A Simple Trick for Estimating the Weight Decay Parameter

Thorsteinn S. Rögnvaldsson

Centre for Computer Architecture (CCA), Halmstad University, P.O. Box 823,
S-301 18 Halmstad, Sweden.
email: denni@cca.hh.se

Abstract. We present a simple trick to get an approximate estimate of the weight decay parameter λ. The method combines early stopping and weight decay, into the estimate

$$\hat{\lambda} = \|\nabla E(\boldsymbol{W}_{es})\| / \|2\boldsymbol{W}_{es}\|,$$

where $\boldsymbol{W}_{es}$ is the set of weights at the early stopping point, and $E(\boldsymbol{W})$ is the training data fit error.
The estimate is demonstrated and compared to the standard cross-validation procedure for λ selection on one synthetic and four real life data sets. The result is that $\hat{\lambda}$ is as good an estimator for the optimal weight decay parameter value as the standard search estimate, but orders of magnitude quicker to compute.
The results also show that weight decay can produce solutions that are significantly superior to committees of networks trained with early stopping.

1 Introduction

A regression problem which does not put constraints on the model used is ill-posed [21], because there are infinitely many functions that can fit a finite set of training data perfectly. Furthermore, real life data sets tend to have noisy inputs and/or outputs, which is why models that fit the data perfectly tend to be poor in terms of out-of-sample performance. Since the modeller's task is to find a model for the underlying function while not overfitting to the noise, models have to be based on criteria which include other qualities besides their fit to the training data.

In the neural network community the two most common methods to avoid overfitting are *early stopping* and *weight decay* [17]. Early stopping has the advantage of being quick, since it shortens the training time, but the disadvantage of being poorly defined and not making full use of the available data. Weight decay, on the other hand, has the advantage of being well defined, but the disadvantage of being quite time consuming. This is because much time is spent with selecting a suitable value for the weight decay parameter (λ), by searching

over several values of λ and estimating the out-of-sample performance using e.g. cross validation [25].

In this paper, we present a very simple method for estimating the weight decay parameter, for the standard weight decay case. This method combines early stopping with weight decay, thus merging the quickness of early stopping with the more well defined weight decay method, providing a weight decay parameter which is essentially as good as the standard search method estimate when tested empirically.

We also demonstrate in this paper that the arduous process of selecting λ can be rewarding compared to simpler methods, like e.g. combining networks into committees [16].

The paper is organized as follows: In section 2 we present the background of how and why weight decay or early stopping should be used. In section 3 we review the standard method for selecting λ and also introduce our new estimate. In section 4 we give empirical evidence on how well the method works, and in section 5 we summarize our conclusions.

2 Ill-Posed Problems, Regularization, and Such Things...

2.1 Ill-Posed Problems

In what follows, we denote the input data by $\boldsymbol{x}(n)$, the target data by $y(n)$, and the model (neural network) output by $f(\boldsymbol{W}, \boldsymbol{x}(n))$, where $\boldsymbol{W}$ denotes the parameters (weights) for the model. We assume a target data generating process of the form

$$y(n) = \phi[\boldsymbol{x}(n)] + \varepsilon(n) \tag{1}$$

where ϕ is the *underlying function* and $\varepsilon(n)$ are sampled from a stationary uncorrelated (IID) zero mean noise process with variance σ^2. We select models f from a model family F, e.g. multilayer perceptrons, to learn an approximation to the underlying function ϕ, based on the training data. That is, we are searching for

$$f^* \equiv f(\boldsymbol{W}^*) \in F \text{ such that } E(f^*, \phi) \leq E(f, \phi) \ \forall \ f \in F, \tag{2}$$

where $E(f, \phi)$ is a measure of the "distance" between the model f and the true model ϕ. Since we only have access to the target values y, and not the underlying function ϕ, $E(f, \phi)$ is often taken to be the mean square error

$$E(f, \phi) \rightarrow E(f, y) = E(\boldsymbol{W}) = \frac{1}{2N} \sum_{n=1}^{N} [y(n) - f(\boldsymbol{W}, \boldsymbol{x}(n))]^2. \tag{3}$$

Unfortunately, minimizing (3) is, more often than not, an ill-posed problem. That is, it does not meet the following three requirements [21]:

- The model (e.g. neural network) can learn the function training data, i.e. there *exists* a solution $f^* \in F$.
- The solution is *unique*.

– The solution is *stable* under small variations in the training data set. For instance, training with two slightly different training data sets sampled from the same process must result in similar solutions (similar when evaluated on e.g. test data).

The first and second of these requirements are often not considered serious problems. It is always possible to find a multilayer perceptron that learns the training data perfectly by using many internal units, since any continuous function can be constructed with a single hidden layer network with sigmoid units (see e.g. [6]), and we may be happy with any solution and ignore questions on uniqueness. However, a network that has learned the training data perfectly will be very sensitive to changes in the training data. Fulfilling the first requirement is thus usually in conflict with fulfilling the third requirement, which is a really important requirement. A solution which changes significantly with slightly different training sets will have very poor generalization properties.

2.2 Regularization

It is common to introduce so-called regularizers[1] in order to make the learning task well posed (or at least less ill-posed). That is, instead of only minimizing an error of fit measure like (3) we augment it with a regularization term $\lambda R(\boldsymbol{W})$ which expresses e.g. our prior beliefs about the solution.

The error functional then takes the form

$$E(\boldsymbol{W}) = \frac{1}{2N}\sum_{n=1}^{N}[y(n) - f(\boldsymbol{W}, \boldsymbol{x}(n))]^2 + \lambda R(\boldsymbol{W}) = E_0(\boldsymbol{W}) + \lambda R(\boldsymbol{W}), \quad (4)$$

where λ is the *regularization parameter* which weighs the importance of $R(\boldsymbol{W})$ relative to the error of fit $E_0(\boldsymbol{W})$.

The effect of the regularization term is to shrink the model family F, or make some models more likely than others. As a consequence, solutions become more stable to small perturbations in the training data.

The term "regularization" encompasses all techniques which make use of penalty terms added to the error measure to avoid overfitting. This includes e.g. weight decay [17], weight elimination [26], soft weight sharing [15], Laplacian weight decay [12] [27], and smoothness regularization [2] [9] [14] . Certain forms of "hints" [1] can also be called regularization.

2.3 Bias and Variance

The benefit of regularization is often described in the context of *model bias* and *model variance*. This originates from the separation of the expected generalization error $\langle E_{gen}\rangle$ into three terms [8]

$$\langle E_{gen}\rangle = \langle \int [y(\boldsymbol{x}) - f(\boldsymbol{x})]^2 \, p(\boldsymbol{x}) d\boldsymbol{x}\rangle$$

[1] Called "stabilizers" by Tikhonov [21].

$$= \int [\phi(\boldsymbol{x}) - \langle f(\boldsymbol{x}) \rangle]^2 \, p(\boldsymbol{x}) d\boldsymbol{x} + \langle \int [f(\boldsymbol{x}) - \langle f(\boldsymbol{x}) \rangle]^2 \, p(\boldsymbol{x}) d\boldsymbol{x} \rangle +$$
$$\langle \int [y(\boldsymbol{x}) - \phi(\boldsymbol{x})]^2 \, p(\boldsymbol{x}) d\boldsymbol{x} \rangle$$
$$= \text{Bias}^2 + \text{Variance} + \sigma^2, \tag{5}$$

where $\langle\rangle$ denotes taking the expectation over an ensemble of training sets. Here $p(\boldsymbol{x})$ denotes the input data probability density.

A high sensitivity to training data noise corresponds to a large model variance. A large bias term means either that $\phi \notin F$, or that ϕ is downweighted in favour of other models in F. We thus have a trade-off between model bias and model variance, which corresponds to the trade-off between the first and third requirements on well-posed problems.

Model bias is weighed versus model variance by selecting both a parametric form for $R(\boldsymbol{W})$ and an optimal[2] value for the regularization parameter λ.

Many neural network practicioners ignore the first part and choose weight decay by default, which corresponds to a Gaussian parametric form for the prior on $\boldsymbol{W}$. Weight decay is, however, not always the best choice (in fact, it is most certainly not the best choice for all problems). Weight decay does not for instance consider the function the network is producing, it only puts a constraint on the parameters. Another, perhaps more correct, choice would be to constrain the higher order derivatives of the network function (which is commonplace in statistics) like in e.g. [14].

2.4 Bayesian Framework

From a Bayesian and maximum likelihood perspective, *prior* information about the model (f) is weighed against the *likelihood* of the training data (D) through Bayes theorem (see [4] for a discussion on this). Denote the probability for observing data set D by $p(D)$, the prior distribution of models f by $p(f)$, and the likelihood for observing the data D, if f is the correct model, by $p(D|f)$. We then have for the posterior probability $p(f|D)$ for the model f given the observed data D

$$p(f|D) = \frac{p(D|f)p(f)}{p(D)} \Rightarrow$$
$$-\ln p(f|D) = -\log p(D|f) - \ln p(f) + \ln p(D) \Rightarrow$$
$$-\ln p(f|D) = \sum_{n=1}^{N} [y(n) - f(\boldsymbol{W}, \boldsymbol{x}(n))]^2 - \ln p(f) + \text{constant}, \tag{6}$$

where Gaussian noise ε is assumed in the last step. If we identify $2N\lambda R(\boldsymbol{W})$ with the negative logarithm of the model prior, $-\ln p(f)$, then maximizing $p(f|D)$ is equivalent to minimizing expression (4).

[2] Optimality is usually measured via cross-validation or some similar method.

From this perspective, choosing $R(\boldsymbol{W})$ is equivalent to choosing a parameterized form for the model prior $p(f)$, and selecting a value for λ corresponds to estimating the parameters for the prior.

2.5 Weight Decay

Weight decay [17] is the neural network equivalent to the Ridge Regression [11] method. In this case $R(\boldsymbol{W}) = \|\boldsymbol{W}\|^2 = \sum_k w_k^2$ and the error functional is

$$E(\boldsymbol{W}) = E_0(\boldsymbol{W}) + \lambda R(\boldsymbol{W}) = \frac{1}{2N}\sum_{n=1}^{N}[y(n) - f(\boldsymbol{W}, \boldsymbol{x}(n))]^2 + \lambda\|\boldsymbol{W}\|^2, \quad (7)$$

and λ is usually referred to as the *weight decay parameter*. In the Bayesian framework, weight decay means implicitly imposing the model prior

$$p[f(\boldsymbol{W})] = \sqrt{\frac{\lambda}{2\pi\sigma^2}}\exp\left(\frac{-\lambda\|\boldsymbol{W}\|^2}{2\sigma^2}\right) \quad (8)$$

where σ^2 is the variance of the noise in the data.

Weight decay often improves the generalization properties of neural network models, for reasons outlined above.

2.6 Early Stopping

Undoubtedly, the simplest and most widely used method to avoid overfitting is to stop training before the training set has been learned perfectly. This is done by setting aside a fraction of the training data for estimating the out-of-sample performance. This data set is called the validation data set. Training is then stopped when the error on the validation set starts to increase. Early stopping often shortens the training time significantly, but suffers from being ill-defined since there really is no well defined stopping point, and wasteful with data, since a part of the data is set aside.

There is a connection between early stopping and weight decay, if learning starts from small weights, since weight decay applies a potential which forces all weights towards zero. For instance, Sjöberg and Ljung [20] show that, if a constant learning rate η is used, the number of iterations n at which training is stopped is related to the weight decay parameter λ roughly as

$$\lambda \sim \frac{1}{2\eta n}. \quad (9)$$

This does not, however, mean that using early stopping is equivalent to using weight decay in practice. Expression (9) is based on a constant learning rate, a local expansion around the optimal stopping point, ignoring local minima, and assumes small input noise levels, which may not reflect the situation when overfitting is a serious problem. The choice of learning algorithm can also affect the

early stopping point, and one cannot expect (9) to hold exactly in the practical case.

Inspired by this connection between early stopping and weight decay, we use early stopping in the following section to estimate the weight decay parameter λ.

3 Estimating λ

From a pure Bayesian point of view, the prior is something we know/assume in advance and do not use the training data to select (see e.g. [5]). There is consequently no such thing as "λ selection" in the pure Bayesian model selection scheme. This is of course perfectly fine if the prior is correct. However, if we suspect that our choice of prior is less than perfect, then we are better off if we take an "empiricial Bayes" approach and use the data to tune the prior, through λ.

Several options for selecting λ have been proposed. Weigend et al. [26] present, for a slightly different weight cost term, a set of heuristic rules for changing λ during the training. Although Weigend et al. demonstrate the use of these heuristics on a couple of time series problems, we cannot get these rules to work consistently to our satisfaction. A more principled approach is to try several values of λ and estimate the out-of-sample error, either by correcting the training error, with some factor or term, or by using cross-validation. The former is done in e.g. [10], [23], and [24] (see also references therein). The latter is done by e.g. [25].

The method of using validation data for estimating the out-of-sample error is robust but slow since it requires training several models. We use cross-validation here because of its reliability.

3.1 Search Estimates

Finding the optimal λ requires the use of a search algorithm, which must be robust because the validation error can be very noisy. A simple and straightforward way is to start at some large λ where the validation error is large, due to the large model bias, and step towards lower values until the out-of-sample error becomes large again, due to the large model variance. In our experience, it often makes sense to do the search in $\log\lambda$ (i.e. with equally spaced increments in $\log\lambda$).

The result of such a search is a set of K values $\{\lambda_k\}$ with corresponding average n-fold cross validation errors $\{\log E_{nCV,k}\}$ and standard deviations $\{\sigma_{nCV,k}\}$ for the validation errors. These are defined as

$$\log E_{nCV,k} = \frac{1}{n}\sum_{j=1}^{n} \log E_{j,k} \tag{10}$$

$$\sigma^2_{nCV,k} = \frac{1}{n-1}\sum_{j=1}^{n} \left(\log E_{j,k} - \log E_{nCV,k}\right)^2 \tag{11}$$

when $\lambda = \lambda_k$. The number of validation data sets is n and $E_{j,k}$ denotes the validation error when $\lambda = \lambda_k$ and we use validation set j. Taking logarithms is motivated by our observation that the validation error distribution looks approximately log-normal and we use this in our selection of the optimal λ value below.

Once the search is finished, the optimal λ is selected. This is not necessarily trivial since a large range of values may look equally good, or one value may have a small average cross-validation error with a large variation in this error, and another value may have a slightly higher average cross-validation error with a small variation in this error. The simplest approach is to look at a plot of the validation errors versus λ and make a judgement on where the optimal λ is, but this adds an undesired subjectiveness to the choice. Another is to take a weighted average over the different λ values, which is what we use here (see Ripley [19] for a discussion on variants of λ selection methods).

Our estimate for the optimal λ is the value

$$\hat{\lambda}_{opt} = \frac{\sum_{k=1}^{K} n_k \lambda_k}{\sum_{k=1}^{K} n_k} \tag{12}$$

where n_k is the number of times λ_k corresponds to the minimum validation error when we sample validation errors from K log-normal distributions with means $\log E_{nCV,k}$ and standard deviations $\sigma_{nCV,k}$, assuming that the validation errors are independent. This is illustrated on a hypothetical example in Figure 1. The choice (12) was done after confirming that it often agrees well with our subjective choice for λ. We refer to this below as a "Monte Carlo estimate" of λ.

3.2 Two Early Stopping Estimates

If $\boldsymbol{W}^*$ is the set of weights when $E(\boldsymbol{W})$ in eq. (4) is minimized, then

$$\nabla E(\boldsymbol{W}^*) = \nabla E_0(\boldsymbol{W}^*) + \lambda \nabla R(\boldsymbol{W}^*) = 0, \tag{13}$$

which implies

$$\lambda = \frac{\|\nabla E_0(\boldsymbol{W}^*)\|}{\|\nabla R(\boldsymbol{W}^*)\|} \tag{14}$$

for the regularization parameter λ. Thus, if we have a reasonable estimate of $\boldsymbol{W}^*$, or of $\|\nabla E_0(\boldsymbol{W}^*)\|$ and $\|\nabla R(\boldsymbol{W}^*)\|$, then we can use this to estimate λ. An appealingly simple way of estimating $\boldsymbol{W}^*$ is to use early stopping, because of its connection with weight decay.

Denoting the set of weights at the early stopping point by $\boldsymbol{W}_{es}$, we have

$$\hat{\lambda}_1 = \frac{\|\nabla E_0(\boldsymbol{W}_{es})\|}{\|\nabla R(\boldsymbol{W}_{es})\|}, \tag{15}$$

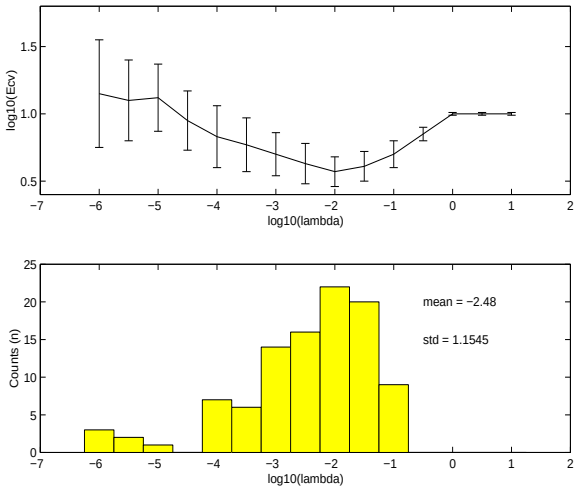

Fig. 1. Illustration of the procedure for estimating $\hat{\lambda}_{opt}$ on a hypothetical example. From the search we have a set of K lognormal distributions with means $\log E_{nCV,k}$ and variances $\sigma^2_{nCV,k}$, which is illustrated in the top plate. From these K distributions, we sample K error values and select the λ corresponding to the minimum error value as "winner". This is repeated several times (100 in the figure but 10,000 times in the experiments in the text) collecting statistics on how often different λ values are winners, and the mean $\log \lambda$ is computed. This is illustrated in the bottom plate, which shows the histogram resulting from sampling 100 times. From this we get $\log \hat{\lambda}_{opt} = -2.48 \pm 1.15$, which gives us $\lambda = 10^{-2.48} = 0.003$ for training the "best" network.

as a simple estimate for λ. A second possibility is to consider the whole set of linear equations defined by (13) and minimize the squared error

$$\begin{aligned} &\|\nabla E_0(\boldsymbol{W}_{es}) + \lambda \nabla R(\boldsymbol{W}_{es})\|^2 = \\ &\|\nabla E_0(\boldsymbol{W}_{es})\|^2 + 2\lambda \nabla E_0(\boldsymbol{W}_{es}) \cdot \nabla R(\boldsymbol{W}_{es}) + \lambda^2 \|\nabla R(\boldsymbol{W}_{es})\|^2 \end{aligned} \tag{16}$$

with respect to λ. That is, solving the equation

$$\frac{\partial}{\partial \lambda} \left\{ \|\nabla E_0(\boldsymbol{W}_{es}) + \lambda \nabla R(\boldsymbol{W}_{es})\|^2 \right\} = 0 \tag{17}$$

which gives

$$\hat{\lambda}_2 = \max \left[0, \frac{-\nabla E_0(\boldsymbol{W}_{es}) \cdot \nabla R(\boldsymbol{W}_{es})}{\|\nabla R(\boldsymbol{W}_{es})\|^2} \right]. \tag{18}$$

The estimate is bound from below since λ must be positive.

The second estimate, $\hat{\lambda}_2$, corresponds to a linear regression without intercept term on the set of points $\{\partial_i E_0(\boldsymbol{W}_{es}), \partial_i R(\boldsymbol{W}_{es})\}$, whereas the first estimate,

$\hat{\lambda}_1$, is closer to the ratio $\max[|\partial_i E_0(\boldsymbol{W}_{es})|]/\max[|\partial_i R(\boldsymbol{W}_{es})|]$. It follows from the Cauchy-Schwartz inequality that

$$\hat{\lambda}_1 \geq \hat{\lambda}_2. \tag{19}$$

For the specific case of weight decay, where $R(\boldsymbol{W}) = \|\boldsymbol{W}\|^2$, expressions (15) and (18) become

$$\hat{\lambda}_1 = \frac{\|\nabla E_0(\boldsymbol{W}_{es})\|}{2\|\boldsymbol{W}_{es}\|}, \tag{20}$$

$$\hat{\lambda}_2 = \max\left[0, \frac{-\nabla E_0(\boldsymbol{W}_{es}) \cdot \boldsymbol{W}_{es}}{2\|\boldsymbol{W}_{es}\|^2}\right]. \tag{21}$$

These estimates are sensitive to the particularities of the training and validation data sets used, and possibly also to the training algorithm. One must therefore average them over different validation and training sets. It is, however, still quicker to do this than to do a search since early stopping training often is several orders of magnitude faster to do than a full minimization of (7).

One way to view the estimates (15) and (18) is as the weight decay parameters that correspond to the early stopping point. However, our aim here is not to imitate early stopping with weight decay, but to use early stopping to estimate the weight decay parameter λ. We hope that using weight decay with this λ value will actually result in better out-of-sample performance than what we get from doing early stopping (the whole exercise becomes rather meaningless if this is not the case).

As a sidenote, we imagine that (15) and (18) could be used also to estimate weight decay parameters in cases when different weight decays are used for weights in different layers. This would then be done by considering these estimates for different groups of weights.

4 Experiments

4.1 Data Sets

We here demonstrate the performance of our algorithm on a set of five regression problems. For each problem, we vary either the number of inputs, the number of hidden units, or the amount of training data to study the effects of the numbers of parameters relative to the number of training data points. The five problems are:

Synthetic bilinear problem. The task is to model a bilinear function of the form

$$\phi(x_1, x_2) = x_1 x_2. \tag{22}$$

We use three different sizes of training data sets, $M \in \{20, 40, 100\}$, but a constant validation set size of 10 patterns. The validation patterns are in addition to the M training patterns. The test error, or generalization error, is computed

by numerical integration over 201×201 data points on a two-dimensional lattice $(x_1, x_2) \in [-1, 1]^2$. The target values (but not the inputs) are contaminated with three different levels of Gaussian noise with standard deviation $\sigma \in \{0.1, 0.2, 0.5\}$. This gives a total of $3 \times 3 = 9$ different experiments on this particular problem, which we refer to as setup A1, A2, ..., and A9 below.

This allows controlled studies w.r.t. noise levels and training set sizes, while keeping the network architecture constant (2 inputs, 8 tanh hidden, and one linear output).

Predicting Puget Sound Power and Light Co. power load between 7 and 8 a.m. the following day. This data set is taken from the Puget Sound Power and Light Co's power prediction competition [3]. The winner of this competition used a set of linear models, one for each hour of the day. We have selected the subproblem of predicting the load between 7 and 8 a.m. 24 hrs. in advance. This hour shows the largest variation in power load. The training set consists of 844 weekdays between January 1985 and September 1990. Of these, 150 days are randomly selected and used for validation. We use 115 winter weekdays, from between November 1990 and March 1992, for out-of-sample testing. The inputs are things like current load, average load during the last 24 hours, average load during the last week, time of the year, etc., giving a total of 15 inputs. Three different numbers of internal units are tried on this task: 15, 10, and 5, and we refer to these experiments as B1, B2, and B3 below.

Predicting daily riverflow in two Icelandic rivers. This problem is tabulated in [22], and the task is to model tomorrow's average flow of water in one of two Icelandic rivers, knowing today's and previous days' waterflow, temperature, and precipitation. The training set consists of 731 data points, corresponding to the years 1972 and 1973, out of which we randomly sample 150 datapoints for validation. The test set has 365 data points (the year 1974). We use two different lengths of lags, 8 or 4 days back, which correspond to 24 or 12 inputs, while the number of internal units is kept constant at 12. These experiments are referred to as C1, C2, C3, and C4 below.

Predicting the Wolf sunspots time series. This time series has been used several times in the context of demonstrating new regularization techniques, for instance by [15] and [26]. We try three different network architectures on this problem, always keeping 12 input units but using 4, 8, or 12 internal units in the network. These experiments are referred to as setup D1, D2, and D3 below. The training set size is kept constant at $M = 221$ (years 1700-1920), out of which we randomly pick 22 patterns for validation. We test our models under four different conditions: Single step prediction on "test set 1" with 35 data points (years 1921-1955), 4-step iterated prediction on "test set 1", 8-step iterated prediction on all 74 available test years (1921-1994), and 11-step iterated prediction on all available test years. These test conditions are coded as s1, m4, m8, and m11.

Estimating the peak pressure position in a combustion engine. This is a data set with 4 input variables (ignition time, engine load, engine speed, and air/fuel ratio) and only 49 training data points, out of which we randomly pick 9 patterns for validation. The test set consists of 35 data points, which have been measured under slightly different conditions than the training data. We try four different numbers of internal units on this task: 2, 4, 8, or 12, and refer to these experiments as E1, E2, E3, and E4.

4.2 Experimental Procedure

The experimental procedure is the same for all problems: We begin by estimating λ in the "traditional" way by searching over the region $\log\lambda \in [-6.5, 1.0]$ in steps of $\Delta\log\lambda = 0.5$. For each λ value, we train 10 networks using the Rprop training algorithm [3] [18]. Each network is trained until the total error (7) is minimized, measured by

$$\log\left[\frac{1}{100}\sum_{i=1}^{100}\frac{|\Delta E_i|}{\|\Delta \boldsymbol{W}_i\|}\right] < -5, \tag{23}$$

where the sum runs over the most recent 100 epochs, or until 10^5 epochs have passed, whichever occurs first. The convergence criterion (23) is usually fulfilled within 10^5 epochs. New validation and training sets are sampled for each of the 10 networks, but the different validation sets are allowed to overlap. Means and standard deviations, $\log E_{nCV,k}$ and $\sigma_{nCV,k}$, for the errors are estimated from these 10 network runs, assuming a lognormal distribution for the validation errors. Figure 2 shows an example of such a search for the Wolf sunspot problem, using a neural network with 12 inputs, 8 internal units, and 1 linear output.

Using the objective Monte Carlo method described above, we estimate an optimal $\hat{\lambda}_{opt}$ value from this search. This value is then used to train 10 new networks with all the training data (no validation set). The test errors for these networks are then computed using the held out test set.

A total of $16 \times 10 = 160$ network runs are thus done to select the $\hat{\lambda}_{opt}$ for each experiment. This corresponds to a few days' or a week's work, depending on available hardware and the size of the problem. Although this is in excess of what is really needed in practice (one could get away with about half as many runs in a real application) the time spent doing this is aggravating. The times needed for doing the searches described in this paper ranged from 10 up to 400 cpu-hours, depending on the problem and the computer[4]. For comparison, the early stopping experiments described below took between 10 cpu-minutes and 14 cpu-hours. There was typically a ratio of 40 between the time needed for a search and the time needed for an early stopping estimate.

[3] Initial tests showed that the Rprop algorithm was considerably more efficient and robust than e.g. backprop or conjugate gradients in minimizing the error. We did not, however, try true second order algorithms like Levenberg-Marquardt or Quasi-Newton.

[4] A variety of computers were used for the simulations, including NeXT, Sun Sparc, DEC Alpha, and Pentium computers running Solaris.

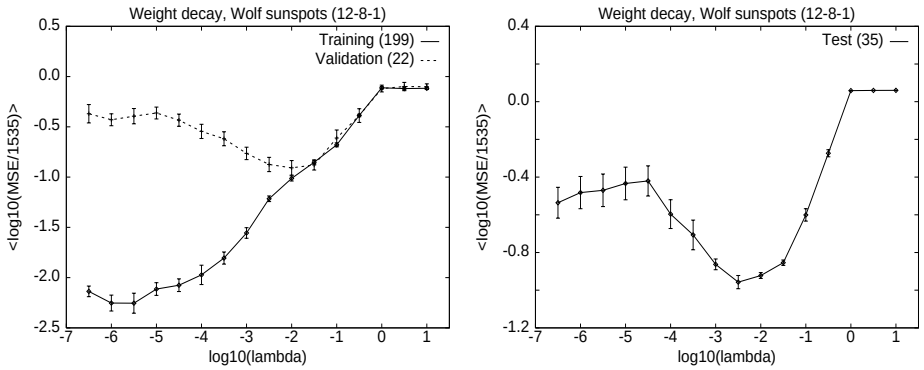

Fig. 2. Left panel: Training and validation errors on the Wolf sunspot time series, setup D2, plotted versus the weight decay parameter λ. Each point corresponds to an average over 10 runs with different validation and training sets. The error bars mark 95% confidence limits for the average validation and training errors, under the assumption that the errors are lognormally distributed. The objective Monte Carlo method gives $\log \hat{\lambda}_{opt} = -2.00 \pm 0.31$. Right panel: The corresponding plot for the test error on the sunpots "test set 1". The network architecture is 12 inputs, 8 tanh internal units, and 1 linear output.

We then estimate $\hat{\lambda}_1$ and $\hat{\lambda}_2$, by training 100 networks with early stopping. One problem here is that the stopping point is ill-defined, i.e. the first observed minimum in the validation error is not necessarily the minimum where one should stop. The validation error quite often decreases again beyond this point. To avoid such problems, we keep a record of the weights corresponding to the latest minimum validation error and continue training beyond that point. The training is stopped when as many epochs have passed as it took to find the validation error minimum without encountering a new minimum. The weights corresponding to the last validation error minimum are then used as the early stopping weights. For example, if the validation error has a minimum at say 250 epochs, we then wait until a total of 500 epochs have passed before deciding on that particular stopping point. From the 100 networks, we get 100 estimates for $\hat{\lambda}_1$ and $\hat{\lambda}_2$. We take the logarithm of these and compute means $\langle \log \hat{\lambda}_1 \rangle$ and $\langle \log \hat{\lambda}_2 \rangle$, and corresponding standard deviations. The resulting arithmetic mean values are taken as the estimates for λ and the standard deviations are used as measures of the estimation error. The arithmetic means are then used to train 10 networks which use all the training data. Figure 3 shows the histograms corresponding to the problem presented in Figure 2.

When comparing test errors achieved with different methods, we use the Wilcoxon rank test [13], also called the Mann-Whitney test, and report differences at 95% confidence level.

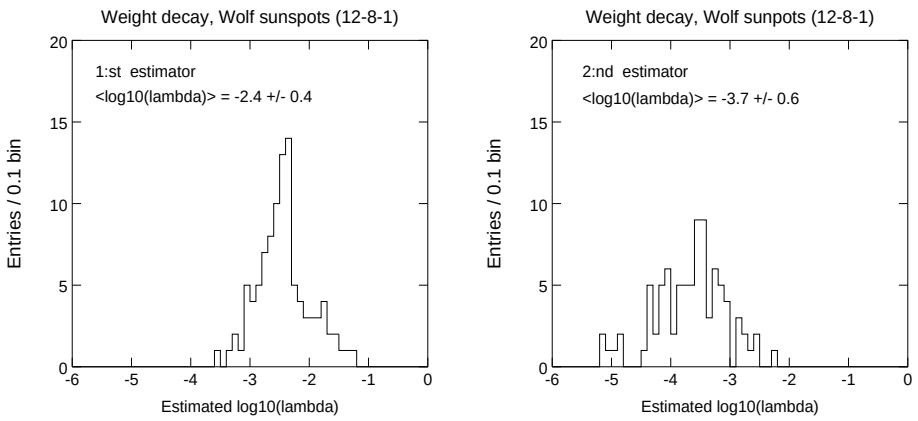

Fig. 3. Left panel: Histogram showing the estimated values $\hat{\lambda}_1$ for 100 different training runs, using different training and validation sets each time. Right panel: Similar histogram for $\hat{\lambda}_2$. The problem (D2) is the same as that depicted in Figure 2.

4.3 Quality of the λ Estimates

As a first test of the quality of the estimates $\hat{\lambda}_1$ and $\hat{\lambda}_2$, we check how well they agree with the $\hat{\lambda}_{opt}$ estimate, which can be considered a "truth". The estimates for all the problem setups are tabulated in table 1 and plotted in Figure 4.

The linear correlation between $\log \hat{\lambda}_1$ and $\log \hat{\lambda}_{opt}$ is 0.71, which is more than three standard deviations larger than the expected correlation between 23 random points. Furthermore, a linear regression with intercept gives the result

$$\hat{\lambda}_{opt} = 0.30 + 1.13\hat{\lambda}_1. \tag{24}$$

Thus, $\hat{\lambda}_1$ is a fairly good estimator of $\hat{\lambda}_{opt}$.

The linear correlation between $\hat{\lambda}_2$ and $\hat{\lambda}_{opt}$ is 0.48, more than two standard deviations from the random correlation. A linear regression gives

$$\hat{\lambda}_{opt} = -0.66 + 0.57\hat{\lambda}_2, \tag{25}$$

and the second estimator $\hat{\lambda}_2$ is clearly a less good estimator of $\hat{\lambda}_{opt}$.

We next compare the out-of-sample performances of these different λ estimates, which is what really matters to the practitioner. Table 2 lists the differences in out-of-sample performance when using the early stopping estimates or the search estimate. A "+" means that using the early stop estimate results in significantly (95% significance level) lower test error than if $\hat{\lambda}_{opt}$ is used. Similarly, a "–" means that the search estimate gives significantly lower test error than the early stopping estimates. A "0" means there is no significant difference. The conclusion from Table 2 is that $\hat{\lambda}_2$ is significantly worse than $\hat{\lambda}_{opt}$, but that there is no consistent difference between $\hat{\lambda}_1$ and $\hat{\lambda}_{opt}$. The two estimates are essentially equal, in terms of test error. In some cases, like the power prediction

Problem	$\log \hat{\lambda}_{opt}$	$\log \hat{\lambda}_1$	$\log \hat{\lambda}_2$
A1 ($M = 20, \sigma = 0.1$)	-2.82 ± 0.04	-2.71 ± 0.66	-3.44 ± 1.14
A2 ($M = 20, \sigma = 0.2$)	-2.67 ± 0.42	-2.32 ± 0.58	-3.20 ± 0.96
A3 ($M = 20, \sigma = 0.5$)	-0.49 ± 1.01	-1.93 ± 0.78	-3.14 ± 1.15
A4 ($M = 40, \sigma = 0.1$)	-2.93 ± 0.49	-2.85 ± 0.73	-3.56 ± 0.87
A5 ($M = 40, \sigma = 0.2$)	-2.53 ± 0.34	-2.41 ± 0.64	-2.91 ± 0.68
A6 ($M = 40, \sigma = 0.5$)	-2.43 ± 0.44	-2.13 ± 0.74	-2.85 ± 0.77
A7 ($M = 100, \sigma = 0.1$)	-3.45 ± 0.78	-3.01 ± 0.86	-3.74 ± 0.93
A8 ($M = 100, \sigma = 0.2$)	-3.34 ± 0.71	-2.70 ± 0.73	-3.33 ± 0.92
A9 ($M = 100, \sigma = 0.5$)	-3.31 ± 0.82	-2.34 ± 0.63	-3.13 ± 1.06
B1 (Power, 15 hidden)	-3.05 ± 0.21	-3.82 ± 0.42	-5.20 ± 0.70
B2 (Power, 10 hidden)	-3.57 ± 0.35	-3.75 ± 0.45	-4.93 ± 0.50
B3 (Power, 5 hidden)	-4.35 ± 0.66	-3.78 ± 0.52	-5.03 ± 0.74
C1 (Jökulsá Eystra, 8 lags)	-2.50 ± 0.10	-3.10 ± 0.33	-4.57 ± 0.59
C2 (Jökulsá Eystra, 4 lags)	-2.53 ± 0.12	-3.15 ± 0.40	-4.20 ± 0.59
C3 (Vatnsdalsá, 8 lags)	-2.48 ± 0.11	-2.65 ± 0.40	-3.92 ± 0.56
C4 (Vatnsdalsá, 4 lags)	-2.39 ± 0.55	-2.67 ± 0.45	-3.70 ± 0.62
D1 (Sunspots, 12 hidden)	-2.48 ± 0.12	-2.48 ± 0.50	-3.70 ± 0.42
D2 (Sunspots, 8 hidden)	-2.00 ± 0.31	-2.43 ± 0.45	-3.66 ± 0.60
D3 (Sunspots, 4 hidden)	-2.51 ± 0.44	-2.39 ± 0.48	-3.54 ± 0.65
E1 (Pressure, 12 hidden)	-3.13 ± 0.43	-3.03 ± 0.70	-4.69 ± 0.91
E2 (Pressure, 8 hidden)	-3.01 ± 0.52	-3.02 ± 0.64	-4.72 ± 0.82
E3 (Pressure, 4 hidden)	-3.83 ± 0.80	-3.07 ± 0.71	-4.50 ± 1.24
E4 (Pressure, 2 hidden)	-4.65 ± 0.78	-3.46 ± 1.34	-4.21 ± 1.40

Table 1. Estimates of λ for the 23 different problem setups. Code A corresponds to the synthetic problem, code B to the Power prediction, code C to the riverflow prediction, code D to the Sunspots series, and code E to the maximum pressure position problem. For the $\log \hat{\lambda}_{opt}$ column, errors are the standard deviations of the Monte Carlo estimate. For the early stopping estimates, errors are the standard deviations of the estimates.

problem, it would have been beneficial to do a small search around the early stop estimate to check for a possibly better value.

The test errors for the combustion engine (setups E) are not included in Tables 2 (and 3) because the test set is too different from the training set to provide relevant results. In fact, no regularized network is significantly better than an unregularized network on this problem.

4.4 Weight Decay versus Early Stopping Committees

Having trained all these early stopping networks, it is reasonable to ask if using them to estimate λ for a weight decay network is the optimal use of these networks? Another possible use is, for instance, to construct a committee [16] from them.

To test this, we compare the test errors for our regularized networks with

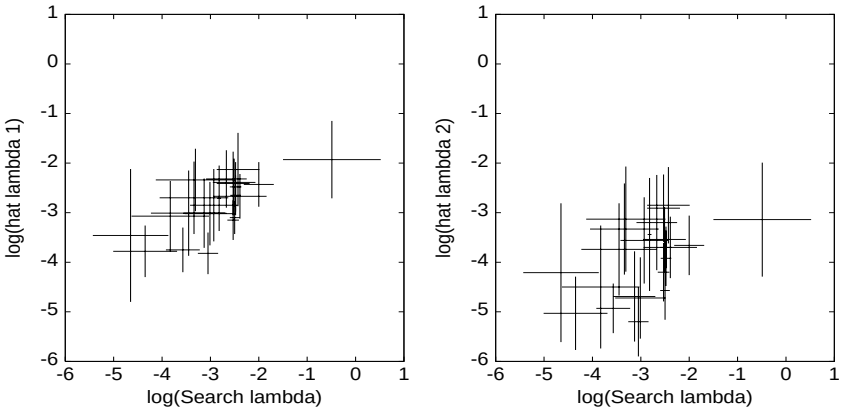

Fig. 4. Plot of the results in Table 1. Left plate: The $\hat{\lambda}_1$ estimate plotted versus $\hat{\lambda}_{opt}$. The linear correlation between $\log \hat{\lambda}_1$ and $\log \hat{\lambda}_{opt}$ is 0.71. Right plate: $\hat{\lambda}_2$ plotted versus $\hat{\lambda}_{opt}$. The linear correlation between $\log \hat{\lambda}_2$ and $\log \hat{\lambda}_{opt}$ is 0.48. The sizes of the crosses correspond to the error bars in Table 1.

those when using a committee of 10 networks trained with early stopping. The results are listed in Table 3.

Some observations from Table 3, bearing in mind that the set of problems is small, are: Early stopping committees seem like the better option when the problem is very noisy (setups A3, A6, and A9), and when the network does not have very many degrees of freedom (setups B3, C4, and D3). Weight decay networks, on the other hand, seem to work better than committees on problems with many degrees of freedom (setups B1 and C3), problems with low noise levels and much data (setup A7), and problems where the prediction is iterated through the network (m4, m8, and m11 setups). We emphasize, however, that these conclusions are drawn from a limited set of problems and that all problems tend to have their own set of weird characteristics.

We also check which model works best on each problem. On the power prediction, the best overall model is a large network (B1) which is trained with weight decay. On the river prediction problems, the best models are small (C2 and C4)) and trained with either weight decay (Jökulsá Eystra) or early stopping and then combined into committees (Vatnsdalsá). On the sunspot problem, the best overall model is a large network (D1) trained with weight decay.

These networks are competitive with previous results on the same data sets. The performance of the power load B1 weight decay networks, using $\hat{\lambda}_{opt}$, are significantly better than what a human expert produces, and also significantly better than the results by the winner of the Puget Sound Power and Light Co. Power Load Competition [7], although the difference is small. The test results are summarized in Figure 5. The performance of the sunspot D1 weight decay

Table 2. Relative performance of single networks trained using the estimates $\hat{\lambda}_1$ and $\hat{\lambda}_2$, for the weight decay parameter, and the performance of single networks trained using the search estimate $\hat{\lambda}_{opt}$. The relative performances are reported as: "+" means that using $\hat{\lambda}_i$ results in a test error which is significantly lower than what the search estimate $\hat{\lambda}_{opt}$ gives, "0" means that the performances are equivalent, and "–" means that using $\hat{\lambda}_{opt}$ results in a lower test error than when using $\hat{\lambda}_i$. All results are reported for a 95% confidence level when using the Wilcoxon test. See the text on why the E results are left out.

Problem Setup	$\hat{\lambda}_1$ vs. $\hat{\lambda}_{opt}$	$\hat{\lambda}_2$ vs. $\hat{\lambda}_{opt}$
A1 ($M = 20, \sigma = 0.1$)	0	0
A2 ($M = 20, \sigma = 0.2$)	0	–
A3 ($M = 20, \sigma = 0.5$)	0	0
A4 ($M = 40, \sigma = 0.1$)	0	0
A5 ($M = 40, \sigma = 0.2$)	0	–
A6 ($M = 40, \sigma = 0.5$)	0	0
A7 ($M = 100, \sigma = 0.1$)	–	0
A8 ($M = 100, \sigma = 0.2$)	0	0
A9 ($M = 100, \sigma = 0.5$)	+	0
B1 (Power, 15 hidden)	–	–
B2 (Power, 10 hidden)	–	–
B3 (Power, 5 hidden)	+	–
C1 (Jökulsá Eystra, 8 lags)	–	–
C2 (Jökulsá Eystra, 4 lags)	0	–
C3 (Vatnsdalsá, 8 lags)	–	–
C4 (Vatnsdalsá, 4 lags)	0	–
D1.s1 (Sunspots, 12 hidden)	0	–
D2.s1 (Sunspots, 8 hidden)	+	–
D3.s1 (Sunspots, 4 hidden)	0	–
D1.m4 (Sunspots, 12 hidden)	0	–
D2.m4 (Sunspots, 8 hidden)	+	–
D3.m4 (Sunspots, 4 hidden)	+	–
D1.m8 (Sunspots, 12 hidden)	0	–
D2.m8 (Sunspots, 8 hidden)	–	–
D3.m8 (Sunspots, 4 hidden)	+	–
D1.m11 (Sunspots, 12 hidden)	0	0
D2.m11 (Sunspots, 8 hidden)	+	–
D3.m11 (Sunspots, 4 hidden)	0	–

network is comparable with the network by Weigend et al., listed in [26]. Figure 6 shows the performance of the D1 network trained with weight decay, $\lambda = \hat{\lambda}_1$, and compares it to the results by Weigend et al. [26]. The weight decay network produces these results using a considerably simpler λ selection method and regularization cost than the one presented in [26].

From these anectodal results, one could be bold and say that weight decay

Table 3. Relative performance of single networks trained using weight decay and early stopping committees with 10 members. The relative performance of weight decay (WD) and 10 member early stopping committees are reported as: "+" means that weight decay is significantly better than committees, "0" means that weight decay and committees are equivalent, and "–" means that committees are better than weight decay. All results are reported for a 95% confidence level when using the Wilcoxon test. See the text on why the E results are left out.

Problem Setup	WD($\hat{\lambda}_{opt}$) vs. Comm.	WD($\hat{\lambda}_1$) vs. Comm.
A1 ($M = 20, \sigma = 0.1$)	0	+
A2 ($M = 20, \sigma = 0.2$)	0	0
A3 ($M = 20, \sigma = 0.5$)	–	–
A4 ($M = 40, \sigma = 0.1$)	0	0
A5 ($M = 40, \sigma = 0.2$)	+	+
A6 ($M = 40, \sigma = 0.5$)	–	–
A7 ($M = 100, \sigma = 0.1$)	+	+
A8 ($M = 100, \sigma = 0.2$)	0	0
A9 ($M = 100, \sigma = 0.5$)	–	0
B1 (Power, 15 hidden)	+	–
B2 (Power, 10 hidden)	0	–
B3 (Power, 5 hidden)	–	–
C1 (Jökulsá Eystra, 8 lags)	+	0
C2 (Jökulsá Eystra, 4 lags)	+	0
C3 (Vatnsdalsá, 8 lags)	+	+
C4 (Vatnsdalsá, 4 lags)	–	–
D1.s1 (Sunspots, 12 hidden)	0	0
D2.s1 (Sunspots, 8 hidden)	–	0
D3.s1 (Sunspots, 4 hidden)	–	–
D1.m4 (Sunspots, 12 hidden)	+	+
D2.m4 (Sunspots, 8 hidden)	+	+
D3.m4 (Sunspots, 4 hidden)	0	+
D1.m8 (Sunspots, 12 hidden)	+	+
D2.m8 (Sunspots, 8 hidden)	+	0
D3.m8 (Sunspots, 4 hidden)	0	0
D1.m11 (Sunspots, 12 hidden)	+	+
D2.m11 (Sunspots, 8 hidden)	0	+
D3.m11 (Sunspots, 4 hidden)	+	+

shows a slight edge over early stopping committees. However, it is fair to say that it is a good idea to try both committees and weight decay when constructing predictor models.

It is emphasized that these results are from a small set of problems, but that these problems (except perhaps for the synthetic data) are all realistic in the sense that the datasets are small and noisy.

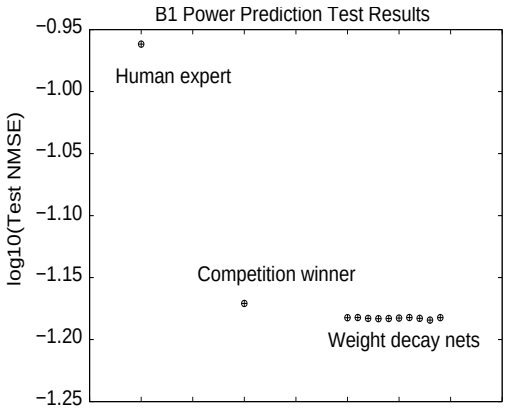

Fig. 5. The performance of the 10 neural networks with 15 inputs, 15 hidden units, and one output unit, trained with weight decay using $\lambda = \hat{\lambda}_{opt}$, on the power prediction problem. "Human expert" denotes the prediction result by the human expert at Puget Sound Power and Light Co., and "Competition winner" denotes the result by the model that won the Puget Sound Power and Light Co.'s Power Prediction Competition.

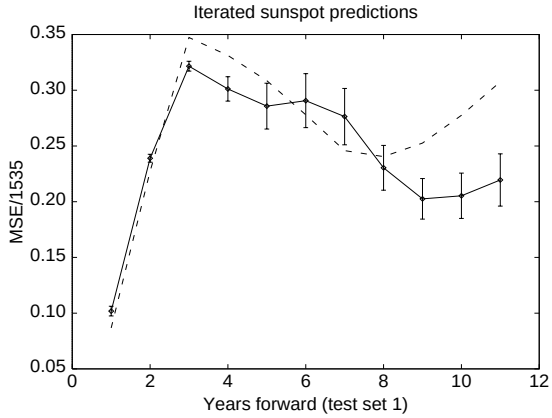

Fig. 6. The performance of a neural network with 12 inputs, 12 hidden units, and one output unit, trained with weight decay using $\lambda = \hat{\lambda}_1$, on iterated predictions for the sunspot problem. The error bars denote one standard deviation for the 10 trained networks. The dashed line shows the results when using the network listed in [26]. Note that these results are achieved with a simple weight decay cost and a very simple method for selecting λ, whereas [26] use weight elimination and a complicated heuristic scheme for setting λ.

5 Conclusions

The established connection between early stopping and weight decay regularization naturally leads to the idea of using early stopping to estimate the weight decay parameter. In this paper we have shown how this can be done and that the resulting λ results in as low test errors as achieved with the standard cross-validation method, although this varies between problems. In practical applications, this means replacing a search which may take days or weeks, with a computation that usually does not require more than a few minutes or hours. This value can also be used as a starting point for a more extensive cross-validation search.

We have also shown that using several early stopping networks to estimate λ can be smarter than combining the networks into committees. The conclusion from this is that although there is a correspondence between early stopping and weight decay under asymptotic conditions this does not mean that early stopping and weight decay give equivalent results in real life situations.

The method unfortunately only works for regularization terms that have a connection with early stopping, like quadratic weight decay or "weight decay like" regularizers where the weights are constrained towards the origin in weight space (but using e.g. a Laplacian prior instead of the usual Gaussian prior). The method does not carry over to regularizers which do not have any connection to early stopping (like e.g. Tikhonov smoothing regularizers).

Acknowledgements: David B. Rosen is thanked for a very inspiring dinner conversation during the 1996 "Machines that Learn" Workshop in Snowbird, Utah. Milan Casey Brace of Puget Sound Power and Light Co. is thanked for supplying the power load data. Financial support is gratefully acknowledged from NSF (grant CDA-9503968), Olle and Edla Ericsson's Foundation, the Swedish Institute, and the Swedish Research Council for Engineering Sciences (grant TFR-282-95-847).

References

1. Y. S. Abu-Mustafa. Hints. *Neural Computation*, 7:639–671, 1995.
2. C. M. Bishop. Curvature-driven smoothing: A learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 4(5):882–884, 1993.
3. M. C. Brace, J. Schmidt, and M. Hadlin. Comparison of the forecast accuracy of neural networks with other established techniques. In *Proceedings of the First International Form on Application of Neural Networks to Power System, Seattle WA.*, pages 31–35, 1991.
4. W. L. Buntine and A. S. Weigend. Bayesian back-propagation. *Complex Systems*, 5:603–643, 1991.
5. P. Cheeseman. On Bayesian model selection. In *The Mathematics of Generalization - The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning*, pages 315–330, Reading, MA, 1995. Addison-Wesley.

6. G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:304–314, 1989.
7. R. Engle, F. Clive, W. J. Granger, R. Ramanathan, F. Vahid, and M. Werner. Construction of the puget sound forecasting model. EPRI Project # RP2919, Quantitative Economics Research Institute, San Diego, CA., 1991.
8. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
9. F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.
10. L. K. Hansen, C. E. Rasmussen, C. Svarer, and J. Larsen. Adaptive regularization. In J. Vlontzos, J.-N. Hwang, and E. Wilson, editors, *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV*, pages 78–87, Piscataway, NJ, 1994. IEEE Press.
11. A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation of nonorthogonal problems. *Technometrics*, 12:55–67, Feb. 1970.
12. M. Ishikawa. A structural learning algorithm with forgetting of link weights. Technical Report TR-90-7, Electrotechnical Laboratory, Information Science Division, 1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan, 1990.
13. M. G. Kendall and A. Stuart. *The Advanced Theory of Statistics*. Hafner Publishing Co., New York, third edition, 1972.
14. J. E. Moody and T. S. Rögnvaldsson. Smoothing regularizers for projective basis function networks. In *Advances in Neural Information Processing Systems 9*, Cambridge, MA, 1997. MIT Press.
15. S. Nowlan and G. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493, 1992.
16. M. P. Perrone and L. C. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *Artificial Neural Networks for Speech and Vision*, pages 126–142, London, 1993. Chapman & Hall.
17. D. Plaut, S. Nowlan, and G. Hinton. Experiments on learning by backpropagation. Technical Report CMU-CS-86-126, Carnegie Mellon University, Pittsburg, PA, 1986.
18. M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In H. Ruspini, editor, *Proc. of the IEEE Intl. Conference on Neural Networks*, pages 586–591, San Fransisco, California, 1993.
19. B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.
20. J. Sjöberg and L. Ljung. Overtraining, regularization, and searching for minimum with application to neural nets. *Int. J. Control*, 62(6):1391–1407, 1995.
21. A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed problems*. V. H. Winston & Sons, Washington D.C., 1977.
22. H. Tong. *Non-linear Time Series: A Dynamical System Approach*. Clarendon Press, Oxford, 1990.
23. J. Utans and J. E. Moody. Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction. In *Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street*. IEEE Computer Society Press, Los Alamitos, CA, 1991.
24. G. Wahba, C. Gu, Y. Wang, and R. Chappell. Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance. In *The Mathematics of Generalization - The Proceedings of the SFI/CNLS Workshop*

on Formal Approaches to Supervised Learning, pages 331–359, Reading, MA, 1995. Addison-Wesley.
25. G. Wahba and S. Wold. A completely automatic french curve. *Communications in Statistical Theory & Methods*, 4:1–17, 1975.
26. A. Weigend, D. Rumelhart, and B. Hubermann. Back-propagation, weight-elimination and time series prediction. In T. Sejnowski, G. Hinton, and D. Touretzky, editors, *Proc. of the Connectionist Models Summer School*, San Mateo, California, 1990. Morgan Kaufmann Publishers.
27. P. M. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7:117–143, 1995.