Appeared in Neural Networks 1998

Automatic Early Stopping Using Cross Validation: Quantifying the Criteria

Lutz Prechelt (prechelt@ira.uka.de)
Fakultät für Informatik; Universität Karlsruhe
D-76128 Karlsruhe; Germany
Phone: +49/721/608-4068, Fax: -7343

December 12, 1997

Abstract

Cross validation can be used to detect when overfitting starts during supervised training of a neural network; training is then stopped before convergence to avoid the overfitting ("early stopping"). The exact criterion used for cross validation based early stopping, however, is chosen in an ad-hoc fashion by most researchers or training is stopped interactively. To aid a more well-founded selection of the stopping criterion, 14 different automatic stopping criteria from 3 classes were evaluated empirically for their efficiency and effectiveness in 12 different classification and approximation tasks using multi layer perceptrons with RPROP training. The experiments show that on the average slower stopping criteria allow for small improvements in generalization (on the order of 4%), but cost about factor 4 longer training time.

1 Training for generalization

When training a neural network, one is usually interested in obtaining a network with optimal generalization performance. Generalization performance means small error on examples not seen during training.

Because standard neural network architectures such as the fully connected multi layer perceptron almost always have too large a parameter space, such architectures are prone to overfitting (Geman, Bienenstock & Doursat, 1992). While the network *seems* to get better and better (the error on the training set decreases), at some point during training it actually begins to get worse again (the error on unseen examples increases).

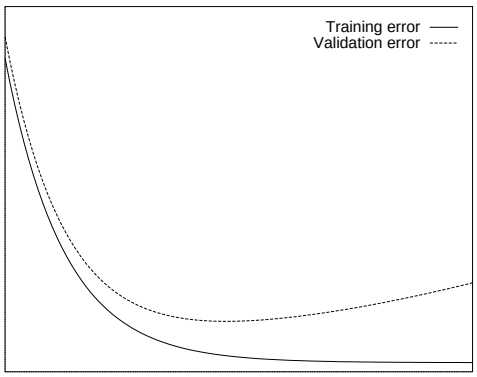
There are basically two ways to fight overfitting: reducing the number of dimensions of the parameter space or reducing the effective size of each dimension. The parameters are usually the connection weights in the network. The corresponding techniques used in neural network training to reduce the number of parameters, i.e., the number of dimensions of the parameter space, are greedy constructive learning (e.g. Fahlman & Lebiere, 1990), pruning (e.g. Le Cun, Denker & Solla, 1990; Hassibi & Stork, 1992; Levin, Leen & Moody, 1994), or weight sharing (e.g. Nowlan & Hinton, 1992). The corresponding NN techniques for reducing the size of each parameter dimension are regularization such as weight decay (e.g. Krogh & Hertz, 1992) and others (e.g. Weigend, Rumelhart & Huberman, 1991) or early stopping (Morgan & Bourlard, 1990). See also (Reed, 1993; Fiesler, 1994) for an overview and (Finnoff, Hergert & Zimmermann, 1993) for an experimental comparison.

Early stopping is widely used because it is simple to understand and implement and has been reported to be superior to regularization methods in many cases, e.g. in (Finnoff, Hergert & Zimmermann, 1993). The method can be used either interactively, i.e., based on human judgement, or automatically, i.e., based on some formal stopping criterion. However, such automatic stopping criteria are usually chosen in an ad-hoc fashion today. The present paper aims at providing some quantitative data to guide the selection among automatic stopping criteria. The means to achieve this goal is an empirical investigation of the behavior of 14 different criteria on 12 different learning problems.

The following sections discuss the problem of early stopping in general, formally introduce three classes of stopping criteria, and then describe the idea, setup and results of the experimental study that measured the efficiency and the effectiveness of the criteria.

2 Ideal and real generalization curves

In most introductory papers on supervised neural network training one can find a diagram similar to the one shown in figure 1. It is claimed to show the evolution over time of the per-example error on the training set and on a test set not used for training (the *training curve* and the *generalization curve*). Given this behavior, it is clear how to do early stopping using cross validation: (1) Split the training data into a training set and a cross validation set, e.g. in a 2 to 1 proportion. (2) Train only on the training set and evaluate the per-example error on the validation set once in a while, e.g. after every fifth epoch.



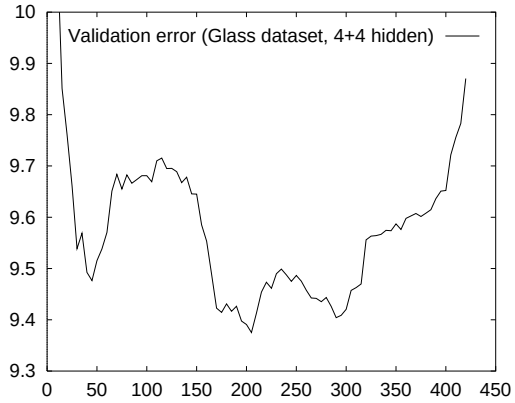


Figure 1: *(left) Idealized training and generalization error curves. Vertical: errors; horizontal: time*

Figure 2: *(right) A real generalization error curve. Vertical: Validation set error; horizontal: time (in training epochs).*

(3) Stop training as soon as the error on the cross validation set is higher than it was the last time it was checked. (4) Use the weights the network had in that previous step as the result of the training run. This approach uses the cross validation set to anticipate the behavior on the test set (or in real use), assuming that the error on both will be similar.

However, the real situation is a lot more complex. Real generalization curves almost always have more than one local minimum. (Baldi & Chauvin, 1991) showed for linear networks with n inputs and n outputs that up to n such local minima are possible; for multi layer networks, the situation is even worse. Thus, it is impossible in general to tell from the beginning of the curve whether the global minimum has already been seen or not, i.e., whether an increase in the generalization error indicates real overfitting or is just intermittent. Such a situation is shown in figure 2. This real generalization curve was measured during training of a two hidden layer network on the *glass1* problem (see below). The curve exhibits as many as 16 local minima in the validation set error before severe overfitting begins at about epoch 400; of these local minima, 4 are the global minimum up to where they occur. The optimal stopping point in this example would be epoch 205. Note that stopping in epoch 400 compared to stopping shortly after the first "deep" local minimum at epoch 45 trades an about sevenfold increase of learning time for an improvement of validation set performance by 1.1% (by finding the minimum at epoch 205). If representative training data is used, the validation error is an optimal estimation of the actual network performance; so we expect a 1.1% decrease of the generalization error in this case. Nevertheless, overfitting might sometimes go undetected because the validation set is not perfectly representative of the problem.

Unfortunately, this or any other generalization curve is not *typical* in the sense that all curves share the same qualitative behavior. Other curves might never reach a better minimum than the first, or than, say, the third; the mountains and valleys in the curve can be of very different width, height, and shape. The only thing all curves seem to have in common is that the differences between the first and the following local minima, if any, are not huge. Theoretical analyses of the error curves cannot yet be done for the interesting cases, e.g. multi layer perceptrons with sigmoid functions; today they are

possible for simpler cases only, namely for linear networks (Baldi & Chauvin, 1991; Wang & Venkatesh, 1994).

As we see, choosing a stopping criterion predominantly involves a tradeoff between training time and generalization error. However, some stopping criteria may typically find better tradeoffs that others. This leads to the question which criterion to use with cross validation to decide when to stop training. The present work provides empirical data in order to give an answer.

3 Actual stopping criteria

There are a number of plausible stopping criteria. This work evaluates three classes of them.

To formally describe the criteria, we need some definitions first. Let E be the objective function (error function) of the training algorithm, for example the squared error. Then $E_{tr}(t)$ is the average error per example over the training set, measured after epoch t. $E_{va}(t)$ is the corresponding error on the validation set and is used by the stopping criterion. $E_{te}(t)$ is the corresponding error on the test set; it is not known to the training algorithm but characterizes the quality of the network resulting from training.

The value $E_{opt}(t)$ is defined to be the lowest validation set error obtained in epochs up to t:

$$E_{opt}(t) = \min_{t' \le t} E_{va}(t')$$

Now we define the *generalization loss* at epoch t to be the relative increase of the validation error over the minimum-so-far (in percent):

$$GL(t) = 100 \cdot \left(\frac{E_{va}(t)}{E_{opt}(t)} - 1 \right)$$

A high generalization loss is one obvious candidate reason to stop training, because it directly indicates overfitting. This leads us to the first class of stopping criteria: stop as soon as the generalization loss exceeds a certain threshold. We define the class GL_α as

$$GL_\alpha : \text{ stop after first epoch } t \text{ with } GL(t) > \alpha$$

However, we might want to suppress stopping if the training is still progressing very rapidly. The reasoning behind this approach is that when the training error still decreases quickly, generalization losses have higher chance to be "repaired"; we assume that overfitting does not begin until the error decreases only slowly. To formalize this notion we define a *training strip of length* k to be a sequence of k epochs numbered $n+1 \ldots n+k$ where n is divisible by k. The training *progress* (in per thousand) measured after such a training strip is then

$$P_k(t) = 1000 \cdot \left(\frac{\sum_{t'=t-k+1}^{t} E_{tr}(t')}{k \cdot \min_{t'=t-k+1}^{t} E_{tr}(t')} - 1 \right)$$

that is, "how much was the average training error during the strip larger than the minimum training error during the strip?" Note that this progress measure is high for unstable phases of training, where the training set error goes up instead of down. This is intended, because many training algorithms sometimes produce such "jitter" by taking inappropriately large steps in weight space. The progress measure is, however, guaranteed to approach zero in the long run unless the training is globally unstable (e.g. oscillating).

Now we can define the second class of stopping criteria using the quotient of generalization loss and progress:

$$PQ_\alpha: \text{ stop after first end-of-strip epoch } t \text{ with } \frac{GL(t)}{P_k(t)} > \alpha$$

In the following we will always assume strips of length 5 and measure the cross validation error only at the end of each strip.

A third class of stopping criteria relies only on the sign of the changes in the generalization error. These criteria say "stop when the generalization error increased in s successive strips":

$$UP_s: \text{ stop after epoch } t \text{ iff } UP_{s-1} \text{ stops after epoch } t-k \text{ and} E_{va}(t) > E_{va}(t-k)$$

$$UP_1: \text{ stop after first end-of-strip epoch } t \text{ with } E_{va}(t) > E_{va}(t-k)$$

The idea behind this definition is that when the validation error has increased not only once but during s consecutive strips(!), we assume that such increases indicate the beginning of final overfitting, independent of how large the increases actually are. The UP criteria have the advantage that they measure change locally so that they can directly be used in the context of pruning algorithms, where errors must be allowed to remain much higher than previous minima over long training periods.

Note that none of these criteria can guarantee termination. We thus complement them by the rule that training is stopped when the progress drops below 0.1 and also after at most 3000 epochs.

All stopping criteria have in common the way they are used: They decide to stop at some time t during training and the result of the training is then the set of weights that exhibited the lowest validation error $E_{opt}(t)$. Note that in order to implement this scheme, only one duplicate weight set is needed.

4 Design of the study

For most efficient use of training time we would be interested in knowing which of these criteria will achieve how much generalization using how much training time on which kinds of problems. However, as said before, no direct mathematical analysis of criteria

with respect to these factors is possible today. Therefore, we resort to studying the criteria empirically.

To achieve a broad coverage, we use multiple different network topologies, multiple different learning tasks, and multiple different exemplars from each stopping criteria class. To keep the experiment feasible, only one training algorithm is used.

We are interested in answering the following questions:

1. Training time: How long will training take with each criterion, i.e., how *fast* or *slow* are they?
2. Efficiency: How much of this training time will be redundant, i.e., will occur after the finally found validation error minimum has been seen?
3. Effectiveness: How good will the resulting network performance be?
4. Tradeoffs: Which criteria provide the best time-performance tradeoff?
5. Quantification: How can the tradeoff be quantified?

To find the answers we record for a large number of runs when each criterion would stop and what the associated network performance would be.

To measure network performance, we partition each dataset into two disjoint parts: *Training data* and *test data*. The training data (and only that) is used for training the network; the test data is used to estimate the network performance after training has finished. The training data is further subdivided into a *training set* of examples used to adjust the network weights and a *validation set* of examples used to estimate network performance during training as required by the stopping criteria. In the setup described below, the validation set was never used for weight adjustment. This decision was made in order to obtain pure stopping criteria results. In a real application this would be a waste of training data and should be changed.

12 different problems were used, all from the PROBEN1 NN benchmark set (Prechelt, 1994). These problems form a sample of a quite broad class of domains, but are of course not universally representative of learning; see (Prechelt, 1994) for a discussion of how to characterize the PROBEN1 domains.

5 Experimental setup

The stopping criteria examined were GL_1, GL_2, GL_3, GL_5, $PQ_{0.5}$, $PQ_{0.75}$, PQ_1, PQ_2, PQ_3, UP_2, UP_3, UP_4, UP_6, and UP_8. A series of simulations using all of the above criteria was run, in which all criteria where evaluated simultaneously, i.e., each single training run returned one result for each of the criteria. This approach reduces the variance of the estimation.

All runs were done using the RPROP training algorithm (Riedmiller & Braun, 1993) using the squared error function and the parameters $\eta^+ = 1.1$, $\eta^- = 0.5$, $\Delta_0 \in 0.05 \ldots 0.2$ randomly per weight, $\Delta_{max} = 50$, $\Delta_{min} = 0$, initial weights $-0.5 \ldots 0.5$ randomly. RPROP is a fast backpropagation variant similar in spirit to quickprop (Fahlman, 1988). It is about

as fast as quickprop but more stable without adjustment of the parameters. RPROP requires epoch learning, i.e., the weights are updated only once per epoch. Therefore, the algorithm is fast without parameter tuning for small training sets but not recommendable for large training sets. That no parameter tuning is necessary for RPROP also helps to avoid the common methodological error of tuning parameters using the performance on the test sets.

The 12 problems have between 8 and 120 inputs, between 1 and 19 outputs, and between 214 and 7200 examples. All inputs and outputs are normalized to range $0 \ldots 1$. 8 of the problems are classification tasks using 1-of-n output encoding (*cancer, card, diabetes, gene, glass, heart, horse, soybean*, and *thyroid*), 3 are approximation tasks (*building, flare*, and *hearta*); all problems are real datasets from realistic application domains.

The examples of each problem were partitioned into training (50%), validation (25%), and test set (25% of examples) in three different random ways, resulting in 36 datasets. Each of these datasets was trained with 12 different feedforward network topologies: one hidden layer networks with 2, 4, 8, 16, 24, or 32 hidden nodes and two hidden layer networks with 2+2, 4+2, 4+4, 8+4, 8+8, or 16+8 hidden nodes in the first+second hidden layer, respectively; all these networks were fully connected including all possible shortcut connections. For each of the network topologies and each dataset, two runs were made with linear output units and one with sigmoidal output units using the activation function $f(x) = x/(1+|x|)$. A popular rule of thumb recommends to always use sigmoidal output units for classification tasks and linear output units for regression (approximation) tasks. This rule was not applied since it is too far from always being good; see (Prechelt, 1994).

Altogether, 1296 training runs were made for the comparison, giving 18144 stopping criteria performance records for the 14 criteria. 270 of these records (or 1.5%) from 125 different runs reached the 3000 epoch limit instead of using the stopping criterion itself.

6 Results and discussion

The results for each stopping criterion averaged over all 1296 runs are shown in table 1. I will now explain and interpret the entries in the table. Please note that much of the discussion is biased by the particular collection of criteria chosen for the study.

Basic definitions: For each run, we define $E_v(C)$ as the minimum validation set error found until criterion C indicates to stop; it is the error after epoch number $t_m(C)$ (read: "time of minimum"). $E_t(C)$ is the corresponding test set error and characterizes network performance. Stopping occurs after epoch $t_s(C)$ (read: "time of stop"). A *best* criterion $\hat{C}$ of a particular run is one with minimum t_s of all those (among the examined) with minimum E_v, i.e., a criterion that found the best validation set error fastest. There may be several best, because multiple criteria may stop at the same epoch. Note that *the* criterion $\hat{C}$ does not really exist as such in general because it changes from run to run. C is called *good* in a particular run if $E_v(C) = E_v(\hat{C})$, i.e., if it is among those that found the lowest validation set error, no matter how fast or slow. We now discuss the five questions raised above.

	training time		efficiency and effectiveness			
C	$S_{\hat{c}}(C)$	$S_{GL_2}(C)$	$r(C)$	$B_{\hat{c}}(C)$	$B_{GL_2}(C)$	$P_g(C)$
UP_2	0.792	0.766	0.277	1.055	1.024	0.587
GL_1	0.956	0.823	0.308	1.044	1.010	*0.680
UP_3	1.010	1.264	0.419	*1.026	1.003	0.631
GL_2	1.237	1.000	0.514	1.034	1.000	*0.723
UP_4	1.243	1.566	0.599	*1.020	0.997	0.666
$PQ_{0.5}$	1.253	1.334	0.663	1.027	1.002	0.658
$PQ_{0.75}$	1.466	1.614	0.863	1.021	0.998	0.682
GL_3	1.550	1.450	0.712	1.025	0.994	*0.748
PQ_1	1.635	1.796	1.038	1.018	0.994	0.704
UP_6	1.786	2.381	1.125	*1.012	0.990	0.737
GL_5	2.014	2.013	1.162	1.021	0.991	*0.772
PQ_2	2.184	2.510	1.636	1.012	0.990	0.768
UP_8	2.485	3.259	1.823	1.010	0.988	0.759
PQ_3	2.614	3.095	2.140	1.009	0.988	0.800

Table 1: *Behavior of stopping criteria. $S_{GL_2}(C)$ is normalized training time, $B_{GL_2}(C)$ is normalized test error (both relative to GL_2). For further description please refer to the text.*

1. *Training time:* The *slowness* of a criterion C in a run, relative to another criterion x is $S_x(C) := t_s(C)/t_s(x)$, i.e., the relative total training time. As we see, the times relative to a fixed criterion as shown in column $S_{GL_2}(C)$ vary by more than a factor of 4. Therefore, the decision for a particular stopping criterion influences training times dramatically, even if one considers only the range of criteria used here. In contrast, even the slowest criteria train only about 2.5 times as long as the fastest criterion of each run that finds the same result, as indicated in column $S_{\hat{C}}(C)$. This shows that the training times are not completely unreasonable even for the slower criteria, but do indeed pay off to some degree.
2. *Efficiency:* The redundancy of a criterion can be defined as $r(C) := (t_s(C)/t_m(C)) - 1$. It characterizes how long the training continues after the final solution has been found. $r(C) = 0$ would be perfect, $r(C) = 1$ means that the criterion trains twice as long as necessary. Low values indicate efficient criteria. As we see, the slower a criterion is, the less efficient it tends to get. Even the fastest criteria "waste" about one fifth of overall training time. The slower criteria train more than twice as long as would be necessary for finding the same solution.
3. *Effectiveness:* We define the *badness* of a criterion C in a run relative to another criterion x as $B_x(C) := E_t(C)/E_t(x)$, i.e., its relative error on the test set. $P_g(C)$ is the fraction of the 1296 runs in which C was a good criterion. This is an estimate for the probability that C is good in a run. As we see from the P_g column, even the fastest criteria are fairly effective. They reach a result as good as the best-of-that-run criteria in about 60% of the cases. On the other hand, even the slowest criteria are not at all infallible; they achieve about 80%. So to obtain the best possible results, a conjunction of all three criteria classes has to be used. However, P_g says nothing about how *far* from the optimum the remaining runs are. Columns $B_{\hat{C}}(C)$ and $B_{GL_2}(C)$

indicate that these differences are usually rather small: column $B_{GL_2}(C)$ shows that even the criteria with the lowest error achieve only about 1% lower error on the average than the relatively fast criterion GL_2. In column $B_{\hat{(}C}(C)$ we see that even several only modestly slow criteria have just about 2% higher error on the average than the best criteria of the same run.

4. *Best tradeoffs:* Despite the common overall trend, some criteria may be more cost-effective than others, i.e., provide better tradeoffs between training time and resulting network performance. Column $B_{\hat{c}}$ of the table suggests that the best tradeoffs between test set performance and training time are (in order of increasing willingness to spend lots of training time) UP_3, UP_4, and UP_6, if one wants to minimize the expected network performance from a single run. If on the other hand one wants to make several runs and pick the network that seems to be best (based on its validation set error), P_g is the relevant metric and the GL criteria are preferable. The criteria with best tradeoffs are marked with a star in the table. Figure 3 illustrates these results. The upper curve corresponds to column $B_{\hat{C}}$ of the table (plotted against column $S_{\hat{C}}$); local minima indicate criteria with the best tradeoffs. The lower curve corresponds to column P_g; local maxima indicate the criteria with the best tradeoffs.

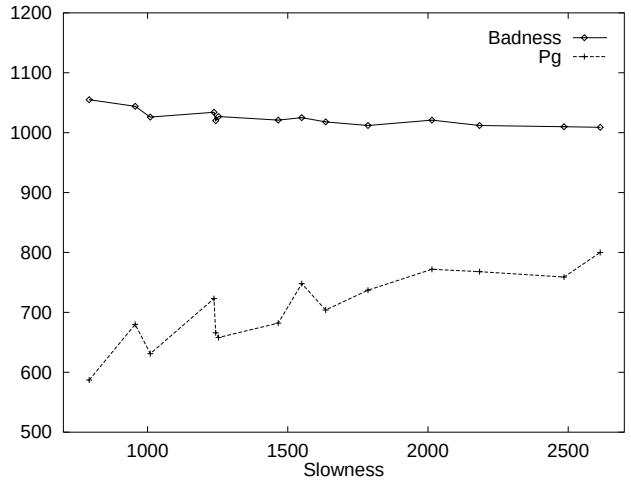


Figure 3: *Badness $B_{\hat{C}}(C)$ and P_g against slowness $S_{\hat{C}}(C)$ of criteria*

5. *Quantification:* From columns $S_{GL_2}(C)$ and $B_{GL_2}(C)$ we can quantify the tradeoff involved in the selection of a stopping criterion as follows: In the range of criteria examined we can roughly trade a 4% decrease in test set error (from 1.024 to 0.988) for an about fourfold increase in training time (from 0.766 to 3.095). Within this range, some criteria are somewhat better than others, but no panacea exists among the criteria examined in this study.

I also tried to find out whether similar results hold for more specialized circumstances such as only large or only small networks, only large or only small data sets or only particular learning problems. To do this, a factor analysis was performed by reviewing appropriate subsets of the data separately. The results indicate that generally the same trends hold for specialized circumstances within the limits of the study. One notable exception was the fact that for very small networks the PQ criteria are more cost-effective than both the GL and the UP criteria for minimizing $B_{\hat{C}}(C)$. An explanation of this lies in the fact

that such small networks do not overfit severely; in this case it is advantageous to take training progress into account as an additional factor to determine when to stop training.

7 Conclusion and further work

This work studied three classes of stopping criteria, namely GL, UP, and PQ on a variety of learning problems. The results indicate that "slower" criteria, which stop later than others, on the average indeed lead to improved generalization compared to "faster" ones. However, the training time that has to be expended for such improvements is significant. Systematic differences between the criteria *classes*, if any, are insignificant.

It remains an open question whether and how the above results apply to other training algorithms, other error functions, and in particular other problem domains. Future work should address these issues in order to provide clear quantitative engineering rules for network construction using early stopping. In particular, a theory should be built that quantitatively explains the empirical data. Such a theory would then have to be validated by further empirical studies. Only such a theory can overcome the inherent limitation of empirical work: the difficulty in generalizing the results to other situations.

For training setups similar to the one used in this work, the following rules can be used to select a stopping criterion:

1. Use fast stopping criteria unless small improvements of network performance (e.g. 4%) are worth large increases of training time (e.g. factor 4).
2. To maximize the probability of finding a "good" solution (as opposed to maximizing the average quality of solutions), use a GL criterion.
3. To minimize the average quality of solutions, use a PQ criterion if the network overfits only very little or an UP criterion otherwise.

References

Baldi, P. and Chauvin, Y. (1991). Temporal evolution of generalization during learning in linear networks. *Neural Computation*, 3:589–603.

Cowan, J. D., Tesauro, G., and Alspector, J., editors (1994). *Advances in Neural Information Processing Systems 6*, San Mateo, CA. Morgan Kaufman Publishers Inc.

Cun, Y. L., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *(Touretzky, 1990)*, pages 598–605.

(efiesler@idiap.ch), Fiesler, E. (1994). Comparative bibliography of ontogenic neural networks. (submitted for publication).

Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Fahlman, S. E. and Lebiere, C. (1990). The Cascade-Correlation learning architecture. In *(Touretzky, 1990)*, pages 524–532.

Finnoff, W., Hergert, F., and Zimmermann, H. G. (1993). Improving model selection by nonconvergent methods. *Neural Networks*, 6:771–783.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

Hanson, S. J., Cowan, J. D., and Giles, C. L., editors (1993). *Advances in Neural Information Processing Systems 5*, San Mateo, CA. Morgan Kaufman Publishers Inc.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *(Hanson et al., 1993)*, pages 164–171.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *(Moody et al., 1992)*, pages 950–957.

Levin, A. U., Leen, T. K., and Moody, J. E. (1994). Fast pruning using principal components. In *(Cowan et al., 1994)*.

Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors (1991). *Advances in Neural Information Processing Systems 3*, San Mateo, CA. Morgan Kaufman Publishers Inc.

Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors (1992). *Advances in Neural Information Processing Systems 4*, San Mateo, CA. Morgan Kaufman Publishers Inc.

Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In *(Touretzky, 1990)*, pages 630–637.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493.

Prechelt, L. (1994). PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany. Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.gz on ftp.ira.uka.de.

Reed, R. (1993). Pruning algorithms — a survey. *IEEE Transactions on Neural Networks*, 4(5):740–746.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA.

Touretzky, D. S., editor (1990). *Advances in Neural Information Processing Systems 2*, San Mateo, CA. Morgan Kaufman Publishers Inc.

Wang, C., Venkatesh, S. S., and Judd, J. S. (1994). Optimal stopping and effective machine complexity in learning. In *(Cowan et al., 1994)*.

Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In *(Lippmann et al., 1991)*, pages 875–882.