Modeling ANNs Performance on Time Series Forecasting

Ranyart R. Suárez
Division de Estudios de Posgrado
Facultad de Ingeniería Electrica
Universidad Michoacana de San Nicolas de Hidalgo
ranyart@dep.fie.umich.mx

Mario Graff
Division de Estudios de Posgrado
Facultad de Ingeniería Electrica
Universidad Michoacana de San Nicolas de Hidalgo
mgraffg@dep.fie.umich.mx

Abstract—Artificial Neural Networks (ANNs) are accurate models that can learn the characteristics of the problem being solved, mainly these are used for classification and regression problems. In this work, the performance of ANNs used in Time Series forecasting is modeled. Modeling the performance of ANNs is specially useful when the algorithm selection problem is being tackled. In order to model different ANNs, only changes in the training parameters are being considered. The influence of the training parameters on the performance is also determined in order to assure that different ANNs are being modeled.

I. INTRODUCTION

An Artificial Neural Network (ANN) is an accurate model mainly used for classification and regression problems. In machine learning, classification is the process of determining the class of an object from a set of categories. An ANN classifies objects based on their features or characteristics, which are presented to the ANN as inputs. Another main use of ANNs is regression problems. We face a regression problem when, given a set of independent variables, we determine the value of the dependent variable. One example of this problem is time series forecasting, which will be addressed in this work.

ANNs are methods that require a great user effort when used in real world applications. This is because their parameters have to be optimized, and this requires an experienced user. They also require higher computational resources than simpler techniques such as linear models, this is because ANNs have a large number of connections whose values have to be set appropriately.

Because ANNs are computationally expensive methods compared to other techniques, the possibility to know a priori how well an ANN will perform over a set of problems becomes a necessity. If a problem can be solved by simpler techniques than ANNs, there is no reason to use them. Therefore, modeling the performance of an ANN gives information that can be used when the user is facing the algorithm selection problem. The algorithm selection problem [1] consists in choosing the algorithm, over a collection of algorithms, that best solves a given problem. The trivial solution, for this problem, is to prove all the algorithms available over the problem and then choose the one that presents the best results. Clearly, this trivial solution would be very expensive, and there are cases, as the one presented in this contribution, where this solution is not even viable. For example, when the problem is to forecast or classify an unknown object, the trivial solution does not provide any useful information to make the selection. In a nutshell, to solve the algorithm selection problem, it is necessary to know a priori what would be performance of the algorithms over the given problem.

The rest of this paper is organized as follows: Section II presents the related work. Section III gives a brief introduction to ANNs. Section IV presents the procedure used to compute the importance of the training algorithm parameters. Section V explains the considerations taken in order to model the ANNs. Section VI presents the results, and finally the conclusions of this contribution are presented in Section VII.

II. RELATED WORK

Due to the fact that ANNs make no assumptions regarding the data presented to them, these are often used to model systems in many different fields: Medicine [2], Electronics [3], Environmental Engineering [4], Hydrologic Engineering [5]. It is also common to use ANN when traditional techniques fail to model complex relationships between data.

Despite that ANNs model different systems, in the literature, there are not many works focused in modeling the performance of ANNs, that is, modeling the behavior of the ANN itself. However, the problem of modelling the performance of forecasters have been address by different researchers, e.g. [6], [7], [8]. Most related to this contribution is [6] there the authors modeled the performance of three forecasters. We share the main objective with this work, i.e., to model the performance of time series forecasters in order to choose the most suitable method, among the available options, for the problem under analysis. Nonetheless, there are considerable differences; one is the forecaster being modelled, here we focus on ANNs whereas in [6] focused on linear forecasters such as ARIMA. Secondly, the structure of the model, in this contribution we decide to use Random Forest to create the model, meanwhile in [6] used a linear regression model.

III. ARTIFICIAL NEURAL NETWORKS

In general an ANNs are mathematical models inspired in the biological structure of animals’ brain cells. ANNs are called networks because it connects simple processing
elements together. The processing element acts like a real Neuron in the nervous system, although the complexity of real neurons is highly abstracted when modeling an ANN. A neuron receives an input (or inputs), processes it, and then delivers an output. Every neuron has a value, namely weight, that interacts with the associated input, the way that a neuron processes these inputs is given in Equation (1),

\[ S = \sum_{i}^{N} w_{i}x_{i}, \]  

where \( x_{i} \) is the \( i \)-th input and \( w_{i} \) is the weight value associated to that connection. The neurons are arranged in layers. One of the most popular ANN models is the 3-layer feed forward model, where neurons are organized in three layers and the information always goes forward. This model can be seen in Figure 1.

![Fig. 1. Three layer ANN](image)

These three layers are organized as follows: the input layer are the neurons that directly receive the inputs, the layer that returns the output of the ANN is called the output layer, and the layer that performs the inner calculations is called the hidden layer.

The number of neurons placed in the input layer and in the output layer are defined by the problem being solved. For example, in classification problems the number of input neurons is defined by the number of features, meanwhile the number of output neurons is related to the number of classes. On the other hand, the number of hidden neurons does depend on the problem being solved; nonetheless, there is no a proved procedure to set its value.

Before an ANN can be used, it has to be trained. The process of training an ANN is commonly referred as the training phase. The ANN’s weights are initialized with small random values typically in the range [0.01, 0.01]. The algorithm that optimize the weight values is called Training algorithm. In this work, we model the performance of two training algorithms: QUICKPROP [9] and RPROP [10] which are improvements of the traditional Back propagation algorithm [11].

IV. PARAMETER INFLUENCE AS A REGRESSION PROBLEM

The first step to model the performance of an ANN is to design the ANN being modeled. That is, to decide its parameters. Here, we would like to model ANNs that have differences in performance, so in order to obtain these, we decided to identify the ANN parameters that affect the most its performance. The other problem that arises in the case one decides to test a small number of different parameters is a combinatorial problem. For example, RPROP has five parameters, let us suppose that we test only four different value for each parameter, then the number of possible combinations is \( 5^4 = 625 \). This number might not seem too large but if we raise the possible values for the parameters this number grows exponentially. Then identifying the most important parameters allows us to explore more different values of these parameters.

This identification process can be solved as a regression problem as follows. The Regression problem consists in finding the relationship between variables, namely explanatory variables, and a response variable. Although this seems quite trivial, it is not. Most of the time, the relationship between the explanatory variables and response variable is not easy to find or easy to model accurately. In this contribution we decide to use Random Forests to solve the regression problem. Now in order to determine the parameter influence, the training-algorithm parameters are used as the explanatory variables and the error of the ANN after the training as the response variable. Finally, the regression algorithm used here sorts the explanatory variables according to their importance.

Random Forest [12] are an ensemble of tree predictors [13], where the generalization error depends on the strength of individual trees in the forest and the correlation between them. Random Forest have several desired qualities: they are robust to outliers and noise, give useful internal information like error, strength, correlation, and variable importance. The Random Forest characteristic that matter the most to this contribution is that it computes the importance of each explanatory variable.\(^1\)

To perform the regression, Random Forest needs two inputs: a matrix that contains the explanatory variables and a vector representing the response variable. Let us suppose that \( \theta \) is a particular combination of training parameters, the matrix used by Random Forests is given by Equation 2, where each row is a particular ANNs configuration, i.e. a set of parameters, and each column contains the different values a particular parameter could have. Each entry of the response vector is a particular ANNs configuration, i.e. a set of parameters, and each column contains the different values a particular parameter could have. Each entry of the response vector is given by the MSE value obtained after training the ANN with a combination of training parameters values.

\[
M = \begin{pmatrix}
\theta_{1}^{1} & \theta_{1}^{2} & \theta_{1}^{3} & \theta_{1}^{4} & \theta_{1}^{5} & \theta_{1}^{6} \\
\theta_{2}^{1} & \theta_{2}^{2} & \theta_{2}^{3} & \theta_{2}^{4} & \theta_{2}^{5} & \theta_{2}^{6} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\theta_{m}^{1} & \theta_{m}^{2} & \theta_{m}^{3} & \theta_{m}^{4} & \theta_{m}^{5} & \theta_{m}^{6}
\end{pmatrix}
\]  

\(^1\)This work uses the sklearn [14] implementation of Random Forest.
In this work the performance of a 3-layer feed forward network is modeled. Besides, this architecture of ANN is trained with 1 of 2 training algorithms: QUICKPROP and RPROP. These two algorithms represent improved versions of the traditional back propagation training algorithm and have more parameters that need to be optimized. Table I show the parameters of these training algorithms.

<table>
<thead>
<tr>
<th>QUICKPROP</th>
<th>RPROP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning factor</td>
<td>Increase factor</td>
</tr>
<tr>
<td>Decay factor</td>
<td>Decrease factor</td>
</tr>
<tr>
<td>Mu</td>
<td>Delta min</td>
</tr>
<tr>
<td></td>
<td>Delta max</td>
</tr>
<tr>
<td></td>
<td>Delta zero</td>
</tr>
</tbody>
</table>

V. MODELING ANNS

Here, we follow the procedure described in [6] to model the performance of an ANN. The only difference is that we decided to use Random Forest to create the model instead of using a linear regression model as done in [6].

The first step is to create a training set of pairs \((M_i, y_i)\), where \(M_i\) is a matrix similar to the one presented on Equation (2) and \(y\) is the performance of the ANN on the different time series. The matrix \(M\) is build using the time series characteristics described by [8] and our own time series characteristics presented on [15] and complemented on [6]. Table II shows the time series features describe by [8].

<table>
<thead>
<tr>
<th>Trend</th>
<th>Seasonality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Correlation</td>
<td>Non-linearity</td>
</tr>
<tr>
<td>Skewness</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>Self-similarity</td>
<td>Chaos</td>
</tr>
<tr>
<td>Periodicity</td>
<td></td>
</tr>
</tbody>
</table>

Now that matrix \(M\) have been computed we need to obtain vector \(y\), this is the performance of the ANN on the different time series used in \(M\). In this contribution, we decided to create two models for each ANN. The first one correspond to the performance of the trained ANN when ask to forecast the values used to train it. That is, we measure the ability of the network in its training phase. The second model, we collect the performance of the ANN when is ask to forecast periods that were not seen during its training. That is, we measure the performance of the ANN on a validation set. In order to create these two models, we collect to vectors \(y_T\) that corresponds to the performance of ANNs on training set and \(y_V\) which is the performance of the ANN on the validation set. By creating these models tells us the ability of the ANN to learn a particular time series and the second tells the ability of the ANN to forecast.

Furthermore, in order to test our model technique, we split the matrix \(M\) and its corresponding vectors \(y_T\) and \(y_V\) into two sets. The first set is used to instantiate the model, i.e., to train the Random Forest and the second is as validation set of the model. That is, it is the ability of our models to generalize. We refer to the first set as known time series and to the second as unknown time series.

VI. RESULTS

In this work, ANNs are used to solve the time series prediction problem. A collection of 1001 time series called \(M1\) [16] is used to illustrate the effectiveness of this approach. \(M1\) is commonly used as benchmark to test different procedures on time series forecasting.

The parameter influence has been determined for QUICKPROP and RPROP in both training and validation stages. Besides the three parameters of QUICKPROP and the five parameters of RPROP an extra parameter has been added in both algorithms. This extra parameter is the number of hidden neurons. We decided to include this parameter because it is related to the learning capabilities of the ANN.

This section is organized in various subsections. Subsections VI-A and VI-B presents the parameter order influence for RPROP and QUICKPROP training algorithms. Subsections VI-D and VI-E presents the results obtained by the modeling of ANNs trained with RPROP and QUICKPROP training algorithms.

A. Parameter Influence for QUICKPROP Algorithm

The tables presenting the results of the influence of the parameters are organized as follows. The columns are number from one to the number of parameters where one indicates the time that particular variable is selected. Furthermore, the value of the column shows the percentage that the variable is selected on that order. For example, in Table III, parameter decay is 90% of time the last parameter selected.

The results for the QUICKPROP algorithm are presented for the training stage first and then for the validation stage. Table III presents the results for the training stage. The learning factor was selected 90% of the times as the most important, followed by the hidden neurons which were chosen 60% of the time. The mu parameter is in third place 54% of the time and the least important parameter is the decay factor, who is in last place 90% of the time.

Table IV presents the results of the QUICKPROP algorithm for the validation phase. There are no changes in the parameter order compared to the training phase. The learning factor is first place 74% of the time, but mu increases its percentage of first place. The number of hidden neurons is selected again in second place 46% of the time, but this time, it is also selected 6% of the time in first place. The parameter mu increases the number of times it is chosen in first place from 8% in the training set to 18% in the validation set. The parameter order in the validation stage does not change compared to the training, because parameter mu was selected fewer times in second place than the number hidden neurons. The decay factor once again is selected 90% of the time in last place.
TABLE III
QUICKPROP TRAINING RESULTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order of Parameters</td>
<td>1</td>
</tr>
<tr>
<td>learning factor</td>
<td>90</td>
</tr>
<tr>
<td>hidden neurons</td>
<td>0</td>
</tr>
<tr>
<td>mu</td>
<td>8</td>
</tr>
<tr>
<td>decay factor</td>
<td>2</td>
</tr>
</tbody>
</table>

TABLE IV
QUICKPROP VALIDATION RESULTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order of Parameters</td>
<td>1</td>
</tr>
<tr>
<td>learning factor</td>
<td>74</td>
</tr>
<tr>
<td>mu</td>
<td>18</td>
</tr>
<tr>
<td>hidden neurons</td>
<td>6</td>
</tr>
<tr>
<td>decay factor</td>
<td>2</td>
</tr>
</tbody>
</table>

B. Parameter Influence for RPROP Algorithm

The results for the RPROP training algorithm are easier to interpret. For the training phase, Table V shows that 100% of the time the increase factor is the most important parameter. The second place is the decrease factor also 100% of the time. The third place is for the hidden neurons with 78% of the time. The parameters delta max and delta zero are selected at fourth and fifth place with 44% and 58%, respectively. The last place in all the occasions is delta min.

Table VI shows the RPROP results on the validation set, it can be observed that there is no change in the order with respect to the training set on the first three places, i.e., these are the increase factor, the decrease factor and the hidden neurons. However, there is a difference in order on the parameters zero and delta max. In this case, delta zero is in fourth place with 68%. The parameter delta min is still in last place again with the 100%.

C. Modeling Results

It is time to present the results of the modeling technique. To measure the quality of the models, we decided to compare the actual performance with the modeled performance using sMAPE which is described in Equation 3.

\[
sMAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{(y_i + \hat{y}_i)/2} \right|
\]  

A sMAPE value close to zero indicates an accurate model, on the other hand a negative or positive value indicates a poor performance of the model. Each plot represents the behaviour of the model of a particular ANN. The solid line in the plot depicts the performance predicted by random forests, whereas the points represents the actual ANN performance. Clearly, if the points are close to the line it is an indication that the model is performing accurate predictions.

D. ANNs trained with QUICKPROP

Figures 2 and 3 shows the performance of the models in the training and validation set, respectively. It can be observed that the points forms a cloud around the line indicating that our modelling technique is performing accurate predictions. For the training stage, it has a sMAPE of 7.32% and in the validation the error is 5.44%. This means that in the training stage the modeling has an accuracy around 93% and around 95% for the validation stage. Note that these values are very high; however, as indicated in the figure, these are obtained using the time series used to create the model. In the next figure, we present the data for unseen time series, these depict the generalization ability of our model.

Fig. 2. Modelling in training stage with known time series (QUICKPROP)
Fig. 3. Modelling in validation stage with known time series (QUICKPROP)

Figures 4 and 5 show the model performance of QUICKPROP in the training and validation set, respectively, in this case the time series have not been seen by the model. The accuracy is around 93% in the training stage and 74% in the validation stage. These results show that it is possible to model an ANN with random forests and use this model to forecast the performance of the ANN on a new time series.

Fig. 4. Modelling in training stage with unknown time series (QUICKPROP)

Fig. 5. Modelling in validation stage with unknown time series (QUICKPROP)

E. ANNs trained with RPROP

Figures 6 and 7 present the modeling results for RPROP algorithm. In these figures we can see an accuracy around 93% in the training stage and around 80% in the validation stage. This time, the results shown are only in the case when new problems are presented to the model.

Fig. 6. Modelling in training stage with unknown time series (RPROP)

Fig. 7. Modelling in validation stage with unknown time series (RPROP)

In summary, we have presented models of ANNs trained with QUICKPROP and RPROP. In both cases, the results show that our models are making accurate predictions in unseen time series, in all the cases the sMAPE is below 21%. This indicates that our models are generalizing well. Furthermore, we have model the performance of the ANN on the training phase and in the validation. That is, the ANN are trained with a part of a time series, then we used the trained ANN to forecast the rest of that time series and the part where it was trained. In both cases, our models have been able to produce accurate predictions.

VII. CONCLUSIONS

The main objective of this work was to model the performance of an ANN. This analysis was done for two training algorithms: QUICKPROP and RPROP. Both algorithms represent improved versions of the traditional Back propagation
algorithm. Unfortunately, because of the number of parameters of each algorithm, and the possible values for their parameters, a combinatorial problem arises. Due to this, before modeling the ANNs, the parameter influence on the performance is determined in order to assure that only ANNs with different performance are being modeled. In all the cases, the ANNs models were used to predict Time Series.

The modeled ANNs were proved with time series used in the instantiation of the models and on new time series (i.e., label unknown) in order to show that this strategy can generalize well, and that can be used in real problems. The main reason to model ANNs is to have the possibility to know a priori, with some degree of confidence, the expected performance of an ANN under a collection of problems. That way, the possibility to choose the best method available increases, this is intimately linked to the solution of the Algorithm selection problem.

A. Future Work

An research avenue that we would like to pursue is to test whether this technique can work on different training algorithms such as SARPROP [17]. Moreover, we would like to use these models to solve the algorithm selection problem and test whether this procedure can have better performance than traditional forecasters, such as ARIMA.

REFERENCES