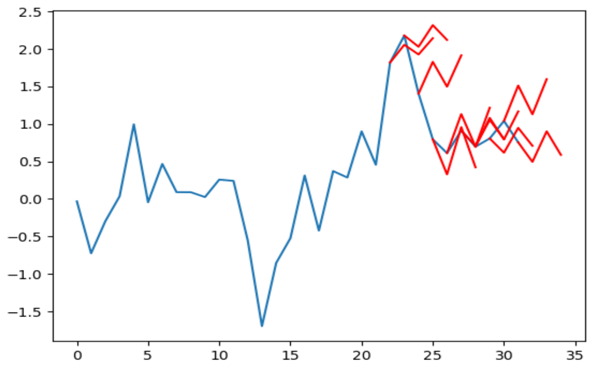

Fig.4 Mean monthly bias plotted alongside the values forecasted for three months ahead. Model ran with 1500 epochs and 1 neuron

The red lines show the three month forecast from each month of the test batch.

The root-mean squared deviation of the model run like this was as follows:

t+1 RMSD: 0.693904
t+2 RMSD: 0.820104
t+3 RMSD: 0.968915

What this shows is that the model performs progressively worse with the increase of the time step – something which we expected. Also, the decline in performance is clear where there is a big difference between biases months-wise.

Next, more experiments were run with more epochs – 3000 epochs (Fig. 5).

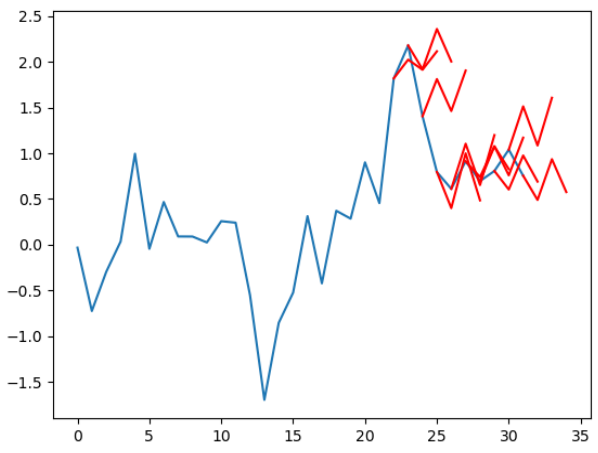

Fig. 5 Mean monthly bias plotted alongside the values forecasted for three months ahead. Model ran with 3000 epochs and 1 neuron

The RMSD with two times the epochs was as follows:

t+1 RMSD: 0.683846
t+2 RMSD: 0.819885
t+3 RMSD: 0.992471

The decrease of RMSD is slight compared to the computing power needed to run two times the epochs, which shows that more epochs is not the way to go to increase the accuracy of the model.

Increasing the neurons also increased the RMSD twofold and gave the model a severe bias when it came to forecasting negative values. However, it did not affect positive values and this is worth investigating. Tuning the model so that increasing the neurons does not affect negative values, looks like it might give out better results than using a single neuron. Multiple neurons were also better at predicting a decrease in bias, where a single neuron predicted an increase or a small decrease (Fig. 6).

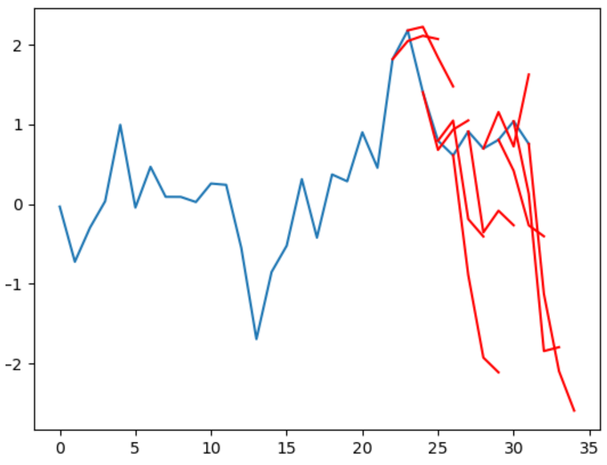

Fig. 6 Mean monthly bias plotted alongside the values forecasted for three months ahead. Model ran with 1500 epochs and 3 neurons

t+1 RMSD: 1.052310
t+2 RMSD: 1.656046
t+3 RMSD: 1.804132

Conclusion

LTSM networks can be used to create large recurrent networks that, in turn, can be employed to address difficult sequence problems. Deep learning can be fundamental in being able to tackle bias and not have it affect forecasting while also working on combating what it is caused by. Finding the right model and tuning it correctly can produce results that are so close to observations; deep learning can also prove useful in predicting severe weather anomalies by analysing data patterns.

Where deep learning might really shine is data assimilation and ensemble forecasting. The reason is that neural networks analysing data can provide new initial conditions and can predict parameters to be assimilated into next runs.

References

Brownlee, Jason. 2020. Deep Learning for Time Series Forecasting. https://machinelearningmastery.com/ category/deep- learning-time-series/

Durai, V. R., R. Bhradwaj. 2014. Evaluation of statistical bias correction methods for numerical weather prediction model forecasts of maximum and minimum temperatures. – Nat. Hazards, 73, 1229–1254. https://doi.org/10.1007/ s11069-014-1136-1

Kamalov, Firuz & Smail, Linda & Gurrib, Ikhlaas. (2021/03/21). Forecasting with Deep Learning: S&P 500 index., 2020 13th International Symposium on Computational Intelligence and Design (ISCID), DOI: 10.1109/ISCID51228.2020.00102, Hangzhou, China

Official Python3.xx documentation https://docs.python.org/3/

Tsenova, Boryana, Andrey Bogatchev. 2020. On the use of atmospheric instability indices based on NWP model production for thunderstorm forecast, Bulgarian Journal of Meteorology and Hydrology, 24/2/2020, Sofia, Bulgaria.

Tsenova, Boryana, Rilka Valcheva. 2020. Verification of the regional numerical weather prediction in Bulgaria, Bulgarian Journal of Meteorology and Hydrology, 24/2/2020, Sofia, Bulgaria.