Identification of MIMO systems using MLP Networks: Comparison between SVR and random initialisation

Hajer Zardoum, Nawel Mensia, Moufida Ksouri
University of Tunis El Manar, National School of Engineering of Tunis
Analysis, Conception and Control of Systems Laboratory (LR-11-ES20)
BP, 37, Le BELVEDERE, 1002 Tunis, Tunisia
(e-mail: hajer_zd@yahoo.fr)

Abstract: Neural network (NN) modelling approach is often used for non-linear system identification. Building a NN for some identification problem starts by choosing its structure and initial weights. There is no exact method to determine the optimal initialisation for a NN, but some authors have used support vector regression (SVR) to initialise a RBFNN which could be considered as a systematic way.

This paper presents a SVR initialisation method for Multi-Layer Perceptron (MLP) NN. The proposed method is based on the analogy between NN and SVR to determine the necessary number of hidden neurons and the initial weights for a given modelling precision. Simulation results for multi-input multi-output (MIMO) system show the feasibility and accuracy of the proposed method.

I. Introduction

The most used neural networks for non-linear system identification are the Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) NNs [1]-[3]. Both NNs are considered as universal non-linear approximators for complex non-linear systems. RBFNNs are less complicated than MLPNNs, but they have less generalisation capabilities.

It is well known that the performance of a given NN architecture critically depends on its structure as well as on its training algorithm. Besides, its convergence, speed, and generalisation depend on its weight initialisation [4], [5].

Before training a neural network, its initial structure must be determined. Several initialisation methods were used in the literature; the simplest and commonly used one is the empirical method which consists in varying the number of neurons in a range of values fixed according to the experience of the designer, and selecting the number that gives the best performance. Another type of method [6] starts from a minimum number of neurons and increases it till having the desired error, or moves from an exhaustive number of neurons and decreases it till obtaining the desired performance. Some theoretical studies [7], [8] tented to give the necessary number of hidden neurons function of the desired approximation order.

Methods cited above are not fast and practical to use especially in complex systems. However, a systematic way to initialise a neural network is proposed in [9], namely the support vector regression (SVR) method. SVR is an identification method that leads to the statistical learning theory; it consists in transforming a non-linear function approximation in to linear expansion identification by using some typical functions called kernels. Support vectors are data points that serve to smooth the function under an approximation tube which represents the error [9]-[12]. SVR method was used in [9] to initialise a RBFNN by determining the number of hidden layer nodes, the initial parameters of the kernel and the initial weights.

In this paper, the SVR method is adopted to initialise a MLPNN, and an instantaneous learning error minimisation algorithm based on the back-propagation gradient descent method is used to train the NN. The paper is organised as follows: in sectionII the SVR approach is presented and its applicability to initialise a MLP neural network is explained. The description of the training algorithm is given in sectionIII. Finally simulation results of the identification of a two-input two-output non-linear system are presented in section IV.

II. Initial structure of the MLP network by the SVR approach

SVR is an identification method extracted from the statistical learning theory introduced by Vapnick. It can approximate an unknown function from its input output samples that has at most deviation in the e-tube and is as flat as possible [10], [11]. The non-linear function to be estimated is written as:

\[ y = \sum_{i=1}^{l} w_i \phi_i(x) + b \] (1)

Parameters of this equation are estimated by minimizing the regularized risk function \( R(C) \) (2).

\[ R(C) = C \frac{1}{N} \sum_{i=1}^{N} L_e(d_i, y_i) + \frac{1}{2} \|w\|^2 \] (2)

\[ E_{risk} = \frac{1}{N} \sum_{i=1}^{N} L_e(d_i, y_i) \] (3)

The regularized risk (2) is composed of two terms and a constant C. The first term (3) called the empirical risk is measured by the insensitive loss function, which select data for regression that is only inside an e-tube.

The second term \( \frac{1}{2} \|w\|^2 \) measures the function flatness.

C is a regularized constant that determines the trade-off between the training error and the model flatness [13]. It was demonstrated in [9]-[12], that the solution of such minimisation problem with the use of Lagrange multipliers \((\alpha_c, \alpha_c^*)\), needs some specified functions called kernels \((k(x_c, x))\). This functions transform non-linear data processing on to a linear problem. The solution is given by the following expression:
The resolution of the identification problem consists in finding the Lagrange multipliers that satisfy the above mentioned constraints (5). It uses a quadratic programming algorithm [10]. It is demonstrated that Lagrange multipliers ($\alpha_i$, $\alpha_i^*$) are null for points lying inside the $\varepsilon$-tube, however points outside the $\varepsilon$-tube have non zeros Lagrange multipliers and are called support vectors (fig1). Support vectors weighted by their corresponding Lagrange multipliers serve to determine the solution.

SVR have many kernels, the commonly used ones being the linear (6), the polynomial (7), the RBF (8) and the sigmoid (9) functions. Among these kernels, only the RBF and the sigmoid are related to neural networks namely the RBF and the MLP NNs.

$$f(x_j) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(x_i,x_j) + b \quad (4)$$

$$0 \leq \alpha_i^* \leq C, \quad i=1,...,l$$

$$\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 \quad (5)$$

The idea proposed by [9] is to initialise a RBFNN by SVR. It gives the number of neurons, the centers of the RBF activation functions and their corresponding weights. It was demonstrated in [9] that this approach gives good results.

But what about MLPNN? Could sigmoid SVR give initialisation to MLPNN? Too few works were done about the relationship between sigmoid SVR and MLPNN. To be a SVR kernel a function must satisfy Mercer’s conditions [11]: it must be PSD (positive semi-defined). It was demonstrated in [14] that sigmoid parameters $\gamma$ and $r$ of (7) must meet some specified conditions to make the sigmoid function PSD and therefore a suitable kernel. This constraint makes it quite difficult to select suitable parameters for the sigmoid kernel [15].

$$y(t+1) = \sum_{i=1}^{N} w_i \tanh(v_i x + b) \quad (10)$$

$$y(t+1) = \sum_{i=1}^{N} w_i \exp\left(-\frac{\|x_m-x_i\|^2}{2\sigma^2}\right) \quad (11)$$

When using a sigmoid function (9) as kernel for the SVR, and comparing it to the MLP activation function (10), the term $\gamma x_i$ which represents the support vectors multiplied by the sigmoid constant $\gamma$, and their corresponding weights can be regarded respectively as the initial weights $v_i$ of the input layer and the initial weights $w_i$ of the hidden layer (fig2). The number of hidden neurons is the number of support vectors.

### III. Back-propagation gradient descent learning algorithm for the MLPNN

The MLP is trained with an instantaneous learning error minimisation algorithm based on the back-propagation gradient descent method. The input data are propagated through the network, the output is determined, and the error between network and system outputs is then calculated. The weights are adjusted by back-propagation of the instantaneous error through the network according to the following equations:

$$J = \frac{1}{2} (y_{NN} - y_{real})^2 \quad (12)$$

$$\Delta \omega_v (k) = -\mu \frac{\partial J(t)}{\partial \omega_v} \quad (13)$$

Where $J$ is the quadratic error between the NN and the system outputs. $y_{NN}$ and $y_{real}$ are the NN and the system outputs respectively.
The identification approach is constructed as follows:
Step 1: Run SVR toolbox [10].
Step 2: Give Initialisation to the MLP with the SVR results (the number of neurons is the number of SVs and the initial weights \(v_i\) and \(w_i\) are respectively \(\gamma SVs\) and their corresponding weights).
Step 3: Train the MLPNN by the back-propagation gradient descent method.

IV. Simulation results

In this section, the accuracy of the MLP initialisation by SVR is tested for a MIMO system.

SVR is a naturally multi-input simple-output (MISO) method, when the system to be identified is MIMO, we apply the method to each output separately. Consequently the NN identification requires constructing NNs as many outputs as the system has.

a. The MIMO Plant:

The MIMO system to be identified here is widely used as a benchmark [6], [9], [16]. It is a two-input two-output plant described by the following equations:

\[
\begin{bmatrix}
  y_1(k+1) \\
  y_2(k+1)
\end{bmatrix} =
\begin{bmatrix}
  \frac{y_1(k)}{1 + y_1^2(k)} \\
  \frac{y_1(k) y_2(k)}{1 + y_2^2(k)}
\end{bmatrix} + \begin{bmatrix}
  x_1(k) \\
  x_2(k)
\end{bmatrix}
\]

\( (x_i(k), x_i(k)) = \left( \sin(\frac{2\pi k}{25}), \cos(\frac{2\pi k}{25}) \right) \) for \(0 \leq k \leq 100\)

b- Initialisation of the NN with sigmoid SVR

After some testing simulations with a grid search for the best parameters, we noticed that the support vector number (nsv) depends on C and \(\varepsilon\) (§II). For a given value of C a too little \(\varepsilon\) leads to a big number of support vectors, and a big \(\varepsilon\) reduces the nsv. So, to have an acceptable nsv which could be taken as the MLP number of hidden neurons \(\varepsilon\) should be set between 0.2 and 0.5.

Table 1 shows the parameters and the results of the SVR for both \(y_1\) and \(y_2\). The number of support vectors nsv obtained for \(\varepsilon=0.3\) is 26.

<table>
<thead>
<tr>
<th>(y_1)</th>
<th>(\Gamma)</th>
<th>(r)</th>
<th>(\varepsilon)</th>
<th>C</th>
<th>nsv</th>
<th>rmse</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
<td>6</td>
<td>6</td>
<td>0.3</td>
<td>8</td>
<td>26</td>
<td>0.2665</td>
</tr>
<tr>
<td>(y_2)</td>
<td>6</td>
<td>6</td>
<td>0.3</td>
<td>5</td>
<td>26</td>
<td>0.4006</td>
</tr>
</tbody>
</table>

Table 1: SVR parameters and results

\(rmse = \sqrt{\frac{1}{N}\sum(y_{NN} - y_{real})^2}\)

\(\gamma\) et \(r\): sigmoid parameters of equation (9).

Simulation results for \(y_1\) and \(y_2\) are shown in fig.3. We notice that SVR can identify the system with the prefixed accuracy. The chosen value of \(\varepsilon=0.3\) is sufficient because the SVR is not used here as a fair identification method but is only used to identify the system partially and rapidly. The SVR identification results are taken as a starting point for a MLP neural network identification.

c- System identification with MLPNN:

As explained in the above sections, both system outputs \(y_1\) and \(y_2\) are modelled with two MLPNNs which are initialised by the SVR and trained with the back-propagation gradient descent learning algorithm. The parameters and the results of the training algorithms for both \(y_1\) and \(y_2\) are summarised in table 2.

The training RMSE plots are shown in fig.5. and the responses of the plant and the identification model for \(y_1\) and \(y_2\) are presented in fig. 4. These results show that the training algorithms initialised with SVR converge and the RMSE values of \(y_1\) and \(y_2\) after 1000 training epochs are found to be 0.053 and 0.042 respectively. The output curves of the system plotted in fig.4 show that the NN model can identify the system with a good accuracy.

<table>
<thead>
<tr>
<th>(y_1)</th>
<th>(\eta_{init})</th>
<th>(rmse_{init})</th>
<th>(rmse_{final})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
<td>26</td>
<td>0.09</td>
<td>0.2665</td>
</tr>
<tr>
<td>(y_2)</td>
<td>26</td>
<td>0.1</td>
<td>0.4006</td>
</tr>
</tbody>
</table>

Table 2: Parameters and results of the NN identification
d- Comparison between SVR and random initialisation methods:

To show the effectiveness of the proposed method, we have compared it to random initialisation of the weights. The number of hidden neurons is the same as taken in the SVR initialisation method as explained in (§II.2).

Since random values of the weights could vary from an operator to another, we made simulations for three different random ensembles of weights. The training RMSE, plotted in fig. 14, shows a good convergence of the training algorithms for the SVR and the different random initialisations of the weights.

Simulation results versus time are shown in fig. 8-13. As it can be noticed, both initialisation methods give a good convergence of the training algorithm. By comparing the zoomed plots of \( y_1 \) and \( y_2 \) in fig.9 and fig.11, we find that responses of the identified model initialised with SVR are closer to the plant ones than those of the model initialised with random weights.

Based on all above results, we can conclude that SVR initialisation for MLP could be a good method where there is no knowledge about the system. It is a systematic way to determine the necessary number of neurons and their respective initial weights.
Fig. 8. Outputs of the plant and the identification model for $y_1$ after 1000 epochs for different initialisations.

Fig. 9. Zoom of fig.8.

Fig. 10. Outputs of the plant and the identification model for $y_2$ after 1000 epochs for different initialisations.

Fig. 11. Zoom of fig.10.

Fig. 12. The plots of $y_1(k)-y_{1in}(k)$ after 1000 epochs for different initialisation methods.

Fig. 13. The plots of $y_2(k)-y_{2in}(k)$ after 1000 epochs for different initialisation methods.

Fig. 14. Training rmse curves of the model for $y_1$ and $y_2$ for different initialisation methods.

V. Conclusion

In the majority of the NN initialisation methods, the number of hidden neurons is taken arbitrarily or progressively, and the initial weights are chosen randomly. We have proposed the use of SVR to initialise a MLPNN. This method allows determining the number
of hidden neurons as well as the initial weights for a given modelling accuracy. The NN is then trained with a classical back-propagation gradient descent learning algorithm. Compared with the random initialisation approach, the proposed method gives almost better identification accuracy with the advantage of determining systematically the necessary number of hidden neurons.

VI. References


