DT0280 Project
Report

Time series forecasting using Long-short-term-memory (LSTM) and accuracy comparison with ARIMA and VAR

Submitted in partial fulfillment of
Machine Learning course

Submitted by

261414 Dagmawi Abraham Seifu

Under the guidance of
Pasquale Caianiello

UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila
L'Aquila, Italy

Winter Semester 2020

Abstract

Time series forecasting has enormous significance because data from many areas like demand and sales, number of visitors to a website, stock price, process and quality control etc. are essentially time series data. Time is a natural element that is always present when data is collected. Time series analysis involves working with time based in order to make predictions about the future. Traditionally, there are several techniques to effectively forecast time series data such as ARIMA and VAR. Nowadays Machine Learning (ML) algorithms have introduced new approaches to prediction problem. Machine Learning, especially Deep Learning (DL) methods are capable of identifying structure and pattern of data such as non-linearity and complexity in time series forecasting. Especially, the temporal structure a time series data presents can be exploited through Recurrent Neural Networks (RNNs) where we explicitly construct a sequential representation of our data. Therefore It's an interesting question to know the accuracy and precision of traditional forecasting techniques when compared to ML (DL) based forecasting methods. Long-Short-Term-Memory (LSTM) method is used due to its use in preserving and training the features of given data for a longer period of time. The main objective of this project is to investigate which forecasting methods offer best prediction with respect to lowest forecast errors and higher accuracy of forecasts. To the best of my knowledge, there is no specific evidence for using LSTM in time series forecasting than traditional forecasting methods such as ARIMA and VAR. With the goal of comparing performance of ARIMA, VAR and LSTM, I conducted a series of experiments on some selected real time series and synthetic chaotic time series data.

Contents

Chapter 1

Introduction

1.1 Time Series

A time series is a sequence of data points (observations) recorded over a continuous regular time intervals, where each data point consists of a time stamp and one or multiple values. Time series analysis is the preparatory step before developing a forecast for the series. Time series is time dependent, so the basic assumption of a linear regression model that the observations are independent does not hold in this case. Time series is said to be stationary if its statistical properties such as mean, variance remain constant over time. Most TS models work on the assumption that the TS is stationary.

Univariate: a series with a single time-dependent variable.
Multivariate: it has more than one-time dependent variable. Each variable depends not only on its past value but also has some dependency on other variables. The dependency is used for forecasting future values.

1.2 Traditional Statistical Forecasting models Used

1.2.1 Auto Regressive Integrated Moving Average (ARIMA)

An ARIMA model is a class of statistical models for analyzing and forecasting time series data.
AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
I: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

1.2.2 Vector Auto Regression (VAR)

VAR model is an extension of univariate regression models to Multivariate time series data. It is a multi-equation system where all the variables are treated as endogeneous (dependent). There is one equation for each variable as dependent variable, i.e each variable is a linear function of past values of itself and past values of all other variables. The right hand side of each equation includes lagged values of all dependent variables in the system.

Chapter 2

Datasets

2.1 Jena Weather Dataset

The dataset contains weather features observation recorded every 10 minutes. The original dataset is from 2009-2016, too large to train a model on my machine, so I used only observations of 2016. The dataset is Multivariate time series with 14 features.
`https://www.bgc-jena.mpg.de/wetter/weather_data.html`

Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
2016-01-01 00:00:00	999.08	-0.01	273.22	-0.44	96.9	6.10	5.91	0.19	3.69	5.92	1271.32	1.16	2.04	192.4
2016-01-01 00:10:00	999.03	0.01	273.25	-0.41	97.0	6.11	5.93	0.18	3.70	5.94	1271.16	1.01	2.12	211.6
2016-01-01 00:20:00	999.07	0.06	273.29	-0.36	97.0	6.13	5.95	0.18	3.71	5.96	1270.97	0.80	1.52	203.8
2016-01-01 00:30:00	999.09	0.07	273.30	-0.36	96.9	6.14	5.95	0.19	3.71	5.96	1270.93	0.77	1.64	184.2
2016-01-01 00:40:00	999.09	-0.05	273.18	-0.50	96.8	6.09	5.89	0.19	3.68	5.90	1271.54	0.84	1.92	200.1

Figure 2.1: Weather dataset

I used dataset both as source of Univariate time series (extracting only the temperature feature, for the sake of comparison on univariate time series) and Multivariate time series.

2.2 Beijing PM2.5 Dataset

Observations of Pm2.5 Air pollutant concentration in beijing. The dataset is multivariate time series with 8 features, recorded from january 1, 2010 to december 31, 2014. The measurement contains hourly observation.
`https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data`

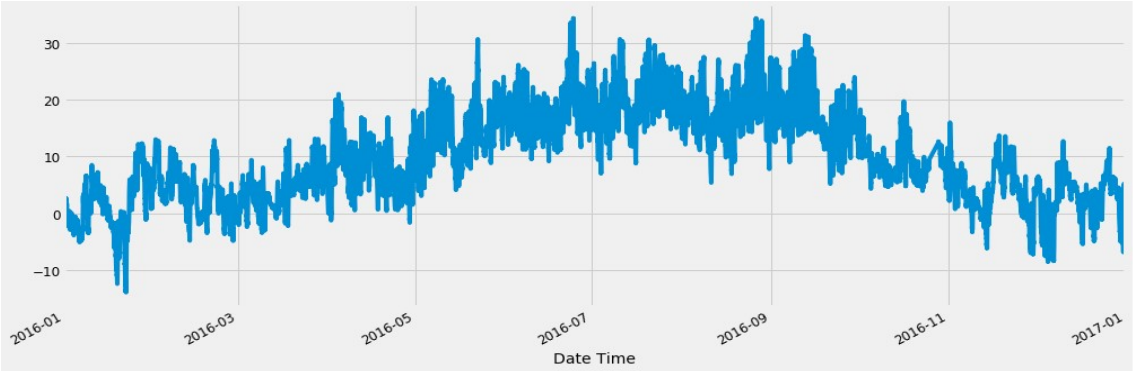

Figure 2.2: Temperature time series extracted from the weather dataset

date	pollution	dew	temp	press	wnd_dir	wnd_spd	snow	rain
2014-12-31 19:00:00	8.0	-23	-2.0	1034.0	NW	231.97	0	0
2014-12-31 20:00:00	10.0	-22	-3.0	1034.0	NW	237.78	0	0
2014-12-31 21:00:00	10.0	-22	-3.0	1034.0	NW	242.70	0	0
2014-12-31 22:00:00	8.0	-22	-4.0	1034.0	NW	246.72	0	0
2014-12-31 23:00:00	12.0	-21	-3.0	1034.0	NW	249.85	0	0

Figure 2.3: PM2.5 Pollution Dataset

2.3 Synthetic chaotic time series

For the purpose of comparison of the forecasting models on nonlinear (rather chaotic) time series, I generated a time series using the famous chaotic dynamical system, the Lorenz differential equation, developed by an MIT meteorologist Edward Lorenz in the 1960s. The Lorenz system described a system of nonlinear differential equations that modeled thermally induced fluid convection in the atmosphere. Lorenz studied the Navier-Stokes equations which describe relationship among pressure, temperature and density in moving fluid. It described a model in which fluid flows in a container whose top and bottom surfaces are cooled and heated bottom surfaces are cooled and heated respectively to create a temperature gradient similar to the atmosphere. As the gradient increased fluid transitioned from stationary to steady to chaotic flow. The system is formally described by three ODES as follows

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

Model demonstrate deterministic chaos at certain parameter values and initial conditions (the famous depiction used, sigma = 10, beta 8/3 and rho = 28). Recurrent Neural Networks by their nature are dynamic and most likely allow for modelling of chaotic behavior. Predicting future behavior of the dynamical system involves recording and analyzing the results of the system over time. I used the python module Scipy.integrate.odeint to integrate the equation at

certain regular time steps. The observations (solutions) will form a chaotic time series data.

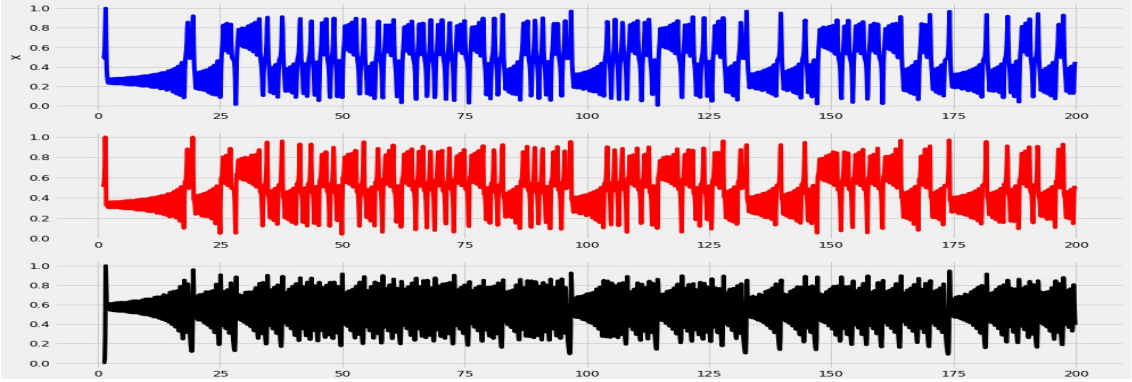

Figure 2.4: Chaotic time series obtained from Lorenz dynamics

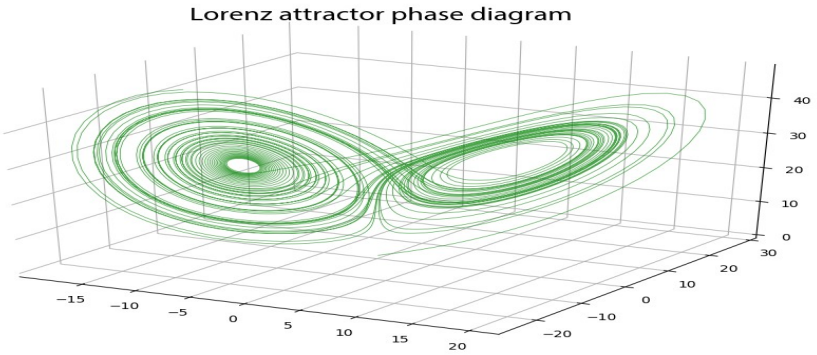

Figure 2.5: Lorenz attractor

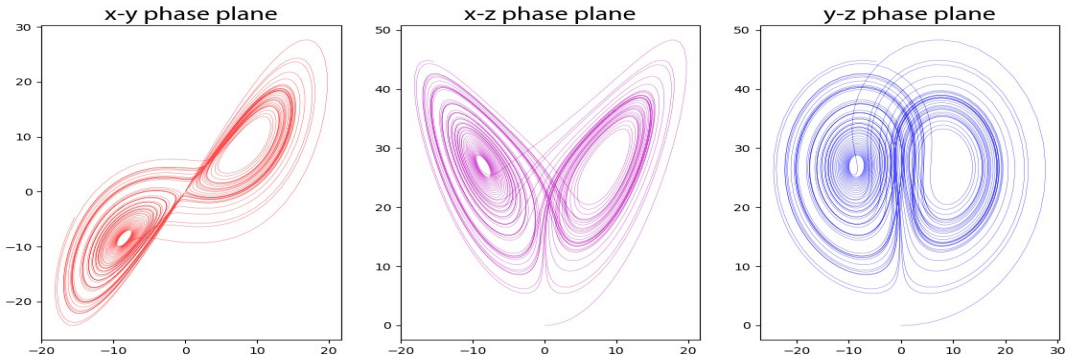

Figure 2.6: Phase portrait

Chapter 3

LSTM, Chosen language and Frameworks

3.1 LSTM in brief

Long-short-term-memory (LSTM) network is kind of recurrent neural network with the capability of remembering the values from earlier stages for the purpose of future use. Recurrent neural networks are class of neural nets that can predict the future. They can work on sequences of arbitrary lengths, rather that on fixed-inputs like feed-forward neural networks. They can take sentences, documents or audio samples as input. A recurrent neural network looks very much like a feed-forward neural network, except it also has connections pointing backwards. Since the output of a recurrent neuron at time step t is a function of all the inputs from the previous time steps, we could say it has a form of memory. Recurrent neural networks work just fine when we are dealing with short-term dependencies. On the other hand, LSTM networks have an edge over conventional feed-forward neural networks and RNN. This is because of their property of selectively remembering patterns for long duration of time.

3.2 Chosen language and Frameworks

The chosen programming language for implementation is Python as it easy to structure and read, and used in fast prototyping. Python is also versatile in a sense of scripting, Object-oriented programming and other myriads of libraries. Some of the Frameworks that will be used in the implementation:

- Data processing: pandas, numpy
- Models and ML functions: sklearn, keras, statsmodels, tensorflow
- Visualization: matplotlib, seaborn

3.3 Assessment metric

The Root-Mean-Square-Error (RMSE) is a measure frequently used for assessing the accuracy of prediction obtained by a model. It measures the differences of residuals between actual and predicted values. It penalizes large errors.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)}$$

Chapter 4

Work Done

4.1 ARIMA

The parameters of the ARIMA model are defined as follows:
p: The number of lag observations included in the model, also called the lag order.
d: The number of times that the raw observations are differenced, also called the degree of differencing.
q: The size of the moving average window, also called the order of moving average.

I created an ARIMA model using the statsmodels python library as follows:

- Define the model by calling ARIMA() and passing in the p, d, and q parameters.
- The model is prepared on the training data by calling the fit() function.
- Predictions can be made by calling the predict() function and specifying the index of the time or times to be predicted.

My approach is using the training set predict one step, then add the predicted value to the training set and use it for the next forecasting. Since the dataset is huge in size to automate this procedure, I used small size for ARIMA prediction.
The **RMSE** score just for a few dataset is **2.06** with this approach

4.2 VAR

A VAR model can be created using the statsmodels python library as follows:

- Define the model by calling VAR() and passing endogeneous(dependent) variables.
- The model is prepared on the training data by calling the fit() function.
- Predictions can be made by calling the predict() function.

4.3 LSTM on Weather dataset

First I tried to implement single step prediction for the temperature time series I extracted from the weather dataset and I found the LSTM got to the actual value in short amount of time.

Multistep prediction on weather dataset

```
rmse value for p (mbar) is :  11.804369955692533
rmse value for T (degC) is :  9.428597650368753
rmse value for Tpot (K) is :  10.069102879033311
rmse value for Tdew (degC) is :  6.967394342566237
rmse value for rh (%) is :  15.566439731262154
rmse value for VPmax (mbar) is :  7.95417083904802
rmse value for VPact (mbar) is :  4.256171595979834
rmse value for VPdef (mbar) is :  3.986428919072173
rmse value for sh (g/kg) is :  2.7190651787947497
rmse value for H2OC (mmol/mol) is :  4.338279497562126
rmse value for rho (g/m**3) is :  53.57435347895098
rmse value for wv (m/s) is :  1.329320635211837
rmse value for max. wv (m/s) is :  2.09695155591035
rmse value for wd (deg) is :  72.36201855498376
```

Figure 4.1: RMSE of VAR for weather dataset

```
rmse value for pollution is :  93.62012501029305
rmse value for dew is :  13.08229187484093
rmse value for temp is :  11.01007224996844
rmse value for press is :  9.25939543046966
rmse value for wnd_spd is :  44.85073038794289
rmse value for snow is :  0.7045787903337586
rmse value for rain is :  1.0703218724022083
```

Figure 4.2: RMSE of VAR for pollution dataset

4.4 LSTM on Pollution dataset

For forecasting on this dataset I created a function ('series_to_supervised') to change the problem into a supervised learning problem, by shifting each features one step further. So we can use the features' t-1 value to predict their values at time t. For LSTM prediction I only used the first feature ('pollution') as an output but the function can create any output.

4.5 LSTM for Chaotic time series

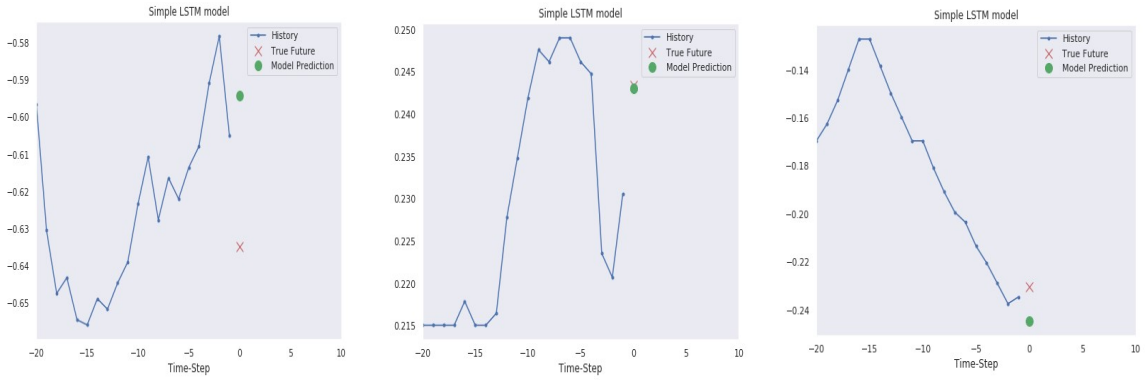

Figure 4.3: LSTM Single step prediction on temperature time series

Figure 4.4: LSTM single step prediction loss on weather dataset

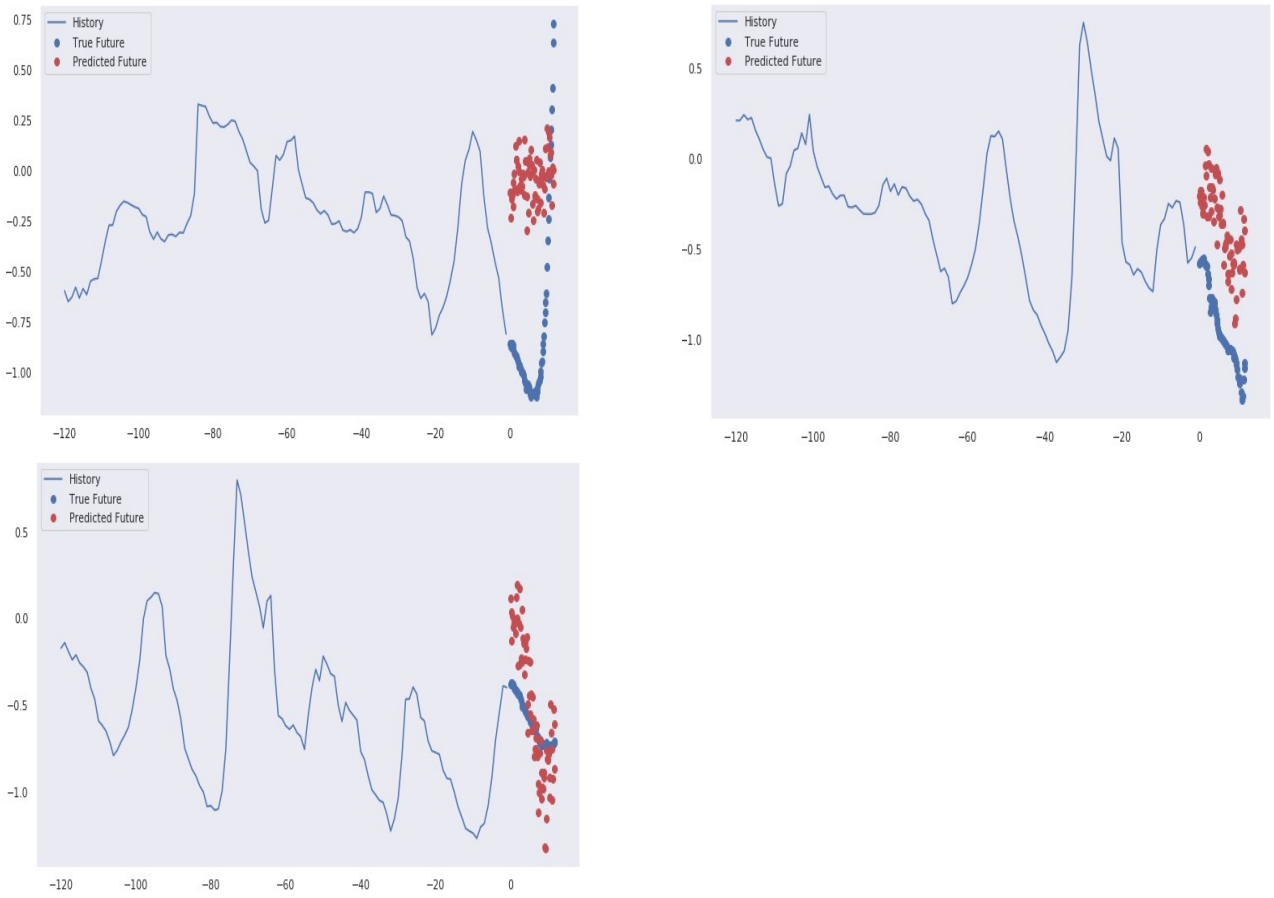

Figure 4.5: LSTM multi step prediction on temperature time series

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var7(t-1)	var8(t-1)	var1(t)
1	0.129779	0.352941	0.245902	0.527273	0.666667	0.002290	0.000000	0.0	0.148893
2	0.148893	0.367647	0.245902	0.527273	0.666667	0.003811	0.000000	0.0	0.159960
3	0.159960	0.426471	0.229508	0.545454	0.666667	0.005332	0.000000	0.0	0.182093
4	0.182093	0.485294	0.229508	0.563637	0.666667	0.008391	0.037037	0.0	0.138833
5	0.138833	0.485294	0.229508	0.563637	0.666667	0.009912	0.074074	0.0	0.109658

Figure 4.6: The reframed pollution dataset

Figure 4.7: Single step loss for Lorenz time series

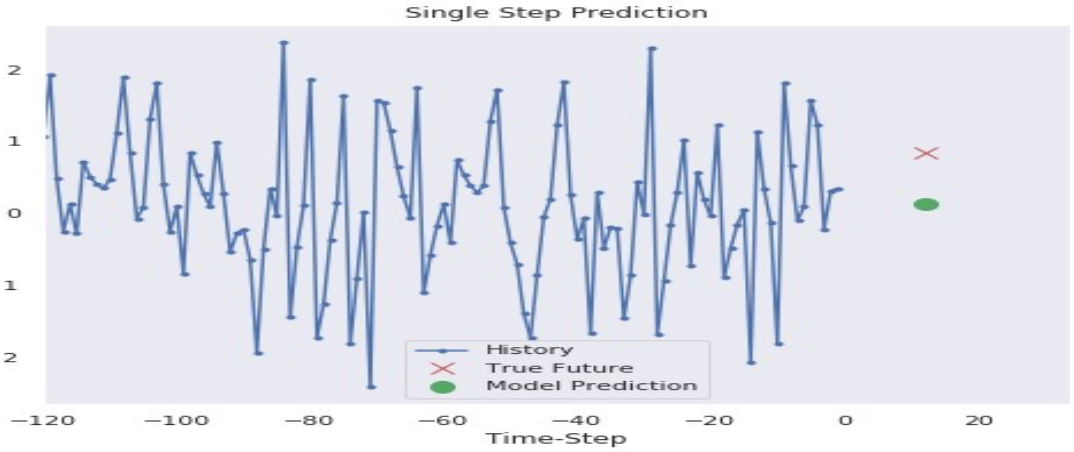

Figure 4.8: Lorenz single step prediction using LSTM

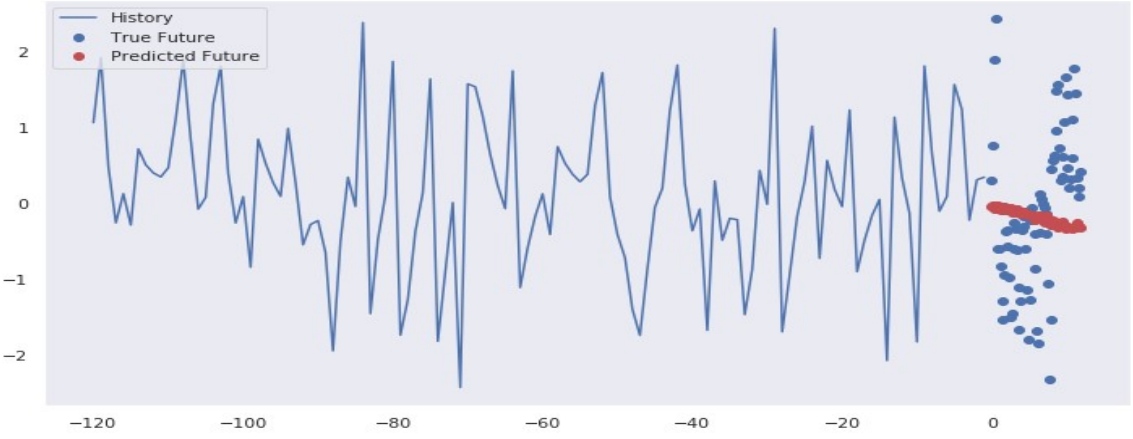

Figure 4.9: Lorenz multistep prediction using LSTM

Chapter 5

Conclusion

ARIMA and LSTM are both good tools in time series prediction, but predicting time series itself is a hard problem. It has a lot of circumstances behind time series, situations may happen where it is out of the model's control.

ARIMA only perform well on stationary time series. Arima requires a series of parameter (p,q,d) which must be calculated based on data, while LSTM does not require setting such parameter. Trying to automate of finding the parameters (p,d,q) takes quite sometime than LSTM. I tried to automate the process of evaluating a large number of hyperparameters for the ARIMA model by using a grid search procedure, but finding optimal hyperparameters took me time considering the size of the data is large. But LSTM can model non-linear function with neural networks in quite short amount of time. From the simulation I carried out LSTM sometime performs well and sometime behaves badly. I believe the sweet spot for using machine learning time series is where classical methods fall down. This maybe with complex univariate time series, if there is nonlinearity in the time series. Otherwise the classical methods ARIMA and VAR both perform well.

Acknowledgments

I thank Professor Pasquale Caianiello for providing me the opportunity to do this project. I owe my deep gratitude to my friends who were very helpful in explaining a bit complicated concepts.

Dagmawi Abraham Seifu

February, 2020
University of L'Aquila

References

https://datasciencetips.com/time-series-as-supervised-learning/
https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.var_model.VAR.html
https://keras.io/layers/recurrent/
https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduct