Assignment 2

Comparative Analysis of Levenberg–Marquardt Algorithm and Particle Swarm Optimisation for Time-Series and Regression Tasks

Ankit Gupta (no. 2118619) and Diogo Freitas (no. 2019214)

Faculty of Exact Sciences and Engineering
University of Madeira

Table of Contents

Introduction

Introduction

- Comparison between the Levenberg–Marquardt backpropagation algorithm (LMA) and the Particle Swarm Optimisation (PSO) algorithm as parameters optimisation algorithms;

Introduction

- Comparison between the Levenberg–Marquardt backpropagation algorithm (LMA) and the Particle Swarm Optimisation (PSO) algorithm as parameters optimisation algorithms;
- Four data sets were used: two time-series and the other two for data fitting applications;

Introduction

- Comparison between the Levenberg–Marquardt backpropagation algorithm (LMA) and the Particle Swarm Optimisation (PSO) algorithm as parameters optimisation algorithms;
- Four data sets were used: two time-series and the other two for data fitting applications;
- Comparison was done taking into consideration the Mean Squared Error (MSE) value and the Regression value (R^2);

Introduction

- Comparison between the Levenberg–Marquardt backpropagation algorithm (LMA) and the Particle Swarm Optimisation (PSO) algorithm as parameters optimisation algorithms;
- Four data sets were used: two time-series and the other two for data fitting applications;
- Comparison was done taking into consideration the Mean Squared Error (MSE) value and the Regression value (R^2);

GitHub:

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

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:
    - Make relations between the independent variable(s) and the dependent variable(s) being often obfuscated;

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:
  - Make relations between the independent variable(s) and the dependent variable(s) being often obfuscated;
  - We will use the X–11 method, introduced by Shiskin in 1965.

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:
  - Make relations between the independent variable(s) and the dependent variable(s) being often obfuscated;
  - We will use the X–11 method, introduced by Shiskin in 1965.
- Training a model for data fitting can lead to the problem of overfitting or underfitting:

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:
    - Make relations between the independent variable(s) and the dependent variable(s) being often obfuscated;
    - We will use the X–11 method, introduced by Shiskin in 1965.
- Training a model for data fitting can lead to the problem of overfitting or underfitting:
    - Ends up training inaccurate and less powerful models for classification or regressions tasks;

Introduction (cont.)

- Time-series analysis requires vital factors to be considered, e.g., autocorrelation, seasonality or stationarity in the data:
  - Make relations between the independent variable(s) and the dependent variable(s) being often obfuscated;
  - We will use the X–11 method, introduced by Shiskin in 1965.
- Training a model for data fitting can lead to the problem of overfitting or underfitting:
  - Ends up training inaccurate and less powerful models for classification or regressions tasks;
  - We will use the variance and the Univariate Feature Selection (based on F-tests).

Methods and Algorithms

Seasonal Adjustment

- 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 (seasonality);

Seasonal Adjustment

- 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 (seasonality);
- The relations between the independent variable(s) and the dependent variable(s) are often obfuscated by these periodic fluctuations;

Seasonal Adjustment

- 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 (seasonality);
- The relations between the independent variable(s) and the dependent variable(s) are often obfuscated by these periodic fluctuations;
- One should remove the seasonal variation of the time-series, in order to obtain a different time-series, but with less dispersion than original time-series;

Seasonal Adjustment

- 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 (seasonality);
- The relations between the independent variable(s) and the dependent variable(s) are often obfuscated by these periodic fluctuations;
- One should remove the seasonal variation of the time-series, in order to obtain a different time-series, but with less dispersion than original time-series;
- The process of removing the seasonal variation is known as **seasonal adjustment**.

Seasonal Adjustment (cont.)

- Time-series are decomposed into three basic patterns: trend (t), seasonality (s) and irregular variations (ϵ);

Seasonal Adjustment (cont.)

- Time-series are decomposed into three basic patterns: trend (t), seasonality (s) and irregular variations (ϵ);
- When the variance does not change over the time-series, one time-series is said to follow an **additive model**;

Seasonal Adjustment (cont.)

- Time-series are decomposed into three basic patterns: trend (t), seasonality (s) and irregular variations (ϵ);
- 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**;

Seasonal Adjustment (cont.)

- Time-series are decomposed into three basic patterns: trend (t), seasonality (s) and irregular variations (ϵ);
- 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**;
- Two types of models: additive model and multiplicative model.

Seasonal Adjustment (cont.)

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

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

Seasonal Adjustment (cont.)

- 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:

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

Seasonal Adjustment (cont.)

- 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) (multiplicative model):

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

Seasonal Adjustment (cont.)

- 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) (multiplicative model):

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

- 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);

Seasonal Adjustment (cont.)

- 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) (multiplicative model):

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

- 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**.

Seasonal Adjustment: The X–11 Method

- The X–11 iterative 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;

Seasonal Adjustment: The X–11 Method

- The X–11 iterative 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;

Seasonal Adjustment: The X–11 Method

- The X–11 iterative 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;

Seasonal Adjustment: The X–11 Method

- The X–11 iterative 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.

Seasonal Adjustment: The X–11 Method (cont.)

- A two-sided moving average (MA) is a statistical procedure identical to a simple average; however, the average is centred in each observation k, acting like a filter, in order to estimate a component;

Seasonal Adjustment: The X–11 Method (cont.)

- A two-sided moving average (MA) is a statistical procedure identical to a simple average; however, the average is centred in each observation k, acting like a filter, in order to estimate a component;

- The MA can be expressed as:

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

  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:

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

Seasonal Adjustment: The X–11 Method (cont.)

- A two-sided moving average (MA) is a statistical procedure identical to a simple average; however, the average is centred in each observation k, acting like a filter, in order to estimate a component;

- The MA can be expressed as:

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

  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:

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

- The Henderson filter is a more elaborated MA mechanism, that considers both symmetrical and asymmetrical weights.

Feature Selection

- 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);

Feature Selection

- 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);
- 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;

Feature Selection

- 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);
- 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: it only considers the relationship between each independent variable and its importance for the output variable(s).

Feature Selection (cont.)

- The importance of each independent feature to the output variable(s) is calculate by the F-statistic, given by:

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

  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;

Feature Selection (cont.)

- The importance of each independent feature to the output variable(s) is calculate by the F-statistic, given by:

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

  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;

- 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.

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 in 1995;

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 in 1995;
- PSO uses particles to explore the d-dimensional search space of the fitness function;

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 in 1995;
- PSO uses particles to explore 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;

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 in 1995;
- PSO uses particles to explore 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;
- 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}).

Particle Swarm Optimisation (cont.)

- The volocity of each particle is updated as:

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

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;

Particle Swarm Optimisation (cont.)

- The volocity of each particle is updated as:

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

  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:

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

Particle Swarm Optimisation (cont.)

- 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;

Particle Swarm Optimisation (cont.)

- 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;
- For finding the optimal weights of an 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;

Particle Swarm Optimisation (cont.)

- 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;
- For finding the optimal weights of an 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.

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;

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;

- 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:

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

being m the number of samples in the data set.

Levenberg–Marquardt Backpropagation (cont.)

- 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 $\mathbf{P} + \delta$;

Levenberg–Marquardt Backpropagation (cont.)

- 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:

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

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

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

Levenberg–Marquardt Backpropagation (cont.)

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

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

Levenberg–Marquardt Backpropagation (cont.)

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

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

- The vectorised form of the above equation is as follows:

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

Levenberg–Marquardt Backpropagation (cont.)

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

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

  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;

Levenberg–Marquardt Backpropagation (cont.)

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

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

  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:

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

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

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

Levenberg–Marquardt Backpropagation (cont.)

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

Levenberg–Marquardt Backpropagation (cont.)

- The λ is adjusted at each iteration of the algorithm to ensure the minimisation of $\mathbf{S}(\mathbf{P})$;
- The LMA is also called the mixture of Gauss–Newton method and Gradient Descent;

Levenberg–Marquardt Backpropagation (cont.)

- The λ is adjusted at each iteration of the algorithm to ensure the minimisation of $\mathbf{S}(\mathbf{P})$;
- The LMA is also called the mixture of Gauss–Newton method and 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 $\mathbf{S}(\mathbf{P})$, 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.

Results

Results

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.

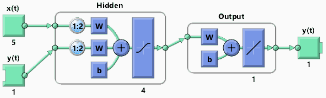

Figure 1: An example of the NARX in open-loop for the urban traffic in São Paulo – Brazil – data set.

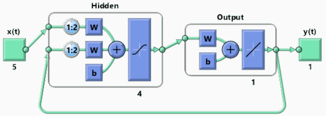

Figure 2: An example of the NARX in closed-loop for the urban traffic in São Paulo – Brazil – data set.

Results (cont.)

- The ANNs were tested ten times for each number of neurons in the hidden layer considered: 4, 7, 10, 12, 15 and 20;

Results (cont.)

- The ANNs were tested ten times for each number of neurons in the hidden layer considered: 4, 7, 10, 12, 15 and 20;
- Initial weights were set in the range $[-2.4/I, 2.4/I]$, where I is the number of inputs;

Results (cont.)

- The ANNs were tested ten times for each number of neurons in the hidden layer considered: 4, 7, 10, 12, 15 and 20;
- 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;

Results (cont.)

- The ANNs were tested ten times for each number of neurons in the hidden layer considered: 4, 7, 10, 12, 15 and 20;
- 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;
- The maximum 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;

Results (cont.)

- 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, with $\omega = 0.9$, $l_1 = 0.5$ and $l_2 = 0.3$;

Results (cont.)

- 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, with $\omega = 0.9$, $l_1 = 0.5$ and $l_2 = 0.3$;

- **Mean Squared Error (MSE):**

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

  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;

Results (cont.)

- 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, with $\omega = 0.9$, $l_1 = 0.5$ and $l_2 = 0.3$;

- **Mean Squared Error (MSE):**

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

  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;

- **Coefficient of Determination (R^2):**

$$R^2 = 1 - \frac{SS_r}{SS_t}, \tag{19}$$

  where SS_r is sum of squared error for the model to be evaluated, and SS_t is the model with slope 0.

Results: Urban Traffic in São Paulo – 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.

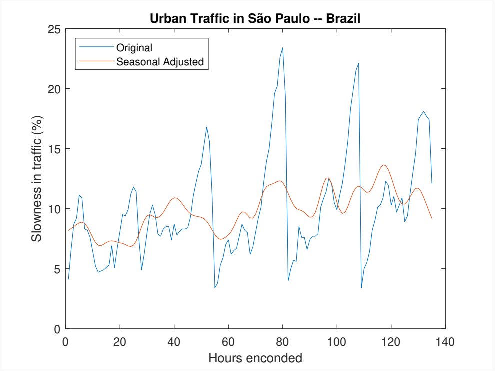

Figure 3: The original and the seasonally adjusted time-series of the urban traffic in São Paulo – Brazil.

Results: Urban Traffic in São Paulo – Brazil (cont.)

Table 1: 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
⋮	⋮

Results: Urban Traffic in São Paulo – Brazil (cont.)

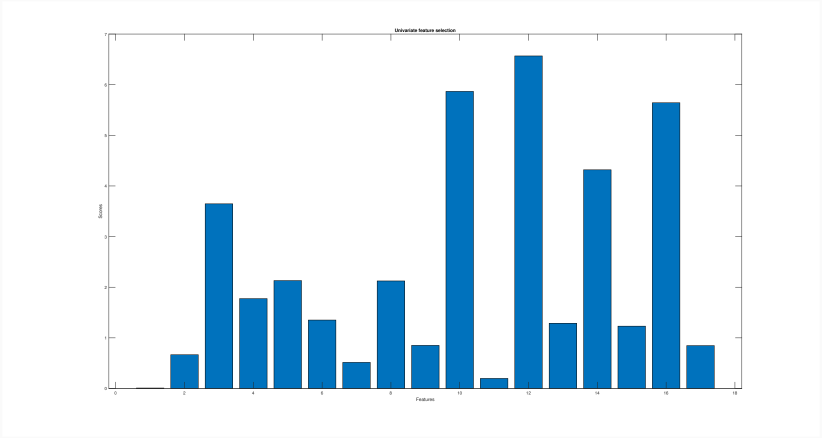

Figure 4: Feature selection of the urban traffic in São Paulo – Brazil – data set.

Selected features: broken truck, lack of electricity, point of flooding, defect in the network of trolleybuses and semaphore off.

Results: Urban Traffic in São Paulo – Brazil (LMA)

Table 2: 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	0.0584	0.9870	1.6688	0.2177	0.8305	0.6104
7	0.2605	0.9253	2.6266	0.8535	0.9939	0.7292
10	0.6885	0.7948	1.1546	0.4983	0.9400	0.1718
12	0.4691	0.8258	2.4525	0.0869	1.0511	0.5918
15	0.5403	0.8142	1.5761	0.7185	1.1118	0.3289
20	0.4573	0.8390	2.6734	0.5107	1.4723	0.7814

The lowest MSE value was obtained using an ANN with four hidden units. When analysing the correlation coefficients, it is possible to verify that a lower number of hidden units is more beneficial to the algorithm, showing always a strong positive relationship.

Results: Urban Traffic in São Paulo – Brazil (LMA) (cont.)

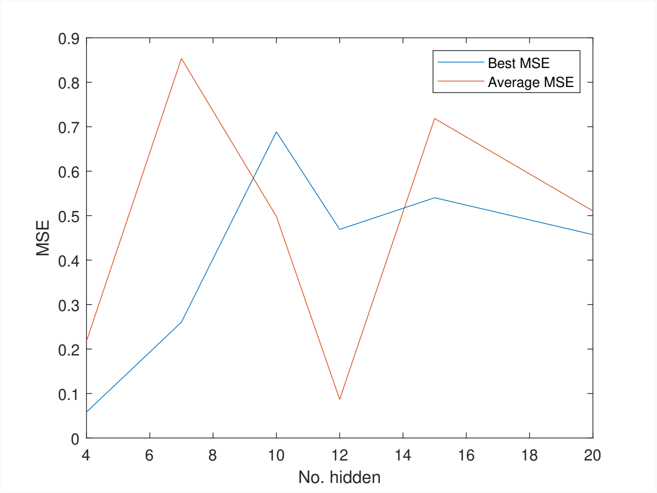

Figure 5: Comparison between the best MSE and average MSE of the urban traffic in São Paulo – Brazil – test data set.

Results: Urban Traffic in São Paulo – Brazil (LMA) (cont.)

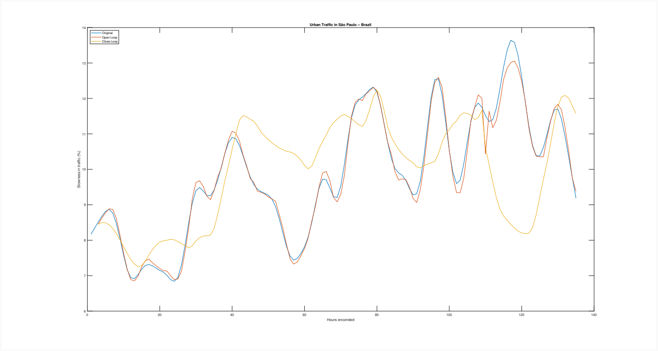

Figure 6: The original, open-loop and closed-loop predictions of the urban traffic in São Paulo – Brazil – test data set.

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.

Results: Urban Traffic in São Paulo – Brazil (PSO)

Table 3: 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	101.7226	0.2068	5109.8107	-0.3671	1562.1835	1767.8574
7	421.4021	-0.1829	5529.2300	0.3162	1935.4416	1609.4869
10	683.9212	0.5967	3672.3913	0.3595	1954.6482	887.6399
12	1976.4206	0.4596	6737.4916	-0.0698	2998.1264	1391.8016
15	1079.4149	-0.1617	7318.6149	0.4696	2807.7580	1926.8093
20	1618.9738	-0.5214	10345.2000	-0.4959	4765.2654	2521.8635

On possible justification for these results 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.

Results: Urban Traffic in São Paulo – Brazil (PSO) (cont.)

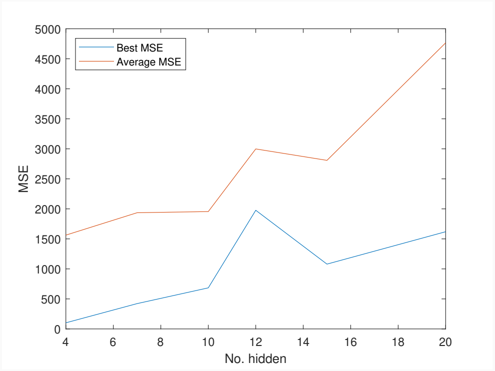

Figure 7: Comparison between the best MSE and average MSE of the urban traffic in São Paulo – Brazil – test data set.

On average, as the number of hidden units increases, the worse the MSE gets!

Results: Monthly Milk Production

- Includes the monthly production of milk per cow from January 1962 to December 1975;

Results: Monthly Milk Production

- Includes the monthly production of milk per cow from January 1962 to December 1975;
- 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.

Results: Monthly Milk Production

- Includes the monthly production of milk per cow from January 1962 to December 1975;
- 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.

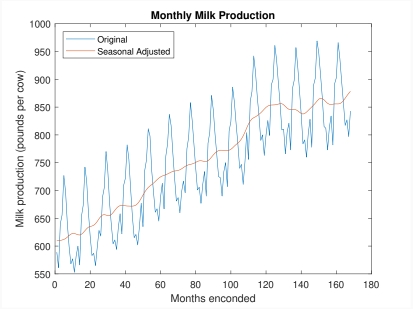

Figure 8: The original and the seasonally adjusted time-series of milk production.

Results: Monthly Milk Production (LMA)

Table 4: 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	1.2323	0.9937	4151.0826	0.8037	509.6915	1301.3939
7	19.5716	0.9892	4405.9362	0.2970	1018.2642	1451.0654
10	11.7560	0.9798	3546.5183	-0.6988	660.5850	1104.5266
12	37.6958	0.5780	8960.1478	0.5213	1639.6111	2800.5876
15	18.5800	0.9146	2253.8357	0.5751	647.8629	814.4541
20	208.5665	0.4306	35310.1453	0.7500	6585.3965	11564.4728

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

Results: Monthly Milk Production (LMA) (cont.)

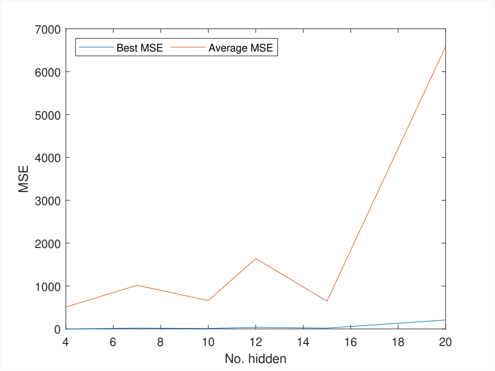

Figure 9: Comparison between the best MSE and average MSE of the monthly milk production test data set.

The STD and average MSE are both 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.

Results: Monthly Milk Production (PSO)

Table 5: 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	12860.8945	-0.8544	671046.2823	-0.0696	208352.4323	237863.3288
7	7813.1833	-0.5541	531818.7168	0.4953	165247.0502	170387.5299
10	6599.4517	0.1143	1163303.5879	0.3458	372083.0204	415800.9392
12	2152.8343	0.8150	894676.5407	0.7162	205431.9951	281434.4598
15	5295.8345	-0.6825	620003.5429	0.4642	134927.0841	193029.7375
20	6088.2057	-0.3090	1325615.3272	-0.1316	325474.5211	440254.7112

In this case, the ANN with the lowest MSE is one with twelve hidden units. We did not find any relationship between the number of hidden units and the accuracy of the algorithm.

Results: Monthly Milk Production (PSO) (cont.)

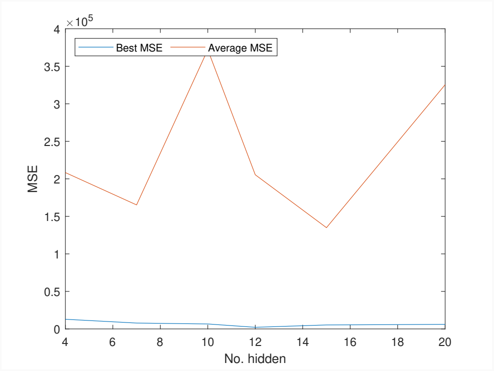

Figure 10: Comparison between the best MSE and average MSE of the monthly milk production test data set.

Overall results indicate that accuracy of the ANN for each number of hidden units is not stable between tests.

Results: QSAR fish toxicity

It includes 908 samples with five features and acute aquatic toxicity for fish as a target attribute.

Table 6: 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

Results: QSAR fish toxicity (cont.)

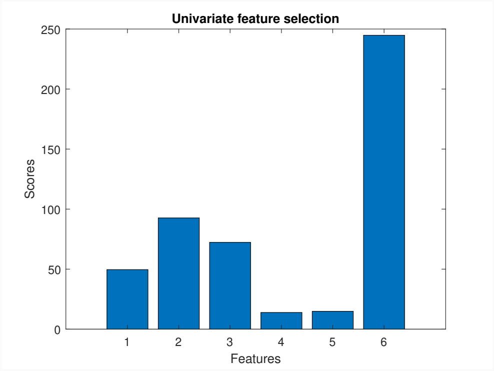

Figure 11: Feature selection of the QSAR fish toxicity data set.

Selected features: MLOGP, CIC0, $SM1_{Dz}$ and GATS1i.

Results: QSAR fish toxicity (LMA)

Table 7: Summary of tests performed with LMA for fish test data set.

No. hidden	**Best MSE**	**R**	**Worst MSE**	**R**	**Average MSE**	**STD**
4	0.6901	0.7446	0.9134	0.6537	0.7628	0.0690
7	0.6969	0.7418	0.8341	0.7007	0.7686	0.0470
10	0.7696	0.7101	0.9520	0.6352	0.8571	0.0604
12	0.8599	0.6794	1.1434	0.5610	0.9529	0.0783
15	0.8293	0.6878	1.0093	0.6098	0.9268	0.0452
20	0.7843	0.7044	1.0096	0.6361	0.9116	0.0711

The best performance was achieved at using the ANN with four hidden units, achieving the minimum best MSE and average MSE, and maximum R value, whereas ANN with twelve neuron units in the hidden layer performs worst achieving the highest MSE and one of the lowest R values. ANNs with four and seven hidden units achieved similar MSE values.

Results: QSAR fish toxicity (LMA) (cont.)

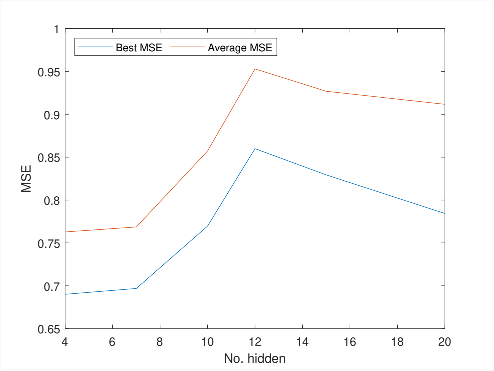

Figure 12: Comparison between the best MSE and average MSE of the fish test data set.

The LMA algorithm was shown in this case to be stable between executions, since a low STD was always found.

Results: QSAR fish toxicity (PSO)

Table 8: Summary of the tests performed with PSO in fish test data set.

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.8711	0.6727	1.2119	0.4921	0.9907	0.1029
7	0.7354	0.7289	3.1363	0.3525	1.1536	0.7176
10	1.0336	0.5988	8.2609	0.1446	2.7976	2.1848
12	1.0483	0.6042	14.7689	0.0228	6.3428	5.4560
15	1.6856	0.4882	18.2857	0.2377	5.6543	5.0045
20	4.9660	0.3646	56.8326	-0.0426	24.3417	17.7986

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.

Results: QSAR fish toxicity (PSO) (cont.)

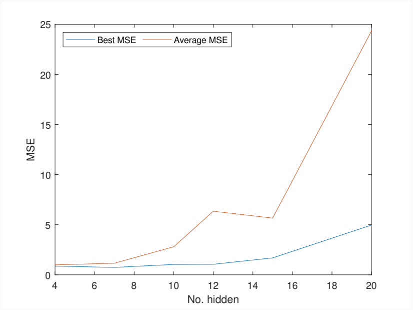

Figure 13: Comparison between the best MSE and average MSE of the fish test data set.

It is noteworthy that the average MSE and STD increase 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.

Results: QSAR fish toxicity (Comparison)

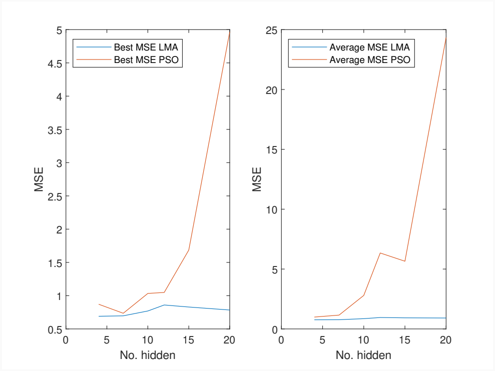

Figure 14: Comparison between LMA and PSO in terms MSE.

It is important to note that both LMA and PSO achieved very similar best MSE values (0.6901 and 0.7354, respectively); however, the LMA algorithm just needed four hidden units, while the PSO required seven.

Results: Wine Quality

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.

Table 9: 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
⋮	⋮

Table 10: 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
⋮	⋮

Results: Wine Quality (cont.)

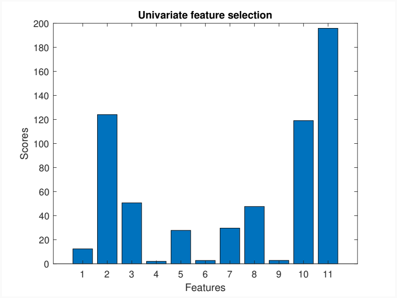

Figure 15: Feature selection of the red wine data set.

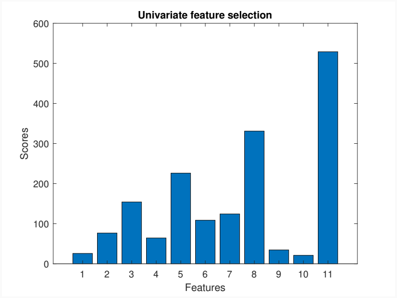

Figure 16: Feature selection of the white wine data set.

Selected features: red wine data set — fixed acidity, volatile acidity, citric acid, chlorides, total sulfur dioxide, density, sulphates and alcohol; white wine data set — citric acid, chlorides, density and alcohol.

Results: Red Wine Quality (LMA)

Table 11: Summary of tests performed with LMA for red wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.5221	0.4901	0.6281	0.3816	0.5759	0.0312
7	0.5222	0.4887	0.5629	0.4490	0.5381	0.0138
10	0.4864	0.5602	0.5727	0.4074	0.5442	0.0300
12	0.5044	0.5080	0.5705	0.4207	0.5392	0.0224
15	0.4813	0.5534	0.5906	0.3901	0.5337	0.0351
20	0.5024	0.5072	0.5484	0.4683	0.5290	0.0160

The least MSE was achieved by fifteen hidden units containing ANN, whereas the worst performance was achieved by ANN with four or seven hidden units, in terms of best MSE; however, the best MSE and R are not able to suggest a single architecture.

Results: Red Wine Quality (LMA) (cont.)

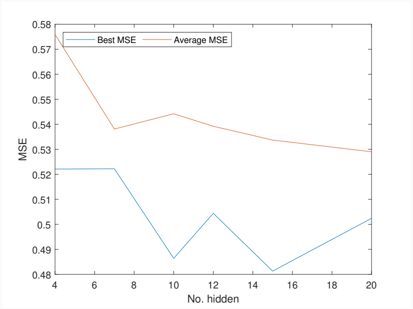

Figure 17: Comparison between the best MSE and average MSE of the red wine data set.

It is possible to identify a trend in the data, since as 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.

Results: Red Wine Quality (PSO)

Table 12: Summary of tests performed with PSO for Red wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.5327	0.4976	0.6331	0.2548	0.5842	0.0328
7	0.4779	0.5430	0.6755	0.2621	0.5662	0.0491
10	0.5292	0.4869	1.0436	0.3861	0.7293	0.2217
12	0.5628	0.4153	9.9891	0.1801	2.3868	2.9127
15	1.3739	0.2977	9.4930	0.1327	4.6305	3.1891
20	6.7498	-0.0856	114.2235	-0.0363	40.0774	29.8391

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.

Results: Red Wine Quality (PSO) (cont.)

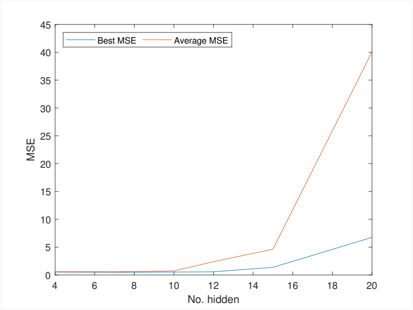

Figure 18: Comparison between the best MSE and average MSE of the red wine data set.

Figure 18 depicts an exponential growth of the best and average MSE, with the increase in the number of hidden units in the hidden layer.

Results: Red Wine Quality (Comparison)

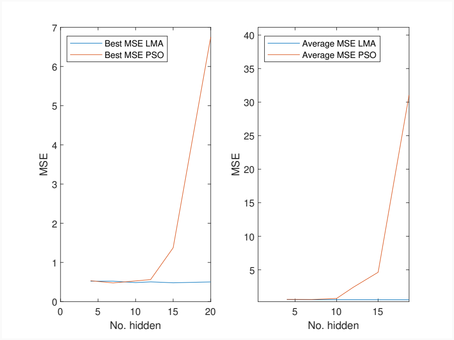

Figure 19: Comparison between LMA and PSO in terms of best MSE.

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 0.4779, whereas LMA required fifteen hidden units to get the best MSE of 0.4813.

Results: White Wine Quality (LMA)

Table 13: Summary of tests performed with LMA for white wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.4863	0.4570	0.5223	0.4277	0.5083	0.0117
7	0.4984	0.4549	0.5253	0.4262	0.5131	0.0102
10	0.4861	0.4774	0.5378	0.3995	0.5136	0.0155
12	0.4917	0.4544	0.5342	0.4085	0.5145	0.0159
15	0.4926	0.4716	0.5364	0.4272	0.5171	0.0127
20	0.4788	0.4660	0.5463	0.4022	0.5109	0.0220

The least MSE was achieved by twenty hidden units containing ANN, whereas the worst performance was achieved by ANN with seven hidden units, in terms of best MSE; however, like the red wine data set, the best MSE and R are not able to suggest a single architecture.

Results: White Wine Quality (LMA) (cont.)

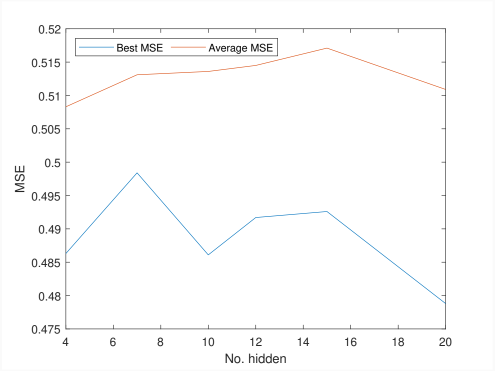

Figure 20: Comparison between the best MSE and average MSE of the white wine data set.

As 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.

Results: White Wine Quality (PSO)

Table 14: Summary of tests performed with PSO for white wine test data set

No. hidden	Best MSE	R	Worst MSE	R	Average MSE	STD
4	0.5070	0.3980	0.8161	0.2374	0.5675	0.0900
7	0.5020	0.4176	0.7615	0.2788	0.5667	0.0801
10	0.5356	0.3480	4.6563	0.0293	1.0719	1.2671
12	0.6051	0.3716	2.8742	0.1766	1.2500	0.7651
15	0.7335	0.2653	7.7688	-0.0237	2.8896	2.3616
20	3.7630	-0.0330	14.4324	0.1248	7.1729	3.4411

The least MSE value was achieved by ANN with seven 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.

Results: White Wine Quality (PSO) (cont.)

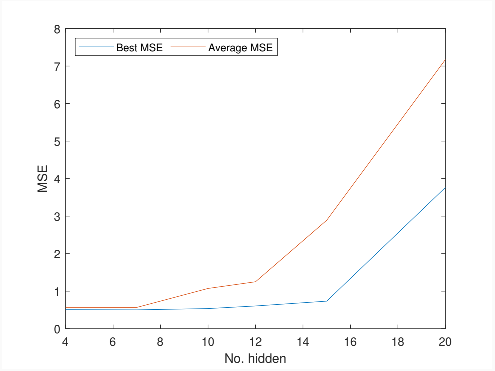

Figure 21: Comparison between the best MSE and average MSE of the white wine data set.

As the number of hidden units increase, the MSE value increases to (unlike the LMA).

Results: White Wine Quality (Comparison)

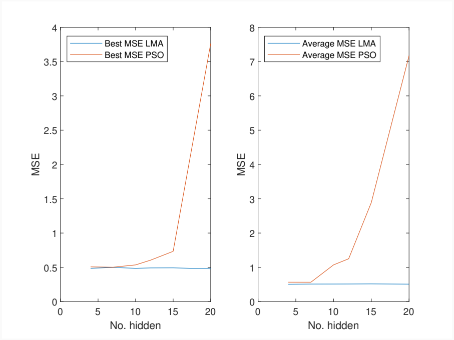

Figure 22: Comparison between LMA and PSO in terms of best MSE.

In this example, to reach to similar best MSE level, PSO needed a less number of hidden units, when compared to LMA. However, PSO is not able to explain the relationship between independent and dependent variables so well.

Conclusion

Conclusion

Table 15: 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	0.0584	0.9870	4	101.7226	0.2068
Milk	4	1.2323	0.9937	12	2152.8343	0.8150
Fish	4	0.6901	0.7446	7	0.7354	0.7289
Red wine	15	0.4813	0.5534	7	0.4779	0.5430
White wine	20	0.4788	0.4660	7	0.5020	0.4176

- 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.

Conclusion

Table 15: 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	0.0584	0.9870	4	101.7226	0.2068
Milk	4	1.2323	0.9937	12	2152.8343	0.8150
Fish	4	0.6901	0.7446	7	0.7354	0.7289
Red wine	15	0.4813	0.5534	7	0.4779	0.5430
White wine	20	0.4788	0.4660	7	0.5020	0.4176

- 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.
- However, for the same accuracy, the PSO algorithm required a lower number of hidden units, as can be seen in Table 15;

Conclusion

Table 15: 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	0.0584	0.9870	4	101.7226	0.2068
Milk	4	1.2323	0.9937	12	2152.8343	0.8150
Fish	4	0.6901	0.7446	7	0.7354	0.7289
Red wine	15	0.4813	0.5534	7	0.4779	0.5430
White wine	20	0.4788	0.4660	7	0.5020	0.4176

- 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.
- However, for the same accuracy, the PSO algorithm required a lower number of hidden units, as can be seen in Table 15;
- **Future work:** test the PSO algorithm with different parameters, different velocity update equations and different topologies of swarms.

Thank You!

GitHub:

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