ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 129-140

Pruning Methods: A Review

Christian Jutten and Olivier Fambon

TIRF-INPG, 46 Av. Felix Viallet, F-38000 Grenoble
E-mail: chris@tirf.inpg.fr

Abstract: In this paper, we propose a commented state-of-the-art of pruning methods. In fact, most of the methods can be interpreted according a statistical point of view, which is very useful for cleaner application: choice of parameters, pruning stopping criterion.

1 Introduction

In the context of evolutive neural networks, not only incrementality, but also decrementality can be considered. As a matter of fact, decreasing the size of a network leads to simpler models. Additionally, constraining a model to be 'simple' in some sense is a way to avoid the overfitting phenomenon, which is often a cause for bad generalisation. While constructive methods try to build up a minimal model by adding successive parameters (or units), the pruning approach starts off with an already built model, and tries to extract its essence by pruning useless parameters. This paper is a commented but non exhaustive state of the art of the field (see also [21]), essentially concerning Multi-Layer Perceptron (MLP) trained with any Least Squares (LS) method, for instance with Back-Propagation (BP). We distinguish two classes of pruning methods: during-learning pruning (Section 2), post-learning pruning (Section 3).

The first class of methods makes use of a special learning scheme that aims at reducing the complexity of the network as the learning process is carried out. The usefulness of parameters is determined as the network learns, and therefore dependently on that process. These methods are mainly weight pruning methods, as the learning occurs on weights.

The second class of methods uses a more modular approach and aims at removing useless parameters after the learning phase has reached some convergence. Pruning is carried out independently, on the resulting net, generally assuming that convergence is reached to simplify the matter.

2 During-learning pruning

2.1 Principles

These methods were introduced in the Neural Network literature in the context of Back-Propagation, and are known as 'Weight Decay' [16] [8] or 'Weight Elimination' [23]. The idea is to balance the goodness of fit (precision) with some other cost term (complexity):

$$O(\mathbf{w}) = E(\mathbf{w}) + \lambda C(\mathbf{w}). \tag{1}$$

$O(\mathbf{w})$ is the objective function that is to be minimized, with respect to $\mathbf{w}$, the vector of parameters. $E(\mathbf{w})$ is the fitting error term, usually taken to be the Mean Squared Error (MSE) over the learning base. $C(\mathbf{w})$ is the so called 'complexity term'. λ is used to fix the trade-off between complexity and goodness of fit. Usually, an increase in complexity of a model (number of parameters for example) will allow for a lower error to be reached, but at higher cost, and will probably lead to overfitting. Thus the terms $E(\mathbf{w})$ and $C(\mathbf{w})$ are antagonist in the minimisation process.

This 'complexity term' can also be seen as a way to embed *a priori* knowledge in the learning scheme, as the notion of 'complexity' will take different forms depending on the properties we wish to give to the model.

Function $O(\mathbf{w})$	Name	Ref
$\sum_i w_i^2$	Weight Decay	[16]
$\sum_i \lvert w_i \rvert$	Weight Decay	[8]
$\sum_i \frac{w_i^2}{1+w_i^2}$	Weight Elimination	[23]

Table 1: Complexity terms used in BP learning.

Table 1 is a summary of complexity terms that have been proposed in the literature. In the table, w_i denotes the i^{th} weight of the weight vector. The three terms will find direct explanation in the theoretical justification given later on.

Regardless of the diversity of functions $C(\mathbf{w})$, λ is a parameter that is difficult to tune. Mackay [19] gave some guidelines about that issue, and many practitioners tune it by hand. This parameter is in fact the weak point of these methods, as an incorrect (too large) value will convey disastrous consequences. It seems that setting λ to zero at the beginning of learning and then progressively increasing it as the network learns is the best rule of thumb. We implemented and evaluated these methods [10], with weight decay and weight elimination terms on a simple function approximation task. Our results corroborate what Crespo [8] and others concluded on weight decay methods used directly, with λ tuned by hand: the balance parameter is hard to tune. At least, BP is very sensitive to that parameter. Moreover, the use of weight decay terms somewhat slows down BP convergence (in number of iterations).

Finally, the two main practical questions are: How to chose the complexity term ? How to tune the parameter λ ? In the following, we answer to the first, one using statistical interpretation proposed by [24].

2.2 Theoretical justification

From a statistical point of view, the weight vector $\mathbf{w}$ is an estimation of the parameter vector, based on the data of the learning base. Then, it can be associated with a density, say $p(\mathbf{w}/\mathbf{y}_d)$ if we call $\mathbf{y}_d$ the desired output (data of the learning base). From Bayes rule, we can write

$$p(\mathbf{w}/\mathbf{y}_d) = \frac{p(\mathbf{y}_d/\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}_d)}. \tag{2}$$

Then, an optimal choice for $\mathbf{w}$ can be defined by maximising $p(\mathbf{w}/\mathbf{y}_d)$ with respect to $\mathbf{w}$ (Maximum A Posteriori equation). This is equivalent to maximize

$$\ln[p(\mathbf{y}_d/\mathbf{w})] + \ln[p(\mathbf{w})]. \tag{3}$$

If we have no prior information concerning $\mathbf{w}$ (i.e setting $p(\mathbf{w}) = \text{constant}$), maximization of (3) leads to Maximum Likelihood estimation. Moreover, supposing observations are corrupted by an additive zero-mean Gaussian noise with variance σ^2

$$y_d(x^i) = y_d^i = y^i + n^i.$$

Then, if n^i are independent, the density is

$$p(\mathbf{y}_d/\mathbf{w}) = K \prod_i \exp(-\frac{(y^i - y_d^i)^2}{2\sigma^2}).$$

And it leads to the well known result that maximize (3) is equivalent (in the Gaussian case) to minimize the Mean Square Error:

$$E(\mathbf{w}) = \frac{1}{2\sigma^2} \sum_i (y^i - y_d^i)^2. \tag{4}$$

But if we do have prior information on weights, or if we decide to constrain the weights to give them a known property, we can set $p(\mathbf{w})$ to a function of $\mathbf{w}$ rather than to a constant. Gaussian or Laplacian distributions are possible choices that lead to classical weight decay terms:

If weight are zero-mean identically distributed Gaussian variables: $N(0, \sigma_w^2)$, then (assuming each weight is independent of each other) the density of the weight vector is: $p(\mathbf{w}) = K \cdot \prod_i \exp(-\frac{w_i^2}{2\sigma_w^2})$ with K constant. Maximization of $p(\mathbf{w})$ leads to the minimisation of $C(\mathbf{w}) = \frac{1}{2\sigma_w^2} \sum w_i^2$.

If weight are Laplacian random variables $L(0, \mu_w)$, then (assuming each weight is independent of each other) the density of the weight vector is: $p(\mathbf{w}) = K \cdot \prod_i \exp(-\frac{|w_i|}{\mu_w})$ with K constant. By similar way, it leads to the minimization of $C(\mathbf{w}) = \frac{1}{\mu_w} \sum |w_i|$.

Similarly, last exemple of Table 1 can be derived from a Cauchy distribution [24].

All these particular distributions assume that weights are independent—this is why the $p(\mathbf{w})$ can be written as a product. This is a strongly limiting hypothesis, because as the learning process goes on, weights are certainly less and less independent.

The constant terms that appear in front of the sums can be included in the final λ of eq. (1). As they correspond to unknown noise level and unknown weight distribution parameter, they can be eliminated using a classical ignorance prior for scale parameters.

This Bayesian interpretation of weight decay methods provided by Williams gives theoretical justification for these methods. The author also justifies the use of a Laplacian prior [24] following Jayne's principles, and gives an alternative to Mackay's way [19] to calculate the balance parameter λ, also justified by theoretical principles.

2.3 Pruning with robust backpropagation

All the previously described methods make use of an additional term in the objective function, and thus constrain the net to 'prune' itself useless parameters.

In a context of fault tolerance, Kerlirzin *et al.* show [17] that, against all odds, MLP trained with BP are not robust models as regards pruning: random pruning of weights (or units) can imply disastrous results. Therefore, they proposed an alternate algorithm for training robust MLP, the so called 'BP with mortality' algorithm.

It is essentially a double stochastic BP, carried out on a probabilised error function. Usually, the error to be minimised is the MSE $E(\mathbf{w})$ taken on the learning base. If we consider that some weights can be eliminated (intentionally or not), this means that only any subset K of the weights is relevant. Let us consider the probability of subset K to occur, say $P(K)$, a probabilised version of the error can be derived:

$$E_P(\mathbf{w}) = \sum_K E(w_i \in K) \cdot P(K).$$

The new stochastic aspect is that configurations (K) are also taken at random, in much the same way patterns are. Sure, that makes the training slower, but this special learning scheme has the desirable effect of sharing the encoded information on weights in a more uniform way than BP does.

Experimental results [17] are very convincing. This gives a simple strategy to prune a MLP: as weights are trained in a very specific way, they can be removed at random without much loss of performance.

3 Post-Learning Pruning

The previously presented methods act as the learning process goes on, possibly performing global optimisation, but at higher expense. Other techniques, some-

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 129-140

times called 'sensitivity based methods' act on a previously trained net, and perform pruning as a post-processing.

3.1 Optimal Brain Damage and Optimal Brain Surgeon

Optimal Brain Damage (OBD) is one of the first pruning method proposed by Le Cun *et al.* in 1990 [18]. Optimal Brain Surgeon (OBS), which is a reffinement of OBD proposed by Hassibi in [15], allows for a one shot update of the weights after pruning, and avoids re-training.

The methods are based on a Taylor expansion of the error $E(\mathbf{w})$ around the minimum reached after training. The idea is that deleting a parameter w_i is understood as bringing it to zero, thus causing a variation $\delta w_i = (0 - w_i)$ of the parameter vector $\mathbf{w}$. Using a Taylor expansion of E around $\mathbf{w}$ relates the variation of the error to the variation of the parameters:

$$\delta E = \sum_i \frac{\partial E}{\partial w_i}\delta w_i + \sum_i \frac{\partial^2 E}{\partial w_i^2}\delta w_i^2 + \sum_{i,j/i\neq j} \frac{\partial^2 E}{\partial w_i \partial w_j}\delta w_i \delta w_j + O(\| \delta \mathbf{w} \|^3). \quad (5)$$

In OBD, making the hypothesis that (i) E is almost quadratic (order 2 development is enough), (ii) the Hessian is diagonal (cross terms vanish), (iii) the network is at a local minimum (first term vanishes), Le Cun *et al.* derive the simplified expression of saliency of weight w_i:

$$s_i = w_i^2 \frac{\partial^2 E}{\partial w_i^2}. \quad (6)$$

This means that deleting parameter w_i (setting it to 0) causes an increase of E of s_i. Then, the w_i's associated with the smallest s_i's are pruned. Retraining is necessary to compensate for the deletion of a parameter.

In OBS, Hassibi formalises the problem as a constrained optimisation. In fact, one wishes to minimise the increase in E given the fact that we delete parameter w_i. This constraint can be written

$$\delta \mathbf{w} \cdot \mathbf{e}_i + w_i = 0,$$

where e_i is the canonical vector selecting the i^{th} coordinate of $\mathbf{w}$. Then, using the matrix expression of (5), a Lagrangian can be written and minimised

$$L(\delta \mathbf{w}) = \frac{1}{2}\delta \mathbf{w}^t \cdot H \cdot \delta \mathbf{w} + \lambda \cdot (\delta \mathbf{w} \cdot \mathbf{e}_i + w_i).$$

Differentiating with respect to $\delta \mathbf{w}$ gives the constrained minimum and corresponding variation of $\mathbf{w}$

$$L_i = \frac{w_i^2}{[H^{-1}]_{ii}}, \quad (7)$$

$$\delta \mathbf{w}^t = -\frac{w_i}{[H^{-1}]_{ii}} H^{-1} e_i. \quad (8)$$

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 129-140

Equation (7) is the generalisation of (6) to the case of a full Hessian. Eq. (8) allows for a one shot update of the weight vector $\mathbf{w}$, automatically setting w_i to zero (constraint) and slightly changing the other parameters, without retraining.

In the two methods, there is no stopping criterion for pruning: How to chose the threshold (on s_i or L_i) which drives the pruning ? A justified stop criterion will be paliated with the method explained in the next section.

3.2 Statistical Stepwise Method

This method was proposed by M. Cottrell *et al.* in [7]. Though it is founded on a statistical point of view, the pruning criterion is very close to OBD and OBS, mainly because the underlying mathematical tools are similar [11].

The Statistical Stepwise Method (SSM) consists in pruning the weights that are statistically non significant, but only if the resulting net is "better" than the previous one. The notion of quality for a network (model) is defined as Akaike's B Information Criterion (BIC) [1].

The key is to consider that the vector of estimated parameters obtained after a LS optimization is a vector of random variables, $\hat{\mathbf{w}}$, known as the LS estimator of the theoretical solution $\mathbf{w}^*$. As a matter of fact, even in the case of non linear LS estimator [2], $\hat{\mathbf{w}}$ is known to be asymptotically Gaussian:

$$\sqrt{T}(\hat{\mathbf{w}} - \mathbf{w}^*) \underset{T\to\infty}{\xrightarrow{\mathcal{L}}} N(0, \sigma_r^2 \Sigma^{-1}) \tag{9}$$

where T is the size of the learning base, σ_r^2 is the residual variance of the model (MSE) and Σ the Hessian of the error E with respect to the parameters at $\mathbf{w}^*$.

Now, we can test (statistically) the nullity of each component of $\hat{\mathbf{w}}$, using a standard Gaussian test, carried out on the mean of the estimator ($\mathbf{w}^*$ from (9)) in order to decide whether if w_i is null or not. The criterion is thus the quantity defined for the test rule:

$$t_i = \left| \frac{\hat{w}_i}{\hat{\sigma}(\hat{w}_i)} \right|, \tag{10}$$

where σ_r^2 is estimated by $\hat{\sigma}_r^2$, and the variance of $\hat{w}_i$ is then estimated by $\hat{\sigma}_r^2$ multiplied by the $i^t h$ diagonal term of the inverse of the Hessian taken in $\hat{\mathbf{w}}$.

For a fixed significance level of 5%, w_i is considered non significant as soon as $t_i < 1.96$. It is actually pruned only if the BIC of the pruned net, *after retraining*, is smaller than the BIC of the current net. If not, pruning is stopped. We recall that the Akaike's B Information Criterion is:

$$BIC = \log(\frac{\sigma_r^2}{T}) + n_p \frac{\log(T)}{T}$$

where n_p is the number of parameters of the network.

3.3 Comparison

It is simple to see from eq. (7) and (10) that the selection criterion are identical. More precisely, $\hat{\sigma}(\hat{w}_i)$ is proportional to the square root of the diagonal term of the inverse of the Hessian. Thus, L_i as well as s_i (with the approximation of diagonal Hessian) are just the square of t_i, and as we seek minima, the three criteria will select the same weights.

SSM provides a theoretically justified threshold for the selection/pruning phase where OBD and OBS do n't. On the other hand, OBS provides a one shot update of the parameter vector, that could be used with profit in SSM. Finally, the idea of post-pruning verification using the BIC criterion could be applied to any pruning procedure.

4 Direct Pruning

In previous sections, we presented methods that perform pruning on weights in a rather indirect way. Other methods take a more direct approach and attempt to remove units directly. This has the potential interest of speeding up the pruning and provides simple interpretation.

4.1 Inside the MLP

Siestmas & Dow proposed a strategy for pruning a MLP [22] that has been properly formulated and automated by Chung & Lee [5]. The principle of their method is to consider a MLP as a succession of layers that project the set of incoming patterns onto a set of output patterns (for each layer). The first set of input patterns is the learning base, that is projected onto a first (internal) output set, that will in turn be the input set to the next layer. From this point of view, a simple study of the input/output relations at a layer level gives an obvious strategy to remove useless units: (i) a unit having stable output for every input pattern can be discarded, (ii) if two units have identical or inverted outputs on the input set, one can be discarded, (iii) if the number of different patterns remains the same between input and output, the full layer can be discarded, (iv) units that are not 'essential' for the generation of the output patterns can be discarded.

This last point is not properly formalised in [22] but implies a notion of orthogonality to express the 'independence' of units. This stage of pruning requires a little more efforts to be automated. The two first points correspond directly to classes (i), (ii) and (iii) in [5]. The last one is related to class (iv).

We will not develop further the explanations and formalisation given by Chung & Lee, for the philosophy has been given yet. The algorithm they give is a direct translation of their results, and requires a pass over the total internal set of patterns, for each layer. Moreover, retraining is performed (possibly locally) between each pruning, which is not very clever, as direct modifications could easily be applied on the remaining weights. It is thus very greedy as regards computing time and memory.

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 129-140

4.2 Local LS

Pelillo & Fanelli [20] proposed a different approach to that problem, though keeping the same line of thought: they wish to suppress one single unit in a layer, and update the remaining units of the layer in order to keep the 'internal inputs' to the next layer identical. Again, a pass over the complete learning base (N patterns) will be required to build the corresponding internal input/output set.

We wish to suppress unit u_{kl}, the k^{th} unit in layer l, and keep the output unchanged by a proper update δ_{ij} of the output weights w_{ijl} on this layer (thresholds are included in the notation as usual). We then write the equations ($\forall i, j$ and pattern p)

$$\sum_j w_{ijl} y_{jl}(p) = \sum_{j \neq k} (w_{ijl} + \delta_{ij}) y_{jl}(p),$$

where $y_{jl}(p)$ is the output of unit u_{jl} on pattern p. This is equivalent to

$$\sum_{j \neq k} \delta_{ij} y_{jl}(p) = w_{ikl} y_{kl}(p),$$

where i spans all the units in layer $l + 1$.

The authors propose to solve this system [1] $Ax = b$ to the unknown vector $x = [\delta_{ij}]$ using an iterative procedure (called pre-conditionned conjugate gradient) that consists in minimising the remainders $r_n = \| Ax_n - b \|$ and requires the definition of a initial point, usually chosen null ($x_0 = 0$). The method stops as soon as $\| x_n - x_{n+1} \| < \varepsilon$ fixed. The problem is (see footnote 1): Which unit shall we prune ? As we minimise remainders, starting from a minimum remainder r_0 sounds reasonable, and leads to chose the unit associated to vector b of minimum norm, as $r_0 = b$ for $x_0 = 0$. This criterion is obviously *ad hoc* and possibly under-optimal, but works well in practice, according to the authors. Moreover, they argue that Siestmas & Dow proposals correspond to a very particular conditioning of the system.

5 Other Methods

Different methods related to the Neural Network field have also been proposed to perform pruning. We will shortly describe their field and make breif commentaries.

5.1 Trees

For classification tasks, not only MLP or Bayes classifiers are efficient. Indeed, classification trees have been proposed earlier in the field of Artificial Intelligence or Computer Science. Recently, a correspondence was made between Trees and

[1] this system depends on the pruned unit —that is not determined yet—, thus on k, l.

Neural Networks [3]. As a matter of fact, mapping a tree onto a Neural Net is easy. And as there exist some well established optimal pruning procedures on trees [4], the Neural Network community could benefit from them.

Some approaches propose hybrid learning [9]. Some directly implant tree procedures in a neural fashion. But in the end, the tree structure is essential to the algorithm. The pruning phase relies on a previously built tree classifier, and gently degrades it. So no proper generalisation can be drawn for other type of classifiers. There is often a need to refer to the tree structure when pruning. This structure is lost when mapping a tree onto a Neural Net. This is why we do not develop this field any further, as it seems to be a closed one.

5.2 Genetic Algorithms

Some authors have proposed to use Genetic Algorithms for pruning —and more generally for designing a Neural Network architecture. We will only cite [13]. This field does not seem to be mature enough to provide efficient tools. Moreover, it often needs a great computational capacity.

5.3 Pruning density estimators

The kernel estimation of density [14] consists in putting a kernel centered at every data point and of tuning the width factor in some optimal way so as to obtain the best estimator. Such estimator is used in Bayesian Neural Classification schemes [6]. As this may become computationally prohibiting with large databases, some pre-processing such as vector quantisation (VQ) is usually performed in order to strongly reduce the size –thus the complexity– of the model. From now on, points or *centers* will refer to centroids that represent each cluster.

If the number of centroids after vector quantization is too large, usually it is necessarry to apply again another vector quantization from the complete data base, which is computationally heavy. An alternative idea would be to prune $n << N$ centroids, from a given kernel estimator based on N points. We derived such a method [12] in the case of fixed kernel estimator (each kernel has the same width). After pruning, we constrain the width factor to the theoretical value and, considering the density before pruning as a target, we move the remaining centroids in order to compensate for the deletion of some kernels.

We define a parameterised expression of the density estimator that will allow for pruning via parameter variation:

$$\phi\underbrace{(C_i,h,p)}_{P} = \sum_{i=1}^{n} \frac{1}{nh^d} K(\frac{x-C_i}{h}) + p \cdot \sum_{i=n+1}^{N} \frac{1}{Nh_N^d} K(\frac{x-C_i^N}{h_N})$$

where C_i, h and p are variable parameters, respectively the new centers, the new width and p is a 'pruning' parameter that is used to simulate the elimination of $N-n$ kernels (last term of ϕ). They form together the parameter vector P. Notice that n is the desired number of kernels, and is linked to h, thus ϕ is a

ESANN'1995 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 19-20-21 April 1995, D-Facto public., ISBN 2-9600049-3-0, pp. 129-140

function of h. The C_i^N are the old centers from estimator $\hat{f}_N$; if N denotes the total number of patterns, the C_i^N are just the patterns themselves.

Then, if we set the C_i's to their old values C_i^N, h to h_N and p to 1, we recover the estimated density $\hat{f}_N$ (call P_N the corresponding parameter vector). If we set p to 0, and h to h_n, we have a n-kernel net (pruned net) that is obtained after a variation of the parameters of ϕ, i.e. the centers C_i of the remaining kernels (first term in ϕ).

We now apply a constrained minimisation scheme inspired by OBS, in which we wish to minimize the error variation δE with constrain on two parameters of P:

$$\delta p = 0 - 1 \quad and \quad \delta h = h_n - h_N.$$

The first constraint indicates that we prune $N - n$ kernels. The second one indicates that we wish to use the 'optimal' width on our resulting estimator.

Then, we minimize the Lagrangian:

$$L(\delta P) = \frac{1}{2}\delta P^t \cdot H \cdot \delta P + \Lambda \cdot (M \delta P + B)$$

where Λ is the unknown Lagrange multiplier $\Lambda = [\lambda_1, \lambda_2]$, H is the Hessian matrix, B is the vector $[h_N - h_n, -1]$, and M is the matrix $M = [e_1, e_2]^t$, where e_1 and e_2 denote the canonical vectors associated with the first and second coordinates. Differentiating and using constraints, we find the constrained minimum and the associated δP:

$$\begin{aligned} Min &= \frac{1}{2} B^t A^{-1} B && (11) \\ \delta P^t &= -B^t A^{-1} M H^{-1} && (12) \end{aligned}$$

where matrix A is a 2×2 matrix defined by $A = MH^{-1}M^t$.

More details and experimental results are proposed in [12]. The method is efficient to prune a small number of kernels and works well in situations for which vector quantization does not. Thus, it can avoid another vector quantization procedure from the complete data base.

6 Conclusion

In this paper, we point out some relations between statistical methods and pruning.

In the during–learning pruning methods (like weight decay or weight elimination), the complexity term can be optimally chosen if the weight distribution is known and if weights are independent, which are very strong assumptions and explain practical difficulties of the methods.

Post–learning pruning methods (OBD and OBS) are based on cancellation of weights having small enough saliency. Surprisingly, such methods can be interpreted as cancelling the weight "statistically equal to zero" (while it is often claimed that small weights can be influent), and a simple hypothesis test provides consistent threshold of the saliency and pruning stopping criterion.

Direct methods, based on cancellation of internal units which can be easily compensated, have also been explored. Although most of the above methods have been used for MLP,but pruning may also be succesfully used with kernel estimators (Radial Basis functions).

Pruning methods are basic approaches to simplify networks and avoid overparametrization, and even overfitting (although it is difficult to prove). For that reason, we recommend to apply pruning under control of a criterion like Akaike's Information Criterion (AIC or BIC), as done in [7]. From a practical point of view, notice that pruning methods need heavy computations, typically involving Hessian computation, or average on the whole data base, etc.

Acknowledgement

This work has been partly supported by Esprit III project ELENA (6891).

References

[1] H. Akaïke. A new look at the statistical model identification. *IEEE Trans on Automatic Control*, 19(6):716–723, December 1974.

[2] T. Ameniya. *Advanced Econometrics.* Basil Blackwell, 1986.

[3] P. Bigot and M. Cosnard. Probabilistic decision trees and MLP. In *ESANN'93*, pages 91–96, 1993.

[4] P. Chou and al. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans on Information Theory*, 35(2):299–315, March 1989.

[5] F.L. Chung and T. Lee. A node pruning algorithm for backpropagation networks. *International Journal of Neural Systems*, 3:301–314, 1992.

[6] P. Comon. Supervised classification, a probabilistic approach. In *ESANN'95*, 1995.

[7] M. Cottrell, B. Girard, Y. Girard, and M. Mangeras. Times series and neural network: a statistical method for weight elimination. In *ESANN'93*, pages 157–164, 1993.

[8] J.L. Crespo and E. Mora. Tests of different regularization terms in small networks. In *IWANN'93*, 1993.

[9] G. Dreyfus, S. Knerr, and L. Personnaz. From theory to silicon: an efficient procedure for design of neural classifiers and its application to automatic recognition of handwritten digits. In *Neuronîmes'91*, pages 149–158, 1991.

[10] O. Fambon. Méthodes d'élagage pour les réseaux de neurones. Rapport de DEA de sciences cognitives, Institut National Polytechnique de Grenoble - France, Septembre 1993.

[11] O. Fambon and C. Jutten. A comparison of two weight pruning methods. In *ESANN'94*, pages 147–152, 1994.

[12] O. Fambon and C. Jutten. Pruning kernel density estimators. In *ESANN'95*, 1995.

[13] F. Gruau. A learning and pruning algorithm for genetic boolean neural networks. In *ESANN'93*, pages 57–63, 1993.

[14] W. H ardle. *Smoothing Techniques, with Implementation in S.* Springer-Verlag, 1990.

[15] B. Hassibi and D. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS 5*, 1993.

[16] G.E. Hinton. Learning distributed reperesentation and concepts. In 8th *annual conference of the cognitive science society*, 1986.

[17] P. Kerlirzin and F. Vallet. Robustness in multilayer perceptrons. In *Neural Computation*, volume 5, pages 473–482, 1993.

[18] Y. Le Cun, J.S. Denker, and S.A. Sola. Optimal brain damage. In *NIPS 2*, pages 598–605, 1990.

[19] D.J.C. Mackay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4:448–472, 1992.

[20] M. Pelillo and A.M. Fanelli. A method of pruning layered feed forward neural networks. In A. Prieto, editor, *IWANN'93*, pages 278–283. Springer-Verlag Lecture Notes in Computer Sciences, 1993.

[21] R. Reed. Pruning Algorithms: A Survey. *IEEE Trans. on Neural Networks*, 4(5):740–747, 1994.

[22] J. Sietsmas and R.J.F. Dow. Neural net prunning: Why and how. In *ICNN 88*, volume 1, pages 325–333, 1988.

[23] A.S. Weigend and al. Generalisation by weight-elimination with applications to forecasting. In *NIPS 3*, 1991.

[24] P. M. Williams. Bayesian Regularisation and Pruning using a Laplace Prior. *Neural Computation*, 7(1):117–143, 1995.