Foreign Exchange Rates forcasting with LSTM

Huy Phung, Tashi Choden, Sahil Pasricha
University of Konstanz

February 2019

Contents

1 Abstract

Foreign Exchange (abbreviation *Forex* or simply *FX*) Market is the decentralized market for currency investment. Forex market is the second most important market, after stock market. Supply and demand in the market determine Forex rate, in which a pair of currency can be exchanged. Forex Rates has been studied in econometrics as a financial timeseries. The purpose of studying Forex rates is to explain the market behaviour or forecast future prices.
In our project, we use statistical models and deep learning model to predict the future rates of one step ahead. Our goal is to compare the effectiveness of LSTM and statistical models (ARIMA and VAR) as timeseries models, in both accuracy and performance.

2 Problem Description

2.1 Forex rates

Foreign Exchange rates (short Forex rates) are decided solely by support and demand of Forex market. Each rate represents the price to buy or sell a pair of currency (e.g. EURUSD) at the moment. The price to buy is called Bid price; the price to sell is called Ask price. The difference between Bid price and Ask price is called Spread. In this project we consider only the Bid price. However, if both Bid and Ask (and therefore Spread) were available, our analysis would be more precise.
Forex brokers update rates according to the market within milliseconds by standardized FIX (Financial Information eXchange) protocol. The time interval between FIX market update messages are not uniform; it may varies from a millisecond to few seconds. Therefore, the timeseries is of continuous time step. In order to simplify our analysis, we convert it to a data form that has discrete, uniform time step, while still keep important information. One possible way to do so is to format the rates into OHLC format. This approach is widely used in financial technical analysis. We partition the timeseries into intervals of uniform time length t. For each interval, we keep only 4 rate: the first (*open*), the last (*close*), the maximum (*high*), the minimum (*low*). Since the time intervals are uniform among dataset, we have the desired discrete, uniform time step for analysis.

Figure 1:

2.2 Prediction

In this project, we concern about the prediction of future Open, High, Low, Close prices. Other features, either originally exists (volume) or later added (mean, median, momentum), are only considered as supporting features. These features are only used for prediction of OHLC features.
The problem we are trying to solve in this project is declared as follow: given history data in OHLC form of Forex rates, namely $x_0, x_1, \ldots, x_k$ where $x_i = (x_i^O, x_i^H, x_i^L, x_i^C)$, predict future rate of *one* step ahead, x_{k+1}.

2.3 Dataset

Acquiring real-time data is expensive, due to the fact that most FIX data providers requires subscription contract. However, Janus [2] collected an EURUSD rate dataset. The dataset consists of OHLC of BID price (no ASK price) of EURUSD rates from 2010 to 2016, thus contains 245444 values. Time interval for OHLC value is uniformly set to 15 minutes. Janus also published a smaller sample subset of the dataset, which contains only 14880 values. We would use the sample dataset later for ARIMA parameter search to reduce computational effort.

2.4 Model selection

2.4.1 Akaike Information Criterion (AIC)

Akaike Information Criterion (AIC) is basically log-likelihood, but it penalizes a model by the number of parameters. AIC is widely used in statistical model selection, not only ARIMA and VAR, but also Hidden Markov Model

and so on.

$$AIC = 2k - 2\ln(\hat{L})$$

in which k is the number of parameters and $\hat{L}$ is the likelihood. Since the log-likelihood is multiplied by -1, lower AIC means the model fits better to the data.

2.4.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error is widely used to measure the difference between values predicted by a model and the actually observed values. Given y represents the actually observed values and $\hat{y}$ represents the values predicted, $RMSE$ is given by:

$$RMSE = \left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)^{\frac{1}{2}}$$

Root Mean Squared Error shows difference between y and $\hat{y}$ regardless the difference is negative or positive. However, since it does not take the range of possible values into account, it would be difficult to intepret the $RMSE$ result without knowing the possible range of predicted and actual values.

2.4.3 Mean Absolute Percentage Error (MAPE)

In order to measure the difference between predicted values and actual values with regarding to the scale, we use Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Compare to RMSE, MAPE is easier to interpret, for it is only a ratio without unit. Knowing the Max-Min range of the data beforehand is not necessary. The drawback could be the absolute value function, which is not continuous and thus make it difficult to take it as a loss function. However, we do not use it as a loss function here.

3 Statistical Models

3.1 Basic of Time series.

Let $\{x_t\}$ is a timeseries with $\mu = \mu(t)$ is its mean and $\sigma = \sigma(t)$ is its variance. **Autocovariance** of lag k is defined as $C_k = E[(x_t - \mu)(x_{t+k} - \mu)]$

and **autocorrelation** is defined as $\rho_k = \frac{C_k}{\sigma^2}$.
A time series $\{e_t\}$ is a *discrete white noises* if its elements e_i are independent, identically distributed, have mean equals to zero and no autocorrelation between any of its values. Formally, $\mu_{\{e_t\}} = \mu_{\{e_t\}}(t) = 0$, $Cor(e_i, e_j) \neq 0, \forall i \neq j$.
A time series $\{x_t\}$ is a *random walk* if it satisfies that $x_t = x_{t-1} + e_t$ where $\{e_t\}$ is a discrete white noise as described above.
The following models we consider are similar to linear regression to some extents. Their rationales is that we find a linear relation between the value at time t and certain *lags* before it . Detailed explanation of the models can be found at [4].

3.2 ARIMA(p,d,q)

3.2.1 Model description

ARIMA(p,d,q) consists of three models: $AR(p)$, $MA(q)$ and integrated series of order d.
A timeseries $\{x_t\}$ is a *Auto-Regression* $AR(p)$ if it satisfies that

$$x_t = \alpha_1 x_{t-1} + \ldots + \alpha_{t-p} x_{t-p} + e_t$$

where $\{e_t\}$ is discrete white noises.
A timeseries $\{x_t\}$ is a *Moving Average* $AR(p)$ if it satisfies that

$$x_t = \alpha_1 e_{t-1} + \ldots + \alpha_{t-p} e_{t-p} + e_t$$

where $\{e_t\}$ is discrete white noises.

3.2.2 Parameters selection

It is important for ARIMA model that we select the proper parameter (p, d, q) so that it covers all the past values which has effect on the current value. In order to do so, we first look at survey the ACF and PACF plot. For $AR(p)$ model , we find the furthest lag with significant autocorrelation in ACF plot. For $MA(q)$, we find at which lag the autocorrelation start to decay.
Another method is that we try all possible combination of p, d and q to find the combination which gives us the lowest AIC. Algorithm for parameters selection:

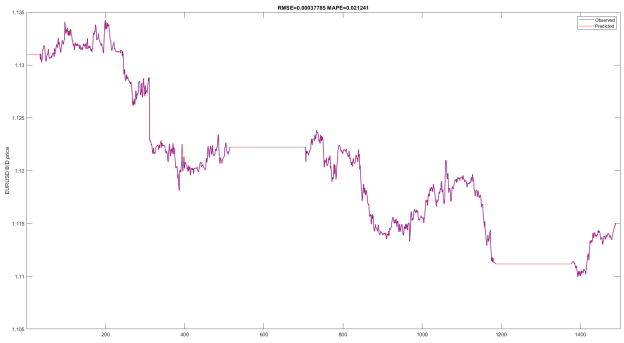

Figure 2:

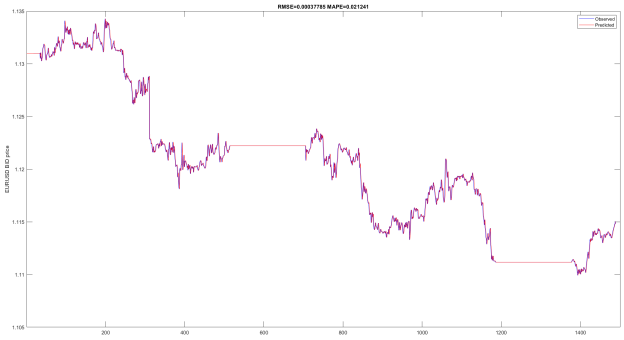

Figure 3:

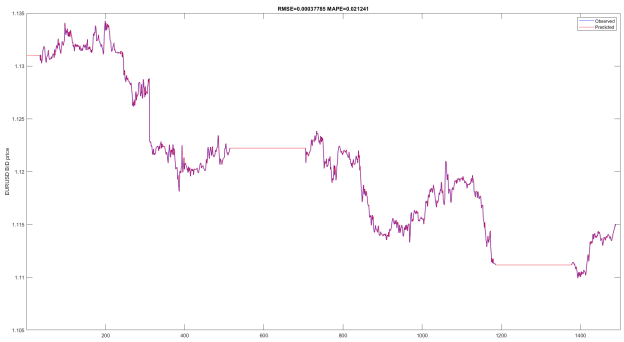

Figure 4:

3.2.3 Results

3.3 VAR(p)

3.3.1 Model description

VAR is applied to multivariate data. It is similar to $AR(p)$ model. However we apply the model for a vector $\curvearrowright_{\approx}$ of multivariate timeseries instead of using a single value.

3.3.2 Parameters selection

Parameters selection for VAR consists only testing a range of lag with AIC.

3.3.3 Results

4 Deep Learning Model

4.1 Recurrent Neural Network

Recurrent Neural Network (RNN) is introduced by [3] to process sequential input. In RNN, each state connects to the followed state to form a directed graph. The structure of RNN makes it capable of handling sequential data with temporal dynamic behaviour.

Challenges in Recurrent Neural Network are *vanishing gradient* and *exploding gradient*.

4.2 Long-Short Term Memory

Hochreiter and Schmidhuber (1997) [1] introduced Long-Short Term Memory neural network architecture architecture to solve both vanishing gradient and exploding gradient problem from RNN Cummins presented

4.3 Proposed Network Topology

Our survey on the data implies that significant autocorrelation may appears even further than 20 timesteps into the past. In our proposed network topology, we

add more hidden units into each LSTM layers, so that the network can learn from data points of higher lags.

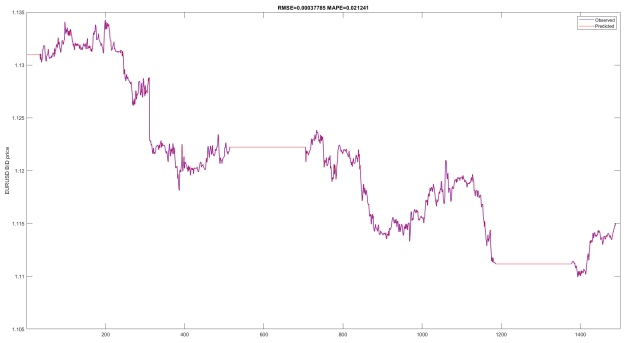

Figure 5:

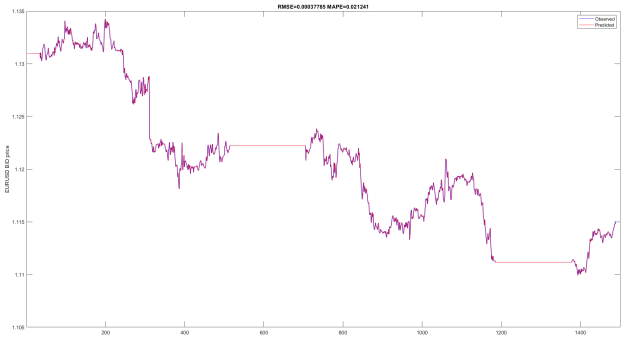

Figure 6:

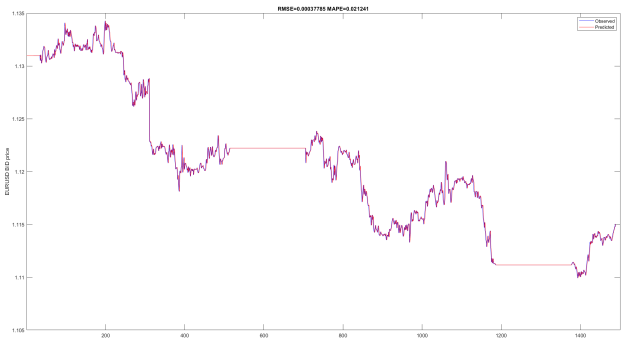

Figure 7:

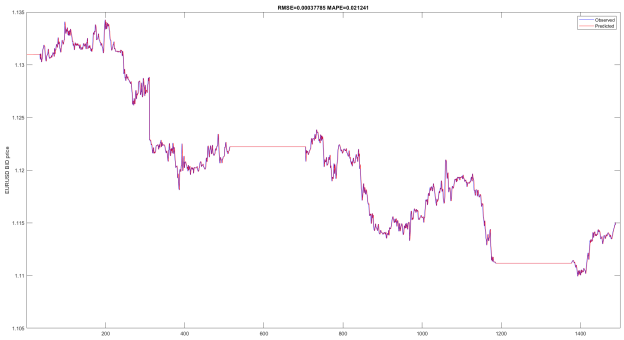

Figure 8:

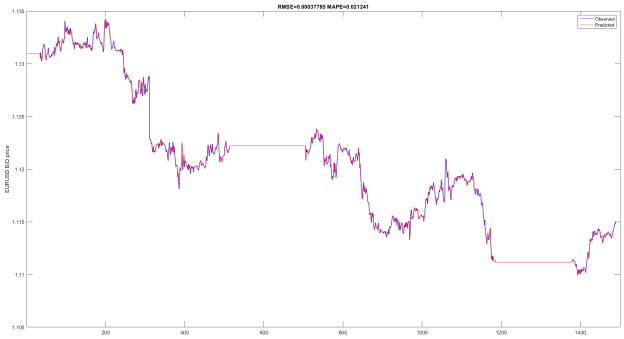

Figure 9:

4.4 Experiments

We consider using LSTM network in two experiments. In the first experiments, we build and train the network to predict future values only by giving to history lags of one timeseries *(univariate)*. For example, we train the network to predict one Open price value in the future given only the Open prices in the past. This experiments is designed to compare the performance of LSTM with $ARIMA(p, d, q)$ model.
In the second experiments, we build and train the network so that it would predict one values ahead of one feature (e.g. one value of Open price ahead), given history values of *all* features in the past. This experiment is designed to compare the performance of LSTM to $VAR(p)$ model.
In both experiments, since we want to predict the future values of all Open, High, Low and Close features, we have to build and train one mode for each feature accordingly. Therefore, in each experiment we have to build and train 4 models.

4.5 Training

As we can see from training progress, after the second epoch, the loss does not decrease any more, meanwhile the

4.6 Results

4.6.1 Single Variate

4.6.2 Multivariate

5 Implementation

6 Conclusion

As the results have shown, ...

However, computational effort spent for training LSTM deep learning model is much higher than ARIMA and VAR. Furthermore the, adding components to neural network topology results in higher computational cost.

References

[1] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. 1999.

[2] Michal Januszewski. *EURUSD-15m-2010-2016*. `https://www.kaggle.com/meehau/EURUSD/data`.

[3] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[4] Ruey S. Tsay. *Analysis of financial time series*. Wiley series in probability and statistics. Wiley-Interscience, Hoboken, NJ, 2. ed. edition, 2005.