On the Use of the LMA and PSO for Time Series and Regression Tasks

Ankit Gupta
Faculty of Exact Sciences and Engineering
University of Madeira
Madeira, Portugal
2118619@student.uma.pt

Diogo Freitas
Faculty of Exact Sciences and Engineering
University of Madeira
Madeira, Portugal
2019214@student.uma.pt

***Abstract*—This paper analyses how the Levenberg–Marquardt backpropagation algorithm (LMA) and the Particle Swarm Optimisation (PSO) can be used as a training algorithm of Artificial Neural Networks. In this view, four data sets (two of them for time-series analysis and the other two for fitting applications) were tested with different ANN architectures. The comparison between the two algorithms was made in terms of lower Mean Squared Error (MSE) value and in terms of the coefficient of correlation (R). Our results showed that the LMA worked better for time-series analysis and PSO for data fitting tasks.**

***Index Terms*—levenberg–marquardt, LMA, particle swarm optimisation, PSO, bio-inspired algorithms, time-series, fitting**

I. Introduction

Neural networks are a set of algorithms which are designed to identify patterns in the data. The resemblance with the human brain allows the neural networks to be used for many real-time applications, e.g., building self-driving cars, spam or non-spam prediction, detection of fraud, anomaly detection. The neural networks can perform a variety of tasks, e.g., classification, clustering and predictive modelling. This study deals with exploring the concepts of predictive analytics by time-series analysis and data fitting.

A. Time-Series Analysis

Time-series is a collection of data based on time. As an instance, a collection of sales data from the past six months daily or the number of patients diagnosed with a disease, each month for a period of the last five years. Time is an independent variable for time-series analysis which is used future forecasting.

Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data. Autocorrelation means similarity in the data after a fixed time lag, whereas seasonality refers to the presence of periodic patterns in the data.

Autocorrelation and seasonality are interrelated to each other in the sense that autocorrelated data ensures data seasonality. On the other hand, stationarity in the data means that the mean and variance in the data do not alter with respect to time.

Time-series modelling requires mitigating the effect of autocorrelation and seasonality and encouraging stationarity in the time-series.

B. Data Fitting

Training a model can lead to the problem of overfitting or underfitting, which ends up training inaccurate and less powerful models for classification or regressions tasks. These problems arise due to lack of optimised learnable parameters.

Data fitting, or curve fitting, helps in alleviating the effects of overfitting or underfitting. It deals with enhancing the quality of mapping between data and targets by optimising the learnable parameters. It allows the selection of appropriate neural network architectures and continuous learning of parameters until the termination condition is reached. The termination conditions can be, e.g., achieving the maximum number of iterations when no convergence is detected during model training. The result of data fitting is the optimised parameters of the model.

The study aims at using two optimisation algorithms: Levnenberg–Maquardt algorithm (LMA) and Particle Swarm Optimisation (PSO) for time-series analysis and data fitting, followed an analysis of the results for both, in terms of Mean Squared Error (MSE) and Coefficient of Correlation (R). Each data set was divided into sequential blocks originating other three data sets: training (70%), validation (15%) and test data set (15%).

This paper is organised as follows: in Section II, the methods and algorithms that will be used in this work to seasonal adjust the time-series and for feature selection are presented. Also, in this section, the LMA is introduced, as well as the PSO algorithm. In turn, Section III presents the results of the comparison between the LMA and PSO. Finally, the last section presents some concluding remarks.

II. Methods and Algorithms

A. Seasonal Adjustment

Data variations on time-series are, in most cases, related with, e.g., the current economic situation, technological and health care developments, and even with the environmental issues. As a consequence, cyclical fluctuations can be present on the time-series over the time considered. These repeated

patterns can happen daily, weekly, monthly or yearly, and are known as seasonal variations.

When one time-series has a strong seasonal component, the relations between the independent variable(s) and the dependent variable(s) are often obfuscated by these periodic fluctuations.

To overcome this problem, one should remove the seasonal variation of the time-series, in order to obtain a different time-series, but with less dispersion than original. Thus, it is possible to better understand the relation between variables. The process of removing the seasonal variation is known as seasonal adjustment.

Before getting into to the details of the method followed for the seasonal adjustment, it essential to decompose the time-series into three basic patterns: trend (t), seasonality (s) and irregular variations (ϵ). In this view, it is possible to distinguish two types of models.

When the variance does not change over the time-series, one time-series is said to follow an additive model. However, when the amplitude increases with the level, the time-series is characterised as being a multiplicative model.

If the model of the time-series is considered to be an additive model, then each observation k can be decomposed as follows:

$$y_k = t_k + s_k + \epsilon_k. \tag{1}$$

On the other hand, when the model of the time-series is considered to be a multiplicative model, the observation k is given by:

$$y_k = t_k \cdot s_k \cdot \epsilon_k. \tag{2}$$

According to Equation (1), the seasonality component (s) can be removed from the times series by taking:

$$y_k - t_k = +s_k + \epsilon_k, \tag{3}$$

or, according to Equation (2):

$$\frac{y_k}{t_k} = s_k \cdot \epsilon_k. \tag{4}$$

Thus, in order to compute an estimative of s_k, one needs to remove the trend component first from the time-series, and then compute the estimative of s_k, according to Equations (3) or (4). One commonly followed approach to identify the different time-series' components is known as the X–11 method [6], and it was introduced by Shiskin in 1965.

1) The X–11 Method: The X–11 method is an iterative approach, for the reason that in order to identify the seasonality, it is necessary to know the trend and to obtain a good estimative of the trend, it is necessary to deseasonalise the original series.

In this view, the X–11 method can be summarised into four main steps:

1) Compute an estimative of the trend component using a two-sided centred moving averages, such as the Henderson filter;
2) Having the estimative of the trend component, it is possible to obtain a new time-series with the seasonal and irregular components;
3) Since this new time-series has both the seasonal and irregular components, a seasonal filter is applied in order to lessen the irregularities in the data;
4) Compute the adjusted data, and return to step 1) using Equation (1) or (2) only once.

A two-sided moving average (MA) is a statistical procedure identical to a simple average; however, the average is centred in each observation k. The objective of MA is to smooth the time-series, acting like a filter, in order to estimate a component. The MA can be expressed as:

$$\text{MA}_k = \frac{1}{L} \sum_{j=-d}^{d} y_{k+j}, \tag{5}$$

where L corresponds to the period of the cycles in the time-series and is an odd number given by $L = 2d + 1$. When L is even, and thus defined by $L = 2d$, the MA can be expressed as:

$$\text{MA}_k = \frac{1}{L} \sum_{j=-d}^{d-1} y_{k+j}. \tag{6}$$

In turn, the Henderson filter is a more elaborated MA mechanism, that considers both symmetrical and asymmetrical weights, especially for the start and end values of the time-series.

B. Feature Selection

Feature selection is an essential task in machine learning algorithms and in statistics. The objective of the feature selection is to select a subset of features, from all features collected, according to their importance to the output variable(s).

By just selecting a subset of features, one is simplifying the model, requiring less computational resources in order to compute it (e.g., execution time and memory), and also avoids that the model learns specific information about the data set, such as noise (i.e., avoid overfitting).

In order to select which subset of features are more relevant to the output(s) variable(s), a myriad of strategies exist, such as from simply use the variance in order to remove features with low variance, to using the univariate feature selection based on F-tests.

Univariate feature selection is considered a filter method, meaning that it only considers the relationship between each independent variable and its importance for the output variable(s). In turn, the importance of each independent feature to the output variable(s) is calculate by the F-statistic, given by:

$$F = \frac{\left(\dfrac{\text{RSS}_1 - \text{RSS}_2}{p_2 - p_1}\right)}{\left(\dfrac{\text{RSS}_2}{n - p_2}\right)}, \tag{7}$$

where RSS_i and p_i are, respectively, the residual sum of square and the number of features of the model i. n, in turn, denotes the number of samples in the data set.

Equation (7) compares the model one with model two, under the null hypothesis that the model two does not prove to be statistically better, in terms of fitting, when compared to model

one. If the null hypothesis is rejected, then the model two performs equal or better than the model one.

Finally, features are sorted according to the computed F-statistic value, and the user/researcher is then responsible for choosing the number of features to be used in the predictive model.

C. Particle Swarm Optimisation

Particle swarm optimisation (PSO) is a stochastic optimisation technique for linear and nonlinear functions (also known as fitness function), suggested by Eberhart and Kennedy [1], [2] in 1995.

PSO starts with random initialisation of particles between the boundaries of the d-dimensional search space of the fitness function. Each particle has its fitness value, which is decided by the fitness function, a position and a velocity, which enables it to move in the problem space.

The PSO algorithm is an iterative process which tends to find the optimal solution by the movement of particles. In each iteration, every particle is updated with its best fitness value p_{best} and global best fitness value achieved by any particle in the swarm g_{best}. These values are then used for updating the particle velocity, such as [5]:

$$v_{\text{new}} = \omega \cdot v + l_1 \cdot r_1 \cdot \Big(p_{\text{best}} - x\Big) + l_2 \cdot r_2 \cdot \Big(g_{\text{best}} - x\Big), \quad (8)$$

where v_{new} is the new velocity, ω is known as the inertia term and controls the influence of the previous velocity in the next particle's velocity v. l_1 and l_2 are the learning parameters, and r_1 and r_2 are random numbers with values ranging between 0 and 1. Finally, x is the current position of the particle.

In turn, the position of the particle is updated as:

$$x_{\text{new}} = x + v_{\text{new}}. \quad (9)$$

The algorithm stops on reaching the maximum number of iterations or when an appropriate solution is found, according to a predefined accuracy, being the output of the algorithm the position of the best particle in the swarm.

In order to use the PSO algorithm to find the optimal weights of an Artificial Neural Network (ANN), each particle can be seen as a different ANN. The weights of the ANN represent the position of the particle in the search space, and the fitness value is given by the MSE of the ANN in the training data set.

Particles move in the search space according to Equations (8) and (9), and the output of the algorithm is the position (i.e., the weights) of the best particle in the swarm in terms of the MSE in the validation data set.

D. Levenberg–Marquardt Backpropagation

The Levenberg–Marquardt algorithm (LMA) is an optimisation algorithm, which is used to solve the least square fitting problem. It was suggested by Kenneth Levenberg and improved later by Donald Marquardt [3], [4].

Mathematically, given a set of pairs of dependent (x_i) and independent variable (y_i) as (x_i, y_i), $i = 1, 2, 3, \cdots, n.$, the objective is to find the optimal set of parameters ($\mathbf{P}$), such that the sum of squared error deviation $\mathbf{S}(\mathbf{P})$ can be minimised as follows:

$$\hat{\mathbf{P}} \in \text{argmin}_{\hat{\mathbf{P}}}\ \mathbf{S}(\mathbf{P}) \equiv \text{argmin}_{\hat{\mathbf{P}}} \sum_{i=1}^{m} \Big[y_i - f(x_i, \mathbf{P})\Big]^2, \quad (10)$$

being m the number of samples in the data set.

The LMA is an iterative procedure which starts with random initialisation of $\mathbf{P}$ (vector or scalar), and at each iteration, $\mathbf{P}$ is updated, i.e., $\mathbf{P}$ is updated to $\mathbf{P} + \delta$.

For δ value determination, $f(x_i, \mathbf{P}+\delta)$ has to be linearised, so the equation, according to the Taylor series expansion, becomes:

$$f(x_i, \mathbf{P} + \delta) \approx f(x_i, \mathbf{P}) + \mathbf{J}_i\delta, \quad (11)$$

where $\mathbf{J}$ is the gradient of f with respect to $\mathbf{P}$ and it is given by (Jacobian matrix):

$$J_i = \frac{\partial f(x_i, \mathbf{P})}{\partial \mathbf{P}} \quad (12)$$

The minimum of $S(\mathbf{P})$ is found at $\mathbf{J}\delta = 0$. The first order approximation for $f(x_i, \mathbf{P} + \delta)$ is:

$$S(\mathbf{P} + \delta) = \sum_{i=1}^{m} \Big[y_i - f(x_i, \mathbf{P}) - \mathbf{J}_i\delta\Big]^2. \quad (13)$$

The vectorised form of the above equation is as follows:

$$S(\mathbf{P} + \delta) = \| \mathbf{y} - \mathbf{f}(\mathbf{P}) - \mathbf{J}\delta \|^2 . \quad (14)$$

Differentiating the above term with respect the δ and equating it to 0 gives:

$$(\mathbf{J}^T\mathbf{J})\delta = \mathbf{J}^T\Big[\mathbf{y} - \mathbf{f}(\mathbf{P})\Big], \quad (15)$$

where J is the singular Jacobian matrix of size $m \times n$, and $\mathbf{y}$ and $\mathbf{f}(\mathbf{P})$ are the vector components for the i^{th} sample y_i and $f(x_i, \mathbf{P})$, respectively.

The above equation is given by Marquardt. In turn, Levenberg added an parameter, $\lambda\mathbf{I}$, making it the damped version:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\delta = \mathbf{J}^T\Big[\mathbf{y} - \mathbf{f}(\mathbf{P})\Big], \quad (16)$$

where $\mathbf{I}$ is an identity matrix and thus:

$$\delta = \Big[\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\Big]^{-1}\mathbf{J}^T\Big[\mathbf{y} - \mathbf{f}(\mathbf{P})\Big]. \quad (17)$$

The λ is adjusted at each iteration of the algorithm to ensure the minimisation of $S(\mathbf{P})$.

The LMA is also called the mixture of Gauss–Newton method and Gradient Descent. If $S(\mathbf{P})$ is minimised at a faster rate, a small change in λ is required. This makes the algorithm to behave like Gauss–Newton method. On the other hand, if the change in λ is not sufficient enough for minimisation, then λ has to be increased, leading to the algorithm to taking the steps in the direction of the negative gradient. This makes the algorithm to behave like Gradient Descent.

The initialisation of λ is random. At the next step, it is decreased by $\frac{\lambda}{v}$, where $v > 1$. If this does not minimise the $S(\mathbf{P})$, then the value of λ can be increased by a factor of $v^k(\lambda v^k)$, being v and k chosen by the user/researcher.

III. Results

This study deals with time-series analysis for the urban traffic in São Paulo – Brazil, monthly milk production and data fitting for QSAR fish toxicity and wine quality data sets. The neural network used for time-series analysis is a nonlinear autoregressive with exogenous input (NARX), whereas for data fitting a two-layered feed-forward network with hidden sigmoid neurons and linear output neurons (`fitnet`) was used.

Two structures of NARX network were considered: open-loop and closed-loop. In Fig. 1 and 2 a visual representation of both structures is given, where $x(t)$ represents the independent variables and $y(t)$ the dependent variable values of a time-series. It also important to note that number of delays was set to be two, meaning that the ANN will be based on the previous two values to predict the following value in the time-series.

In an open-loop architecture, the real output is used instead of feeding back the estimated output given by the ANN, as it happens in the closed-loop architecture. For the following tests, the ANN were trained using an open-loop architecture. Nevertheless, the same ANN was also used to test the closed-loop accuracy.

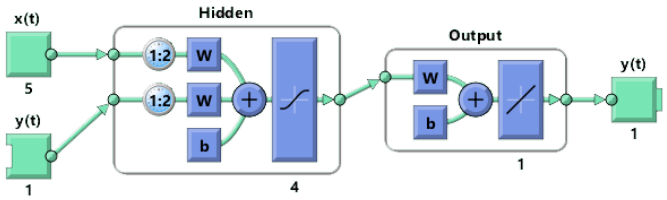

Fig. 1: An example of the NARX in open-loop for the urban traffic in São Paulo – Brazil – data set.

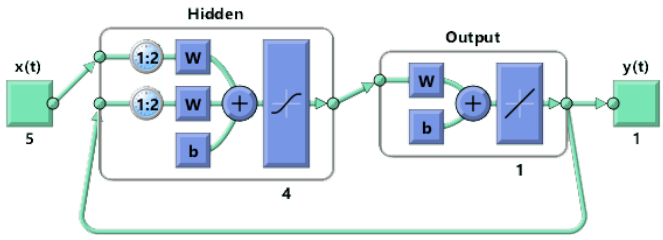

Fig. 2: An example of the NARX in closed-loop for the urban traffic in São Paulo – Brazil – data set.

The ANNs for time-series analysis and data fitting were tested ten times for each number of neurons in the hidden layer considered: 4, 7, 10, 12, 15 and 20. Besides that, the initial weights were set in the range $[-2.4/I, 2.4/I]$, where I is the number of inputs.

Each data set was split into a training (70%), validation (15%) and test data set (15%), being divided using blocks of sequential indices.

On the other hand, the number of epochs was defined to be 5 000, and the error limit was set to be zero. For this set of tests, the early stopping of the algorithm was not used.

The LMA algorithm was parameterised so that the initial λ was set to be equal to 1, with increments of 10 and decrements of 0.1. On the other hand, 24 particles were used with PSO. Besides that, $\omega = 0.9$, $l_1 = 0.5$ and $l_2 = 0.3$.

Tests were executed from the above-mentioned data sets, under different experimental conditions. In these experiments, it was included a comparison between the performance of the LMA and PSO. The performance metrics used for the evaluation of models are discussed in the next subsection and were used only in the test data set.

A. Performance metrics

1) Mean Squared Error (MSE): It is important to note that the MSE is computed as follows:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(Y_i - \hat{Y}_i)^2, \tag{18}$$

where m is the number of samples in the data set, Y_i is the observed value and $\hat{Y}_i$ the predicted value computed by the model. Of course that, the closer $\hat{Y}_i$ is to Y_i, the lower the MSE value will be.

2) Coefficient of Correlation R: The coefficient of correlation measures the degree of correlation between the actual and predicted target values. R is mathematically defined as:

$$R_{Y\hat{Y}} = \frac{\sigma_{Y\hat{Y}}}{\sigma_Y \sigma_{\hat{Y}}}, \tag{19}$$

where $\sigma_{Y\hat{Y}}$ is the population covariance, and σ_Y and $\sigma_{\hat{Y}}$ are the population standard deviations

If $\sigma_{Y\hat{Y}} \approx \sigma_Y \sigma_{\hat{Y}}$, then the R value will be approximately equal to 1 or -1, indicating that the model has a strong positive or negative relationship with the real values. Whereas when $\sigma_{Y\hat{Y}} >> \sigma_Y \sigma_{\hat{Y}}$ or $\sigma_{Y\hat{Y}} << \sigma_Y \sigma_{\hat{Y}}$, R can stay away from 1, indicating that the model can not find a relationship between the real target and the computed targets.

B. Urban Traffic in São Paulo – Brazil

This data set consists of the urban traffic behaviour of the São Paulo, a city in Brazil. It includes the urban traffic data from December. 14, 2009, to December. 18, 2009, between 7:00 to 20:00, with a time interval of 30 minutes. The data set contains 135 samples and 18 attributes: hour, immobilised bus, broken truck, vehicle excess, accident victim, running over, fire vehicles, occurrence involving freight, incident involving dangerous freight, lack of electricity, fire, point of flooding, manifestations, defect in the network of trolleybuses, tree on the road, semaphore off, intermittent semaphore, and slowness in traffic (the target variable).

1) Seasonal Adjustment: Before analysing the data set, the seasonal adjustment procedure (X–11 method) was conducted in the time-series. In this case, the repeated patters can happen daily, every thirty minutes.

The original and seasonally adjusted time-series can be found on Fig. 3. As can be seen in the original time-series, the amplitude increases with the level, meaning that the

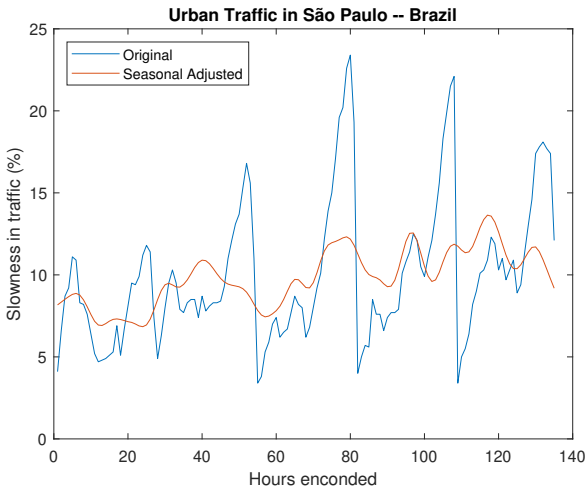

Fig. 3: The original and the seasonally adjusted time-series of the urban traffic in São Paulo – Brazil.

time-series is characterised as being a multiplicative model (Equation (2)).

The final time-series is a more smooth line, when compared with the original time-series, that contains only the approximately constant trend (t) and the irregular variations (ϵ) components of the original time-series.

2) Feature Selection: When analysing the variance of the features of the data set presented in Table I, it is possible to observe that variables have a minimal variation, being the broken truck, point of flooding and defect in the network of trolleybuses the features with the highest variance. However, since the seasonal component (i.e., the hour) was removed from the time-series, it is expected that this variable contributes very little to the variation of the dependent variable.

TABLE I: Variance of the features of the urban traffic in São Paulo – Brazil – data set.

Feature	Variance
Broken Truck	1.215
Defect in the network of trolleybuses	0.671
Point of flooding	0.508
Accident victim	0.485
Immobilised bus	0.435
Lack of electricity	0.255
Semaphore off	0.215
Running over	0.120
Manifestations	0.050
Tree on the road	0.043
Vehicle excess	0.029
Intermittent Semaphore	0.015
Fire Vehicles	0.007
Occurrence involving freight	0.007
Incident involving dangerous freight	0.007
Fire	0.007

Following the univariate feature selection approach, the features with the highest importance for the target variable were the presence of a broken truck, lack of electricity, point of flooding, defect in the network of trolleybuses and semaphore off, as can be seen in Fig. 4, that depicts the scores according to its importance to the target variable.

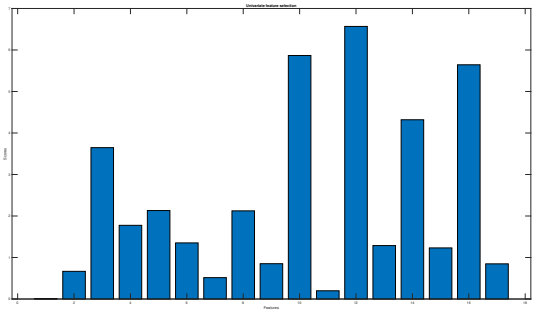

Fig. 4: Feature selection of the urban traffic in São Paulo – Brazil – data set.

3) Levenberg–Marquardt Backpropagation: As already mentioned, the LMA algorithm was tested using a different number of hidden neurons and, for each number of hidden units, ten tests were executed. Table II summarises the results obtained with the LMA algorithm, presenting the number of hidden neurons, the best and the worst MSE achieved, and the respective R values, the average MSE and its standard deviation (STD) only for the test portion of the data set.

It is important to note that slowness in traffic of the previous two days was used to compute the prediction of the current day. Thus, the predicted time-series has two missing points in its beginning.

As can be analysed from Table II, the lowest MSE value was obtained using an ANN with four hidden units. However, a contrast was found, since one of the worst MSE values was also obtained with ANN with four hidden units, thus justifying its high STD.

When analysing the correlation coefficients, it is possible to verify that, regardless of the number of hidden units, a strong positive relationship has always found. Nevertheless, as the number of hidden units increases, the average MSE also increases.

TABLE II: Summary of the tests performed with the LMA in the urban traffic in São Paulo – Brazil – test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.037	0.994	0.866	0.843	0.363	0.267
7	0.073	0.988	0.687	0.876	0.340	0.188
10	0.276	0.955	0.483	0.916	0.371	0.073
12	0.170	0.971	0.593	0.896	0.345	0.123
15	0.230	0.961	0.704	0.881	0.434	0.143
20	0.363	0.940	0.896	0.892	0.611	0.186

Two types of predictions were used in order to assess the fitness of the model with the best ANN found (four hidden units): open-loop and closed-loop. It is noteworthy that the ANNs, whose results were presented in Table II, were trained using an open-loop architecture. Thus, the results of the open-loop prediction are better than the open-loop, as can be seen in Fig. 5.

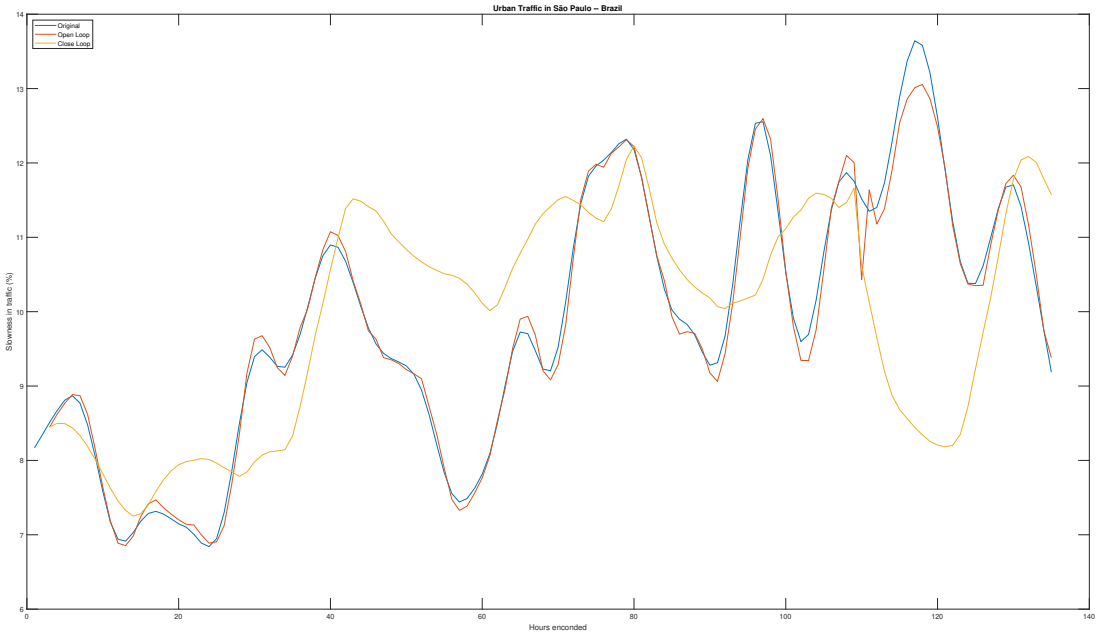

Fig. 5: The original, open-loop and closed-loop predictions of the urban traffic in São Paulo – Brazil – test data set.

4) Particle Swarm Optimisation: Using the same seasonally adjusted time-series and the same ANN architectures, the authors tested the use of the PSO algorithm, as an alternative to using the LMA to find the optimal ANN weights that minimise the MSE.

The results of the tests performed with the PSO are presented in Table III, in which it is possible to observe that the PSO is not at all a viable alternative for finding the optimal weights for this time-series, since in all cases the algorithm did not converge, leading to a high MSE value and a low correlation coefficient.

On possible justification for this problem may be related with the choice of parameters used for the PSO, e.g., the number of particles, swarm architecture, and the number of iterations, since PSO may require more iterations in order to converge when compared to the LMA.

Not disregarding this, once again the best MSE value found belongs to an ANN with four hidden units in the hidden layer and, on average, as the number of hidden units increases, the worse the MSE gets. The values of the STD, in turn, suggest that the use of the PSO algorithm for this data set is not stable.

C. Monthly Milk Production

The data set includes the monthly production of milk per cow from January 1962 to December 1975. It contains 168 samples of milk production with three attributes: year, month and the milk production in pounds per cow of that specific month. This study aimed at analysing the data through time-series analysis.

TABLE III: Summary of tests performed with the PSO algorithm of the urban traffic in São Paulo – Brazil – test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	139.813	0.277	5301.177	-0.119	1558.653	1789.866
7	479.310	0.171	5672.562	0.216	1992.805	1841.456
10	505.743	0.411	3116.445	0.046	1823.125	815.772
12	1816.650	-0.252	7883.120	0.138	3194.680	1764.955
15	1028.200	0.032	9178.211	-0.063	3209.100	2420.504
20	1433.744	-0.385	9820.723	-0.300	4813.707	2680.271

In this case, feature selection is not required, since the seasonal adjustment will remove the monthly cycle, and with this the month feature. Thus, only the year remains as input variable.

1) Seasonal Adjustment: Contrarily to the traffic data set, this time-series follows a multiplicative model, since the amplitude increases with the level. Thus, the seasonal adjustment was done taking into consideration the Equation (2).

As already mentioned, the repeated patters can happen monthly. The original and the final seasonally adjusted series can be found on Fig. 6, where it is possible to analyse that the time-series has a approximately linear growing trend.

2) Levenberg–Marquardt Backpropagation: Following the same structure of tests used in the traffic data set, each ANN architecture was tested ten times with different number of hidden units in the single hidden layer. The results of these tests are presented in Table IV.

According to Table IV, the lowest MSE value was found, like the traffic data set, by an ANN with four hidden units. However, when we consider the average MSE, ANNs with

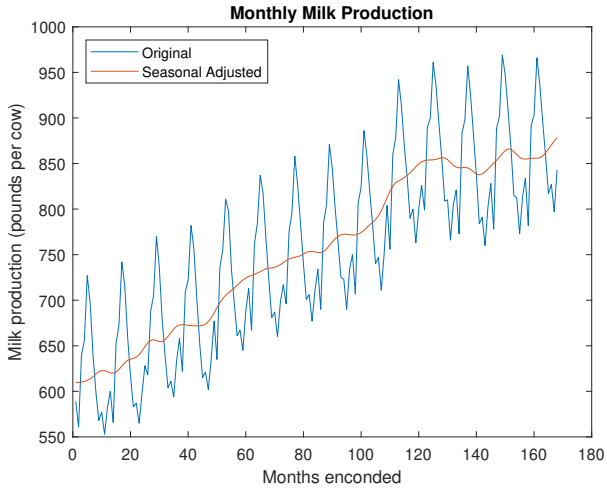

Fig. 6: The original and the easonally adjusted time-series of milk production.

fifteen hidden units found in most of the tests the best architecture.

Besides that, the STD is high in the six number of hidden neurons considered, meaning that one can obtain good results in some tests, but other tests may also perform poorly, especially for the case of four and twenty hidden units, that also showed the highest average MSE value.

In either case, the correlation coefficient performed well, having values higher than 0.990 for the best ANN of each number of hidden units.

TABLE IV: Summary of the tests performed with the LMA in the milk production test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.711	0.999	4569.515	0.676	726.043	1586.540
7	3.804	0.999	688.122	0.967	179.710	262.210
10	2.554	0.999	555.447	0.984	134.616	214.408
12	5.863	0.999	3007.064	0.936	414.066	925.790
15	4.452	0.999	335.660	0.984	105.529	120.156
20	47.604	0.997	5156.289	0.904	1041.943	1650.181

Using the best ANN with four hidden units, the authors computed the prediction using the open and closed-loop. The visual result is presented in Fig. 7, where is possible to very that the ANN in open-loop architecture can predict with high accuracy the values of the time-series. As expected, the closed-loop does not perform as good as the open-loop.

Like the traffic data set, since the data was divided into sequential blocks, the first 70% of time-series was used to train the network, the next 15% to validation, and the last 15% can be seen as a test to the ANN's predictive capacity.

3) Particle Swarm Optimisation: Once again, the PSO algorithm did not perform as well as the LMA, as can be seen by the best and the worst MSE values presented in Table V.

Interestingly, the ANN with the lowest MSE is one with twenty hidden units, i.e., the highest number of neuronal units considered. In terms of average MSE value, ANNs with ten or fifteen hidden units performed similarly, but overall results indicate that accuracy of the ANN for each number of hidden units is not stable between tests.

TABLE V: Summary of the tests performed with the PSO algorithm in the milk production test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	17253.295	-0.531	626369.059	-0.474	201543.068	225819.681
7	11397.214	0.359	543440.476	0.231	170995.183	166423.953
10	11879.112	0.308	1215495.604	0.474	376017.283	414875.998
12	13490.709	0.556	817049.406	-0.427	208999.135	250281.472
15	8338.686	0.493	573488.400	-0.446	143309.806	180550.263
20	7599.916	-0.412	1224740.967	-0.475	328536.823	409440.885

As a result, the PSO algorithm is not, by any means, a good alternative for time-series fitting. In the following tests, a comparison between the LMA and PSO will be presented, taking into consideration regression tasks.

D. QSAR fish toxicity

QSAR fish toxicity data set includes 908 samples with five features and acute aquatic toxicity for fish as a target attribute. This data set is used to fit the data using two-layered feed-forward neural network.

1) Feature Selection: Table VI shows the variance of features for predicting the acute aquatic toxicity towards the fish. Variance also gives an insight into the outliers present in the data. It is clear from the variance values that the data is well scaled, except the MLOGP feature, that has the highest variance, suggesting its use on the feature set.

The univariate feature selection performs F-tests individually for each feature with the predictor variable followed by the ranking of features using the p-value of F-test statistics. Fig. 8 presents the score of each feature, showing that NdsCH and NdssC have relatively very less score and hence can be excluded from the feature set.

TABLE VI: Variance of features of the fish data set.

Feature	Variance
MLOGP	2.054
NdssC	0.741
CIC0	0.571
NdsCH	0.366
$SM1_{Dz}$	0.183
GATS1i	0.155

2) Levenberg–Marquardt Backpropagation: Data fitting requires fitting of parameters at every iteration. This fitting is done using an optimisation algorithm which guides the process of updating the learnable parameters, i.e., the weights of the ANN.

Table VII shows the MSE and R values for the different number of neuron units in the hidden layer. The best performance was achieved at using the ANN with twelve hidden units, achieving the minimum MSE and maximum R value, whereas ANN with fifteen neuron units in the hidden layer performs worst achieving the highest MSE and one of the lowest R values.

Nevertheless, ANNs with seven units perform, on average and in terms of MSE value, better than other types of ANN architectures.

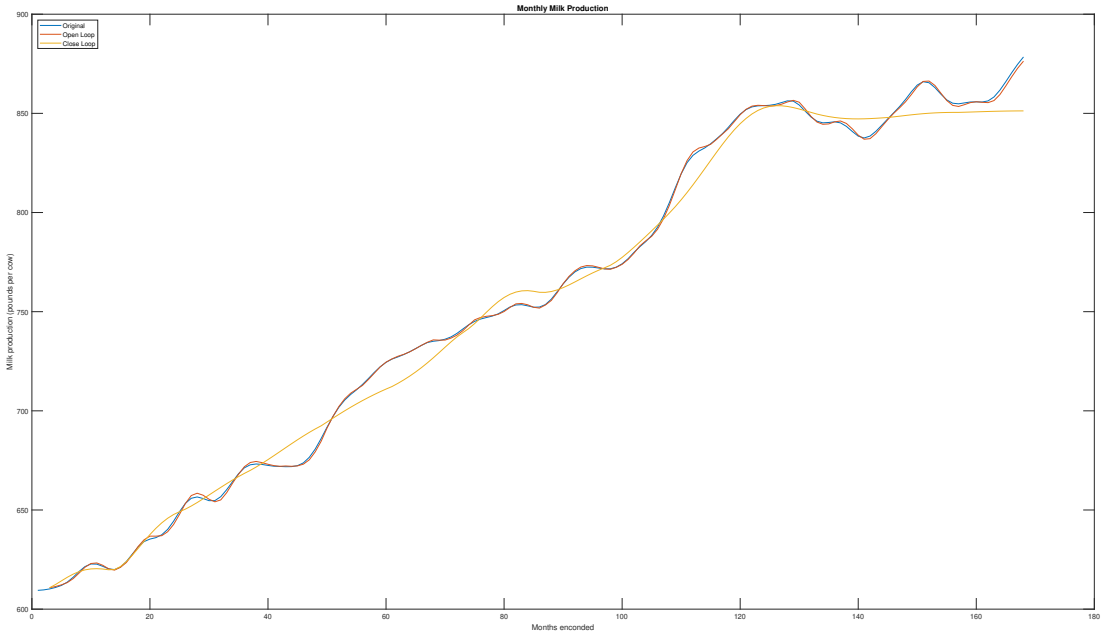

Fig. 7: The original, open-loop and closed-loop predictions of the milk production data set.

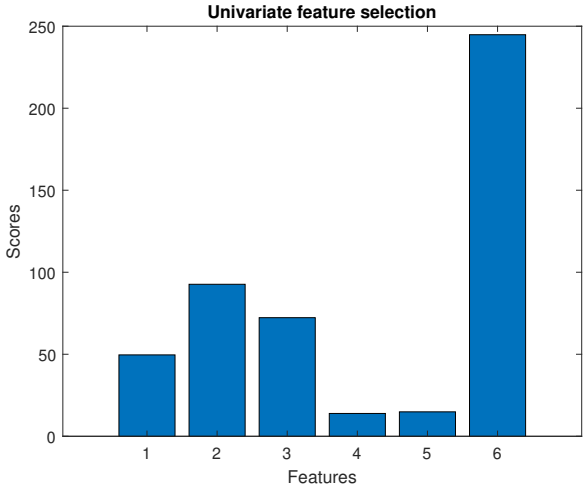

Fig. 8: Feature selection of the QSAR fish toxicity data set.

TABLE VII: Summary of tests performed with LMA for fish test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.821	0.782	1.001	0.726	0.907	0.048734645
7	0.832	0.779	0.932	0.748	0.885	0.040
10	0.819	0.783	1.077	0.700	0.915	0.089
12	0.777	0.795	1.014	0.722	0.917	0.069
15	0.833	0.779	1.039	0.716	0.932	0.067
20	0.817	0.784	1.042	0.714	0.930	0.073

3) Particle Swarm Optimisation: The PSO algorithm was also used as an optimisation algorithm for data fitting of the fish data set.

The MSE and R results for various hidden neuron units are presented in Table VIII.

ANN with seven hidden units achieved the lowest MSE, and highest MSE was achieved by twenty hidden units containing ANN, depicting the worst performance. Thus, the highest R value was also achieved by ANN with seven hidden units, whereas the twenty hidden units performed worst, achieving the lowest R value.

TABLE VIII: Summary of the tests performed with PSO in fish test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.871	0.672	1.211	0.492	0.990	0.102
7	0.735	0.728	3.136	0.352	1.153	0.717
10	1.033	0.598	8.260	0.144	2.797	2.184
12	1.048	0.604	14.768	0.022	6.342	5.455
15	1.685	0.488	18.285	0.237	5.654	5.004
20	0.966	0.364	56.832	-0.042	24.341	17.798

It is important to note that both LMA and PSO achieved very similar best MSE values (0.777 and 0.735, respectively); however, the PSO algorithm just needed seven hidden units, while the LMA required twelve. It is also noteworthy that the STD increases with the increase in the number of neurons in the hidden layer. This leads to the conclusion that a smaller number of neurons in the hidden layer is more beneficial for PSO.

E. Wine Quality

The data set was prepared for red and white wine, with the same number of attributes. It consists of total 4898 (1600 for red and 3298 for white wine) samples and twelve features, out of which the quality is used as a response variable based on other attributes in the data set.

1) Feature Selection: The wine data set consist of data related to 2 types of wine: red and white with the same number of features.

The variance of both data sets is shown in Table IX and X, respectively, where it is possible to verify that the total and free sulfur dioxide for both the data sets have a substantial variance.

The Fig. 9 and 10 shows the selection of useful features for data fitting tasks. Thus, for the red wine data set, the chosen features are: fixed acidity, volatile acidity, citric acid, chlorides, total sulfur dioxide, density, sulphates and alcohol. For the white wine, a smaller set of features was used: citric acid, Chlorides, density and alcohol.

TABLE IX: Variance of features for the red wine data set.

Feature	Variance
Total sulfur dioxide	1806.085
Free sulfur dioxide	289.242
Residual sugar	25.725
Density	8.945
Slcohol	1.514
Fixed acidity	0.712
pH	0.022
Citric acid	0.014
Sulphates	0.013
Volatile acidity	0.01
Chlorides	0

TABLE X: Variance of features for the white wine data set.

Feature	Variance
Total sulfur dioxide	1082.102
Free sulfur dioxide	109.414
Density	3.562
Fixed acidity	3.031
Residual sugar	1.987
Alcohol	1.135
Citric acid	0.037
Volatile acidity	0.032
Sulphates	0.028
pH	0.023
Chlorides	0.002

2) Levenberg–Marquardt Backpropagation: Like the fish data set, the LMA was used as an optimisation algorithm for parameter optimisation. Table XI and Table XII show the MSE and R values for the different values of hidden units in the ANNs.

Red Wine Analysis:

The least MSE is achieved by fifteen hidden units containing ANN, whereas the worst performance was achieved by ANN with four hidden units. Thus, the best performance was achieved by ANN with fifteen hidden units and worst-performing ANN was 4 hidden units containing ANN, in terms of R value.

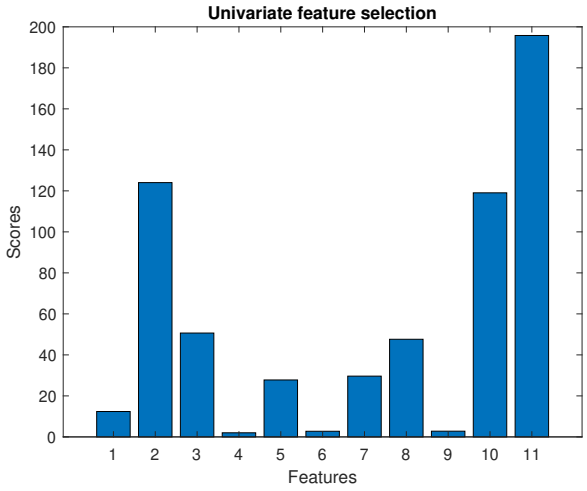

Fig. 9: Feature selection of the red wine data set.

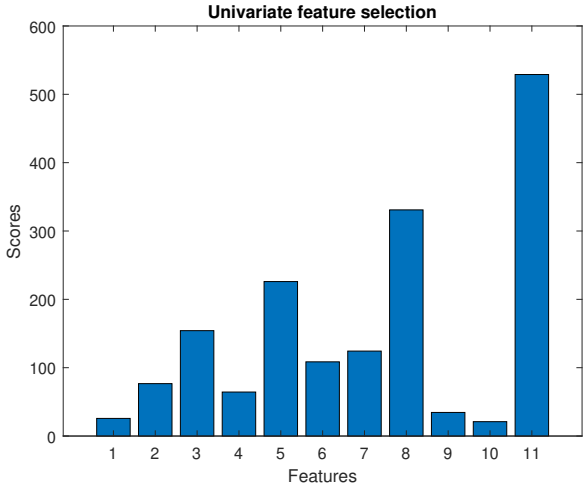

Fig. 10: Feature selection of the white wine data set.

It is important to note that in the case where MSE and R is not able to suggest a single architecture, the means MSE can also be used as a part of the analysis. From the table it is clear that least average MSE is for fifteen hidden units in the hidden layer network, so for red wine data set fifteen hidden units in the hidden layer of the ANN model can be used for further analysis. Nevertheless, regardless of the number of neurons in the hidden layer, the average was found to be similar.

TABLE XI: Summary of tests performed with LMA for red wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.465	0.536	0.535	0.424	0.484	0.0213
7	0.434	0.580	0.507	0.472	0.468	0.0261
10	0.411	0.609	0.490	0.499	0.458	0.0295
12	0.414	0.606	0.486	0.506	0.460	0.0236
15	0.410	0.611	0.489	0.499	0.445	0.0267
20	0.446	0.563	0.476	0.521	0.459	0.0105

White Wine Analysis:

The least MSE was achieved by twenty hidden units containing ANN, whereas the worst performance was achieved by ANN with 4 hidden units. The best performance was also achieved by ANN with twenty hidden units and worst-performing ANN was 4 hidden units containing ANN, in terms of R value. Thus, twenty hidden units containing architecture can be used for further analysis since it has the highest R value and least average MSE value.

In this case, it is possible to identify a trend in the data, since has the number of hidden units increase, the MSE value decreases, indicating that the use of a higher number of hidden units may be more beneficial for the LMA algorithm.

From the three data sets above, is clear that, giving the low STD values, the LMA is a more stable training algorithm when it comes to fitting applications than for time-series analysis.

TABLE XII: Summary of tests performed with LMA for white wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.598	0.487	0.624	0.455	0.610	0.007
7	0.597	0.490	0.613	0.468	0.605	0.006
10	0.591	0.498	0.616	0.465	0.602	0.007
12	0.590	0.498	0.622	0.455	0.603	0.010
15	0.578	0.514	0.621	0.458	0.599	0.011
20	0.576	0.518	0.613	0.466	0.590	0.012

3) Particle Swarm Optimisation: The MSE and R value for different ANN architectures using PSO as an optimisation algorithm is shown in Table XIII and XIV.

Red Wine analysis:

Table XIII shows the MSE and R values for different neural units for the red wine data set. The least MSE and highest R value was achieved by ANN with seven hidden units, whereas ANN with twenty hidden units showed the highest MSE and lowest R value and thereby the worst performance. Choosing the ANN architecture is straightforward in this case since MSE, and R value analysis are pointing to the same network architecture.

When compared with the PSO algorithm, for similar MSE values, the PSO required a lower number of hidden units, i.e., the PSO required seven hidden units and obtained the best value of MSE of 0.477, whereas LMA required fifteen hidden units to get the best MSE of 0.411.

TABLE XIII: Summary of tests performed with PSO for Red wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.532	0.497	0.633	0.254	0.584	0.032
7	0.477	0.543	0.675	0.262	0.566	0.049
10	0.529	0.486	1.043	0.386	0.729	0.221
12	0.562	0.415	9.989	0.180	2.386	2.912
15	1.373	0.297	9.492	0.132	4.630	3.189
20	6.749	-0.085	114.223	-0.036	40.077	29.839

4) White wine analysis: Table XIV shows the MSE and R values for different neural units for the red wine data set. The least MSE value was achieved by ANN with four hidden units, whereas ANN with twenty hidden units showed the highest MSE. The best performing model in terms of R value is the ANN architecture with seven hidden units and worst with fifteen hidden units. In this scenario, the best network is a network with seven hidden units.

In all tests, the PSO performed better, in terms of best MSE, than the LMA, and required a lower number of hidden units to achive similiar accuracy in terms of best MSE values.

TABLE XIV: Summary of tests performed with PSO for white wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.507	0.398	0.816	0.237	0.567	0.090
7	0.501	0.417	0.761	0.278	0.566	0.080
10	0.535	0.347	4.656	0.029	1.071	1.267
12	0.605	0.371	2.874	0.176	1.249	0.765
15	0.733	0.265	7.768	-0.023	2.889	2.361
20	3.762	-0.032	14.432	0.124	7.172	3.441

IV. Conclusion

This study highlights two important concepts: time-series analysis and data fitting. It includes using two data sets: milk production and urban traffic behaviours in the São Paulo data set for time-series analysis and QSAR fish toxicity and wine data set for data fitting.

The experiments were conducted altering the number of neurons in the hidden layer. Two optimisation algorithms (Levenberg-Maquardt algorithm and Particle Swarm Optimisation) were used for both tasks. LMA worked better for time-series analysis and PSO for data fitting tasks. This could be due to the ability of PSO to handle high dimensionality of data than LMA.

On the one hand, the LMA algorithm was more stable than the PSO algorithm, especially in regression tasks; however, for the same accuracy, the PSO algorithm required a lower number of hidden units, as can be seen in Table XV.

TABLE XV: Best MSE values and respective R coefficients of tests performed with LMA and PSO for each data set.

	LMA			PSO		
Data set	No. Hidden	Best MSE	R	No. Hidden	Best MSE	R
Traffic	4	0.037	0.994	4	139.813	0.277
Milk	4	0.711	0.999	20	7599.916	-0.412
Fish	12	0.777	0.795	7	0.735	0.728
Red wine	15	0.410	0.611	7	0.477	0.543
White wine	20	0.576	0.518	7	0.501	0.417

As future work, we are going to test the PSO algorithm with different parameters, different velocity update equations and different topologies of swarms.

Before using the algorithms to find the optimal ANN's weights, feature selection methods were used for this study and were based on the variance of each feature and on its rank based on its p-values of F-test statistics.

Although for other data sets, combination of the average MSE and R value have suggested neural networks in the way that both were pointing to the same network architecture, this may not always be the case. Finally, it can be concluded that selection of algorithm is dependent on the type of task, as seen in the study.

The code-base of this paper will be available in the following GitHub repository: https://github.com/Dntfreitas/PSORNN

REFERENCES

[1] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. 6th Int. Symp. Micromachine and Human Science, Nagoya, Japan, Oct. 4–6 1995*, pages 39–43.

[2] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Netw., Perth, Australia, Nov. 27–Dec. 1 1995*, volume 4, pages 1942–1948.

[3] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[4] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[5] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Proc. IEEE World Congr. Comput. Intell., Anchorage, AK, USA, May 4–9 1998*, pages 69–73.

[6] Julius Shiskin. *The X-11 variant of the census method II seasonal adjustment program*. Number 15. US Government Printing Office, 1965.