Pruning Using Parameter and Neuronal Metrics

Pi van de Laar and Tom Heskes
RWCP*, Theoretical Foundation, SNN,
Department of Medical Physics and Biophysics,
University of Nijmegen, The Netherlands

Abstract

In this article, we introduce a measure of optimality for architecture selection algorithms for neural networks: the distance from the original network to the new network in a metric that is defined by the probability distributions of all possible networks. We derive two pruning algorithms, one based on a metric in parameter space and another one based on a metric in neuron space, which are closely related to well-known architecture selection algorithms, such as GOBS. Furthermore, our framework extends the theoretically range of validity of GOBS and therefore can explain results observed in previous experiments. In addition, we give some computational improvements for these algorithms.

1 Introduction

A neural network trained on a problem for which its architecture is too small to capture the underlying data structure, will not yield satisfactory training and testing performance. On the other hand, a neural network with too large an architecture can even fit the noise in the training data, leading to good training but rather poor testing performance. Unfortunately, the optimal architecture is not known in advance for most real-world problems. The goal of architecture selection algorithms is to find this optimal architecture. These algorithms can be grouped according to their search strategy or definition of optimality. The most widely known search strategies are growing and pruning, although other strategies exist, see e.g. (Fahlman & Lebiere 1990), (Reed 1993) and (Hirose et al. 1991). The optimality of an architecture can be measured by, for example, minimum description length (Rissanen 1978), an information criterion (Akaike 1974, Ishikawa 1996), a network information criterion (Murata et al. 1994), error on the training set (LeCun et al. 1990, Hassibi & Stork 1993) or error on an independent test set (Pedersen et al. 1996). In this article, another measure

*Real World Computing Partnership

Foundation for Neural Networks

of optimality for pruning algorithms will be introduced: the distance from the original architecture in a predefined metric.

We briefly describe the problem of architecture selection and the general framework of our pruning algorithms based on metrics in section 2. In sections 3 and 4, we introduce two pruning algorithms; one based on a metric in parameter space and the other on a metric in neuron space. We relate these algorithms to other well-known architecture selection algorithms. In section 5 we discuss some of the computational aspects of these two algorithms, and in section 6 we compare the performance of the algorithms. We end with conclusions and a discussion in section 7.

2 Architecture selection

For a given neural network with weights represented by a W -dimensional vector $\mathbf{w}$, there are $2^W - 1$ possible subsets in which one or more of the weights have been removed. Therefore, a procedure which estimates the relevance of the weights based on the performance of every possible subset of weights, is only feasible if the number of weights is rather small. When the number of weights is large, one has to use approximations, such as backward elimination, forward selection, or stepwise selection [see e.g. (Draper & Smith 1981, Kleinbaum et al. 1988)]. In the neural network literature, pruning is identical to backward elimination and growing to forward selection. Although the results of this search strategy already provide insight in the importance of the different connections in the original architecture, for real-world applications one needs a final model. A possibility is to select from all evaluated architectures the optimal architecture, see e.g. (van de Laar et al. 1997). Of course, many different definitions of optimality are possible. For example, the error on the training set (Hassibi & Stork 1993, Castellano et al. 1997) or the generalization error on an independent test set (Pedersen et al. 1996). Another possibility is to use an ensemble of architectures instead of a single architecture, see e.g. (Breiman 1996).

In the following two sections we will construct pruning algorithms based on two different metrics. In these sections, we will concentrate on the definition of the metric and the comparison of the resulting algorithms with other well-known architecture selection algorithms.

3 Parameter metric

In this section we start by defining a metric in parameter space. Let $\mathcal{D}$ be a random variable with a probability distribution specified by $\mathcal{P}(\mathcal{D}|\mathbf{w})$, where $\mathbf{w}$ is a W -dimensional parameter vector. The Fisher information metric is the natural geometry to be introduced in the manifold formed by all such distributions (Amari 1998)

$$\mathcal{F}_{ij}(\mathbf{w}) = \int d\mathcal{D} \mathcal{P}(\mathcal{D}|\mathbf{w}) \frac{\partial \log \mathcal{P}(\mathcal{D}|\mathbf{w})}{\partial w_i} \frac{\partial \log \mathcal{P}(\mathcal{D}|\mathbf{w})}{\partial w_j}. \quad (1)$$

Although we can perform pruning using this Fisher information metric for any model that defines a probability distribution over the data, we will restrict ourselves to multilayer perceptrons (MLPs). We will adopt the terminology of the literature about MLPs. For example, the parameters of an MLP will be called weights.

For an MLP, the random variable $\mathcal{D}$ can be divided into an N -dimensional input vector ($\mathbf{X}$) and a K -dimensional target (also called desired output) vector ($\mathbf{T}$). The probability distribution in the input space of an MLP does not depend on the weights, therefore

$$\mathcal{P}(\mathbf{X}, \mathbf{T} | \mathbf{w}) = \mathcal{P}(\mathbf{T} | \mathbf{w}, \mathbf{X}) \mathcal{P}(\mathbf{X}) . \quad (2)$$

When an MLP minimizes the sum-squared error between actual and desired output, the following probability distribution in the target space given the inputs and weights can be assumed [the additive Gaussian noise assumption (MacKay 1995)]

$$\mathcal{P}(\mathbf{T} | \mathbf{w}, \mathbf{X}) = \prod_{k=1}^K \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp^{-\frac{(t_k - O_k)^2}{2\sigma_k^2}} , \quad (3)$$

where O_k , the k^{th} output of the MLP, is a function of the input and weights, and σ_k is the standard deviation of the k^{th} output.

Furthermore, since an MLP does not define a probability distribution of its input space, we assume that the input distribution is given by delta-peaks located on the data, i.e.

$$\mathcal{P}(\mathbf{X}) = \frac{1}{P} \sum_{\mu=1}^P \delta^N(\mathbf{X} - \mathbf{X}^\mu) . \quad (4)$$

Inserting equations (3) and (4) in equation (1) leads to the following Fisher information metric for an MLP:

$$\mathcal{F}_{ij}(\mathbf{w}) = \frac{1}{P} \sum_{\mu=1}^P \sum_{k=1}^K \frac{1}{\sigma_k^2} \frac{\partial O_k^\mu}{\partial w_i} \frac{\partial O_k^\mu}{\partial w_j} . \quad (5)$$

With this metric we can determine the distance D from one MLP to another MLP of exactly the same architecture by

$$D^2 = \frac{1}{2} \delta \mathbf{w}^T \mathcal{F} \delta \mathbf{w} , \quad (6)$$

where $\delta \mathbf{w}$ is the difference in weights of the two MLPs. For small $\delta \mathbf{w}$, D is an approximation of the Riemannian distance. Although the Riemannian distance is symmetric with respect to the two MLPs A and B , equation (6) is only symmetric up to $\mathcal{O}(|\delta \mathbf{w}|^2)$. The asymmetry is due to the dependence of the Fisher information matrix on the weights of the original MLP. So, like for the

Kullback-Leibler divergence, the distance from MLP A to B is not identical to the distance from MLP B to A .

Since there is no natural ordering of the hidden units of an MLP, one would like to have a distance measure which is insensitive to a rearrangement of the hidden units and corresponding weights. Unfortunately, the distance between two functionally identical but geometrically different MLPs according to equation (6) is, in general, non-zero. Therefore, this distance measure can best be described as local. Thus, this metric based approach is only valid for sequences of relatively small steps from a given architecture.

Since the deletion of a weight is mathematically identical to setting its value to zero, the deletion of weight q can be expressed as $\delta w_q = -w_q$, and this metric can also be used for pruning. We have to determine for every possible smaller architecture¹ its optimal weights with respect to the distance from the original MLP. Finally, we have to select, from all possible smaller MLPs with optimal weights, our final model.

With the assumption that the output noises are independent, i.e. $\sigma_k \equiv \sigma$, this pruning algorithm will select the same architectures as Generalized Optimal Brain Surgeon (GOBS) (Hassibi & Stork 1993, Stahlberger & Riedmiller 1997). GOBS is derived using a Taylor series expansion up to second order of the error of an MLP trained to a (local or global) minimum. Since the first order term vanishes at a minimum, only the second order term, which contains the Hessian matrix, needs to be considered. The inverse of the Hessian matrix is then calculated under the approximation that the desired and actual output of the MLP are almost identical. Given this approximation, the Hessian and Fisher information matrix are identical. (Hassibi & Stork 1993) already mentioned the close relationship with the Fisher information matrix, but they did not provide an interpretation.

Unlike (Hassibi & Stork 1993), our derivation of GOBS does not assume that the MLP has to be trained to a minimum. Therefore, we can understand why GOBS performs so well on “stopped” MLPs, i.e. MLPs which have not been trained to a local minimum (Hassibi et al. 1994).

4 Neuronal metric

In this section we will define a metric which, unlike the previously introduced metric, is specific for neural networks. The metric will be defined in neuron space. Why would one like to define such a metric? Assuming that a neural network has constructed a good representation of the data in its layers to solve the task, one would like smaller networks to have a similar representation and, consequently, similar performance on the task. As in the previous section, we will restrict ourselves to MLPs.

In pruning there are two reasons why the activity of a neuron can change. The first reason is the deletion of a weight leading to this neuron. The second

¹As already described in section 2 this approach becomes computationally intensive for large MLPs and other search strategies might be preferred.

reason is a change in activity of an incoming neuron. For example, when a weight between input and hidden layer is deleted in an MLP, this does not only change the activity of a hidden neuron but also the activities of all neurons which are connected to the output of that hidden neuron.

To find the MLP with neuronal activity as close as possible to the neuronal activity of the original MLP, one should minimize

$$D^2 = \frac{1}{2\mathcal{N}P} \sum_{i=1}^{\mathcal{N}} \sum_{\mu=1}^P (O_i^\mu - \bar{O}_i^\mu)^2, \quad (7)$$

where $\mathcal{N}$ denotes the number of neurons (both hidden and output), $\mathbf{O} = f(\mathbf{w}^T \mathbf{X})$ the original output, $\bar{\mathbf{O}} = f((\mathbf{w} + \delta \mathbf{w})^T \bar{\mathbf{X}})$ the new output, f the transfer function, and $\mathbf{X}$ and $\bar{\mathbf{X}}$ the original and new input of a neuron. Equation (7) is rather difficult to minimize since the new output of the hidden neurons also appears as the new input of other neurons.

Equation (7) can be approximated by incorporating the layered structure of an MLP, i.e. the calculations start at the first layer and proceed up to the last layer. In this case the input of a layer is always known, since it has been calculated before, and the solution of the layer can be determined. Therefore starting at the first hidden layer and proceeding up to the output layer, one should minimize with respect to the weights for each neuron

$$D_i^2 = \frac{1}{2P} \sum_{\mu=1}^P (O_i^\mu - \bar{O}_i^\mu)^2. \quad (8)$$

Due to the non-linearity of the transfer function, the solution of equation (8) is still somewhat difficult to find. Using a Taylor series expansion up to first order $\bar{O}_i^\mu = O_i^\mu + \frac{\partial O_i^\mu}{\partial z} \delta z + \mathcal{O}(\delta z^2)$, the previous equation can be approximated by

$$D_i^2 \approx \frac{1}{2P} \sum_{\mu=1}^P \left(\frac{\partial O_i^\mu(z)}{\partial z} \Big|_{z=O_i^\mu} \right)^2 (o_i^\mu - [\bar{\mathbf{w}}^T \bar{\mathbf{X}}^\mu]_i)^2, \quad (9)$$

where $o = \mathbf{w}^T \mathbf{X}$ is the original incoming activity of the neuron. This distance can be easily minimized with respect to the new weights $\bar{\mathbf{w}}$ by any algorithm for least squares fitting. The complexity of this minimization is equal to an inversion of a matrix with the dimension equal to the number of inputs of the neuron.

This pruning algorithm based on the neuronal metric is closely related to other well-known architecture selection algorithms. If the contribution of the scale factor $\frac{\partial O_i^\mu(z)}{\partial z} \Big|_{z=O_i^\mu}$ can be neglected² this pruning algorithm is identical to a pruning algorithm called “partial retraining” (van de Laar et al. 1998).

²For an error analysis of this assumption see (Moody & Antsaklis 1996).

Another simplification is to ignore the second reason of change in activity of a neuron, i.e. ignore the change of activity due to a change in activity of an incoming neuron. When the input of a neuron does not change, equation (8) can be simplified to

$$D_i^2 \approx \frac{1}{2} \delta \mathbf{w}^T \mathcal{F}_i \delta \mathbf{w} , \quad (10)$$

with $[\mathcal{F}_i]_{jk} = \frac{1}{P} \sum_{\mu=1}^P \frac{\partial O_i^\mu}{\partial w_j} \frac{\partial O_i^\mu}{\partial w_k}^T$. The optimal weight change for this problem can be easily found and will be described in section 5.

When both simplifications, i.e. neglecting both the contribution of the scale factor and the second reason of change in activity of a neuron, are applied simultaneously, one derives the architecture selection algorithm as proposed by (Egmont-Petersen 1996) and (Castellano et al. 1997).

5 Computational aspects

A number of different computational approaches exists to find the minimal distance from the original network to a smaller network, as given by

$$D^2 = \frac{1}{2} \delta \mathbf{w}^T \mathcal{F} \delta \mathbf{w} . \quad (11)$$

5.1 Lagrange

One could apply Lagrange's method to calculate this distance [see also (Hassibi & Stork 1993, Stahlberger & Riedmiller 1997)]: The Lagrangian is given by

$$L = \frac{1}{2} \delta \mathbf{w}^T \mathcal{F} \delta \mathbf{w} + \lambda^T (\mathbf{w}_{\mathbf{D}} + \delta \mathbf{w}_{\mathbf{D}}) , \quad (12)$$

with λ a vector of Lagrange multipliers, $\mathbf{D}$ the set that contains the indices of all the weights to be deleted, and $\mathbf{w}_{\mathbf{D}}$ the sub-vector of $\mathbf{w}$ obtained by excluding all remaining weights.

Assuming that the semi-positive Fisher information matrix and the submatrix $[\mathcal{F}^{-1}]_{\mathbf{D}\mathbf{D}}$ of the inverse Fisher information matrix are invertible, the resulting minimal distance from the original network in the given metric is given by

$$D^2 = \frac{1}{2} \mathbf{w}_{\mathbf{D}}^T ([\mathcal{F}^{-1}]_{\mathbf{D}\mathbf{D}})^{-1} \mathbf{w}_{\mathbf{D}} , \quad (13)$$

and the optimal change in weights is equal to

$$\delta \mathbf{w} = -[\mathcal{F}^{-1}]_{\cdot\mathbf{D}} ([\mathcal{F}^{-1}]_{\mathbf{D}\mathbf{D}})^{-1} \mathbf{w}_{\mathbf{D}} . \quad (14)$$

5.2 Fill in

One could fill in the known weight changes $\delta \mathbf{w}_D = -\mathbf{w}_D$ and minimize the resulting distance with respect to the remaining weights:

$$D^2 = \frac{1}{2} \left(\mathbf{w}_D^T \mathcal{F}_{DD} \mathbf{w}_D - \delta \mathbf{w}_R^T \mathcal{F}_{RD} \mathbf{w}_D - \mathbf{w}_D^T \mathcal{F}_{DR} \delta \mathbf{w}_R + \delta \mathbf{w}_R^T \mathcal{F}_{RR} \delta \mathbf{w}_R \right), \quad (15)$$

where $\mathbf{R}$ and $\mathbf{D}$ denote the sets that contain the indices of all remaining and deleted weights, respectively. When the matrix $\mathcal{F}_{RR}$ is invertible the minimal distance from the original MLP is achieved for the following change in weights

$$\delta \mathbf{w}_R = [\mathcal{F}_{RR}]^{-1} \mathcal{F}_{RD} \mathbf{w}_D \quad (16)$$

and is equal to

$$D^2 = \frac{1}{2} \left(\mathbf{w}_D^T \mathcal{F}_{DD} \mathbf{w}_D - \mathbf{w}_D^T \mathcal{F}_{DR} [\mathcal{F}_{RR}]^{-1} \mathcal{F}_{RD} \mathbf{w}_D \right). \quad (17)$$

5.3 Inverse updating

One could use the fact that the inverse of the Fisher information matrix with less variables can be calculated from the inverse of the Fisher information matrix that includes all variables (Fisher 1970), i.e.

$$\delta \mathcal{F}^{-1} = -[\mathcal{F}^{-1}]_{\mathbf{D}} ([\mathcal{F}^{-1}]_{\mathbf{DD}})^{-1} [\mathcal{F}^{-1}]_{\mathbf{D}}. \quad (18)$$

For example, when weights are iteratively removed, updating the inverse of Fisher information matrix in each step using equation (18) makes the matrix inversions in equations (13) and (14) trivial, since the matrices to be inverted are always of 1×1 dimension.

5.4 Comparison

All these three approaches give, of course, the same solution. For the first two approaches this can be easily seen since for any invertible matrix

$$\mathcal{F}_{RR} [\mathcal{F}^{-1}]_{RD} + \mathcal{F}_{RD} [\mathcal{F}^{-1}]_{DD} = 0, \quad (19)$$

and where it is assumed that the submatrices $\mathcal{F}_{RR}$ and $[\mathcal{F}^{-1}]_{DD}$ are invertible [see equations (13) and (16)]. Also the first and third approach yield the same solution since equation (14) can be rewritten as

$$\delta \mathbf{w} = (\delta \mathcal{F}^{-1}) \mathcal{F} \mathbf{w}. \quad (20)$$

The matrix inversion in the first two approaches is the most computationally intensive part. Therefore, when a given set of variables has to be deleted, one

Figure 1: The calculation time needed to iteratively prune all weights of a randomly generated MLP versus the number of weights using (1) Lagrange [i.e. GOBS as proposed by (Stahlberger & Riedmiller 1997)], (2) Lagrange and fill in (i.e. selecting the smallest matrix inversion), and (3) Inverse updating (i.e. updating the weights and inverse Fisher information matrix in each step). The solutions of these three approaches were, of course, identical.

Figure 2: The average estimated order versus the number of weights using the same three approaches as described in figure 1. The estimated order is calculated as given by equation (21), where ΔW was chosen to be equal to 25. The error bars show the standard deviation over 10 trials. The figure seems to confirm that the first two approaches are of fifth order and the last approach is only of third order.

should prefer the first approach if the number of variables is less than half of all weights. If one has to remove more than half of all variables, the second approach should be applied. When backward elimination or an exhaustive search should be performed one should use the third approach. For example, with the third approach, GOBS removes all weights using backward elimination³ in $\mathcal{O}(W^3)$ time steps, while with the first or second approach $\mathcal{O}(W^5)$ time steps are needed, where W is the number of weights. To verify these theoretical predictions, we determined the calculation time needed to iteratively prune all weights of a randomly generated MLP as a function of the number of weights (see figure 1). Furthermore, we estimated the order of the different approaches by

$$o(W) \approx \frac{\log t(W + \Delta W) - \log t(W)}{\log(W + \Delta W) - \log(W)}, \quad (21)$$

where $t(W)$ is the calculation time needed to iteratively prune all W weights (see Figure 2). The accuracy of this estimation improves with the number of weights (W); asymptotically it yields the order of the approach.

Of course, one can apply algorithms such as conjugate gradient instead of matrix inversion to optimize equation (11) directly in all three approaches [see for example (Castellano et al. 1997)].

6 Comparison

In this article, we proposed two pruning algorithms based on different metrics. In this section we will try to answer the question: What is the difference in accuracy between these two algorithms? To answer this question we have chosen a number of standard problems: the artificial Monk classification tasks (Thrun et al. 1991), the real-world Pima Indian diabetes classification task (Prechelt 1994), and the real-world Boston housing regression task (Belsley et al. 1980). After training an MLP on a specific task, its weights will be removed by backward elimination, i.e. the weight that results in the architecture with the smallest distance, according to our metric, from our original network will be iteratively removed until no weight is left.

The inverse of the Fisher matrix was calculated as described in (Hassibi & Stork 1993). But unlike (Hassibi & Stork 1993), the small constant α was chosen to be 10^{-4} times the largest singular value of the Fisher matrix⁴. This value of α penalizes large candidate jumps in parameter space, and thus insures that the weight changes are local given the metric. The inverse of the Fisher matrix was not recalculated after removing a weight, but updated as previously described in section 5.3.

³This algorithm is not identical to OBS as described by (Hassibi & Stork 1993), since OBS calculates after each weight removal the inverse Fisher information anew. In other words, OBS changes the metric after the removal of every weight while GOBS keeps the original metric.

⁴The actual value of α was most of the times within the range of $10^{-4} \leq \alpha \leq 10^{-8}$ as was given in (Hassibi & Stork 1993).

Problem	original	parameter metric	neuronal metric
Monk 1	58	15	19
Monk 2	39	15	18
Monk 3	39	5	4

Table 1: The original number of weights and the remaining weights after pruning using the algorithm based on the parameter and neuronal metric on the MLPs trained on the three Monk’s problems (Thrun et al. 1991).

6.1 Monk

Each Monk problem (Thrun et al. 1991) is a classification problem based on six attributes. The first, second, and fourth attribute have three possible values, the third and sixth are binary attributes, and the fifth attribute has four possible values. The different attributes in the Monk’s problem are not equally important. The target in the first Monk problem is $(a_1 = a_2) \cup (a_5 = 1)$. In the second Monk problem, the target is only true if exactly two of the attributes are equal to their first value. The third Monk problem has 5% noise in its training examples, and without noise the target is given by $(a_5 = 3 \cap a_4 = 1) \cup (a_5 \neq 4 \cap a_2 \neq 3)$. Since neural networks cannot easily handle multiple-valued attributes, the Monk problems are usually rewritten to seventeen binary inputs. Each of the seventeen inputs codes a specific value of a specific attribute. For example, the sixth input is only active if the second attribute has its third value.

For each Monk problem, (Thrun et al. 1991) trained an MLP with a single hidden layer. Each of these MLPs had seventeen input neurons and one continuous output neuron. The number of hidden units of the MLP in the three Monk problems was three, two and four, respectively. The transfer function of both the hidden and the output layer of the MLP was a sigmoid, in all three problems. The MLPs were trained using backpropagation on the sum-squared error between the desired output and the actual output. An example is classified as true, if the network’s output exceeds a threshold (0.5), and false otherwise.

We used the trained MLPs as described in (Thrun et al. 1991) to test the algorithm based on the parameter metric and the one based on the neuronal metric. From these three MLPs, we iteratively removed the least relevant weight until the training and test performance deteriorated using both algorithms. In either case, pruning these three MLPs resulted in a large reduction in the number of weights, as can be seen in Table 1.

Although the pruning algorithm based on the neuronal metric is a good pruning algorithm, it is outperformed by the pruning algorithm based on the parameter metric, which removes a few weights more from the same three MLPs. We will show using a toy problem that this difference in performance is (partly) caused by the redundancy in the encoding of the multiple-valued attributes and the ability of the pruning algorithm based on the parameter metric to change its hidden-layer representation.

A_1	A_2	A_3	T
1	0	0	-1
0	1	0	1
0	0	1	1

Table 2: The training data of $A \neq 1$.

Figure 3: A linear MLP to be pruned.

Suppose an attribute A has three possible values and is encoded similarly to the attributes in the Monk problems (Thrun et al. 1991). A linear MLP which implements the function $A \neq 1$ is given in Figure 3 and its training data is given in Table 2. Both pruning algorithms will now be applied to prune weights from this linear MLP.

When the algorithm based on the neuronal metric determines the importance of the connection between A_3 and H (as defined in Figure 3), it first calculates the new weights between input and hidden layer such that the hidden-layer representation is approximated as well as possible, which results in $\bar{w}_1 = 0$ and $\bar{w}_2 = 1$. Unfortunately, this results in the hidden-layer activity 0 if attribute A has value 1 or 3, and activity 1 if attribute $A = 2$. Based on this hidden-layer activity, it is not possible to find new weights for the second layer ($\bar{v}$ and $\bar{b}$) such that $A \neq 1$ is implemented, and w_3 will not be deleted. The same argumentation holds for the deletion of w_2 . The only weight which will be deleted by this algorithm from the MLP given in Figure 3 is w_1 .

When the algorithm based on the parameter metric calculates the relevance of the connection between A_3 and H , all other weights are reestimated simultaneously. This algorithm might (since the Fisher information matrix in this toy-problem is singular) end up with $\bar{w}_1 = -1$, $\bar{w}_2 = 0$, $\bar{v} = 2$, and $\bar{b} = 1$, which exactly implements $A \neq 1$. This algorithm can remove w_3 , and afterwards w_2 , whose value is then equal to zero. So, since this algorithm is able to change the hidden-layer representation from $((A = 2) \cup (A = 3))$ to $A \neq 1$, it can remove one weight more than the algorithm based on the neuronal metric.

Summarizing, although both pruning algorithms find smaller architectures with identical performance, the algorithm based on the neuronal metric removes a few weights less than the algorithm based on the parameter metric. This is caused by the fact that the algorithm based on the neuronal metric is, by definition, restricted to single layers and is therefore necessarily weaker than the algorithm based on the parameter metric, which can “look” across layers to find more efficient hidden-layer representations.

6.2 Diabetes in Pima Indians

The diabetes dataset contains information about 768 females of Pima Indian heritage of at least 21 years old. Based on 8 attributes, such as the number of times pregnant, the diastolic blood pressure, age, and the body mass index, one should predict whether this patient tested positive for diabetes. This dataset is considered very difficult and even state-of-the-art neural networks still misclassify about 25% of the examples. For more information about this dataset see, for example, (Prechelt 1994). After normalization of the input data, e.g. each input variable had zero mean and unit standard deviation, the 768 examples were randomly divided into three sets, the estimation (192), validation (192) and test set (384). For prediction, we use MLPs with 8 inputs, 5 hidden units, one output, and a hyperbolic tangent and linear transfer function of the hidden and output layer, respectively. The MLPs were trained using backpropagation of the sum-squared error on the estimation set, and training was stopped when the sum-squared error on the validation set increased. Similar as in the Monk problems (Thrun et al. 1991), an example was classified as non-diabetic, when the network’s output exceeded a threshold (0.5), and diabetic otherwise.

As the baseline we define the percentage of errors made in classifying the examples in the test set, when they are classified as the most often occurring classification in the train set. For example, if 63% of the training examples are diabetic, all test examples are all labelled diabetic, leading, if the training set is representative, to an error rate of 37%.

In figure 4 the baseline and the percentage of misclassifications of the pruning algorithms based on the parameter and neuronal metric are plotted as function of the number of remaining weights (W) of the MLP. Although the pruning algorithm based on the parameter metric is in the beginning at least as good as the pruning algorithm based on the neuronal metric, after the removal of a number of weights its performance becomes worse than that of the pruning algorithm based on the neuronal metric. With a few weights remaining the pruning algorithm based on the parameter metric even has a performance which is worse than the baseline performance while the pruning algorithm based on the neuronal metric has still a rather good performance.

6.3 Boston housing

The Boston housing dataset (Belsley et al. 1980) contains 506 examples of the median value of owner-occupied homes as function of thirteen input vari-

Figure 4: The mean value and standard deviation (based on 15 runs) of the percentage of misclassifications on the test set of the Pima Indian diabetes dataset of the pruning algorithms based on the parameter and neuronal metric versus the number of remaining weights of the MLP. For comparison, the baseline error has also been drawn.

Figure 5: The mean value and standard deviation (based on 15 runs) of the sum-squared error on the test set of the Boston housing dataset of the pruning algorithms based on the parameter and neuronal metric versus the number of remaining weights of the MLP. For comparison, the baseline error has also been drawn.

ables, such as nitric oxides concentration squared, average number of rooms per dwelling, per capita crime rate, and pupil-teacher ratio by town. For our simulations, we first normalized the data, such that each variable (both input and output) had zero mean and unit standard deviation. Then, we randomly divided the 506 examples into a training and test set which both contained 253 examples. The MLPs were trained using cross validation, therefore the training set was split into an estimation and validation set of 127 and 126 examples, respectively. The MLPs had thirteen inputs, three hidden units, one output, and a hyperbolic tangent and linear transfer function of the hidden and output layer, respectively.

The baseline is the (average) error made in predicting the housing prices of the examples in the test set, when they are predicted as the mean housing price of the training set. The value of the baseline will be close to one due to the normalization of the output.

In figure 5 the baseline and the performance of the pruning algorithms based on the parameter and neuronal metric are plotted as function of the number of remaining weights (W) of the MLP. Similar to the simulations of the diabetes dataset, the pruning algorithm based on the neuronal metric remains close to the original performance, even after removing 85% of the weights in the original network, while the performance of the pruning algorithm based on the parameter metric deteriorates earlier and even becomes worse than the baseline performance.

7 Conclusions and discussion

In this article, we have introduced architecture selection algorithms based on metrics to find the optimal architecture for a given problem. Based on a metric in parameter space and neuron space, we derived two algorithms that are very close to other well-known architecture selection algorithms. Our derivation has enlarged the understanding of these well-known algorithms. For example, we have shown that GOBS is also valid for MLPs which have not been trained to a (local or global) minimum, as was already experimentally observed (Hassibi et al. 1994). Furthermore, we have described a variety of approaches to perform these well-known algorithms and discussed which of the approaches should be preferred given the circumstances.

Although the pruning algorithm based on the parameter metric is theoretically more powerful than the pruning algorithm based on the neuronal metric, as was illustrated by a small example, simulations of real-world problems showed that the stability of the pruning algorithm based on the parameter metric is inferior to the stability of the pruning algorithm based on the neuronal metric. (Hassibi & Stork 1993) already observed this instability of the pruning algorithm based on the parameter metric and suggested to improve the stability by retraining the MLP after removing a number of weights.

We expect that the use of metrics for architecture selection is also applicable to other architectures than the MLP such as, for example, Boltzmann Machines and Radial Basis Functions Networks. Furthermore, based on the similarity between the deletion and addition of a variable (Cochran 1938), we think that this approach can also be applied for growing algorithms instead of pruning.

Acknowledgements

We would like to thank David Barber, Stan Gielen and two anonymous referees for their useful comments on an earlier version of this paper.

References

- Akaike, H. (1974). A new look at the statistical model identification, *IEEE Transactions on Automatic Control* **19**(6): 716–723.

- Amari, S.-I. (1998). Natural gradient works efficiently in learning, *Neural Computation* **10**(2): 251–276.
- Belsley, D. A., Kuh, E. & Welsch, R. E. (1980). *Regression diagnostics: identifying influential data and sources of collinearity*, Wiley Series in Probability and Mathematical Statistics, Wiley, New York.
- Breiman, L. (1996). Bagging predictors, *Machine Learning* **24**(2): 123–140.
- Castellano, G., Fanelli, A. M. & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks, *IEEE Transactions on Neural Networks* **8**(3): 519–531.
- Cochran, W. G. (1938). The omission or addition of an independent variate in multiple linear regression, *Supplement to the Journal of the Royal Statistical Society* **5**(2): 171–176.
- Draper, N. R. & Smith, H. (1981). *Applied Regression Analysis*, Wiley Series in Probability and Mathematical Statistics, second edn, Wiley, New York.
- Egmont-Petersen, M. (1996). *Specification and assessment of methods supporting the development of neural networks in medicine*, PhD thesis, Maastricht University, Maastricht.
- Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture, in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2: Proceedings of the 1989 Conference*, Morgan Kaufmann, San Mateo, pp. 524–532.
- Fisher, R. A. (1970). *Statistical Methods for Research Workers*, fourteenth edn, Oliver and Boyd, Edinburgh.
- Hassibi, B. & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon, in S. J. Hanson, J. D. Cowan & C. L. Giles (eds), *Advances in Neural Information Processing Systems 5: Proceedings of the 1992 Conference*, Morgan Kaufmann, San Mateo, pp. 164–171.
- Hassibi, B., Stork, D. G., Wolff, G. & Watanabe, T. (1994). Optimal Brain Surgeon: Extensions and performance comparisons, in J. D. Cowan, G. Tesauro & J. Alspector (eds), *Advances in Neural Information Processing Systems 6: Proceedings of the 1993 Conference*, Morgan Kaufmann, San Francisco, pp. 263–270.
- Hirose, Y., Yamashita, K. & Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units, *Neural Networks* **4**(1): 61–66.
- Ishikawa, M. (1996). Structural learning with forgetting, *Neural Networks* **9**(3): 509–521.

- Kleinbaum, D. G., Kupper, L. L. & Muller, K. E. (1988). *Applied Regression Analysis and Other Multivariable Methods*, The Duxbury series in statistics and decision sciences, second edn, PWS-KENT Publishing Company, Boston.
- LeCun, Y., Denker, J. S. & Solla, S. A. (1990). Optimal Brain Damage, in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2: Proceedings of the 1989 Conference*, Morgan Kaufmann, San Mateo, pp. 598–605.
- MacKay, D. J. C. (1995). Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks, *Network: Computation in Neural Systems* **6**(3): 469–505.
- Moody, J. O. & Antsaklis, P. J. (1996). The dependence identification neural network construction algorithm, *IEEE Transactions on Neural Networks* **7**(1): 3–15.
- Murata, N., Yoshizawa, S. & Amari, S.-I. (1994). Network information criterion - determining the number of hidden units for an artificial neural network model, *IEEE Transactions on Neural Networks* **5**(6): 865–872.
- Pedersen, M. W., Hansen, L. K. & Larsen, J. (1996). Pruning with generalization based weight saliencies: γ OBD, γ OBS, in D. S. Touretzky, M. C. Mozer & M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference*, MIT Press, Cambridge, pp. 521–527.
- Prechelt, L. (1994). PROBEN1 - a set of neural network benchmark problems and benchmarking rules, *Technical Report 21/94*, Fakultät für Informatik, Universität Karlsruhe.
- Reed, R. (1993). Pruning algorithms - a survey, *IEEE Transactions on Neural Networks* **4**(5): 740–747.
- Rissanen, J. (1978). Modeling by shortest data description, *Automatica* **14**: 465–471.
- Stahlberger, A. & Riedmiller, M. (1997). Fast network pruning and feature extraction using the unit-OBS algorithm, in M. C. Mozer, M. I. Jordan & T. Petsche (eds), *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, MIT Press, Cambridge, pp. 655–661.
- Thrun, S. B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Džeroski, S., Fahlman, S. E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R. S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J. & Zhang, J. (1991). The MONK's problems: A performance comparison of different learning algorithms, *Technical Report CMU-CS-91-197*, Carnegie Mellon University.

- van de Laar, P., Gielen, S. & Heskes, T. (1997). Input selection with partial retraining, *in* W. Gerstner, A. Germond, M. Hasler & J.-D. Nicoud (eds), *Artificial Neural Networks - ICANN'97*, Vol. 1327 of *Lecture Notes in Computer Science*, Springer, pp. 469–474.
- van de Laar, P., Heskes, T. & Gielen, S. (1998). Partial retraining: A new approach to input relevance determination. Submitted.