Improving neural networks generalization with new constructive and pruning methods

Marcelo Azevedo Costa, Antônio Pádua Braga and Benjamin Rodrigues de Menezes
Department of Electronics Engineering, Federal University of Minas Gerais, Campus da UFMG (Pampulha), Caixa Postal 209, CEP 30 161-970, Belo Horizonte, MG, Brazil
E-mail: {mcosta,apbraga,brm}@cpdee.ufmg.br

Abstract. This paper presents a new constructive method and pruning approaches to control the design of Multi-Layer Perceptron (MLP) without loss in performance. The proposed methods use a multi-objective approach to guarantee generalization. The constructive approach searches for an optimal solution according to the pareto set shape with increasing number of hidden nodes. The pruning methods are able to simplify the network topology and to identify linear connections between the inputs and outputs of the neural model. Topology information and validation sets are used.

1. Introduction

Neural networks generalization has been associated in the literature within the context of fitting model to the normally unknown problem complexity. Oversized networks are likely to result on overfitting and poor generalization if learning is based solely on training set error minimization. Model complexity is normally associated to network size, so pruning and constructive algorithms remove and add network nodes and weights in order to fit neural network complexity to the problem complexity. Another approach to generalization control are the validation methods, such as early stopping [1] and cross-validation [2], for which any model that minimizes the validation set error is good, regardless of its size. So, the number of parameters is not the issue in the validation approach. A more formal view for the generalization problem has been described by Geman et al. [3], who demonstrated that the mean squared error can be written as the sum of bias and variance errors. The bias error indicates how far from the data expected value is the average of the neural models and the variance error is a measure of the model variance in relation to the data expected value. A large variance is an indication of an oversized model and a large bias is an indication of a large output error. A proper balance between bias and variance is expected to result on a good generalization model.

Techniques for improving generalization response to unknown patterns include validation [4], cross-validation [2], pruning [5], network growing techniques [6], support-vector machines [7] and multi-objective optimization [8]. Pruning and network growing algorithms regard fitness as adjusting network complexity to problem complexity, where the concept of network complexity is associated to the number of free adjustable parameters (weights) in the network. Validation techniques do not consider the idea of complexity directly, since any network size could be selected, as long as it yields a small validation set error. All these algorithm aim at obtaining an appropriate balance between the bias and the variance of the model [4] in order to avoid under and over-fitting. Other important recent advances in the field have been given by Bartlett [9] and Teixeira et al. [8] who demonstrated that generalization is much more related to the norm of the weight vectors than to the actual number of network parameters.

The Multi-objective algorithm (MOBJ) for training neural networks [8] searches for a restricted set known as the Optimal Pareto set [10] that is obtained within the plane of two objectives: training set error ($E$) and norm of the weight vectors ($||w||$). The appropriate balance between the bias and variance can be achieved with the corresponding balance between the training set error and the norm of the weights. Therefore, solution with
good generalization performance could be obtained regardless of network size [9]. The solutions generated by the MOBJ algorithm can be tuned to be oversized and yet with good generalization performance. In this paper, oversized networks generated by the MOBJ algorithm will be used as initial solutions to be pruned by the algorithms that are proposed. Since the networks are known to be oversized in advance, their structure can be shrunked, what is appropriate to test the proposed algorithms. In addition to the pruning algorithms, a constructive algorithm based on the MOBJ algorithm is also proposed.

Pruning and constructive algorithms are presented in this paper based on the following principles: linearization of the hidden nodes, random sampling of weights, similarity of hidden nodes responses, hessian calculation and a mixture of some of these methods. The algorithms are compared with each other and with the original oversized MOBJ models.

2. Multi-objective network growing algorithm

The multi-objective network growing algorithm searches for a solution with improved generalization performance within the three-dimensional space defined by the following functions:

1. Error function \( E(w) \)
2. Norm function \( \|w\| \)
3. Number of Hidden nodes \( H \)

The method aims at controlling the bias and variance dilemma based on the multi-objective approach [8] but the third objective restricts the number of hidden nodes. Therefore, the method is able to find a solution with generalization performance and reduced network topology.

The optimization problem can be described according to Eq. (1).

\[
\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^{N(H)}} \left\{ \frac{E(\mathbf{w})}{\|\mathbf{w}\|} \right\}_H
\]

where the final weight vector, \( \mathbf{w}^* \), is \( N(H) \)-dimensional expressing that its number of coordinates or elements depends on the number of hidden nodes. The optimal solution represents the appropriate balance between the error, the euclidean norm of the weights and the networks topology (number of hidden nodes).

Figure 1 shows the tree-dimensional space and the optimal solutions represented by the Optimal Pareto sets for the discrete number of hidden nodes. The validation set is used to obtain the final solution. The method starts with one hidden node topology, which is gradually increased and for each new topology the Pareto set defined by error and norm functions are obtained using the MOBJ algorithms [8,11]. As the number of nodes increases, the Pareto set (Error \( \times \) Norm) converges to a stable form regardless of network complexity. The final solution represents the network topology with minimum number of hidden nodes and a Pareto set stable form.

3. Elimination of hidden nodes with linear responses

The idea of this method is to identify those (non-linear) hidden nodes whose response is similar to a node with linear activation function for the current range of input data. According to Fig. 2, a non-linear node with hyperbolic tangent activation function may have a linear output response. For a limited range of the outputs, \( f(-\psi) < f(x) < f(+\psi) \), the sigmoidal activation function could be replaced by a linear one. For every network node, the output range is calculated and, if its response is within the pre-established range \( f(-\psi) < f(x) < f(+\psi) \), it is substituted by a linear function within that range. The effect of linearizing hidden nodes is that the mapping between network input and output can be performed by a single linear transformation, simplifying network structure. The resulting network has a mixed structure with non-linear and linear hidden nodes. This method can be applied after training with a multi-objective algorithm, the network that represents also a solution with good generalization is then simplified. In some cases, such as the Card Data Set from PROBEN [12], that will be discussed further, the whole network structure was linearized. For that problem, that would be normally solved with a two-layer MLP, the linearization with the proposed method showed that the problem can be solved with a single layer network.

The corresponding MLP output equation for a two layer perceptron network with \( H \) hidden nodes, \( p \) outputs and \( N \) inputs, is

\[
y_p = f_p \left\{ \sum_{i=1}^{H} w_{2i} f_i \left[ \sum_{j=1}^{N} (x_j w_{1j}) + b_{1i} \right] \right\} + b_{2p}
\]

(2)
where $f_p$ and $f_i$ are activation functions, $w1, b1, w2$ and $b2$ are the respective input and output layers weights and biases.

Equation (2) can be expanded when a linear node $t$ is identified.

$$y_p = f_p \left\{ \sum_{i=1, i \neq t}^{H} \left( w_{2ip}.f_i \left( \sum_{j=1}^{N} (x_j.w_{1ji}) + b_1 \right) \right) + \sum_{j=1}^{N} (x_j.w_{1jt}.w_{2tp}) + (b_{1t}.w_{2tp} + b_{2p}) \right\}$$

An equivalent equation with linear connections between network inputs and outputs can be established:

$$y_p = f_p \left\{ \sum_{i=1, i \neq t}^{H} w_{2ip}.f_i \left( \sum_{j=1}^{N} (x_j.w_{1ji}) + b_1 \right) + \sum_{j=1}^{N} (x_j.w_{lnjp}) + b_{2p}' \right\}$$

An equivalent equation with linear connections between network inputs and outputs can be established:

$$\|\vec{f}_r^H - \vec{f}_t^H\| \approx 0 \quad (5)$$

Considering a MLP training set output response:

$$\vec{y}_p = f_p$$

$$\left\{ \sum_{i=1}^{H} w_{2ip}.\vec{f}_r^H + w_{2tp}.\vec{f}_t^H + w_{2tp}.\vec{f}_t^H + b_{2p} \right\}$$

based on Eq. (6), a vector $\vec{v}_p$ can be defined with the sum of the similar nodes outputs multiplied by their correspondent output weights.

4. Similar nodes response identification

This method works by identifying hidden nodes with similar responses, that could be substituted by a single one. In order to measure the similarity between two hidden nodes $r$ and $t$, the norm of their responses to the training set input vectors is calculated according to Eq. (5).
M.A. Costa et al. / Improving neural networks generalization with new constructive and pruning methods

\[ \vec{v}_p = w_{2rp} \vec{f}_r^H + w_{2tp} \vec{f}_t^H \]  

(7)

Quasi-similar nodes may be represented by the following relations: \( \vec{f}_r^H = \vec{a}, \vec{f}_t^H = \vec{a} + \vec{\xi} \), where \( \vec{\xi} \) consists on the error difference vector (\( \vec{\xi} = \vec{f}_r^H - \vec{f}_t^H \)). Vector \( \vec{f}_r^H \) is the reference vector \( \vec{a} \) in further calculations.

The vector \( \vec{v}_p \), which represents the sum of the two hidden nodes with minimum distance (Eq. 5), is defined by the reference vector, \( \vec{a} \), and the error difference vector, \( \vec{\xi} \):

\[ \vec{v}_p = (w_{2rp} + w_{2tp}) \vec{a} + w_{2tp} \vec{\xi} \]  

(8)

Based on Eq. 8, the vector \( \vec{v}_p \) presents an error difference vector that is multiplied by the non-reference vector output weight. In order to minimize the error vector influence, the reference vector is defined according to the largest output weight so the error vector is multiplied by the least output weight. The reference output weight is updated with the sum of the non-reference weight. For MLPs with more than one output, the reference vector is chosen based on the norm of the hidden nodes output weights.

The method consists on finding the pair of hidden nodes with minimum distance between their outputs for the whole training set. The norms of the weight vectors that connect each one of the two nodes to the output layer is then computed and the one with the smallest norm is chosen to be pruned and the remaining output weights are updated. The stop criterion is based on the validation error. Pruning stops when it starts to increase.

5. Elimination of randomly selected weights

This method consists on pruning randomly selected weights. If the extraction results on validation error decrease, the node is pruned, otherwise it is maintained. For large MLPs, the performance may decrease, depending on the number of weights to be pruned, that is a user-defined parameter. Due to its simplicity, this method has a good balance between algorithm complexity and network performance.

6. Saliency based pruning

The hessian based network pruning is described in different forms in the literature [13–15]. It is based on the second order derivatives of the error surface to trade-off network complexity and training error. It allows the identification of network weights that produce minimum error increase when pruned. It is normally carried on with a Taylor series local approximation of
the error surface. Quadratic approximation of the error surface is also assumed in order to simplify calculations, as shown in Eq. (9).

\[
\Delta E(w) = g^T(w)\Delta w + \frac{1}{2}\Delta w^T H \Delta w
\]  

(9)

where \( \Delta E(w) \) is the error function variation

\[
\Delta E(w) = E(w + \Delta w) - E(w),
\]

\( g(w) \) is the gradient vector and \( H \) is the hessian matrix evaluated for the current weight vector \( w \). Considering a MLP stuck at a local minimum, which means that \( g(w) = 0 \), the error function variation depends only on the hessian matrix and the weight vector variation \( \Delta w \). The new error function variation approximation is described in Eq. (10).

\[
\Delta E(w) = \frac{1}{2}\Delta w^T H \Delta w
\]  

(10)

The Optimal Brain Surgeon algorithm (OBS) [13] aims to extract weights in order to minimize the error variation, as described in the following equation:
The OBS pruning objective can be seen as an optimization problem that consists on minimizing Eq. (10) subject to Eq. (11). In order to solve this optimization problem, the Lagrangian operation can be applied:

\[
S = \frac{1}{2} \Delta w^T H \Delta w - \lambda (1_i^T \Delta w + w_i)
\]

where \( \lambda \) is the Lagrange multiplier.

The derivative of the Lagrangian with \( \Delta w \) constrained by Eq. (11) yields the direction of the optimal change in the weight vector, obtained as the optimal value of the Lagrangian \( S \) for the \( w_i \) connection.

\[
\Delta w = -\frac{w_i}{[H^{-1}]_{i,i}} H^{-1} 1_i
\]

\[
S_i = \frac{w_i^2}{2[H^{-1}]_{i,i}}
\]

where \( H^{-1} \) is the inverse of the hessian matrix and \( [H^{-1}]_{i,i} \) its ii-th element.

The lagrangian \( S_i \) is also known as the saliency of \( w_i \) and represents the error function increase resulted from \( w_i \) removal.

The proposed saliency based pruning consists on calculating the saliency of the weights and on removing the weight with minimum value at each iteration. The validation set error defines the stopping criterion. The remaining weights are not adjusted. Despite the fact that this method presents a well-based mathematic formulation it has a high computational cost due to the hessian inverse calculation.

7. Combination of methods for network simplification

A combination of the previously described methods is used to simplify network structures. The methods are executed in the following order:

1. Pruning randomly selected weights;
2. Similar nodes response;
3. Linearization.

The methods are tested in the sequence above and, for each algorithm, weights and nodes are pruned.

The final topology may have linear connections between inputs and outputs and also reduced number of hidden nodes with pruned connections.

8. Results

This section presents the results using the 6 methods described previously. Five data sets consisting on three classification and two function approximation problems were used to test the methods. The data sets cancer, card, gene, picked-up from PROBEN [12], were used for classification. Artificially generated data from the sinc function and also the building data set from PROBEN were used for approximation. The data sets were divided into training and validation sets with 60% and 40% of the whole data, respectively.

Figure 4 shows the Pareto set variation for Method 1 and the sinc problem as the number of hidden nodes increases. The pareto set with 15 hidden nodes produced the best solution.

Table 1 presents the results obtained for oversized network with the MOBJ algorithm, without pruning. It shows the best generalization solutions obtained without reducing the network number of weights and nodes, what can be achieved with the MOBJ algorithm. These initial solutions will be used further as a reference for the pruning methods. The pruning methods should reduce these initial over-sized networks without loss in performance.

Results for Method 2 (linearization) are presented in Table 2. The Final Topology presents the number of linear and non-linear nodes between inputs and outputs of the final network. For the card data set, all the hidden nodes of the original network were linearized, what reduced the network into a single layer network with non-linear nodes. The final network replaced the original two-layers non-linear network, without reduction in generalization performance. For the gene data set, no linearization was possible, whereas for the sinc function only one hidden node was linearized.

Similar nodes response identification results (Method 3) are presented in Table 3. The network for the cancer data set was reduced to a 9-6-2 topology, what represented a reduction of 9 nodes in the hidden layer. For the card data set the reduction was of 11 nodes in the hidden layer. There was no reduction for the gene data
Fig. 4. Pareto set results with increasing hidden nodes.

Table 2
Linear/Non-linear node response simplification results

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$E_t$</th>
<th>$E_v$</th>
<th>Initial Topology</th>
<th>Final Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancer</td>
<td>96.42%</td>
<td>98.57%</td>
<td>9-15-2</td>
<td>9-2-2</td>
</tr>
<tr>
<td>card</td>
<td>88.89%</td>
<td>88.04%</td>
<td>51-15-2</td>
<td>51-2</td>
</tr>
<tr>
<td>gene</td>
<td>98.85%</td>
<td>89.53%</td>
<td>120-15-3</td>
<td>120-15-3</td>
</tr>
<tr>
<td>sinc</td>
<td>0.0754</td>
<td>0.0110</td>
<td>1-50-1</td>
<td>1-48-1</td>
</tr>
<tr>
<td>building</td>
<td>0.0234</td>
<td>0.0224</td>
<td>14-15-3</td>
<td>14-7-3</td>
</tr>
</tbody>
</table>

set and reduction of 4 nodes for the sinc function and only one node for the building data set.

Table 3
Similar nodes response identification results

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$E_t$</th>
<th>$E_v$</th>
<th>Initial Topology</th>
<th>Final Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancer</td>
<td>96.18%</td>
<td>98.57%</td>
<td>9-15-2</td>
<td>9-6-2</td>
</tr>
<tr>
<td>card</td>
<td>88.89%</td>
<td>88.04%</td>
<td>51-15-2</td>
<td>51-4-2</td>
</tr>
<tr>
<td>gene</td>
<td>98.85%</td>
<td>89.53%</td>
<td>120-15-3</td>
<td>120-15-3</td>
</tr>
<tr>
<td>sinc</td>
<td>0.0730</td>
<td>0.0127</td>
<td>1-50-1</td>
<td>1-46-1</td>
</tr>
<tr>
<td>building</td>
<td>0.0189</td>
<td>0.0227</td>
<td>14-15-3</td>
<td>14-14-3</td>
</tr>
</tbody>
</table>

The results for Method 4 (randomly selected weights). Topologies with pruned weights are indicated with an asterisk (*). The result for the cancer data set reduced the network to a similar structure to the one obtained by linearization and Method 3. In addition to the final 9-4-2 topology, weights were pruned throughout the network. The results for the card data set showed that some input variables are not relevant for solving the problem, since the number of inputs was reduced from 51 to 14. The final network was also reduced in the hidden layer to 9 nodes, in contrast with the 15 nodes of the original network. A similar result was obtained for the gene data set, that was reduced from 120 inputs to only 44. The number of hidden nodes was reduced to 12. The network for the building data set had also a significant reduction in size and number of inputs (inputs were reduced to 4), whereas the network for the sinc function had no significant reduction in size, although weights were pruned.

Table 5 shows the results for Method 5 (Saliency based pruning). The results for the cancer and sinc data sets reduced the network weights without topology changing. The results for the card data set showed that
The problem could be solved with a single hidden layer withods that combine different paradigms for neural net-pruned for the number of inputs, whereas only 6 hidden nodes were building the original 120 variables was reduced to only 20. The best reduction in the number of inputs of all methods: The method was not able to run the total validation set. The original network had 51 inputs. A single input variable is able to classify 87.68% of the data set, due to the high computational cost of hessian calculations. As shown in Table 6, Method 6 (mixed method) had the best results. This can be justified by the fact that it can take the best features of Methods 2, 3 and 4. For the cancer data set, 13 hidden nodes were pruned; the final network has also 3 linear hidden nodes. There was an impressive reduction in the card data set. The problem could be solved with a single hidden layer with 8 input variables. The gene data set problem had the best reduction in the number of inputs of all methods: the original 120 variables was reduced to only 20. The building data set had also significant reduction in the number of inputs, whereas only 6 hidden nodes were pruned for the sinc function.

9. Conclusions

This paper presented constructive and pruning methods that combine different paradigms for neural net-works generalization control. The constructive multi-objective method stops network growing when its size is enough to obtain a good response with the MOBJ algorithm. The constructive and the MOBJ algorithms are representatives of two different approaches for solving the problem. The first one controls generalization by finding the appropriate number of parameters do fit the data. The second one controls generalization by smoothing the network response within the limits of the Pareto set, regardless of the number of network nodes and weights. The combination of these two methods resulted in an algorithm that is robust and that yields good generalization response.

The linearization approaches the problem by controlling complexity according to the number of network nodes, although linear responses of hidden nodes can be obtained by smoothing the network output with a method such as the MOBJ. We can see both aspects of generalization control in this method as well. The method is not only able to reduce network size, but allows also to study problem complexity. The card network was reduced to a single layer of network nodes, with the complete linearization of the whole intermediate layer. This result shows that the problem can be completely solved with linear separation functions, instead of non-linear functions, that would normally be used.

All methods resulted on significant reduction on network size for all data sets, except for the sinc function, that is known to be approximated with a network with 15 nodes (growing method). The explanation for this may be related to the initial MOBJ solution and to the fact that the sinc function has a very peculiar characteristic of having a peak at $x = 0$ and having smooth variations in the rest of $x$ axis. The analysis of the initial MOBJ solution showed that the network weights are very small and a few of them have large values. This suggests that the MOBJ solution resulted on a small number of large weights specialized in the peak region and a large number of small weights specialized on the smooth region. This suggests that very few weights could then be pruned, since they are specialized on regions within the function definition. This is a very important aspect of the MOBJ algorithm that allows different degrees of freedom for each one the network weights.

The best results were obtained by the combination of methods and paradigms, with networks with mixed structures of linear and non-linear nodes, with reduction in the input size and in the number of weights and with smoothing control. These experimental results suggest
that a good balance between the two main paradigms for
generalization control may result on improved neural
network performance.

Acknowledgements

The authors would like to thank CNPq and CAPES
for the financial support.

References