Modeling the USD/MXN Exchange Rate using Neural Networks and Random Forest

Manuel Alejandro García Acosta

University of Pittsburgh

mag385@pitt.edu

November 15, 2020

Motivation

- Forecasting the evolution of exchange rates is important for people, companies and governments as exchange rates influence several macroeconomic variables such as volume of imported and exported goods, governmental budget (for countries with oil reserves), and remittances.
- It also affects decisions made in economic and foreign politics.

Motivation

- 'Traditional statistical methods, used by economists in the past years, seem to fail to capture the discontinuities, the nonlinearities and the high complexity of financial time series' (Theofilatos, 2012).
- The authors state that Machine learning techniques such as Artificial Neural Networks (ANN), Random Forest (RF) or Support Vector Machines (SVM) provide more capacity to capture the non-linear models dominant in the financial markets.

Who has done it

What I want to do

- Investigate the performance of machine learning techniques in forecasting the USD/MXN exchange rate using data from the Bank of Mexico (BANXICO).
- To do the above, I trained ANN, Random Forest and ARIMA models to compare their forecasting accuracy using the mean absolute percentage error (MAPE).

Artificial Neural Networks

ANN are methods based on mathematical models of the brain.

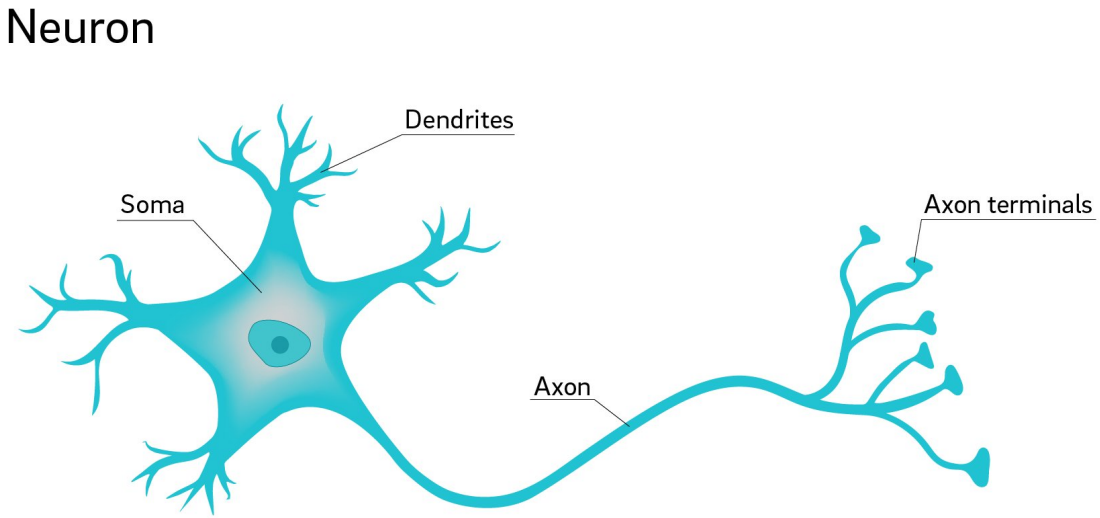

Figure: A neuron.

Single-layered ANNs

A simple neural network contains no hidden layers and only has as many nodes as its outputs and inputs.

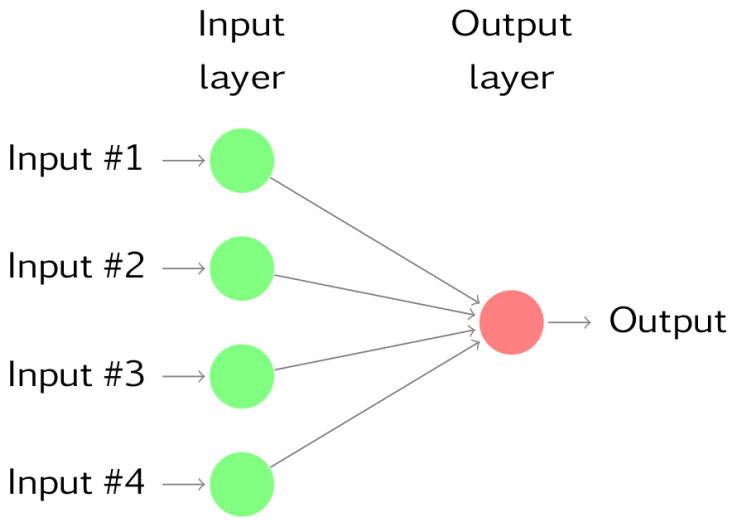

Figure: Neural network with no hidden layers (Hyndman & Athanasopoulos, 2018).

Single-layered ANNs

We can write this network as

$$\hat{y}_i = b_0 + \sum_{k=1}^{4} w_k x_{ik}, \quad x_i \in \mathbb{R}^4 \tag{1}$$

where w_k are the weights, $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4})$ and b_0 is a term called bias (the equivalent of an intercept in a linear model).

In an autoregression setting, (1) turns into

$$\hat{y}_t = b_0 + \sum_{k=1}^{4} w_k y_{t-k}\,. \tag{2}$$

The terms y_{t-k} are commonly known as “lagged entries”.

Multi-layered ANNs

We can add more layers of nodes between the input and the output, which will be called “hidden layers”. These new layers will allow the neural network to work with non-linear functions.

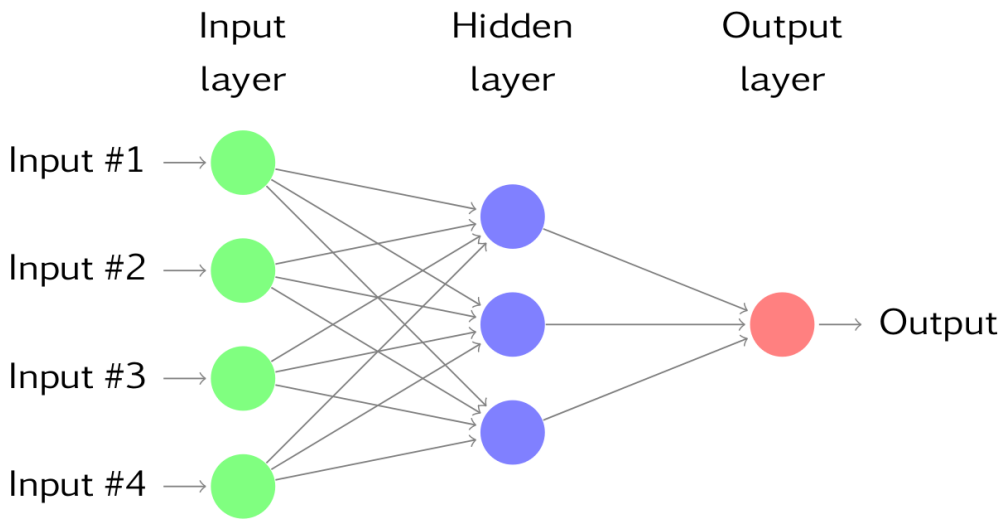

Figure: Neural network with one hidden layer (Hyndman & Athanasopoulos, 2018).

Multi-layered ANNs

The output of the j-th hidden node/neuron can be written as

$$z_{ij} = b_j + \sum_{k=1}^{4} w_{kj} x_{ik}, \quad j \in \{1, 2, 3\}\,. \tag{3}$$

In an autoregression setting, model (3) turns into

$$z_{tj} = b_j + \sum_{k=1}^{4} w_{kj} y_{t-k}\,. \tag{4}$$

Multi-layered ANNs

A function known as the "activation function", $\phi(z)$, will be applied to the nodes' output (z_{ij}).

$$\phi(z) = \frac{1}{1 + e^{-z}}. \tag{5}$$

A linear combination of $\phi(z_{ij})$ will be taken to produce the final output. In the autoregressive model, the output will look like

$$\hat{y}_t = a + \sum_{j=1}^{3} \left[h_j \phi \left(b_j + \sum_{k=1}^{4} w_{kj} y_{t-k} \right) \right]. \tag{6}$$

Data and procedure

- The data series employed was the USD/MXN FIX exchange rate from January 2017 to January 2020, there are 260 observations per year.
- I used the USD/MXN dataset as follows; I used the data from January 2017 to December 2019 as the *training* set and the observations for January 2020 as the *test* set. The training dataset has 782 observations, and the test dataset has 23 observations.

Data and procedure

Performance of the models was measured using the Mean Absolute Percentage Error (MAPE). The percentage errors are defined as

$$p_t = \frac{100 e_t}{y_t},$$

where $e_t = y_t - \hat{y}_t$. The MAPE is defined as

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} |p_t|. \tag{7}$$

Results

The ARIMA forecast obtained a $MAPE$ error of 0.3819605, which is higher than the others. I conclude that this technique was weaker in a simple application, it would be necessary to analyze and transform the series in detail to achieve a better forecast.

Results

For the ANN model I used the nnetar() function from the **forecast** package in R. This function fits a neural network with a given number of lagged entries (p) and one hidden layer with a given number ($size$) of nodes. Due to computational limitations, I trained neural networks with $p \in \{1, \ldots, 20\}$ lagged entries and size $m \in \{1, \ldots, 2p\}$.
From all the trained neural networks, the one with the minimum MAPE used 5 lagged entries and had an architecture of 9 nodes in the hidden layer. The MAPE was $MAPE_{ANN} = 0.3135339$.

Results

The ‘out of the bag’ random forest (with 5 lagged entries) performed surprisingly (or not) well. The MAPE for Random Forest with 5 lagged entries was $MAPE_{OOB} = 0.3506094$. After tuning p, the number of lagged entries, I got that the best p was 12 with corresponding error $MAPE_{RF} = 0.3030751$. The number of variables randomly sampled as candidates at each split, or $mtry$, was left as $p/3$. Further tuning of $mtry$ is expected to improve the performance of the random forest.

Artículos


H. Fujita, On some nonexistence and nonuniqueness theorems of non linear parabolic equations, en: Nonlinear Functional Analysis, Providence, R.I.,1970, Proc. Sympos. Pure Math. 18(1) (1968) 105-113.


M. Dozzi, J.A. López-Mimbela, Finite time blowup and existence of global positive solutions of a semi-linear SPDE, Stochastic Process. Appl. 120(6)(2010)767-776.


M. Niu, B. Xie, Impacts of Gaussian noises on the blow-up times of nonlinear stochastic partial differential equations, Nonlinear Analysis: Real World Applications 13(3)(2012), 1346-1352.