FLAT MINIMUM SEARCH FINDS SIMPLE NETS
Technical Report FKI-200-94

Sepp Hochreiter
hochreit@informatik.tu-muenchen.de

Jürgen Schmidhuber
schmidhu@informatik.tu-muenchen.de

Fakultät für Informatik, H2
Technische Universität München
80290 München, Germany

December 31, 1994

Abstract

We present a new algorithm for finding low complexity neural networks with high generalization capability. The algorithm searches for a "flat" minimum of the error function. A flat minimum is a large connected region in weight-space where the error remains approximately constant. An MDL-based argument shows that flat minima correspond to low expected overfitting. Although our algorithm requires the computation of second order derivatives, it has backprop's order of complexity. Automatically, it effectively prunes units, weights, and input lines. Various experiments with feedforward and recurrent nets are described. In an application to stock market prediction, flat minimum search outperforms (1) conventional backprop, (2) weight decay, (3) "optimal brain surgeon" / "optimal brain damage".

1 INTRODUCTION

Our algorithm finds a large region in weight space with the property that each weight vector in that region leads to *similar* small error. Such a region is called a "flat minimum" [15]. To get an intuitive feeling for why a flat minimum is interesting, consider this (see also [50]): a "sharp" minimum (see figure 2) corresponds to weights which have to be specified with high precision. A flat minimum (see figure 1) corresponds to weights many of which can be given with low precision. In the terminology of the theory of minimum description length (MDL [45, 39]), fewer bits of information are required to describe a flat minimum (corresponding to a "simple" or low complexity-network). The MDL principle suggests that low network complexity corresponds to high generalization performance.

Unlike e.g. Hinton and van Camp's method [14], our algorithm does not depend on the choice of a "good" prior. It finds a "flat" minimum by searching for weights that minimize both training error and weight precision. This requires the computation of the Hessian. However, by using Pearlmutter's and Møller's efficient second order method [35, 26], we obtain conventional backprop's order of computational complexity. *Automatically, the method effectively reduces numbers of units, weights, and input lines, as well as output sensitivity with respect to remaining weights and units. Unlike e.g. simple weight decay, the method treats/prunes units and weights in different layers in different reasonable ways.*

Outline. Section 2 introduces basic concepts. Section 3 describes the novel algorithm. Excellent experimental generalization results are reported in section 4. Section 5 mentions relations to previous work. A detailed theoretical justification of our approach is presented in the appendix.

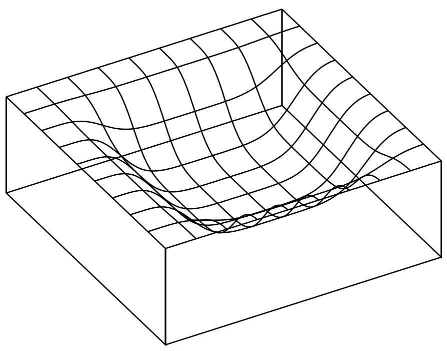

Figure 1: *Example of a "flat" minimum.*

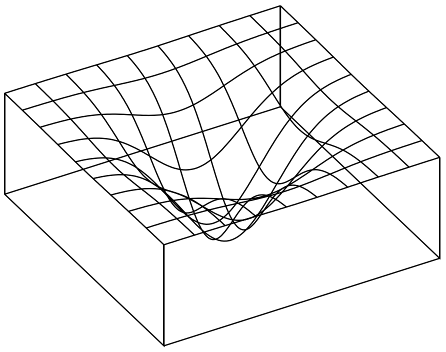

Figure 2: *Example of a "sharp" minimum.*

2 TASK / ARCHITECTURE / BOXES

Generalization task. The task is to approximate an unknown relation $\bar{D} \subset X \times Z$ between a set of possible inputs $X \subset R^N$ and a set of possible outputs $Z \subset R^K$. $\bar{D}$ is taken to be a function. A relation D is obtained from $\bar{D}$ by adding noise to the outputs (see appendix A.1). All training information is given by a finite relation $D_0 \subset D$. D_0 is called the *training set*. The pth element of D_0 is denoted by an input/target pair (x_p, d_p).

Architecture. For simplicity, we will focus on a standard feedforward net (but in the experiments, we will use recurrent nets as well). The net has N input units, K output units, W weights, and differentiable activation functions. It maps input vectors $x_p \in R^N$ to output vectors $o_p \in R^K$. The weight from unit j to i is denoted by w_{ij}. The W-dimensional weight vector is denoted by w.

Training error. We use mean squared error $E_q(w, D_0) := \frac{1}{|D_0|} \sum_{(x_p,d_p) \in D_0} \parallel d_p - o_p \parallel^2$, where $\parallel . \parallel$ denotes the Euclidian norm, and $|.|$ denotes the cardinality of a set.

Tolerable error. To define a region in weight space with the property that each weight vector from that region has "*similar* small error", we introduce the tolerable error E_{tol}, a positive constant. "Small" error is defined as being smaller than E_{tol}. $E_q(w, D_0) > E_{tol}$ implies "underfitting".

Boxes. Each weight w satisfying $E_q(w, D_0) \leq E_{tol}$ defines an "acceptable minimum". **We are interested in a large region of connected acceptable minima. Such a region is called a flat minimum.** They are associated with low expected generalization error (see appendix A.2). To simplify the algorithm for finding a large connected region (see below), we do not consider maximal connected regions but focus on so-called *"boxes"* within regions: for each acceptable minimum w, its *box* M_w in weight space is a W-dimensional hypercuboid with center w. For simplicity, each edge of the box is taken to be parallel to one weight axis. Half the length of the box edge in direction of the axis corresponding to weight w_{ij} is denoted by Δw_{ij}, which is the maximal (positive) value such that for all pairs (i, j), all positive $\kappa_{ij} \leq \Delta w_{ij}$ can be added to or subtracted from the corresponding w_{ij} simultaneously without violating $E_q(., D_0) \leq E_{tol}$ (Δw_{ij} gives the precision of w_{ij}). M_w's *box volume* is defined by $V(\Delta w) := 2^W \prod_{i,j} \Delta w_{ij}$, where Δw denotes the vector with components Δw_{ij}.

3 THE ALGORITHM

Starting with a random initial weight vector, flat minimum search (FMS) finds a w defining a box M_w with maximal box volume $V(\Delta w)$ and minimal $\tilde{B}(w, D_0) := -\log(\frac{1}{2^W} V(\Delta w)) = \sum_{i,j} -\log \Delta w_{ij}$. Note the relationship to MDL ($\tilde{B}$ is the number of bits required to describe the weights, see appendix A.3).

In A.3, we derive the following algorithm. We use gradient descent to minimize $E(w, D_0) = E_q(w, D_0) + \lambda B(w, D_0)$, where

$$B(w, D_0) = \frac{1}{2}\left(-W \log \epsilon + \sum_{i,j} \log \sum_k \left(\frac{\partial o^k}{\partial w_{ij}}\right)^2 + W \log \sum_k \left(\sum_{i,j} \frac{\left|\frac{\partial o^k}{\partial w_{ij}}\right|}{\sqrt{\sum_k \left(\frac{\partial o^k}{\partial w_{ij}}\right)^2}}\right)^2\right) . \qquad (1)$$

Here o^k is the activation of the kth output unit, ϵ is a constant, and λ is a positive variable ensuring either $E_q(w, D_0) \leq E_{tol}$ or an expected decrease of $E_q(., D_0)$ during learning (see [47] for adjusting λ). To minimize $B(w, D_0)$, we have to compute

$$\frac{\partial B(w, D_0)}{\partial w_{uv}} = \sum_{k,i,j} \frac{\partial B(w, D_0)}{\partial \left(\frac{\partial o^k}{\partial w_{ij}}\right)} \frac{\partial^2 o^k}{\partial w_{ij} \partial w_{uv}} \text{ for all } u, v . \qquad (2)$$

See details in appendix A.5.1. It can be shown that by using Pearlmutter's and Møller's efficient second order method [35, 26], the gradient of $B(w, D_0)$ can be computed in $O(W)$ time (see details in A.5). **Therefore, our algorithm has the same order of computational complexity as standard backprop.**

4 EXPERIMENTAL RESULTS

4.1 EXPERIMENT 1 – noisy classification.

The first experiment is taken from Pearlmutter and Rosenfeld [36]. The task is to decide whether the x-coordinate of a point in 2-dimensional space exceeds zero (class 1) or doesn't (class 2). Noisy training examples are generated as follows: data points are obtained from a Gaussian with zero mean and stdev 1.0, bounded in the interval $[-3.0, 3.0]$. The data points are misclassified with probability 0.05. Final input data is obtained by adding a zero mean Gaussian with stdev 0.15 to the data points. In a test with 2,000,000 data points, it was found that the procedure above leads to 9.27 per cent misclassified data. *No* method will misclassify less than 9.27 per cent, due to the inherent noise in the data. The training set is based on 200 fixed data points (see figure 3). The test set is based on 120,000 data points.

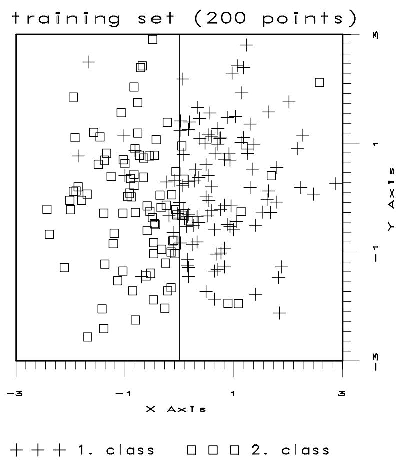

Figure 3: *The 200 input examples of the training set. Crosses represent data points from class 1. Squares represent data points from class 0.*

	Backprop		FMS			Backprop		FMS	
	MSE	dto	MSE	dto		MSE	dto	MSE	dto
1	0.220	1.35	0.193	0.00	6	0.219	1.24	0.187	0.04
2	0.223	1.16	0.189	0.09	7	0.215	1.14	0.187	0.07
3	0.222	1.37	0.186	0.13	8	0.214	1.10	0.185	0.01
4	0.213	1.18	0.181	0.01	9	0.218	1.21	0.190	0.09
5	0.222	1.24	0.195	0.25	10	0.214	1.21	0.188	0.07

Table 1: *10 comparisons of conventional backprop (BP) and flat minimum search (FMS). The second row (labeled "MSE") shows mean squared error on the test set. The third row ("dto") shows the difference between the percentage of misclassifications and the optimal percentage (9.27). The remaining rows provide the analoguous information for FMS, which clearly outperforms backprop.*

Results. 10 conventional backprop (BP) nets were tested against 10 equally initialized networks trained by flat minimum search (FMS). *After 1,000 epochs, the weights of our nets essentially stopped changing (automatic "early stopping"), while backprop kept changing weights to learn the outliers in the data set and overfit.* In the end, our approach left a single hidden unit h with a maximal weight of 30.0 or -30.0 from the x-axis input. Unlike with backprop, the other hidden units were effectively pruned away (outputs near zero). So was the y-axis input (zero weight to h). It can be shown that this corresponds to an "optimal" net with minimal numbers of units and weights. Table 1 illustrates the superior performance of our approach.

Parameters:
Learning rate: 0.1.
Architecture: (2-20-1).
Number of training epochs: 400,000.
With FMS: $E_{tol} = 0.0001$.
See section 4.6 for parameters common to all experiments.

4.2 EXPERIMENT 2 – recurrent nets.

The method works for continually running fully recurrent nets as well. At every time step, a recurrent net with sigmoid activations in $[0,1]$ sees an input vector from a stream of randomly chosen input vectors from the set $\{(0,0),(0,1),(1,0),(1,1)\}$. The task is to switch on the first output unit whenever an input $(1,0)$ had occurred two time steps ago, and to switch on the second output unit without delay in response to any input $(0,1)$. The task can be solved by a single hidden unit.

Non-weight-decay-like results. With conventional recurrent net algorithms, after training, both hidden units were used to store the input vector. Not so with our new approach. We trained 20 networks. All of them learned perfect solutions. Like with weight decay, most weights to the output decayed to zero. But *unlike* with weight decay, **strong inhibitory** connections (-30.0) switched off one of the hidden units, effectively pruning it away.

Parameters:
Learning rate: 0.1.
Architecture: (2-2-2).
Number of training examples: 1,500.
$E_{tol} = 0.0001$.
See section 4.6 for parameters common to all experiments.

4.3 EXPERIMENT 3 – stock market prediction (1).

We predict the DAX[1] (the German stock market index) using fundamental indicators. Following Rehkugler and Poddig [38], the net sees the following indicators: (a) German interest rate (*"Umlaufsrendite"*), (b) industrial production divided by money supply, (c) business sentiments (*"IFO Geschäftsklimaindex"*). The input (scaled in the interval [-3.4,3.4]) is the difference between data from the current quarter and last year's corresponding quarter. The goal is to predict the sign of next year's corresponding DAX difference.

The training set consists of 24 data vectors from 1966 to 1972. Positive DAX tendency is mapped to target 0.8, otherwise the target is -0.8. The test set consists of 68 data vectors from 1973 to 1990. Flat minimum search (FMS) is compared against: (1) Conventional backprop (BP8) with 8 hidden units, (2) Backprop with 4 hidden units (BP4) (4 hidden units are chosen because pruning methods favor 4 hidden units, but 3 is not enough), (3) Optimal brain surgeon (OBS [12]), with a few improvements (see section 4.6), (4) Weight decay (WD) according to [47] (WD and OBS were chosen because they are well-known and widely used). Three different confidence measures are used: network output exceeding 0.0 (0.6, 0.9) is interpreted as a prediction of positive tendency. Network output below 0.0 (-0.6, -0.9) is interpreted as a prediction of negative tendency. Other outputs don't count.

Since wrong predictions lead to loss of money, performance is measured as follows. The sum of confidently but incorrectly predicted DAX changes is subtracted from the sum of confidently and correctly predicted DAX changes. The result is divided by the sum of absolute DAX changes. The most profitable network predicts everything confidently and correctly. With two networks with equal performance, the one making fewer confident predictions is preferrable.

Results. See table 2. **Our method outperforms the other methods.** Only with confidence measure 0.9, BP4 and BP8 exhibit better performance *per prediction*, but considerably fewer predictions are made. Not much money can be made this way. Using FMS, the number of confindent predictions does not change a lot if the confindence measure is changed. Thus, predicition quality does not depend strongly on the confindence measure to be chosen in advance.

Table 3 shows **upper performance bounds** for early stopping methods. They are obtained by cheating: test set performance is monitored during learning the training set. Column "b" shows the minimal number of millions of training examples needed to achieve small test set error, column "e" shows the corresponding maximal number (where test set error starts growing again). The corresponding interval I tends to be relatively small (only the best result within I is shown). Mean performance on I approximately equals the performance of WD and FMS based on confidence measures 0.0/0.6, and is below the performance of WD and FMS based on confidence measure 0.9. The probability of early stopping within I, but *without* cheating, is small.

[1] Raw DAX version according to *Statistisches Bundesamt* (federal office of statistics). Other data are from the same source (except for business sentiment). Collected by Christian Puritscher, for a diploma thesis in industrial management at LMU, Munich.

	train				test				removed		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	w	u	confidence: 0.0/0.6/0.9		
BP8													
1	0.004	0	0	0	1.002	25	25	22			25.74-68	31.38-65	22.59-29
2	0.003	0	0	0	1.002	29	25	23			44.81-68	34.76-62	20.22-23
3	0.003	0	0	0	0.999	30	25	21			47.33-68	30.97-59	21.31-30
4	0.003	0	0	0	0.860	29	22	17			39.26-68	30.92-56	24.97-35
5	0.003	0	0	0	0.886	30	22	18			39.26-68	35.19-56	24.97-35
6	0.003	0	0	0	0.896	29	24	17			29.15-68	35.80-56	23.61-35
7	0.000	0	0	0	0.966	29	24	20			38.78-68	42.61-59	24.02-34
$\bar{x}$	0.003	0	0	0	0.945	29	24	20			37.76-68	34.52-59	23.10-32
performance per prediction:											0.56-68	0.59-59	0.73-32

to be continued on next page

continued from previous page

	train				test				removed		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	w	u	confidence: 0.0/0.6/0.9		
BP4													
1	0.004	0	0	0	1.070	31	28	23			42.02-68	35.07-60	17.35-27
2	0.004	0	0	0	1.072	31	28	23			42.02-68	35.07-60	17.35-27
3	0.004	0	0	0	1.069	31	28	23			42.02-68	35.07-60	17.35-27
4	0.004	0	0	0	1.071	31	28	23			42.02-68	35.07-60	17.35-27
5	0.004	0	0	0	1.069	31	28	23			42.02-68	35.07-60	17.35-27
6	0.004	0	0	0	1.065	31	28	24			42.02-68	34.47-61	17.35-27
7	0.131	4	2	0	1.043	35	30	21			42.02-68	34.47-61	17.35-27
$\bar{x}$	0.043	1	0	0	1.066	32	28	23			42.02-68	34.90-60	17.35-27
performance per prediction:											0.62-68	0.58-60	0.64-27
OBS													
1	0.016	0	0	0	1.081	27	24	21	9	1	45.18-68	32.04-62	32.98-58
2	0.017	0	0	0	1.073	28	24	21	14	3	45.25-68	35.99-57	28.69-55
3	0.386	8	7	0	1.049	28	24	21	16	4	48.89-68	40.53-61	31.72-55
4	0.158	4	2	2	1.154	27	24	23	15	3	44.67-68	35.61-64	8.39-55
5	0.016	0	0	0	1.181	32	28	25	14	3	32.82-68	32.04-61	36.13-52
6	0.016	0	0	0	1.071	29	24	22	14	2	48.14-68	37.01-61	31.19-56
7	0.014	1	0	0	1.009	26	25	22	17	4	27.17-68	36.23-64	15.82-50
$\bar{x}$	0.089	2	1	0	1.088	28	25	22	14	3	41.73-68	35.64-61	26.42-54
performance per prediction:											0.61-68	0.58-61	0.49-54
WD													
1	0.061	3	0	0	1.123	25	24	24	24	5	44.47-68	44.79-66	39.46-60
2	0.080	3	2	0	1.122	25	24	24	23	5	44.47-68	44.06-67	38.95-59
3	0.147	6	3	0	1.102	32	26	21	24	6	36.47-68	23.20-57	19.56-49
4	0.142	6	3	0	1.018	30	25	23	20	5	46.41-68	39.19-61	30.00-57
5	0.084	3	1	0	1.111	27	25	22	22	5	43.65-68	38.41-63	32.19-60
6	0.073	3	1	0	1.126	25	24	24	22	5	44.47-68	44.06-67	39.00-60
7	0.088	4	1	0	1.115	25	24	22	22	5	44.47-68	45.39-65	38.80-59
$\bar{x}$	0.096	4	2	0	1.102	27	25	23	22	5	43.49-68	39.87-64	33.99-58
performance per prediction:											0.64-68	0.62-64	0.59-58
FMS													
1	0.037	0	0	0	1.179	27	25	24	24	4	47.60-68	45.80-65	35.82-62
2	0.053	0	0	0	1.149	26	24	22	26	5	47.74-68	35.75-58	36.28-56
3	0.037	0	0	0	1.151	25	24	24	23	4	40.89-68	38.54-67	35.58-63
4	0.040	0	0	0	1.146	26	24	23	22	3	40.89-68	36.89-65	38.78-63
5	0.037	0	0	0	1.213	25	25	25	26	4	47.60-68	47.60-68	45.25-67
6	0.039	0	0	0	1.158	27	25	23	20	3	39.70-68	37.39-64	37.39-64
7	0.040	0	0	0	1.140	26	24	23	24	4	40.89-68	43.92-65	37.45-61
$\bar{x}$	0.040	0	0	0	1.162	26	24	23	24	4	43.62-68	40.84-65	38.08-62
performance per prediction:											0.64-68	0.63-65	0.61-62

Table 2: *7 comparisons of conventional backprop (BP4, BP8), optimal brain surgeon (OBS), weight decay (WD), and flat minimum search (FMS). All nets except BP4 start out with 8 hidden units. Column "MSE" shows mean squared error, "F1" ("F2","F3") indicates number of errors greater than 0.2 (0.8, 1.4). "F2" indicates number of incorrect predictions (68 at most) using confindence measure 0.0, "F3" indicates number of incorrect confident predictions using confidence measure 0.6. The difference between "F1" and 68 is the corresponding number of correct confident predictions. Column "w" shows the number of pruned weights, column "u" shows the number of pruned units, the final 3 rows ("0.0", "0.6", "0.9") list pairs of performance (see text) and number of confident predictions. $\bar{x}$ denotes the mean of 7 trials. Note that test MSE is insignificant for performance evaluations (this is due to targets 0.8/-0.8, as opposed to the "real" DAX targets).* **With all confidence measures, our method outperforms all other methods.**

	train				test				interval I		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	b	e	confidence: 0.0/0.6/0.9		
ES													
1	0.004	0	0	0	0.978	25	23	21	11.7	12.4	45.28-68	45.69-60	21.30-27
2	0.006	0	0	0	0.983	26	23	22	1.22	1.59	45.28-68	43.94-64	20.75-29
3	0.007	0	0	0	1.000	26	24	23	1.60	1.97	45.25-68	45.52-65	26.74-33
4	0.004	0	0	0	0.930	26	24	19	8.11	8.27	45.25-68	36.69-61	18.62-26
5	0.006	0	0	0	0.939	27	23	19	1.47	1.66	46.71-68	47.02-59	20.84-26
6	0.005	0	0	0	0.973	26	24	21	0.88	3.73	45.25-68	46.71-63	19.54-27
7	0.007	0	0	0	0.993	26	24	22	1.03	1.16	45.25-68	46.80-64	22.88-29
$\bar{x}$	0.007	0	0	0	0.971	26	24	21			45.47-68	44.62-62	21.52-28
performance per prediction:											0.67-68	0.72-62	0.76-28

Table 3: *Upper performance bounds for arbitrary early stopping (ES) methods. They are obtained by cheating by looking at test set performance during learning. All nets start out with 8 hidden units. Column "MSE" shows mean squared error, "F1" ("F2","F3") indicates number of errors greater than 0.2 (0.8, 1.4). "F2" indicates number of incorrect predictions (68 at most) using confindence measure 0.0, "F3" indicates number of incorrect confident predictions using confidence measure 0.6. The difference between "F1" and 68 is the corresponding number of correct confident predictions. Column "b" shows the minimal number of millions of training examples needed to achieve small test set error, column "e" shows the corresponding maximal number (where test set error starts growing again). The final 3 rows ("0.0", "0.6", "0.9") list pairs of performance (see text) and number of confident predictions. $\bar{x}$ denotes the mean of 7 trials.*

Parameters:
Learning rate: 0.01.
Architecture: (3-8-1), except BP4 with (3-4-1).
Number of training examples: 20,000,000.
Method specific parameters:

- FMS: $E_{tol} = 0.13$; $\Delta\lambda = 0.001$.
- WD: like with FMS, but $w_0 = 0.2$.
- OBS: $E_{tol} = 0.015$ (the same result was obtained with higher E_{tol} values, e.g. 0.13).

See section 4.6 for parameters common to all experiments.

4.4 EXPERIMENT 4 – stock market prediction (2).

We predict the DAX again, using the basic set-up of the experiment in section 4.3. However, the following modifications are introduced:

- There are two additional inputs: (d) dividend rate, (c) foreign orders in manufacturing industry.
- Monthly predictions are made. The net input is the difference between the current month's data and last month's data. The goal is to predict the sign of next month's corresponding DAX difference.
- There are 228 training examples and 100 test examples.
- The target is the percentage of DAX change scaled in the interval [-1,1] (outliers are ignored).
- Network output exceeding 0.0 (0.6, 0.8) is interpreted as a prediction of positive tendency. Network output below 0.0 (-0.6, -0.8) is interpreted as a prediction of negative tendency. Other outputs don't count.

- Performance of WD and FMS is also tested on networks "spoiled" by conventional backprop ("WDR" and "FMSR" – the "R" stands for *R*etraining).

Results are shown in table 4. **Average performance of our method almost always exceeds the ones of weight decay, OBS, and conventional backprop.** The only exception is weight decay with confidence measure 0.6 (but with the other comparisons, flat minimum search outperforms weight decay). Table 4 also shows superior performance of our approach when it comes to retraining "spoiled" networks (note that OBS is a retraining method by nature). FMS led to the best improvements in generalization performance.

Like table 3, table 5 shows **upper performance bounds** for early stopping methods. Again, they are obtained by cheating: test set performance is monitored during learning the training set. Columns "b" and "e" are analoguous to those in table 3. Mean performance on interval I approximately equals the performance of WD and FMS based on confidence measures 0.0/0.8, and exceeds the performance of all investigated methods based on confidence measure 0.6. Recall that the probability of early stopping within I, but *without* cheating, is small.

	train				test				removed		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	w	u	confidence: 0.0/0.6/0.8		
BP													
1	0.170	154	37	12	0.613	81	30	31			38.51-100	21.79-43	11.46-13
2	0.183	142	42	18	0.430	78	25	20			55.10-100	19.29-28	10.42-12
3	0.186	150	48	16	0.580	73	35	31			20.69-100	16.60-22	15.51-13
4	0.176	135	37	16	0.459	83	26	22			57.33-100	14.40-21	6.19-8
5	0.180	148	45	13	0.600	80	36	27			24.43-100	20.77-34	8.50-13
6	0.184	156	40	12	0.558	84	33	29			39.05-100	27.88-30	2.71-7
7	0.187	152	44	14	0.529	81	29	30			49.16-100	25.19-25	2.19-4
8	0.173	140	37	12	0.552	74	27	31			41.27-100	13.18-24	8.97-10
9	0.189	150	41	13	0.528	78	29	28			42.50-100	16.04-33	18.05-18
10	0.186	143	41	16	0.503	82	30	25			48.06-100	11.42-27	9.30-11
$\bar{x}$	0.181	147	41	14	0.535	79	30	27			41.61-100	18.67-29	9.33-11
performance per prediction:											0.42-100	0.65-29	0.85-11
OBS													
1	0.217	158	40	16	0.515	80	29	32	18	2	39.09-100	19.89-38	9.91-10
2	0.213	164	39	14	0.491	79	30	32	17	2	40.88-100	22.72-25	10.72-11
3	0.245	170	51	22	0.463	81	29	26	12	0	50.78-100	13.93-18	10.68-8
4	0.215	162	41	17	0.525	76	31	31	15	1	32.20-100	21.61-27	13.82-14
5	0.212	165	42	14	0.498	83	34	28	14	1	34.50-100	17.18-25	10.84-7
6	0.235	162	50	24	0.503	79	29	28	13	0	40.55-100	14.31-25	8.23-8
7	0.217	154	43	18	0.525	84	29	28	17	2	43.60-100	10.99-26	4.55-8
8	0.213	162	45	15	0.497	84	33	27	14	1	36.58-100	17.83-24	4.57-6
9	0.213	160	39	15	0.498	81	28	28	18	2	40.36-100	21.52-28	8.52-7
10	0.212	167	41	19	0.500	83	30	25	13	1	45.72-100	20.39-26	8.23-10
$\bar{x}$	0.219	162	43	17	0.502	81	30	29	15	1	40.43-100	18.04-26	9.01-9
performance per prediction:											0.40-100	0.68-26	1.01-9
WDR													
1	0.169	155	35	12	0.626	80	33	31	0	0	34.52-100	21.90-44	16.44-15
2	0.181	145	42	18	0.446	77	25	20	0	0	55.10-100	19.19-28	10.56-12
3	0.185	150	48	17	0.584	73	34	32	0	0	24.58-100	15.45-23	14.33-12
4	0.175	137	37	15	0.451	82	23	21	0	0	62.54-100	17.14-23	6.33-9
5	0.179	146	45	13	0.612	80	36	26	0	0	23.70-100	20.33-34	9.48-11
6	0.183	156	39	13	0.544	82	31	29	0	0	36.16-100	28.25-32	5.95-7
7	0.186	152	43	14	0.524	83	30	28	0	0	50.37-100	25.42-25	2.19-4
8	0.171	138	36	13	0.560	76	30	32	0	0	39.28-100	13.64-23	8.97-10

to be continued on next page

continued from previous page

	train				test				removed		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	w	u	confidence: 0.0/0.6/0.8		
9	0.189	150	44	14	0.529	78	32	28	0	0	39.59-100	18.91-33	17.53-19
10	0.186	142	41	15	0.504	85	32	27	0	0	45.88-100	13.72-29	7.79-10
$\bar{x}$	0.180	147	41	14	0.538	80	31	27	0	0	41.17-100	19.39-29	9.95-11
performance per prediction:											0.41-100	0.66-29	0.91-11
FMSR													
1	0.168	154	35	11	0.627	80	32	32	0	0	36.14-100	23.79-40	16.44-15
2	0.180	146	42	18	0.449	76	26	21	0	0	53.86-100	19.19-28	10.56-12
3	0.185	148	48	16	0.583	74	34	32	0	0	24.58-100	15.45-23	15.51-13
4	0.174	136	36	15	0.447	81	22	21	0	0	64.07-100	19.24-24	8.15-10
5	0.179	143	44	13	0.611	80	35	26	0	0	23.70-100	22.74-33	9.48-11
6	0.182	156	39	13	0.546	81	32	29	0	0	34.52-100	26.77-31	5.95-7
7	0.185	151	43	14	0.525	82	29	28	0	0	50.53-100	27.71-25	2.19-4
8	0.170	138	36	13	0.559	77	29	32	0	0	40.69-100	12.83-24	8.97-10
9	0.188	156	40	14	0.535	76	31	29	0	0	39.92-100	24.30-34	17.09-18
10	0.186	143	40	15	0.542	84	32	27	0	0	47.34-100	11.37-27	7.79-10
$\bar{x}$	0.180	147	40	14	0.542	79	29	28	0	0	41.57-100	20.34-29	10.21-11
performance per prediction:											0.42-100	0.70-29	0.93-11
WD													
1	0.235	159	54	23	0.435	80	29	25	18	3	37.04-100	18.51-13	1.06-1
2	0.235	158	54	23	0.429	81	27	25	18	3	43.43-100	18.98-13	1.06-1
3	0.235	159	55	23	0.435	82	28	24	18	3	44.47-100	18.51-13	1.96-2
4	0.235	164	55	25	0.454	79	27	24	13	2	54.04-100	12.61-17	0.51-1
5	0.235	158	54	23	0.438	82	29	25	18	3	36.50-100	18.51-13	1.96-2
6	0.235	158	48	24	0.464	89	28	32	18	3	50.53-100	20.77-11	0.90-1
7	0.235	164	56	26	0.512	83	32	28	14	2	35.60-100	15.27-17	9.01-6
8	0.235	166	54	24	0.471	81	33	24	18	3	32.03-100	20.12-14	7.65-4
9	0.235	158	54	23	0.442	78	29	26	18	3	36.81-100	17.88-12	1.06-1
10	0.235	158	54	23	0.440	80	30	26	18	3	37.04-100	17.88-12	1.06-1
$\bar{x}$	0.235	160	54	24	0.452	82	29	26	17	3	40.75-100	17.90-12	2.62-2
performance per prediction:											0.41-100	1.32-12	1.31-2
FMS													
1	0.236	156	47	21	0.464	78	27	26	18	3	54.11-100	17.51-13	10.84-6
2	0.223	167	51	22	0.466	79	30	26	18	3	48.73-100	18.20-15	4.52-4
3	0.237	150	49	21	0.490	75	27	24	17	3	49.77-100	16.09-16	14.35-7
4	0.253	166	57	24	0.461	83	30	29	24	4	44.26-100	18.40-11	9.38-5
5	0.245	170	47	20	0.501	83	29	25	18	3	39.49-100	17.58-11	0.71-1
6	0.247	164	58	25	0.486	81	30	27	24	4	31.12-100	24.00-16	16.33-10
7	0.237	161	53	24	0.478	81	31	27	18	3	36.04-100	12.14-10	0.82-1
8	0.252	166	57	24	0.459	84	29	28	21	4	45.38-100	17.83-12	1.38-1
9	0.235	163	56	24	0.453	82	29	24	18	3	43.81-100	16.61-14	1.06-1
10	0.235	158	44	23	0.462	82	27	26	18	3	51.26-100	17.62-13	0.43-1
$\bar{x}$	0.240	162	52	23	0.472	81	29	26	19	3	44.40-100	17.60-13	5.98-4
performance per prediction:											0.44-100	1.24-13	1.62-4

Table 4: *10 comparisons of conventional backprop (BP), optimal brain surgeon (OBS), weight decay after spoiling the net with BP (WDR), flat minimum search after spoiling the net with BP (FMSR), weight decay (WD), flat minimum search (FMS). All nets start out with 8 hidden units. Column "MSE" shows mean squared error, "F1" ("F2","F3") indicates number of errors greater than 0.2 (0.8, 1.4). Column "w" shows the number of pruned weights, column "u" shows the number of pruned units. The final 3 rows ("0.0", "0.6", "0.8") list pairs of performance (see text) and number of confident predictions. $\bar{x}$ denotes the mean of 10 trials. In one case (confidence measure 0.6), weight decay does slightly better than FMS.* **But in all other cases, flat minimum search outperforms all other methods, including WD.**

	train				test				interval I		performance–nr.-pred.		
nr.	MSE	F1	F2	F3	MSE	F1	F2	F3	b	e	confidence: 0.0/0.6/0.8		
ES													
1	0.209	159	54	17	0.487	77	29	26	0.96	10.4	42.77-100	22.87-22	12.99-8
2	0.206	153	50	17	0.405	74	25	20	1.12	1.24	56.12-100	22.52-19	9.86-8
3	0.209	154	47	21	0.530	74	29	33	1.20	1.32	42.93-100	19.95-24	14.33-9
4	0.198	149	46	19	0.436	76	24	25	1.52	2.24	61.71-100	11.67-17	9.36-9
5	0.227	160	55	20	0.467	77	28	25	0.52	0.80	48.38-100	20.41-19	9.03-7
6	0.234	163	54	20	0.444	77	28	24	0.44	0.72	43.85-100	22.06-19	10.44-5
7	0.222	157	55	19	0.425	78	24	23	0.60	0.96	60.75-100	25.13-21	4.51-4
8	0.233	160	58	19	0.460	79	28	24	0.44	0.64	43.23-100	24-08-18	1.38-1
9	0.236	158	58	21	0.430	83	26	24	0.36	0.48	50.83-100	14.88-13	2.44-2
10	0.232	162	58	20	0.441	80	28	25	0.36	0.52	44.66-100	15.99-19	2.44-2
$\bar{x}$	0.221	158	54	19	0.453	78	27	25			49.52-100	19.95-19	7.68-5.5
performance per prediction:											0.50-100	1.04-19	1.40-5.5

Table 5: *Upper performance bounds for arbitrary early stopping (ES) methods. They are obtained by cheating by looking at test set performance during learning. All nets start out with 8 hidden units. Column "MSE" shows mean squared error, "F1" ("F2","F3") indicates number of errors greater than 0.2 (0.8, 1.4). Column "b" shows the minimal number of millions of training examples needed to achieve small test set error, column "e" shows the corresponding maximal number (where test set error starts growing again). The final 3 rows ("0.0", "0.6", "0.8") list pairs of performance (see text) and number of confident predictions. $\bar{x}$ denotes the mean of 10 trials.*

Parameters:
Learning rate: 0.01.
Architecture: (5-8-1).
Number of training examples: 20,000,000.
Method specific parameters:

- FMS: $E_{tol} = 0.235$; $\Delta\lambda = 0.0001$; if $E_{\text{average}} < E_{tol}$ then $\Delta\lambda$ is set to 0.001.
- WD: like with FMS, but $w_0 = 0.2$.
- FMSR: like with FMS, but $E_{tol} = 0.15$; number of retraining examples: 5,000,000.
- WDR: like with FMSR, but $w_0 = 0.2$.
- OBS: $E_{tol} = 0.235$.

See section 4.6 for parameters common to all experiments.

4.5 EXPERIMENT 5 – stock market prediction (3).

This time, we predict the DAX using weekly *technical* (as opposed to fundamental) indicators. The data (DAX values and 35 technical indicators) was provided by *Bayerische Vereinsbank.*

To analyze the data, we computed: (1) The pairwise correlation coefficients of the 35 technical indicators. (2) The maximal pairwise correlation coefficients of all indicators and all linear combinations of two indicators. This analysis reveiled that only 4 indicators are not highly correlated. For such reasons, our nets see only the 8 most recent DAX-changes and the following technical indicators: (a) the DAX value, (b) change of 24-week relative strength index ("RSI") – the relation of increasing tendency to decreasing tendency, (c) "5 week statistic", (d) "MACD" (smoothened difference of exponentially weighted 6 week and 24 week DAX).

The final network input is obtained by scaling the values (a-d) and the 8 most recent DAX-changes in $[-2, 2]$. The training set consists of 320 data points (July 1985 to August 1991). The targets are the actual DAX changes scaled in $[-1, 1]$.

The following methods are applied to the training set: (1) Conventional backprop (BP), (2) weight decay (WD) according to [47], (3) flat minimum search (FMS). The resulting nets are evaluated on a test set consisting of 100 data points (August 1991 to July 1993).

Like in section 4.3, three different confidence measures are used: network output exceeding 0.0 (0.2, 0.4) is interpreted as a prediction of positive tendency. Network output below 0.0 (-0.2, -0.4) is interpreted as a prediction of negative tendency. Other outputs don't count. Performance is measured like in section 4.3.

Results. Table 6 shows the results. **Again, our method outperforms the other methods.**

	train					test					rem.		performance–nr.-pred.		
	MSE	F1	F2	F3	F4	MSE	F1	F2	F3	F4	w	u	confidence: 0.0/0.2/0.4		
BP															
1	0.15	21	121	19	13	1.10	44	85	54	21			5.65-100	4.86-96	8.60-91
2	0.14	23	136	13	12	1.03	44	85	51	26			11.07-100	12.32-93	8.46-85
3	0.14	27	132	18	15	0.96	39	82	47	14			12.66-100	14.12-89	15.59-86
4	0.10	24	102	13	13	1.19	47	84	59	24			5.86-100	5.75-94	4.64-92
5	0.13	22	119	16	12	1.07	48	88	52	24			13.75-100	12.35-94	11.13-92
6	0.13	34	134	17	21	1.21	52	87	60	21			-16.7-100	-13.9-92	-13.2-84
7	0.12	20	131	14	12	1.08	48	87	53	21			9.67-100	12.06-92	10.16-88
8	0.13	24	116	19	16	0.90	37	81	44	16			28.45-100	28.67-92	28.13-89
9	0.13	16	116	15	10	1.22	53	90	63	28			-11.0-100	-14.0-94	-5.00-88
10	0.12	23	108	12	16	1.00	44	82	50	21			21.53-100	20.12-95	21.00-93
$\bar{x}$	0.13	23	122	16	14	1.08	46	85	53	22			8.08-100	8.23-93	8.95-89
	performance per prediction:												0.081-100	0.088-93	0.101-89
WD1															
1	0.51	148	275	125	85	0.330	43	68	24	22	109	8	18.69-100	0.0-0	0.0-0
2	0.51	149	275	124	86	0.330	39	71	23	20	110	8	25.60-100	0.0-0	0.0-0
3	0.51	148	276	124	85	0.342	48	70	23	22	109	8	-6.88-100	0.0-0	0.0-0
4	0.51	145	276	126	84	0.339	48	70	24	21	111	8	4.97-100	0.0-0	0.0-0
5	0.51	147	275	125	85	0.331	39	71	23	20	111	8	23.44-100	0.0-0	0.0-0
6	0.51	147	275	124	85	0.343	48	70	23	22	110	8	-6.88-100	0.0-0	0.0-0
7	0.51	146	276	126	84	0.330	38	71	23	21	110	8	26.84-100	0.0-0	0.0-0
8	0.51	145	276	127	84	0.337	47	69	23	22	110	8	-3.28-100	0.0-0	0.0-0
9	0.51	146	275	125	85	0.330	38	70	24	22	110	8	25.84-100	0.0-0	0.0-0
10	0.51	150	275	124	87	0.332	44	68	24	22	110	8	21.40-100	0.0-0	0.0-0
$\bar{x}$	0.51	147	275	125	85	0.334	43	70	23	21	110	8	12.97-100	0.0-0	0.0-0
	performance per prediction:												0.130-100	0.0-0	0.0-0
WD2															
1	0.34	84	215	60	46	0.425	44	74	26	20	64	3	22.04-100	13.08-53	9.01-25
2	0.34	106	230	61	63	0.422	50	81	25	24	75	3	1.75-100	3.95-42	4.70-21
3	0.34	75	239	57	43	0.453	47	83	27	26	76	3	5.67-100	5.53-52	8.97-33
4	0.34	89	231	61	36	0.410	45	79	21	18	69	3	5.64-100	6.60-53	5.97-17
5	0.34	86	232	50	47	0.409	46	77	23	23	77	3	5.61-100	10.93-70	14.59-21
6	0.34	89	244	66	50	0.527	46	79	29	19	67	3	5.26-100	-7.20-62	-1.18-35
7	0.34	84	230	60	48	0.401	44	74	24	22	80	3	9.75-100	3.56-44	0.53-24
8	0.34	86	242	62	59	0.481	40	75	21	21	72	3	5.79-100	7.63-69	-14.0-28
9	0.34	90	234	51	47	0.463	47	76	25	26	68	2	12.32-100	20.61-70	0.23-32
10	0.34	93	234	68	53	0.362	46	72	20	23	67	3	15.98-100	19.98-36	9.90-18
$\bar{x}$	0.34	88	233	60	49	0.435	46	77	24	22	72	3	8.98-100	8.47-55	3.88-25
	performance per prediction:												0.090-100	0.154-55	0.153-25
FMS1															
1	0.38	111	231	87	60	0.391	44	76	25	22	103	7	18.82-100	5.97-23	7.46-19
2	0.45	127	258	96	73	0.349	45	81	24	22	102	7	23.26-100	5.14-20	8.55-18
3	0.47	126	276	109	77	0.356	45	79	16	23	103	7	26.11-100	9.42-13	9.42-13

to be continued on next page

continued from previous page															
	train					test					rem.		performance–nr.-pred.		
	MSE	F1	F2	F3	F4	MSE	F1	F2	F3	F4	w	u	confidence: 0.0/0.2/0.4		
4	0.48	126	276	109	71	0.339	43	79	23	18	103	7	26.46-100	9.68-13	8.59-12
5	0.46	122	274	108	75	0.344	42	78	24	22	103	7	27.49-100	8.43-13	4.86-11
6	0.48	129	270	116	76	0.343	46	76	24	22	103	7	18.09-100	6.64-6	6.64-6
7	0.47	124	274	108	76	0.352	43	77	24	22	103	7	24.53-100	8.82-12	4.96-9
8	0.48	126	276	109	71	0.330	44	79	22	20	103	7	26.46-100	9.68-13	8.59-12
9	0.48	129	273	114	77	0.340	46	76	24	22	103	7	18.09-100	5.60-7	5.60-7
10	0.47	121	274	101	65	0.340	43	78	26	20	103	7	29.72-100	11.42-27	10.91-24
$\bar{x}$	0.46	124	268	106	72	0.348	44	78	23	21	103	7	21.26-100	8.08-15	6.70-13
performance per prediction:													0.213-100	0.550-15	0.511-13
FMS2															
1	0.34	105	218	64	65	0.400	40	73	19	21	52	4	14.60-100	13.61-39	5.56-11
2	0.34	103	223	65	59	0.362	39	71	20	20	52	4	18.88-100	14.55-33	5.91-9
3	0.34	94	222	64	58	0.421	36	75	22	16	51	4	20.54-100	11.15-40	3.83-12
4	0.34	97	228	65	53	0.370	42	79	20	19	52	4	22.07-100	23.44-32	7.50-10
5	0.34	89	243	59	49	0.631	54	82	36	25	52	4	2.02-100	-7.08-68	-3.77-43
6	0.34	102	219	64	58	0.359	39	70	19	20	52	4	20.08-100	17.40-36	8.80-11
7	0.34	98	227	64	54	0.368	42	77	19	19	52	4	22.07-100	26.38-33	7.50-10
8	0.34	90	231	57	53	0.605	48	85	38	23	52	4	3.71-100	-14.8-55	-12.8-41
9	0.34	98	228	65	54	0.371	42	78	19	19	52	4	22.07-100	23.44-32	7.50-10
10	0.34	95	217	73	54	0.410	46	75	27	21	52	4	9.75-100	1.49-13	0.59-12
$\bar{x}$	0.34	97	226	64	56	0.430	43	76	24	20	52	4	15.58-100	10.95-38	3.06-17
performance per prediction:													0.156-100	0.287-38	0.181-17

Table 6: *10 comparisons of conventional backprop (BP), optimal brain surgeon (OBS), weight decay (WD), flat minimum search (FMS). All nets start out with 9 hidden units. Column "MSE" shows mean squared error, "F1" indicates number of tendency errors, "F2" ("F3") indicates number of errors greater than 0.2 (0.8), "F4" indicates number of tendency errors for changes from increasing DAX tendency to decreasing DAX tendency (or vice versa). Column "w" shows the number of pruned weights, column "u" shows the number of pruned units, the final 3 rows ("0.0", "0.2", "0.4") list pairs of performance (see text) and number of confident predictions. $\bar{x}$ denotes the mean of 10 trials.* **With all confidence measures, our method outperforms BP and WD.**

Parameters:
Learning rate: 0.01.
Architecture: (12-9-1).
Training time: 10,000,000 examples.
Method specific parameters:

- FMS1: $E_{tol} = 0.34$; $\Delta\lambda = 0.003$.
- FMS2: like with FMS1, but $\Delta\lambda = 0.0005$. If $E_{\text{average}} < E_{tol}$ then $\Delta\lambda$ is set to 0.001.
- WD1: like with FMS1, but $w_0 = 0.2$.
- WD2: like with FMS2, but $w_0 = 0.2$.

See section 4.6 for parameters common to all experiments.

4.6 DETAILS / PARAMETERS

With exception of the experiment in section 4.2, all units are sigmoid in the range of $[-1.0, 1.0]$. Weights are constrained to $[-30, 30]$ and initialized in [-0.1,0.1]. The latter ensures high first order derivatives in the beginning of the learning phase. WD is set up to hardly punish weights below $w_0 = 0.2$. E_{average} is the average error on the training set, approximated using exponential decay: $E_{\text{average}} \longleftarrow \gamma E_{\text{average}} + (1-\gamma)E_q(w, D_0)$, where $\gamma = 0.85$. All nets with the same number (see tables 1 - 6) start with the same weight initialization.

FMS details. To control $B(w, D_0)$'s influence during learning, its gradient is normalized and multiplied by the length of $E_q(w, D_0)$'s gradient (same for weight decay, see below). λ is computed like in [47] and initialized with 0. Absolute values of first order derivatives are replaced by 10^{-20} if below this value. We ought to judge a weight w_{ij} as being pruned if δw_{ij} (see equation (11) in appendix A.3) exceeds the length of the weight range. However, the unknown scaling factor ϵ (see equations (9) and (11) in appendix A.3) is required to compute δw_{ij}. Therefore, we judge a weight w_{ij} as being pruned if, with arbitrary ϵ, δw_{ij} is much bigger than the corresponding δ's of the other weights (typically, there are clearly separable classes of weights with high and low δ's).

If all weights to and from a particular unit are very close to zero, the unit is lost: due to tiny derivatives, the weights will never again increase significantly. Sometimes, it is necessary to bring lost units back into the game. For this purpose, every n_{init} time steps (typically, n_{init} = 500,000), all weights w_{ij} with $0 \leq w_{ij} < 0.01$ are randomly re-initialized in $[0.005, 0.01]$; all weights w_{ij} with $0 \geq w_{ij} > -0.01$ are randomly initialized in $[-0.01, -0.005]$, and λ is set to 0.

Weight decay details. We used the weight decay term in [47]: $D(w, w_0) = \sum_{i,j} \frac{w_{ij}^2/w_0}{1+w_{ij}^2/w_0}$. Like with FMS, $D(w, w_0)$'s gradient was normalized and multiplied by the length of $E_q(w, D_0)$'s gradient. λ was adjusted like with FMS. Lost units were brought back like with FMS.

Modifications of OBS. Typically, most weights exceed 1.0 after training. Therefore, higher order terms of δw in the Taylor expansion of the error function do not vanish. Hence, OBS is not fully theoretically justified. Still, we used OBS to delete high weights, assuming that higher order derivatives are small if second order derivatives are. To obtain reasonable performance, we modified the original OBS procedure (notation following Hassibi and Stork [12]):

- To detect the weight that deserves deletion, we use both $L_q = \frac{w_q^2}{[H^{-1}]_{qq}}$ (the original value used by Hassibi et al.) and $T_q := -\frac{\partial E}{\partial w_q} w_q + \frac{1}{2}\frac{\partial^2 E}{\partial w_q^2} w_q^2$. Here H denotes the Hessian and H^{-1} its approximate inverse. We delete the weight causing minimal training set error (after tentative deletion).
- Like with OBD [20], to prevent numerical errors due to small eigenvalues of H, we do: if $L_q < 0.0001$ or $T_q < 0.0001$ or $\parallel I - H^{-1}H \parallel > 10.0$ (bad approximation of H^{-1}), we only delete the weight detected in the previous step – the other weights remain the same. Here $\parallel . \parallel$ denotes the sum of the absolute values of all components of a matrix.
- If OBS' adjustment of the remaining weights leads to at least one absolute weight change exceeding 5.0, then δw is scaled such that the maximal absolute weight change is 5.0. This leads to better performance (also due to small eigenvalues).
- If $E_{\text{average}} > E_{tol}$ after weight deletion, then the net is retrained until either $E_{\text{average}} < E_{tol}$ or the number of training examples exceeds 800,000. Practical experience indicates that the choice of E_{tol} hardly influences the result.
- OBS is stopped if $E_{\text{average}} > E_{tol}$ after retraining. The most recent weight deletion is countermanded.

5 RELATION TO PREVIOUS WORK

Most previous algorithms for finding low complexity networks with high generalization capability are based on more prior assumptions than our approach (see appendix A.2). They can be broadly classified into two categories (see [40], however, for an exception):

(1) Assumptions about the prior weight distribution. Hinton and van Camp [14] and Williams [49] assume that pushing the posterior distribution (after learning) close to the prior leads to "good" generalization (see more details below). Weight decay (e.g. [11, 18]) can be derived e.g. from Gaussian or Laplace priors. Nowlan and Hinton [34] assume that networks with many similar weights generated by Gaussian mixtures are "better" *a priori*. MacKay's priors [23] are implicit in additional penalty terms, which embody the assumptions made.

(2) Prior assumptions about how theoretical results on early stopping and network complexity carry over to practical applications. Such assumptions are implicit in methods based on validation sets [30, 43, 7, 13], e.g. "generalized cross validation" [6, 9], "final prediction error" [1], "generalized prediction error" [29, 28]. See also Holden [16], Wang et al. [46], Amari and Murata [2], and Vapnik's "structural risk minimization" [10, 44].

Constructive algorithms / pruning algorithms. Other architecture selection methods are less flexible in the sense that they can be used only either before or after weight adjustments. Examples are "sequential network construction" [8, 3, 27], input pruning [28, 37], unit pruning [48, 31, 21], weight pruning, e.g. "optimal brain damage" (OBD [20]), "optimal brain surgeon" (OBS [12]).

Hinton and van Camp [14]. They minimize the sum of two terms: the first is conventional error plus variance, the other is the distance $\int p(\alpha \mid D_0) \log \frac{p(\alpha|D_0)}{p(\alpha)} d\alpha$ between posterior $p(\alpha \mid D_0)$ and prior $p(\alpha)$. The problem is to choose a "good" prior. In contrast to their approach, our approach does not require a "good" prior given in advance. Furthermore, Hinton and van Camp have to compute variances of weights and unit activations, which (in general) *cannot* be done using linear approximation. Intuitively speaking, their weight variances are related to our Δw_{ij}. Our approach, however, *does* justify linear approximation, as seen in appendix A.4.

Wolpert [50]. His (purely theoretical) analysis suggests an interesting different additional error term (taking into account local flatness in all directions): the logarithm of the Jacobi determinant of a functional from weight space to the space of possible nets. This term is small if the net output (based on the current weight vector) is locally flat in weight space (if many weights lead to the same net function in the space of possible net functions). It is not clear, however, how to derive a practical algorithm (e.g. a pruning algorithm) from this.

Murray and Edwards [32]. They obtain additional error terms consisting of weight squares and second order derivatives. Unlike our approach, theirs *explicitly* prefers weights near zero. In addition, their approach appears to require much more computation time (due to second order derivatives in the error term).

APPENDIX – THEORETICAL JUSTIFICATION

Contents:

A.1. THE PROBABILITY DISTRIBUTIONS

Outline. To measure overfitting error (in section A.2), we need probability distributions on data and weights. The targets are partly due to noise (see section 2). The probability of matching the original, noise-free target component of an element in $\bar{D}$ (see section 2) depends on the weights and the input. Using Bayes' rule, we obtain an *a posteriori* distribution of the weights.

Formal details. In what follows, α denotes a weight vector, and $\breve{D} = \{(x_p, d_p) \mid p \geq 1\} \subseteq D$ (in section A.2 we will consider the special cases $\breve{D} = D$ and $\breve{D} = D_0$). In analogy to section 2, we define $E_q(\alpha, \breve{D}) := \frac{1}{|\breve{D}|} \sum_{(x_p, d_p) \in \breve{D}} \| d_p - o_p(\alpha, x_p) \|^2$ for $\breve{D}$ with finite cardinality. For $\breve{D}$ with infinite cardinality, we define $E_q(\alpha, \breve{D}) := \lim_{n \to \infty} E_q(\alpha, D_n)$, if this limes is finite and equal for all $\{D_n \mid n \geq 1\}$ with $\lim_{n \to \infty} D_n = \breve{D}$, where $|D_n| = n$ and $D_n \subset \breve{D}$. Otherwise $E_q(w, \breve{D})$ is undefined.

Let $p(\breve{D} \mid \alpha)$ denote the probability distribution according to which the network makes predicitions[2] of $\bar{D}$'s output values in the input range of $\breve{D}$, given weights α. We assume Gaussian noise[3] with deviation σ on the noise-free targets. With $\beta := \frac{1}{\sigma^2}$, we obtain (see [23, 52])

$$p(\breve{D} \mid \alpha) = \frac{\exp\left(-\beta E_q(\alpha, \breve{D})\right)}{Z(\beta)} \ . \tag{3}$$

Here $Z(\beta)$ is a normalizing constant. $Z(\beta) = \int_{\Re} \exp(-\beta E_q(\alpha, \tilde{D})) d\tilde{D}$, where $\Re$ is the K-dimensonal vector space of possible data $\tilde{D} \in \Re$. $Z(\beta)$ does not depend on α, due to translation invariance of the error in $\Re$. $Z(\beta) = \left(\sqrt{\frac{\pi}{4\beta}}\right)^K$ holds.[4]

Using e.g. gradient descent algorithms, the prior does *not* coincide with the distribution used for weight initialization. The "true" prior $p_A(\alpha)$ depends on the nature of the learning algorithm A. $p_A(\alpha)$ gives the probability of α after training with A[5]. From practical experience we infer that weight initialization does not heavily influence the posterior. From Bayes' rule we obtain the posterior weight distribution, given the observations $\breve{D}$:

[2]Following [23, 51], $p(\breve{D} \mid \alpha)$ is just a shorthand expression for $p(\{d_p\} \mid \{x_p\}, \alpha)$.

[3]Alternatively, instead of assuming Gaussian noise, equation (3) can be obtained by following Levin, Tishby and Solla [22]. They derive it as the only solution of a functional equation, where β is a constant expressing the sensitivity of $p(\breve{D} \mid \alpha)$ on the error.

[4]It may be useful to define $\bar{E}(\beta) := \int E_q(\alpha, \tilde{D}) p(\tilde{D} \mid \alpha) d\tilde{D} = \frac{1}{Z(\beta)} \int E_q(\alpha, \tilde{D}) \exp(-\beta E_q(\alpha, \tilde{D})) d\tilde{D} = \left(\frac{1}{2\beta}\right)^K$. $\bar{E}(\beta)$ is a function of the derivative of $Z(\beta)$: $\log \bar{E}(\beta) = K \log\left(-\frac{1}{K}\frac{d \log Z(\beta)}{d\beta}\right) = -K \log(2\beta)$. $\bar{E}(\beta)$ is the mean squared error over all possible data when using a fixed β, thus being a measure of the "acceptable error level". See Levin et al. [22]. Why should $\bar{E}(\beta)$ be of interest? With an estimate of β we are able to estimate E_{tol} as well.

[5]For instance, MacKay [23] obtains $p_A(\alpha)$ equal to a multidimensional Gaussian distribution with mean 0, by adding $\lambda \| \alpha \|^2$ to his error function. Similar statements can be made about Hinton and van Camp's approach [14].

$$p_A(\alpha \mid \breve{D}) = \frac{p(\breve{D} \mid \alpha) p_A(\alpha)}{p_A(\breve{D})} \; . \tag{4}$$

$p_A(\breve{D}) := \int p(\breve{D} \mid \alpha) p_A(\alpha) d\alpha$ tells us about learning algorithm A's usefulness for predicting $\bar{D}$'s output values in $\breve{D}$'s input range. $p_A(\alpha) = \int p_A(\alpha \mid \tilde{D}) p_A(\tilde{D}) d\tilde{D}$ holds. With problem class P and $\breve{D} \in P$, $p_A(\breve{D}) = p_P(\breve{D})$ defines properties of an "optimal" learning algorithm A with respect to P, where $p_P(\tilde{D})$ is the probability of choosing $\tilde{D}$ from P.

With uniform $p_P(.)$, no algorithm can outperform error reducing algorithms (see [51]). Typical problem distributions in the real world are *not* uniform, however – the real world appears to favor solutions with low algorithmic complexity. See e.g. [40]. See [41] for a universal "self-improving" system which continually attempts to incrementally modify its prior based on experience with previous problems.

A.2. OVERFITTING ERROR

Outline. In analogy to [46] and [10], we decompose the generalization error into an "overfitting" error and an "underfitting" error. There is no significant underfitting error (corresponding to Vapnik's empirical risk) if $E_q(w, D_0) \leq E_{tol}$. Some thought is required, however, to define the "overfitting" error. We do this in a novel way.

Intuition. Since we do not know the relation D, we cannot know $p(\alpha \mid D)$, the "optimal" posterior weight distribution we would obtain by training the net on D ($\rightarrow$ "sure thing hypothesis"). But, for theoretical purposes, suppose we *did* know $p(\alpha \mid D)$. Then we could use $p(\alpha \mid D)$ to initialize weights before learning the training set D_0. Using the Kullback-Leibler distance [19], we measure the information (due to noise) conveyed by D_0, but not by D (see figure 4). In conjunction with the initialization above, this provides the conceptual setting for defining an overfitting error measure. But, the initialization does not really matter, because it does not heavily influence the posterior (see section A.1).

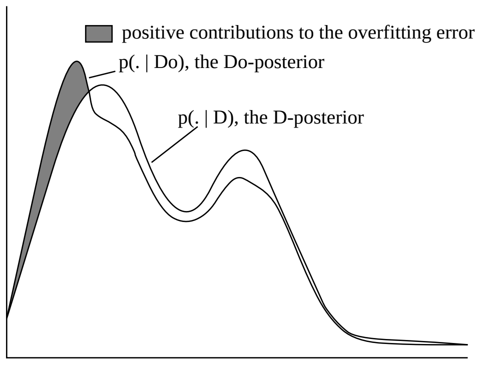
positive contributions to the overfitting error
p(. | Do), the Do-posterior
p(. | D), the D-posterior

Figure 4: *Positive contributions to the overfitting error $E_o(D, D_0)$.*

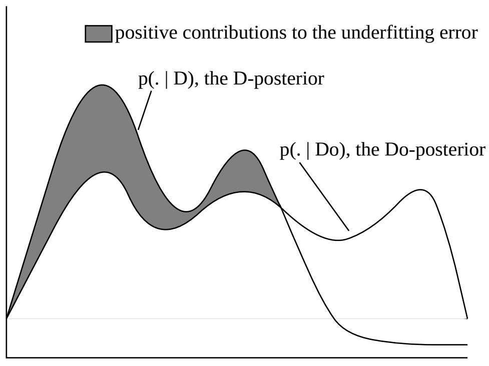
positive contributions to the underfitting error
p(. | D), the D-posterior
p(. | Do), the Do-posterior

Figure 5: *Positive contributions to the underfitting error $E_u(D_0, D)$.*

Formal details. We assume a conventional gradient based learning algorithm A adjusting the weights to locally minimize the error. Then $p_A(\alpha)$ is uniform and $p(\alpha \mid \breve{D}) := p_A(\alpha \mid \breve{D}) = p(\breve{D} \mid \alpha)$. See equation (4). The overfitting error is the Kullback-Leibler distance of the posteriors:

$$E_o(D, D_0) = \int p(\alpha \mid D_0) \log \frac{p(\alpha \mid D_0)}{p(\alpha \mid D)} d\alpha = \int p(\alpha \mid D_0) \log p(D_0 \mid \alpha) d\alpha -$$
$$\int p(\alpha \mid D_0) \log p(D \mid \alpha) d\alpha = \beta \left(\int p(\alpha \mid D_0) E_q(\alpha, D) d\alpha - \bar{E}_q(D_0) \right) , \tag{5}$$

where $\bar{E}_q(D_0) := \int p(\alpha \mid D_0) E_q(\alpha, D_0) d\alpha$ is the mean error after learning D_0, and $E_o(D, D_0)$ is the expectation of $\log \frac{p(\alpha|D_0)}{p(\alpha|D)} = -\log p(\alpha \mid D) - (-\log p(\alpha \mid D_0))$ (the expected difference of the minimal description [42] of α with respect to D and D_0, after learning D_0).

Now we measure the **expected overfitting error relative to** M_w (see section 2) by computing the expectation of $\log \frac{p(\alpha|D_0)}{p(\alpha|D)}$ in the range M_w:

$$E_{ro}(w) = E_{ro}(D, D_0, M_w) = \beta \left(\int_{M_w} p_{M_w}(\alpha \mid D_0) E_q(\alpha, D) d\alpha - \bar{E}_q(D_0, M_w) \right) . \tag{6}$$

Here $p_{M_w}(\alpha \mid D_0) := \frac{p(\alpha|D_0)}{\int_{M_w} p(\tilde{\alpha}|D_0) d\tilde{\alpha}}$ is the posterior of D_0 scaled to obtain a distribution within M_w, and $\bar{E}_q(D_0, M_w) := \int_{M_w} p_{M_w}(\alpha \mid D_0) E_q(\alpha, D_0) d\alpha$ is the mean error in M_w with respect to D_0.

Clearly, we would like to pick w such that $E_{ro}(w)$ is minimized. How to do that? Actually, we will minimize an approximation of $E_{ro}(w)$ (sections A.2.3, A.2.4). Towards this purpose, we need to make two additional *prior assumptions*, which are actually implicit in most previous approaches (which make additional stronger assumptions, see section 5): *(1) "Closeness assumption":* every minimum of $E_q(., D_0)$ is "close" to a maximum of $p(\alpha|D)$ (see formal definition below). Intuitively, "closeness" ensures that D_0 can indeed tell us something about D, such that training on D_0 may indeed reduce the error on D. *(2) "Flatness assumption":* the peaks of $p(\alpha|D)$'s maxima are not sharp. This MDL-like assumption holds if not *all* weights have to be known exactly to model D. It ensures that there are regions with low error on D. Let us have a closer look at both assumptions.

A.2.1. "CLOSENESS" OF D_0 TO D

Intuition. In analogy to section A.2, equation (5), we measure the underfitting error – the information conveyed by D, but not by D_0 (note the exchange of D and D_0 – see figure 5). This information is determined by the D_0-error at the most likelihood weight of D. The concept of underfitting error allows for defining "closeness". Closeness implies that the most likelihood weight of D and the most likelihood weight of D_0 are "close" together in weight space.

Formal details. The underfitting error is

$$E_u(D_0, D) = \int p(\alpha \mid D) \log \frac{p(\alpha \mid D)}{p(\alpha \mid D_0)} d\alpha = \int p(\alpha \mid D) \log p(D \mid \alpha) d\alpha -$$
$$\int p(\alpha \mid D) \log p(D_0 \mid \alpha) d\alpha = \beta \left(\int p(\alpha \mid D) E_q(\alpha, D_0) d\alpha - \bar{E}_q(D) \right) . \tag{7}$$

Here the errors are defined in analogy to the corresponding definitions for the overfitting error, see equation (5). Let $\tilde{w}$ be a most likelihood weight with respect to D. $p(\alpha \mid D)$ has its peak at (or near) $\tilde{w}$. Thus, the underfitting error is dominated by $E_q(\tilde{w}, D_0)$. Let $\hat{w}$ be a most likelihood weight with respect to D_0, and let $U_{E_{tol}}(\hat{w})$ be a connected region around $\hat{w}$, such that $\alpha \in U_{E_{tol}}(\hat{w})$ implies $E_q(\alpha, D_0) < E_{tol}$.

Definition: $\tilde{w}$ is *near* $\hat{w}$ with respect to E_{tol}, iff $\tilde{w} \in U_{E_{tol}}(\hat{w})$. D_0 is **close** to D, iff for all most likelihood weights $\hat{w}$ of D_0, there exists a most likelihood weight $\tilde{w}$ of D such that $\tilde{w}$ is *near* $\hat{w}$.

A.2.2. "FLATNESS" OF D

Intuition. D has a "flat" error surface or a "flat" posterior if (1): at least one weight is not needed to generate D, or (2): there exists a *big* box with respect to D, such that all weight vectors within the box yield about the same error as the most likelihood weight vector of D (which yields minimal error). Hence, there exists a "good" weight vector which can be described with low precision (and few bits).

Formal details. Let $U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$ be a connected region around $\tilde{w}$, where $\alpha \in U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$ implies $E_q(\alpha, D) < E_q(\tilde{w}, D) + \rho$, where ρ is a positive constant near zero.

Definition. D is **flat**, if (1) there exists a $\bar{w} \in U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$, such that there exists at least one w_{ij}, such that for all c with $abs(c)$ below maximal weight value the following holds: $\bar{w} + ce_{ij} \in U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$, where e_{ij} is the unit vector in the direction corresponding to i, j.

Or, D is **flat**, if (2) $V(\bar{\Delta}\tilde{w}) \gg 0$ (i.e. $\tilde{w} = \bar{w}$) or, otherwise, there exists $\bar{w} \in U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$ such that $V(\bar{\Delta}\bar{w}) \gg V(\bar{\Delta}\tilde{w})$. Here $V(\bar{\Delta}w) := max\{2^W \prod_{i,j} \tau_{ij} \mid \forall \kappa_{uv} : 0 \leq \kappa_{uv} \leq \tau_{uv} \leq$ maximal weight value $: w + \sum_{uv} \pm\kappa_{uv}e_{uv} \in U_{E_q(\tilde{w},D)+\rho}(\tilde{w})\}$. $V(\bar{\Delta}w)$ is the volume of the box G_w. Note the similarity to the definition of M_w and $V(\Delta w)$ in section 2: within $U_{E_{tol}}(\hat{w})$, M_w is the biggest box with center w, given D_0. Within $U_{E_q(\tilde{w},D)+\rho}(\tilde{w})$, G_w is the biggest box with center w, given D. Note: (1) implies (2). Therefore, in what follows we focus on (2).

A.2.3. APPROXIMATION AND MINIMUM OF E_{ro}

M_w is defined to have maximal box volume within $U_{E_{tol}}(\hat{w})$. Within the box, the error $E_q(\alpha, D_0)$ is close to E_{tol}. Having found a very flat box (see flatness-condition 2 in A.3), we may assume that $p_{M_w}(\alpha \mid D_0) = \frac{1}{V(\Delta w)}$, i.e. p_{M_w} is uniform. Now we can approximate E_{ro}:

$$E_{ro}(w) \approx \beta \left(\frac{\int_{M_w} E_q(\alpha, D) d\alpha}{V(\Delta w)} - E_{tol} \right) . \tag{8}$$

$\frac{1}{\beta}$, the variance of the target noise, and the tolerable error are fixed parameters of the relative overfitting error. This error is determined by $\frac{\int_{M_w} E_q(\alpha,D)d\alpha}{V(\Delta w)} =: mean(E_q(\alpha, D), M_w)$, which is approximately the mean value of $E_q(\alpha, D)$ within a flat box. Due to flatness, there exists at least one $\bar{w}$ such that $mean(E_q(\alpha, D), G_{\bar{w}}) \approx E_q(\tilde{w}, D)$. Due to $\tilde{w}$ being the most likelihood weight of D, $mean(E_q(\alpha, D), G_w) \geq E_q(\tilde{w}, D)$ for every weight w.

We would like to find some $\bar{w}$ such that the overfitting error is (nearly) minimal both at $\bar{w}$ and on $G_{\bar{w}}$. Near $\tilde{w}$, the vector $\bar{w}$ defines the biggest box $G_{\bar{w}}$, given D. Since only D_0 is available, direct search for $G_{\bar{w}}$ is not possible – all we can do is to look for a big M_w. The problem is: not every w defining a big M_w defines a big G_w as well. To see this, suppose D contains a subset containing no elements of D_0. The next section shows how to deal with this problem.

A.2.4. WHY DOES SEARCH FOR BIG D_0-BOXES WORK?

Outline. If E_{tol} is small enough (this implies "enough" closeness), we *do* have a chance to find a $\bar{w}$ defining a big $G_{\bar{w}}$. Since flatness implies the existence of a big D-box, this subsection justifies our search for big D_0-boxes by showing that a big D-box implies a big D_0-box. Is it possible to end up with a "wrong" big D_0-box (one that is not implied by a big D-box)? Not if there is enough "closeness", as will be seen below.

Formal details. Let's suppose there exists a $\breve{w} \in U_{E_{tol}}(\hat{w})$, such that $V(\Delta\breve{w}) > V(\Delta\bar{w})$ for all $\bar{w}$ (see A.2.2. on flatness), but $mean(E_q(\alpha, D), M_{\breve{w}}) \gg E_q(\tilde{w}, D)$. Then $\breve{w}$ cannot define a big $G_{\breve{w}}$ ($\breve{w}$ defines a "wrong" D_0-box). Now assume that the inner product of $\nabla_w V(\Delta w)|_{w=\hat{w}}$ and the vector pointing to the center of the biggest D_0-box close to $\hat{w}$ is always positive. Then we cannot end up with a $\bar{w}$ as above. Due to flatness, network outputs are about equal on $G_{\bar{w}}$. From this we infer (*): $mean(E_q(\alpha, D_0), M_{\bar{w}}) \approx E_q(\tilde{w}, D_0)$, because of $E_q(\bar{w}, D_0) \approx E_q(\breve{w}, D_0)$.

The validity of our "enough closeness assumption" is equivalent to the validity of the assumption that E_{tol} does go towards $E_q(\hat{w}, D_0)$. The latter implies $E_q(\tilde{w}, D_0) \rightarrow E_q(\hat{w}, D_0)$ (due to closeness relative to E_{tol}). Now let us assume $E_q(\breve{w}, D_0) > E_q(\hat{w}, D_0)$ (this holds if $|D_0| > W$). This implies that $V(\Delta\breve{w})$ decreases if E_{tol} decreases. From (*) (see last paragraph) we deduce that $V(\Delta\bar{w})$ does not decrease. Hence, $V(\Delta\breve{w}) < V(\Delta\bar{w})$ for at least one $\bar{w}$. Alas, we can find a $\bar{w}$ by a gradient based method for searching big box volumes, provided there is enough closeness (E_{tol} is small enough). Now we see: **maximizing $V(\Delta w)$ (thus minimizing E_{ro}) is equivalent to a search for acceptable minima with "flat" network output** (the error depends on the output only). The next subsection shows how to find networks with flat output.

A.3. HOW TO FLATTEN THE NETWORK OUTPUT

Outline. To find nets with flat outputs, two conditions will be defined to specify $B(w, D_0)$ (see section 3). The first condition ensures flatness. The second condition enforces "equal flatness" in all weight space directions, and can be justified using an MDL-based argument. In both cases, linear approximations will be made (to be justified in A.4).

Formal details. We are looking for weights (causing tolerable error) that can be perturbed without causing significant output changes. Perturbing the weights w by δw (with components δw_{ij}), we obtain $ED(w, \delta w) := \sum_k (o^k(w + \delta w) - o^k(w))^2$, where $o^k(w)$ expresses o^k's dependence on w (in what follows, however, w often will be suppressed for convenience). Linear approximation (justified in A.4) gives us **"Flatness Condition 1":**

$$ED(w, \delta w) \approx ED_l(\delta w) := \sum_k (\sum_{i,j} \frac{\partial o^k}{\partial w_{ij}} \delta w_{ij})^2 \leq ED_{l,max}(\delta w) := \sum_k (\sum_{i,j} |\frac{\partial o^k}{\partial w_{ij}}||\delta w_{ij}|)^2 \leq \epsilon, \quad (9)$$

where $\epsilon > 0$ defines tolerable output changes within a box and is small enough to allow for linear approximation (it does not appear in $B(w, D_0)$'s gradient, see section 3). ED_l is the linear approximation of ED, and $ED_{l,max}$ is $max\{ED_l(w, \delta v) | \forall_{ij} : \delta v_{ij} = \pm \delta w_{ij}\}$. Flatness condition 1 is a "robustness condition" (or "fault tolerance condition", or "perturbation tolerance condition" – see e.g. [25, 32, 33, 24, 4, 17, 5]).

The validity of approximation (8) depends on how flat box M_w is. Many M_w satisfy flatness condition 1. To select a particular, very flat M_w, the following **"Flatness Condition 2"** uses up degrees of freedom left by equation (9):

$$\forall i, j, u, v : (\delta w_{ij})^2 \sum_k (\frac{\partial o^k}{\partial w_{ij}})^2 = (\delta w_{uv})^2 \sum_k (\frac{\partial o^k}{\partial w_{uv}})^2 \ . \quad (10)$$

Flatness condition 2 enforces equal "directed errors"
$ED_{ij}(w, \delta w_{ij}) = \sum_k (o^k(w_{ij} + \delta w_{ij}) - o^k(w_{ij}))^2 \approx \sum_k (\frac{\partial o^k}{\partial w_{ij}} \delta w_{ij})^2$, where $o^k(w_{ij})$ has the obvious meaning. Linear approximation is justified by the choice of ϵ in equation (9).

How to derive the algorithm from flatness conditions 1 and 2. We first solve equation (10) for $|\delta w_{ij}| = |\delta w_{uv}| \sqrt{\frac{\sum_k \left(\frac{\partial o^k}{\partial w_{uv}}\right)^2}{\sum_k \left(\frac{\partial o^k}{\partial w_{ij}}\right)^2}}$ (fixing u, v for all i, j). Then we insert $|\delta w_{ij}|$ into equation (9) (replacing the second "$\leq$" in (9) by "$=$"). This gives us an equation for the $|\delta w_{ij}|$ (which depend on w, but this is notationally suppressed):

$$|\delta w_{ij}| = \frac{\sqrt{\epsilon}}{\sqrt{\sum_k (\frac{\partial o^k}{\partial w_{ij}})^2} \sqrt{\sum_k \left(\sum_{i,j} \frac{|\frac{\partial o^k}{\partial w_{ij}}|}{\sqrt{\sum_k (\frac{\partial o^k}{\partial w_{ij}})^2}} \right)^2}} \ . \quad (11)$$

The $|\delta w_{ij}|$ approximate the Δw_{ij} from section 2. The box M_w is approximated by AM_w, the box with center w and edge length $2\delta w_{ij}$. M_w's volume $V(\Delta w)$ is approximated by AM_w's box volume $V(\delta w) := 2^W \prod_{ij} |\delta w_{ij}|$. Thus, $\tilde{B}(w, D_0)$ (see section 3) can be approximated by

$B(w, D_0) := -\log \frac{1}{2^W} V(\delta w) = \sum_{i,j} - \log |\delta w_{ij}|$. This immediately leads to the algorithm given by equation (1).

How can this approximation be justified? **The learning process itself enforces its validity (see A.4).** Initially, the conditions above are valid only in a very small environment of an "initial" acceptable minimum. But during search for new acceptable minima with more associated box volume, the corresponding environments are enlarged. The next section will prove this for feedforward nets (experiments indicate that this appears to be true for recurrent nets as well).

MDL-based justification of flatness condition 2. Let us assume a sender wants to send a description of some box center w to a receiver who knows the inputs but not the targets. The MDL principle suggests that the sender wants to minimize w's expected description length. Let $ED_{mean}(w, D_0)$ denote the mean value of ED on the box. Expected description length is approximated by $ED_{mean}(w, D_0) + B(w, D_0)$. One way of seeing this is to apply Hinton and van Camp's "bits back" argument [14] to a uniform prior (ED_{mean} corresponds to Hinton and vanCamp's output variance). However, we prefer to use a different argument: we encode each weight w_{ij} of the box center w by a bitstring according to the following procedure (Δw_{ij} is given):

```
(0) Define a variable interval $I_{ij} \subset R$.
(1) Make $I_{ij}$ equal to the interval constraining possible weight values.
(2) While $I_{ij} \not\subset [w_{ij} - \Delta w_{ij}, w_{ij} + \Delta w_{ij}]$:
        Divide $I_{ij}$ into 2 equally-sized disjunct intervals $I_1$ and $I_2$.
        If $w_{ij} \in I_1$ then $I_{ij} \leftarrow I_1$; write '1'.
        If $w_{ij} \in I_2$ then $I_{ij} \leftarrow I_2$; write '0'.
```

The final set $\{I_{ij}\}$ corresponds to a "bit-box" within our box. This "bit-box" contains M_w's center w and is described by a bitstring of length $\tilde{B}(w, D_0) + c$, where the constant c is independent of the box M_w. From $ED(w, w_b - w)$ (w_b is the center of the "bit-box") and the bitstring describing the "bit-box", the receiver can compute w as follows: he chooses an initialization weight vector within the "bit-box" and uses gradient descent to decrease $B(w_a, D_0)$ until $ED(w_b, w_a - w_b) = ED(w, w_b - w)$, where w_a in the bit-box denotes the receiver's current approximation of w (w_a is constantly updated by the receiver). This is like "FMS without targets" – recall that the receiver knows the inputs x_p of D_0. Since w corresponds to the weight vector with the highest degree of local flatness within the "bit-box", the receiver will find the correct w.

$ED(w, w_b - w)$ is described by a Gaussian distribution with mean zero. Hence, the description length of $ED(w, w_b - w)$ is $ED(w, w_b - w)$. w_b, the center of the "bit-box", cannot be known before training. However, we do know the *expected* description length of the box center, which is $ED_{mean} + \tilde{B}(w, D_0) + k$ (k is a constant independent of w). Let us approximate ED_{mean}:
$ED_{l,mean}(w, \delta w) := \frac{1}{V(\delta w)} \int_{AM_w} ED_l(w, \delta v) d\delta v =$
$\frac{1}{V(\delta w)} 2^W \frac{1}{3} \sum_{i,j} \left((\delta w_{ij})^3 \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2 \prod_{u,v \text{ with } u,v \neq i,j} \delta w_{uv} \right) = \frac{1}{3} \sum_{i,j} (\delta w_{ij})^2 \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2.$

Among those w that lead to equal $B(w, D_0)$ (the negative logarithm of the box volume plus $W \log 2$), we want to find those with minimal description length. Using Lagrange multiplicators, **it can be shown that $ED_{l,mean}$ is minimal under the condition $B(w, D_0) = \mathit{constant}$ iff flatness condition 2 holds**. To conclude: with given box volume, we need flatness condition 2 to minimize the expected description length of the box center.

Comments. Flatness condition 2 influences the algorithm as follows: (1) The algorithm prefers to increase the δw_{ij}'s of weights which currently are not important to generate the target output. (2) The algorithm enforces equal sensitivity of all output units with respect to the weights. Hence, if certain hidden units help to compute targets only for a subset of the output units, then the sensitivity of the remaining output units with respect to these hidden units will be reduced: the algorithm tends to group hidden units according to their relevance for groups of output units. Flatness condition 2 is essential: flatness condition 1 by itself corresponds to nothing more but first order derivative reduction (ordinary sensitivity reduction).

Automatically, the algorithm treats units and weights in different layers differently, and takes the nature of the activation functions into account.

A.4. WHY DOES THE HESSIAN DECREASE?

Outline. This section shows that second order derivatives of the output function vanish during flat minimum search. This justifies the linear approximations in A.3 and approximation (8) of E_{ro} in A.3.

Intuition. We show that the algorithm tends to suppress the following values: (1) unit activations, (2) first order activation derivatives, (3) the sum of all contributions of an arbitary unit activation to the net output. Since weights, inputs, activation functions, and their first and second order derivatives are bounded, the entries in the Hessian decrease where the corresponding $|\delta w_{ij}|$ increase.

Formal details. For simplicity, in what follows we use the same activation function f for all units. We consider a strictly layered feedforward network with K output units and g layers. We obtain

$$\frac{\partial y^l}{\partial w_{ij}} = f'(s_l) \left\{ \begin{array}{cc} y^j & \text{for } i = l \\ \sum_m w_{lm} \frac{\partial y^m}{w_{ij}} & \text{for } i \neq l \end{array} \right\} , \qquad (12)$$

where y^a denotes the activation of the a-th unit, and $s_l = \sum_m w_{lm} y^m$.

The last term of equation (1) (the "regulator") expresses output sensitivity (to be minimized) with respect to simultaneous perturbations of all weights. "Regulation" is done by equalizing the sensitivity of the output units with respect to the weights. The "regulator" does not influence the same particular units or weights for each training example. It may be ignored for the purposes of this section. Of course, the same holds for the first (constant) term in (1). We are left with the second term. With (12) we obtain:

$$\begin{aligned}
\sum_{i,j} \log \sum_k (\frac{\partial o^k}{\partial w_{ij}})^2 = \\
2 \sum_{\text{unit } k \text{ in the } g \text{ th layer}} (\text{fan-in of unit } k) \log |f'(s_k)| + \\
2 \sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| + \\
\sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-in of unit } j) \log \sum_k (f'(s_k) w_{kj})^2 + \\
2 \sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-in of unit } j) \log |f'(s_j)| + \\
2 \sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| + \\
\sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-in of unit } j) \log \sum_k \left(f'(s_k) \sum_l f'(s_l) w_{kl} w_{lj} \right)^2 + \\
2 \sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-in of unit } j) \log |f'(s_j)| + \\
2 \sum_{\text{unit } j \text{ in the } (g-3)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| + \\
\sum_{i,j, \text{ where unit } i \text{ in a layer } <(g-2)} \log \sum_k \left(f'(s_k) \sum_{l_1} f'(s_{l_1}) w_{kl_1} \sum_{l_2} w_{l_1 l_2} \frac{\partial y^{l_2}}{\partial w_{ij}} \right)^2
\end{aligned} \qquad (13)$$

Let us have a closer look at this equation. We observe:

(1) Activations of units decrease in proportion to their fan-outs.

(2) First order derivatives of the activation functions decrease in proportion to their fan-ins.

(3) A term of the form $\sum_k \left(f'(s_k)\sum_{l_1} f'(s_{l_1})w_{kl_1}\sum_{l_2}\cdots\sum_{l_r} f'(s_{l_r})w_{l_{r-1}l_r}w_{l_rj}\right)^2$ expresses the sum of unit j's squared contributions to the net output. Here r ranges over $\{0,1,\ldots,g-2\}$, and unit j is in the $(g-1-r)$th layer (for the special case $r=0$, we get $\sum_k \left(f'(s_k)w_{kj}\right)^2$). These terms also decrease in proportion to unit j's fan-in. Analogously, equation (13) can be extended to the case of additional layers.

Comment. Let us assume that $f'(s_j)=0$ and $f(s_j)=0$ is "difficult to achieve" (can be achieved only by fine-tuning all weights on connections to unit j). Instead of minimizing $|f(s_j)|$ or $|f'(s_j)|$ by adjusting the net input of unit j (this requires fine-tuning of many weights), our algorithm prefers pushing weights w_{kl} on connections to output units towards zero (other weights are less affected). On the other hand, if $f'(s_j)=0$ and $f(s_j)=0$ is **not** "difficult to achieve", then, **unlike weight decay, our algorithm does not necessarily prefer weights close to zero.** Instead, it prefers (possibly very strong) weights which push $f(s_j)$ or $f'(s_j)$ towards zero (e.g. with sigmoid units active in [0,1]: strong inhibitory weights are preferred; with Gaussian units: high absolute weight values are preferred). See the experiment in section 4.2.

How does this influence the Hessian? The entries in the Hessian corresponding to output o^k can be written as follows:

$$\frac{\partial^2 o^k}{\partial w_{ij}\partial w_{uv}} = \frac{f''(s_k)}{(f'(s_k))^2}\frac{\partial o^k}{\partial w_{ij}}\frac{\partial o^k}{\partial w_{uv}} + f'(s_k)\left(\sum_l w_{kl}\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}} + \bar{\delta}_{ik}\frac{\partial y^j}{\partial w_{uv}} + \bar{\delta}_{uk}\frac{\partial y^v}{\partial w_{ij}}\right) , \quad (14)$$

where $\bar{\delta}$ is the Kronecker-Delta. Searching for big boxes, we run into regions of acceptable minima with o^k's close to target (section 2). Thus, by scaling the targets, $\frac{f''(s_k)}{(f'(s_k))^2}$ can be bounded. Therefore, the first term in equation (14) decreases during learning.

According to the analysis above, the first order derivatives in the second term of (14) are pushed towards zero. So are the w_{kl} of the sum in the second term of (14).

The only remaining expressions of interest are second order derivatives of units in layer $(g-1)$. The $\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}}$ are bounded if (a) the weights, (b) the activation functions, (c) their first and second order derivatives, and (d) the inputs are bounded. This is indeed the case, as will be shown for networks with one or two hidden layers:

Case 1: For unit l in a single hidden layer ($g=3$), we obtain

$$\left|\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}}\right| = |\bar{\delta}_{li}\bar{\delta}_{lu}f''(s_l)y^j y^v| < C_1 , \quad (15)$$

where y^j, y^v are the components of an input vector x_p, and C_1 is a positive constant.

Case 2: For unit l in the third layer of a net with 2 hidden layers ($g=4$), we obtain

$$\left|\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}}\right| = |f''(s_l)(w_{li}y^j + \bar{\delta}_{il}y^j)(w_{lu}y^v + \bar{\delta}_{ul}y^v) +$$
$$f'(s_l)\left(w_{li}\bar{\delta}_{iu}f''(s_i)y^j y^v + \bar{\delta}_{il}\bar{\delta}_{uj}f'(s_j)y^v + \bar{\delta}_{ul}\bar{\delta}_{iv}f'(s_v)y^j\right)| < C_2 , \quad (16)$$

where C_2 is a positive constant. Analoguously, the boundedness of second order derivatives can be shown for additional hidden layers.

Conclusion: As desired, our algorithm makes the $H^k_{ij,uv}$ decrease where $|\delta w_{ij}|$ or $|\delta w_{uv}|$ increase.

A.5. EFFICIENT IMPLEMENTATION OF THE ALGORITHM

Outline. We first explicitly compute the derivatives of (1). Then we show how to use Pearlmutter and Møller's algorithm [26, 35] to speed up the computation of second order terms (A.5.2).

A.5.1 EXPLICIT DERIVATIVE OF EQUATION (1)

The derivative of the right-hand side of (1) is:

$$\frac{\partial B(w,D_0)}{\partial w_{uv}} = \sum_{i,j} \frac{\sum_k \frac{\partial o^k}{\partial w_{ij}} \frac{\partial^2 o^k}{\partial w_{ij} \partial w_{uv}}}{\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2} +$$
$$W \frac{\sum_k \left(\sum_{i,j} \frac{|\frac{\partial o^k}{\partial w_{ij}}|}{\sqrt{\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2}} \sum_{i,j} \left(\operatorname{sign}(\frac{\partial o^k}{\partial w_{ij}}) \frac{\frac{\partial^2 o^k}{\partial w_{ij} \partial w_{uv}} \sum_m (\frac{\partial o^m}{\partial w_{ij}})^2 - \frac{\partial o^k}{\partial w_{ij}} \sum_m \frac{\partial o^m}{\partial w_{ij}} \frac{\partial^2 o^m}{\partial w_{ij} \partial w_{uv}}}{(\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2)^{\frac{3}{2}}} \right) \right)}{\left(\sum_k \left(\sum_{i,j} \frac{|\frac{\partial o^k}{\partial w_{ij}}|}{\sqrt{\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2}} \right) \right)^2} . \quad (17)$$

To compute (2), we need

$$\frac{\partial B(w,D_0)}{\partial (\frac{\partial o^k}{\partial w_{ij}})} = \frac{\frac{\partial o^k}{\partial w_{ij}}}{\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2} +$$
$$W \frac{\sum_m \left(\sum_{l,r} \left(\frac{|\frac{\partial o^m}{\partial w_{lr}}|}{\sqrt{\sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{lr}})^2}} \right) \operatorname{sign}(\frac{\partial o^m}{\partial w_{ij}}) \frac{\bar{\delta}_{mk} \sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{ij}})^2 - \frac{\partial o^m}{\partial w_{ij}} \frac{\partial o^k}{\partial w_{ij}}}{(\sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{ij}})^2)^{\frac{3}{2}}} \right)}{\left(\sum_m \left(\sum_{l,r} \frac{|\frac{\partial o^m}{\partial w_{lr}}|}{\sqrt{\sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{lr}})^2}} \right) \right)^2} , \quad (18)$$

where $\bar{\delta}$ is the Kronecker-Delta. Using the nabla operator and (18), we can compress (17):

$$\nabla_{uv} B(w, D_0) = \sum_k H^k (\nabla_{\frac{\partial o^k}{\partial w_{ij}}} B(w, D_0)) , \quad (19)$$

where H^k is the Hessian of the output o^k. Since the sums over l, r in (18) need to be computed only once (the results are reusable for all i, j), $\nabla_{\frac{\partial o^k}{\partial w_{ij}}} B(w, D_0)$ can be computed in $O(W)$ time. The product of the Hessian and a vector can be computed in $O(W)$ time (see next section). With constant number of output units, **the computational complexity of our algorithm is** $O(W)$.

A.5.2. FAST MULTIPLICATION BY THE HESSIAN

Pearlmutter and Møller compute the product of a vector and the Hessian of the error in $O(W)$ time [26, 35]. Using Pearlmutter's notation, we do the same with the Hessian of the output. An operator R is defined as follows:

$$R_y\{g(x)\} \equiv \frac{\partial}{\partial t} g(x + ty) \mid_{t=0} . \quad (20)$$

The Hessian of the kth output o^k of a feedforward net is computed in 3 successive passes:

1. First backward pass ($y^l = o^k$):

$$\frac{\partial y^l}{\partial y^i} = \left\{ \begin{array}{cc} 1 & \text{for } i = l \\ \sum_j w_{ji} \frac{\partial y^l}{\partial s_j} & \text{for } i \neq l \end{array} \right\} , \quad (21)$$

$$\frac{\partial y^l}{\partial s_i} = f_i'(s_i)\frac{\partial y^l}{\partial y^i} \ , \tag{22}$$

$$\frac{\partial y^l}{\partial w_{ji}} = y^i \frac{\partial y^l}{\partial s_j} \ . \tag{23}$$

2. First forward pass:

$$R\{s_i\} = \sum_j (w_{ij} R\{y^j\} + \frac{\partial B(w, D_0)}{\partial(\frac{\partial o^k}{\partial w_{ij}})} y^j) \ , \tag{24}$$

$$R\{y^i\} = \left\{ \begin{array}{cc} 0 & \text{for } y^i \text{ input} \\ R\{s_i\} f_i'(s_i) & \text{otherwise} \end{array} \right\} \ . \tag{25}$$

3. Second backward pass ($y^l = o^k$):

$$R\{\frac{\partial y^l}{\partial y^i}\} = \left\{ \begin{array}{cc} 0 & \text{for } y^i \text{ in layers not below } y^l \\ \sum_j \left(w_{ji} R\{\frac{\partial y^l}{\partial s_j}\} + \frac{\partial B(w,D_0)}{\partial(\frac{\partial o^k}{\partial w_{ji}})} \frac{\partial y^l}{\partial s_j} \right) & \text{for } y^i \text{ in layers below } y^l \end{array} \right\} \ , \tag{26}$$

$$R\{\frac{\partial y^l}{\partial s_i}\} = f_i'(s_i) R\{\frac{\partial y^l}{\partial y^i}\} + R\{s_i\} f_i''(s_i) \frac{\partial y^l}{\partial y^i} \ , \tag{27}$$

$$R\{\frac{\partial y^l}{\partial w_{ji}}\} = y^i R\{\frac{\partial y^l}{\partial s_j}\} + R\{y^i\} \frac{\partial y^l}{\partial s_j} \ . \tag{28}$$

The elements of the vector $H^k(\nabla_{\frac{\partial o^k}{\partial w_{ij}}} B(w, D_0))$ are $R\{\frac{\partial o^k}{\partial w_{ji}}\}$; see equation (19). Using the technique in [35], recurrent networks can be dealt with as well.

References

[1] H. Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203–217, 1970.

[2] S. Amari and N. Murata. Statistical theory of learning curves under entropic loss criterion. *Neural Computation*, 5(1):140–153, 1993.

[3] T. Ash. Dynamic node creation in backpropagation neural networks. *Connection Science*, 1(4):365–375, 1989.

[4] C. Bishop. Curvature-driven smoothing in backpropagation neural networks. In *Proceedings of the International Joint Conference on Neural Networks IJCNN-90*, volume 2, pages 749–752. Lawrence Erlbaum, Hillsdale, N.J., 1990.

[5] M. J. Carter, F. J. Rudolph, and A. J. Nucci. Operational fault tolerance of CMAC networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 340–347. San Mateo, CA: Morgan Kaufmann, 1990.

[6] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403, 1979.

[7] R. L. Eubank. Spline smoothing and nonparametric regression. In S. Farlow, editor, *Self-Organizing Methods in Modeling*. Marcel Dekker, New York, 1988.

[8] S. E. Fahlman and C. Lebiere. The cascade-correlation learning algorithm. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 525–532. San Mateo, CA: Morgan Kaufmann, 1990.

[9] G. Golub, H. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–224, 1979.

[10] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 471–479. San Mateo, CA: Morgan Kaufmann, 1992.

[11] S. J. Hanson and L. Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 177–185. San Mateo, CA: Morgan Kaufmann, 1989.

[12] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. San Mateo, CA: Morgan Kaufmann, 1993.

[13] T. J. Hastie and R. J. Tibshirani. Generalized additive models. *Monographs on Statisics and Applied Probability*, 43, 1990.

[14] G. E. Hinton and D. van Camp. Keeping neural networks simple. In *Proceedings of the International Conference on Artificial Neural Networks, Amsterdam*, pages 11–18. Springer, 1993.

[15] S. Hochreiter and J. Schmidhuber. Simplifying nets by discovering flat minima. In *Advances in Neural Information Processing Systems 7*. San Mateo, CA: Morgan Kaufmann, 1995. To appear.

[16] S. B. Holden. *On the Theory of Generalization and Self-Structuring in Linearly Weighted Connectionist Networks*. PhD thesis, Cambridge University, Engineering Department, 1994.

[17] P. Kerlirzin and F. Vallet. Robustness in multilayer perceptrons. *Neural Computation*, 5(1):473–482, 1993.

[18] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. San Mateo, CA: Morgan Kaufmann, 1992.

[19] S. Kullback. *Statistics and Information Theory*. J. Wiley and Sons, New York, 1959.

[20] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. San Mateo, CA: Morgan Kaufmann, 1990.

[21] A. U. Levin, T. K. Leen, and J. E. Moody. Fast pruning using principal components. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 35–42. Morgan Kaufmann, San Mateo, CA, 1994.

[22] E. Levin, N. Tishby, and S. Solla. A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE*, 78(10):1568–1574, 1990.

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

[24] K. Matsuoka. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440, 1992.

[25] A. A. Minai and R. D. Williams. Perturbation response in feedforward networks. *Neural Networks*, 7(5):783–796, 1994.

[26] M. F. Møller. Exact calculation of the product of the Hessian matrix of feed-forward network error functions and a vector in O(N) time. Technical Report PB-432, Computer Science Department, Aarhus University, Denmark, 1993.

[27] J. E. Moody. Fast learning in multi-resolution hierarchies. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*. San Mateo, CA: Morgan Kaufmann, 1989.

[28] J. E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 847–854. San Mateo, CA: Morgan Kaufmann, 1992.

[29] J. E. Moody and J. Utans. Architecture selection strategies for neural networks: Application to corporate bond rating prediction. In A. N. Refenes, editor, *Neural Networks in the Capital Markets*. John Wiley & Sons, 1994.

[30] F. Mosteller and J. W. Tukey. Data analysis, including statistics. In G. Lindzey and E. Aronson, editors, *Handbook of Social Psychology, Vol. 2*. Addison-Wesley, 1968.

[31] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 107–115. San Mateo, CA: Morgan Kaufmann, 1989.

[32] A. F. Murray and P. J. Edwards. Synaptic weight noise during MLP learning enhances fault-tolerance, generalisation and learning trajectory. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 491–498. San Mateo, CA: Morgan Kaufmann, 1993.

[33] C. Neti, M. H. Schneider, and E. D. Young. Maximally fault tolerant neural networks. In *IEEE Transactions on Neural Networks*, volume 3, pages 14–23, 1992.

[34] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight sharing. *Neural Computation*, 4:173–193, 1992.

[35] B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 1994.

[36] B. A. Pearlmutter and R. Rosenfeld. Chaitin-Kolmogorov complexity and generalization in neural networks. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 925–931. San Mateo, CA: Morgan Kaufmann, 1991.

[37] A. N. Refenes, G. Francis, and A. D. Zapranis. Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks*, 1994.

[38] H. Rehkugler and T. Poddig. Statistische Methoden versus Künstliche Neuronale Netzwerke zur Aktienkursprognose. Technical Report 73, University Bamberg, Fakultät Sozial- und Wirtschaftswissenschaften, 1990.

[39] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[40] J. H. Schmidhuber. Discovering problem solutions with low Kolmogorov complexity and high generalization capability. Technical Report FKI-194-94, Fakultät für Informatik, Technische Universität München, 1994.

[41] J. H. Schmidhuber. On learning how to learn learning strategies. Technical Report FKI-198-94, Fakultät für Informatik, Technische Universität München, November 1994.

[42] C. E. Shannon. A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423, 1948.

[43] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Roy. Stat. Soc.*, 36:111–147, 1974.

[44] V. Vapnik. Principles of risk minimization for learning theory. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 831–838. San Mateo, CA: Morgan Kaufmann, 1992.

[45] C. S. Wallace and D. M. Boulton. An information theoretic measure for classification. *Computer Journal*, 11(2):185–194, 1968.

[46] C. Wang, S. S. Venkatesh, and J. S. Judd. Optimal stopping and effective machine complexity in learning. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 303–310. Morgan Kaufmann, San Mateo, CA, 1994.

[47] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 875–882. San Mateo, CA: Morgan Kaufmann, 1991.

[48] H. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[49] P. M. Williams. Bayesian regularisation and pruning using a Laplace prior. Technical report, School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton, 1994.

[50] D. H. Wolpert. Bayesian backpropagation over i-o functions rather than weights. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 200–207. San Mateo, CA: Morgan Kaufmann, 1994.

[51] D. H. Wolpert. The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. Technical Report SFI-TR-03-123, Santa Fe Institute, NM 87501, 1994.

[52] R. S. Zemel. *A minimum description length framework for unsupervised learning.* PhD thesis, University of Toronto, 1993.