Time Series Forecasting Using Neural Networks

Bruno Soalheira
bruno.soalheira@tecnico.ulisboa.pt

Instituto Superior Técnico

October 2019

Artificial neural networks are a powerful machine learning technique that is used for several ends; time series forecasting is one of them. Several of these techniques are explored with special attention to recurrent neural networks, which are one of the methods most commonly used in time series forecasting. All these methods are presented with the goal of better understanding recurrent neural networks. There is no predefined better algorithm to solve this type of problems; in this field experimentation is required and only then we can draw conclusions.

In this case, feedforward neural networks and recurrent neural networks will be used to obtain models capable of predicting time series. Two related data sets but with some important differences that will affect the results are used in these experimentations. The main objective is to compare the two types of networks on both data sets, presenting the limitations, drawbacks and advantages of each one of them.

In most cases, recurrent neural networks outperform feedforward neural networks for time series forecasting problems.

Keywords: machine learning; recurrent neural networks; feedforward neural networks; time series forecasting.

1. Introduction

1.1 Motivation

Time series forecasting is an area of machine learning that has been fascinating data scientists for the last few decades. The field of machine learning has brought us innumerous advantages, being one of them the capability of being able to predict what is going to happen and the only thing we need is information. No matter how hard and random a data set may seem, these algorithms are capable of finding patterns that are not possible to identify using other techniques.

Neural networks are an amazing method using for different purposes; time series forecasting is only one of them. They can also be used in many different other areas like in customer research, data validation and risk management.

2. Related Work

2.1 Time Series

Characterized as a set of data points obtained from existing records, usually a sequence of equally spaced points in time, tabulated or plotted in time order. In a more mathematical way, using the definition suggested by Robert H. Shumway and David S. Stoffer in *Time Series Analysis and Its Applications* [1] *"…*we may consider a time series as a sequence of random variables $x1, x2, x3, \ldots$, where the variable $x1$ denotes the value taken by the series at the first time point, the variable $x2$ denotes the value for the second time period, $x3$ denotes the value for the third time period, and so on."

2.2 Time Series Forecasting

Mainly, it consists of predicting future events by applying models to time series. It is an extremely important area of machine learning because there are several prediction problems involving time components. Every time series problem is very specific and with its own data characteristics, and according to that the most fitting model(s) must be chosen to analyze the data.

2.3 Neural Networks

2.3.1 Biology

The brain consists of a large number of connected elements known as neurons. Considering, for our own convenience, that neurons have four principal components: the dendrites, the axon, the cell body and the synapses. The dendrites is short and branched and is an extension of the nerve cell, along which impulses received from other cells at synapses are transmitted to the cell body. The axons are the long threadlike parts of the nerve cells, along which impulses are conducted from the cell body to other cells. The cell body is the spherical part of the neuron that contains the nucleus and is connected to both the axon and the dendrites. The junction or point of connection between two nerve cells, more specifically the connection between the axon of one cell and the dendrites of the next cell is called synapse.

2.3.2 Definition and Structure

An artificial neural network, commonly referred as simply neural network is a type of machine learning, just like all the methods presented in the last chapter. In its most general form, a neural network is a machine that is designed to model the way in which the brain preforms a particular task; the network is usually implemented by using electronic components or is simulated in software on a digital computer.

It is helpful to know the structure of a neural network in order to better understand it. Neural networks are, usually, composed by the following elements:

- *Input layer.* The input layer is the very beginning of the workflow for artificial neural networks. It consists of artificial input neurons and brings the initial data into the system for further processing by subsequent layers of artificial neurons.
- *Hidden layer.* The hidden layer is located between the input layer and output layer, where artificial neurons take in a set of weighted inputs and produce an output through the use of an activation function.
- *Output layer.* The output layer is the last layer of a neural network that produces the outputs for the program.
- *Synapse.* The synapse is the strength of the connection between two artificial neurons.

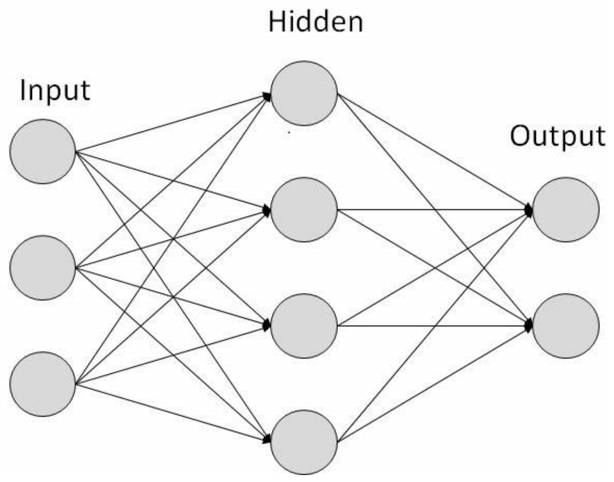


Figure 1 A Typical Neural Network

2.3.3 Learning Methods

Neural networks have a learning phase which is crucial for the potential results. There are three methods for learning strategies:

- *Supervised learning*. It is the machine learning task of learning a function that maps an input to output based on example input-output pairs. The example input-output pairs are usually called training data and each of these consists of an input object (usually a vector) and an output value. The new inputs that will be tested are called test data. This algorithm analyses the training data and infer a function that will be used to calculate the output values for the new test data values [2][3].

- *Unsupervised Learning.* This method has to be used when there is no training data to learn from. The neural network analyses the data, and then a cost function tells the neural network how far it was from the correct result. Basically, the model learns from the test data and keeps adapting according to this data.
- *Reinforced Learning.* This algorithm reinforces the neural network in the presence of positive results and punishes it for negative results, forcing the neural network to learn over time.

2.3.4 Activation Function

In neural networks, an activation function is what defines the output of a node given an input or set of inputs. This output is used as input of the next node and so on until a desired solution to the original problem is found. It maps the resulting values into a desired range (0 to 1 or -1 to 1, for instance) [4].

There are two types of activation functions:

- *Linear Activation Function.* The function is linear. Therefore, the output of the functions will not be confined between any range. These functions don't really help with the usual problems and data that we analyze using neural networks.
- *Non-Linear Activation Function.* The function is not linear. This type of functions is commonly used because it can adapt to complex types of data. For example, we have the sigmoid activation function, which is used to obtain values between 0 and 1 (commonly used in probability problems), we have tanh activation function, which provides values ranging from -1 to 1 (commonly used for classification between two classes), among many others.

2.3.5 Backpropagation

In the beginning, the weights of a neural network are initialized randomly, and these values need to be adjusted until we obtain the values that fit our model the best. One of the most common ways to accomplish this is using the backpropagation technique. It is a method used to calculate a gradient that is needed in the calculation of the weights used in a neural network. It is more used in situations where there is more than a hidden layer in a neural network [5].

Backpropagation requires three things to work properly:

- *A dataset* consisting of input-output pairs.
- *A feedforward neural network*, meaning that there are no connections between nodes in the same layer and that layers are fully connected.
- *An error function,* which defines the error between the desired output and the obtained output.

This method is used in any feedforward networks to learn a training set of inputs and outputs.

The main idea behind backpropagation is to update the weights' values in such a way that the error becomes minimum. First, we figure out if we have whether to increase or decrease the values of the weights. Once we know which way to go, we keep making smaller and smaller adjustments until the network provides the desired results.

2.3.6 Recurrent Neural Networks

Recurrent Neural Networks have been deeply studied since the 1990's. They are commonly used to learn sequential or time varying patterns [6].
There are two types of architectures of recurrent neural networks.

The first ones are named fully connected neural networks. Fully connected networks do not have distinct input layers of nodes, and each node has input from all the other nodes.

The second ones are named partially connected neural networks. Although some nodes are part of a feedforward structure, other nodes provide the sequential context and receive information from other nodes.

An Elman network contains three layers and a set of context units. The context units are connected to the hidden layer with the fixed weight of one. At each time step, the input is fed-forward and a learning rule is applied. The context units save a copy of the previous values of the hidden layer nodes with the goal of maintaining a sort of a state, allowing this model to perform tasks like sequence-prediction [7][8].

Let h_t be the hidden layer vector, y_t the output vector, x_t the input vector, W, U and b the parameter matrices and vector and σ_h and σ_y the activation functions. The hidden layer and the output vector are calculated the following way:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{1}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{2}$$

Jordan networks are similar to Elman networks with the small difference of the context units containing information about the output layer instead of the hidden layer. Considering the same variable from the previous two formulas, we can obtain the value for the hidden layer and the output vector for the Jordan networks:

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \tag{3}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{4}$$

2.3.7 Advantages of Neural Networks

Artificial neural networks bring some benefits that overcome some of the limitations referred in the algorithms mentioned in the previous chapter. It is apparent that a neural network derives its computing power through its massively parallel distributed structure and its ability to learn and therefore generalize. Generalization consists on the process of the neural network obtaining reasonable results for inputs that were not part of the training [9].

The following points are just some of the benefits and capabilities that neural networks offer us:

- *Nonlinearity.* Artificial neurons can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear. This is an important characteristic because it allows us to deal with a higher variety of problems, particularly nonlinear problems.
- *Input-Output Mapping.* Supervised learning is one of the methods to train neural networks. Each data sample in the training data consists of an input and a desired output. The network is trained by receiving these data samples randomly and then the network's synaptic weights keep being modified in order to minimize the network's result and the desired result. Providing the data samples in a different order also helps training the network and the train keeps going until the network reaches a steady state, where there are no significant changes in the synaptic weights. Thus, the neural network learns from the data contained in the training set by constructing an input-output mapping for the problem at hand.
- *Adaptivity.* Neural networks have the capability of changing synaptic weights according to the goals we have in hands. In some cases, when a neural network is trained to operate in a specific environment it can easily be retrained to deal with minor changes. Moreover, when a neural network is operating in a nonstationary environment it should be trained to deal with constant changes and the synaptic weights must adapt in real time. The ability to adapt a neural network according to the situation we

have in hands is indeed a huge advantage, but this will affect the robustness of the same network. As the robustness level increases the adaptive level tends to decrease and vice-versa.

- *Evidential Response.* In the context of pattern classification, a neural network usually provides information about which pattern to select and also the confidence in the decision made. The confidence can be used to reject ambiguous patterns and thereby improve the classification performance of the network.
- *Contextual Information.* Knowledge is represented by the very structure and activation state of a neural network. Every neuron is potentially affected by the global activity of the other neurons in the network. Consequently, neural networks deal with contextual information very naturally.
- *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently fault tolerant. In other words, in the presence of adverse operating conditions the neural network's performance will degrade gracefully.
- *Uniformity of Analysis and Design.* The same notation is used in different problems and environments where neural networks are used.
- *Neurobiological Analogy.* As previously mentioned, neural networks are inspired by the analogy with the brain. Neurobiologists look to artificial networks as an interpretation tool to better understand neurobiological phenomena. On the other hand, we have engineers that try to develop more complex models using neurobiology as inspiration.

2.3.8 Applications Neural Networks

Neural networks have several applications. Some applications improved its results thanks to neural networks and I'm going to mention two of the most relevant:

- *Image Processing and Character Recognition.* Neural network's capability of receiving several inputs, process them and deal with non-linear relationships is playing a big role in image processing and character recognition. Character recognition is used in several problems like automatic number plates recognition, in airports, for passport recognition and information extraction and converting handwriting in real time to control a computer, for instance. Image processing and recognition are used in facial recognition, cancer detection and satellite imagery for agricultural and defense usage.
- *Forecasting.* This is an extremely important application of neural networks and is used in many different contexts. For instance, neural networks are used in weather forecasting, earthquake prediction, mathematical finance, astronomy, among others. All these problems are extremely complex and that's why they rely on neural networks.

3. Thesis Plan

3.1 Introduction and plans

For this project, I always had in mind working with data sets that were related with a big issue that is becoming more and more evident as time goes by; the climate change.

I started researching and looking for data sets that looked interesting to work with and the initial plan was to work with a multivariate data set that aimed to predict if one day is either going to be an ozone day (meaning the ozone level is above a certain predefined threshold) or a normal day.

I ended up deviating from the initial plan because I decided that I wanted to see the differences of how feedforward neural networks and recurrent neural networks deal with seasonal and non-seasonal data sets. I had in mind that feedforward neural networks would perform well on seasonal data sets because there is no need of memory since the pattern is very clear.

Still with the climate change subject in mind, I found two interesting related univariate data sets. The first one was the seasonal data set and it contained information regarding the mean monthly global temperatures for a period of 65 years over the twentieth century. The second one was the non-seasonal data set and it contained information regarding the yearly number of sunspots over a period of 314 years; a factor that greatly affects the temperatures and one that we can't control.

Basically, I want to predict both data sets using feedforward neural networks and recurrent neural networks and then compare the performance of each of the algorithms. During the recurrent neural networks experimentations I want to compare Elman networks and Jordan network and check how impactful their small differences in structure affect the results.

3.2 Technologies used

At first I wanted to develop this project in Python, which is a language that I am comfortable with and it is commonly used to solve this types of problems. After a discussion with my supervisor, professor Andreas Wichert, who recommended me to use R, I decided to change and use this R language because it is a language more directed to implement machine learning techniques than Python. I was not as comfortable with R as I was with Python but the learning process was fine; it didn't take me a long time to get used to the language. I was also familiar with the language from the SAD course (Sistemas de Apoio à Decisão) where I used this language to perform similar activities like data pre-processing and application of several data mining algorithms like SVM (Support Vector Machine), kNN (k-Nearest Neighbours), LVQ (Learning Vector Quantization), Decision Trees and even Neural Networks.

For IDE I chose R Studio, which I had some familiarity with it because I used it during the SAD course.

3.3 Evaluation Method

As mentioned the before, I plan on comparing the performance of both recurrent neural networks and feedforward neural networks and to accomplish this I will use measures of accuracy. The main measure used will be RMSE (root-mean-squared-error) and, when needed, I will recur to other measures to dissipate the doubts.

The evaluation process will consist of dividing the data sets in two parts: a training data set and a test data set. The training data set will be used to train our neural networks. Then we will compare the predictions from our models against the test data sets. For this, we will compare the results graphically and numerically using the previously mentioned measurement of accuracy and others if needed.

4. Comparing feedforward neural networks with recurrent neural networks

4.1 Choosing the data sets

To start off with my experiments, I decided to pick two related data sets; a simpler one where the seasonality is very clear and there are not many deviations from the regular values and a more complex one where there are clearly more deviations from the pattern. Both data sets were obtained from datamarket.com and the first one is called "mean-monthly-temperature-1907-1972" and it contains 792 observations of the mean monthly global temperatures in Fahrenheit starting on January, 1907 and ending on December, 1972. The second data set is called "yearly-mean-total-sunspot-number" and it contains 315 observations of the mean yearly number of sunspots starting on 1700 and ending on 2014. I decided to choose these two data sets because there is a clear relation between them as I explained in the previous chapter. The number of sunspots is a huge factor on the temperatures around the planet.

4.2 Pre-processing

It is important to mention this process since it can greatly affect the results. If we don't perform a correct pre-processing of the data it will negatively affect the results. This process is a data mining technique that involves transforming raw data into an understandable format that can be used for several purposes, including analysis. Real world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends; therefore it is likely to contain many errors. Some of the steps of data pre-processing include checking out for missing values, values normalization/ standardization, checking out the consistency of the data types, splitting the data into training and test data sets, among others.

4.3 Experimentations introduction and notes

The results I'm about to present are in compliance with what I expected in the beginning with exception of one or two points that I will mention later on.

Before I present the results and conclusions I obtained, it is important to mention an important fact about the experimentations on this project; it is known that neural networks have hidden layers and these layers will be the main concern during our experimentations. There are no rules for choosing the number of hidden layers and hidden nodes and the only way to figure them out is by experimenting several values for them and analyzing if the results are improving or not. By default, the nnetar function, has a number of hidden layers of one (which cannot be changed) and a number of hidden nodes predefined as well. This predefined number of hidden nodes will be my starting point and I will proceed by testing with smaller and higher number of nodes in order to figure out which is the best number to obtain the best results possible. For the Elman and Jordan networks the process will be similar but I can also choose the number of hidden layers of the network.

4.4 Experimentations conclusion

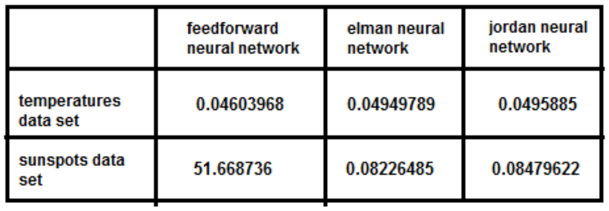
From these experimentations, I can conclude that for the seasonal dataset the feedforward neural network performed sufficiently well. On the other hand, for the non-seasonal dataset the results were not even close to satisfactory. Using the recurrent neural networks, either Jordan network or Elman network, the results were positive for both the seasonal and non-seasonal dataset.

The main conclusion we can take is that recurrent neural networks are a better approach when we are dealing with time series.

5. Conclusion

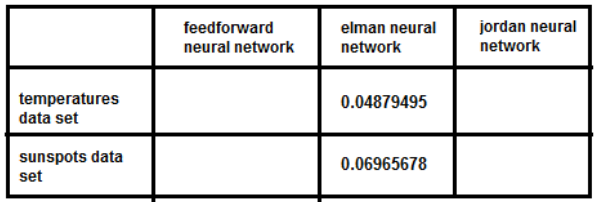
5.1 Results

To sum up the obtained results I present them in the following tables of Figures 2 and 3. In the first table we have the RMSE values corresponding to the networks with one hidden layer.



	feedforward neural network	elman neural network	jordan neural network
temperatures data set	0.04603968	0.04949789	0.0495885
sunspots data set	51.668736	0.08226485	0.08479622

Figure 2 Comparison between all the networks with one hidden layer



	feedforward neural network	elman neural network	jordan neural network
temperatures data set		0.04879495	
sunspots data set		0.06965678	

Figure 3 Results from the Elman network with two hidden layers

So, the main conclusion we can infer from these results are:

- Feedforward neural networks are worth taking into account when dealing with seasonal time series data sets; they might even outperform recurrent neural networks or other techniques.

- Elman neural networks slightly outperformed Jordan neural networks but the difference is not significant so it is worth taking both into account when dealing with these problems.
- The deeper the neural network is the better it will perform with complex problems. It is important to note that increasing layers will also increase the cost and the time consumed, and sometimes it is not worth because the results might not be that much better or even, in some cases, they might get worse.

5.2 Improvements

Even though we chose to work with neural networks, other algorithms (some of them described in Chapter 2) would be able to solve this type of problems and maybe with even better results. Regarding the neural networks, we could try different activation functions.

We used training and test data sets containing 80% and 20% of the data, respectively. We could improve the results by trying different sizes for these partitions. We could also use different parts of the data for these partitions; for example the training data set being the last years of the data set and the test data set being the beginning or even split the test data set and make part of the prediction in the beginning and part of the prediction in the end.

5.3 Difficulties

As previously mentioned, it took me some time to get used to the R language.

The pre-processing was also a part that was time consuming; it was hard to figure out some things like, for instance, that some variable were not defined as the type they should be and that I had to turn my vectors into time series objects.

During the model training part, the hardest parts were figuring out what were the best accuracy measures to compare the results and researching on the neural networks function and the meaning of their parameters.

Certainly, I faced more problems, but these are the ones that come to mind and the ones that I struggled the most with.

References

1. Shumway, R.H. and Stoffer, D.S. (2000). *Time Series Analysis and Its Applications*. Springer International Publishing.
2. Russel, S.J. and Norving, P (2010). *Artificial Intelligence: A Modern Approach.* Prentice Hall.
3. Geman, S.; Bienenstock, E. and Doursat, R. (1992). *Neural networks and the bias/variance dilemma.* Neural Computation.
4. Hinkelmann, K. *Neural Networks.* University of Applied Science Northwestern Switzerland.
5. Nielsen, M.A. (2015). Chapter 6. *Neural Networks and Deep Learning.*
6. Medsker, L.R. and Jain, L.C. (1999, Dec 20). *Recurrent Neural Networks: Design and Applications.* CRC Press.
7. Sak, H.; Senior, A. and Beaufays, F. (2014). *Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling.*
8. Milos, M. (2012). *Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction.* Indian Journal of Computer and Engineering.
9. Shmueli, G. and Lichtendahl Jr., K.C. (2015, Jul 17). *Pratical Time Series Forecasting with R: A Hands-On Guide.* Axelrod Schnall Publisher.