Communicated by Christopher Bishop

Flat Minima

Sepp Hochreiter
Fakultät für Informatik, Technische Universität München, 80290 München, Germany

Jürgen Schmidhuber
IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland

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, Bayesian argument suggests that flat minima correspond to "simple" networks and low expected overfitting. The argument is based on a Gibbs algorithm variant and a novel way of splitting generalization error into underfitting and overfitting error. Unlike many previous approaches, ours does not require gaussian assumptions and does not depend on a "good" weight prior. Instead we have a prior over input-output functions, thus taking into account net architecture and training set. Although our algorithm requires the computation of second-order derivatives, it has backpropagation'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 conventional backprop, weight decay, and "optimal brain surgeon/optimal brain damage."

1 Basic Ideas and Outline

Our algorithm tries to find a large region in weight space with the property that each weight vector from that region leads to similar small error. Such a region is called a flat minimum (Hochreiter and Schmidhuber 1995). To get an intuitive feeling for why a flat minimum is interesting, consider this: A sharp minimum (see Fig. 2) corresponds to weights that have to be specified with high precision. A flat minimum (see Fig. 1) corresponds to weights, many of which can be given with low precision. In the terminology of the theory of minimum description (message) length (MML, Wallace and Boulton 1968; MDL, Rissanen 1978), 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. Similarly, the standard Bayesian

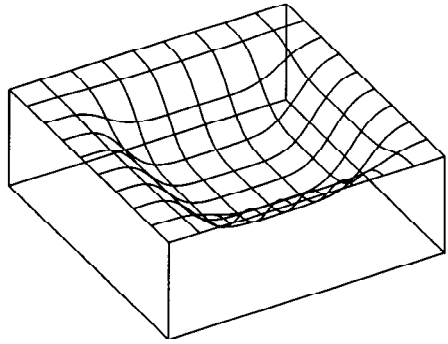

Figure 1: Example of a flat minimum.

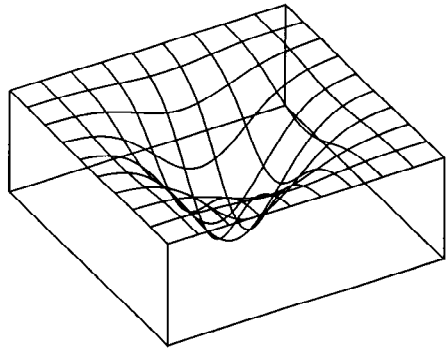

Figure 2: Example of a sharp minimum.

view favors "fat" maxima of the posterior weight distribution (maxima with a lot of probability mass; see, e.g., Buntine and Weigend 1991). We will see that flat minima are fat maxima.

Unlike, e.g., Hinton and van Camp's method (1993), our algorithm does not depend on the choice of a "good" weight 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 an efficient second-order method (Pearlmutter 1994; Møller 1993), we obtain conventional backpropagation'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 simple weight decay, our method automatically treats and prunes units and weights in different layers in different reasonable ways.

The article is organized as follows:

- Section 2 formally introduces basic concepts, such as error measures and flat minima.
- Section 3 describes the novel algorithm, flat minimum search (FMS).
- Section 4 formally derives the algorithm.
- Section 5 reports experimental generalization results with feedforward and recurrent networks. For instance, in an application to stock market prediction, flat minimum search outperforms the following widely used competitors: conventional backpropagation, weight decay, and "optimal brain surgeon / optimal brain damage."
- Section 6 mentions relations to previous work.
- Section 7 mentions limitations of the algorithm and outlines future work.
- The Appendix presents a detailed theoretical justification of our approach. Using a variant of the Gibbs algorithm, Section A.1 defines generalization, underfitting and overfitting error in a novel way. By defining an appropriate prior over input-output functions, we postulate that the most probable network is a flat one.
  Section A.2 formally justifies the error function minimized by our algorithm.
  Section A.3 shows how to compute derivatives required by the algorithm.

2 Task Architecture and Boxes

2.1 Generalization Task. The task is to approximate an unknown function $f \subset X \times Y$ mapping a finite set of possible inputs $X \subset R^N$ to a finite set of possible outputs $Y \subset R^K$. A data set D is obtained from f (see Section A.1). All training information is given by a finite set $D_0 \subset D$. D_0 is called the training set. The pth element of D_0 is denoted by an input-target pair (x_p, y_p).

2.2 Architecture and Net Functions. 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, L weights, and differentiable activation functions. It maps input vectors $x \in R^N$ to output vectors $o(w, x) \in R^K$, where w is the L-dimensional weight vector, and the weight on the connection from unit j to i is denoted w_{ij}. The net func-

tion induced by w is denoted $net(w)$: for $x \in R^N$, $net(w)(x) = o(w,x) = \left(o^1(w,x), o^2(w,x), \ldots, o^{K-1}(w,x), o^K(w,x)\right)$, where $o^i(w,x)$ denotes the ith component of $o(w,x)$, corresponding to output unit i.

2.3 Training Error. We use squared error $E(net(w), D_0) := \sum_{(x_p,y_p)\in D_0} \|y_p - o(w,x_p)\|^2$, where $\|\cdot\|$ denotes the Euclidean norm.

2.4 Tolerable Error. To define a region in weight space with the property that each weight vector from that region leads to small error and similar output, we introduce the tolerable error E_{tol}, a positive constant (see Section A.1 for a formal definition of E_{tol}). "Small" error is defined as being smaller than E_{tol}. $E(net(w), D_0) > E_{tol}$ implies underfitting.

2.5 Boxes. Each weight w satisfying $E(net(w), D_0) \le E_{tol}$ defines an acceptable minimum (compare $M(D_0)$ in Section A.1). We are interested in a large region of connected acceptable minima, where each weight w within this region leads to almost identical net functions $net(w)$. Such a region is called a flat minimum. We will see that flat minima correspond to low expected generalization error. 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 an L-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 the direction of the axis corresponding to weight w_{ij} is denoted by $\Delta w_{ij}(X)$. The $\Delta w_{ij}(X)$ are the maximal (positive) values such that for all L-dimensional vectors κ whose components κ_{ij} are restricted by $|\kappa_{ij}| \le \Delta w_{ij}(X)$, we have: $E(net(w), net(w+\kappa), X) \le \epsilon$, where $E(net(w), net(w+\kappa), X) = \sum_{x\in X} \|o(w,x) - o(w+\kappa, x)\|^2$, and ϵ is a small positive constant defining tolerable output changes (see also equation 3.1). Note that $\Delta w_{ij}(X)$ depends on ϵ. Since our algorithm does not use ϵ, however, it is notationally suppressed. $\Delta w_{ij}(X)$ gives the precision of w_{ij}. M_w's box volume is defined by $V(\Delta w(X)) := 2^L \prod_{i,j} \Delta w_{ij}(X)$, where $\Delta w(X)$ denotes the vector with components $\Delta w_{ij}(X)$. Our goal is to find large boxes within flat minima.

3 The Algorithm

Let $X_0 = \{x_p \mid (x_p, y_p) \in D_0\}$ denote the inputs of the training set. We approximate $\Delta w(X)$ by $\Delta w(X_0)$, where $\Delta w(X_0)$ is defined like $\Delta w(X)$ in Section 2.5 (replacing X by X_0). For simplicity, we will abbreviate $\Delta w(X_0)$ as Δw. Starting with a random initial weight vector, flat minimum search (FMS) tries to find a w that not only has low $E(net(w), D_0)$ but also defines a box M_w with maximal box volume $V(\Delta w)$ and, consequently, minimal $\tilde{B}(w, X_0) := -\log(\frac{1}{2^L} V(\Delta w)) = \sum_{i,j} -\log \Delta w_{ij}$. Note the relationship to

MDL: $\tilde{B}$ is the number of bits required to describe the weights, whereas the number of bits needed to describe the y_p, given w (with $(x_p, y_p) \in D_0$), can be bounded by fixing E_{tol} (see Section A.1). In the next section we derive the following algorithm. We use gradient descent to minimize $E(w, D_0) = E(net(w), D_0) + \lambda B(w, X_0)$, where $B(w, X_0) = \sum_{x_p \in X_0} B(w, x_p)$, and

$$B(w, x_p) = \frac{1}{2}\left(-L \log \epsilon + \sum_{i,j} \log \sum_k \left(\frac{\partial o^k(w, x_p)}{\partial w_{ij}}\right)^2 + L \log \sum_k \left(\sum_{i,j} \frac{\left|\frac{\partial o^k(w, x_p)}{\partial w_{ij}}\right|}{\sqrt{\sum_k \left(\frac{\partial o^k(w, x_p)}{\partial w_{ij}}\right)^2}}\right)^2\right). \tag{3.1}$$

Here $o^k(w, x_p)$ is the activation of the kth output unit (given weight vector w and input x_p), ϵ is a constant, and λ is the regularization constant (or hyperparameter) that controls the trade-off between regularization and training error (see Section A.1). To minimize $B(w, X_0)$, for each $x_p \in X_0$ we have to compute

$$\frac{\partial B(w, x_p)}{\partial w_{uv}} = \sum_{k,i,j} \frac{\partial B(w, x_p)}{\partial \left(\frac{\partial o^k(w, x_p)}{\partial w_{ij}}\right)} \frac{\partial^2 o^k(w, x_p)}{\partial w_{ij} \partial w_{uv}} \text{ for all } u, v. \tag{3.2}$$

It can be shown that by using Pearlmutter's and Møller's efficient second-order method, the gradient of $B(w, x_p)$ can be computed in $O(L)$ time. Therefore, our algorithm has the same order of computational complexity as standard backpropagation.

4 Derivation of the Algorithm

4.1 Outline. We are interested in weights representing nets with tolerable error but flat outputs (see Sections 2 and A.1). To find nets with flat outputs, two conditions will be defined to specify $B(w, x_p)$ for $x_p \in X_0$ and, as a consequence, $B(w, X_0)$ (see Section 3). The first condition ensures flatness. The second condition enforces equal flatness in all weight space directions, to obtain low variance of the net functions induced by weights within a box. The second condition will be justified using an MDL-based argument. In both cases, linear approximations will be made (to be justified in Section A.2).

4.2 Formal Details. We are interested in weights causing tolerable error (see "acceptable minima" in Section 2) that can be perturbed without

causing significant output changes, thus indicating the presence of many neighboring weights leading to the same net function. By searching for the boxes from Section 2, we are actually searching for low-error weights whose perturbation does not significantly change the net function.

In what follows we treat the input x_p as fixed. For convenience, we suppress x_p, abbreviating $o^k(w, x_p)$ by $o^k(w)$. 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; we abbreviate $o^k(w)$ by o^k). Linear approximation (justified in Section A.2) gives us flatness condition 1:

$$ED(w, \delta w) \approx ED_l(\delta w) := \sum_k \left(\sum_{i,j} \frac{\partial o^k}{\partial w_{ij}} \delta w_{ij} \right)^2 \leq ED_{l,max}(\delta w) \tag{4.1}$$

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

where $\epsilon > 0$ defines tolerable output changes within a box and is small enough to allow for linear approximation (it does not appear in $B(w, x_p)$'s and $B(w, D_0)$'s gradients; see Section 3). ED_l is ED's linear approximation, and $ED_{l,max}$ is $max\{ED_l(w, \delta v) | \ \forall_{ij}\colon \delta v_{ij} = \pm \delta w_{ij}\}$. Flatness condition 1 is a "robustness condition" (or "fault tolerance condition" or "perturbation tolerance condition"; see, e.g., Minai and Williams 1994; Murray and Edwards 1993; Neti *et al.* 1992; Matsuoka 1992; Bishop 1993; Kerlirzin and Vallet 1993; Carter *et al.* 1990).

Many boxes 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 inequality 4.1:

$$\forall i, j, u, v : (\delta w_{ij})^2 \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2 = (\delta w_{uv})^2 \sum_k \left(\frac{\partial o^k}{\partial w_{uv}} \right)^2 . \tag{4.2}$$

Flatness condition 2 enforces equal "directed errors"

$$ED_{ij}(w, \delta w_{ij}) = \sum_k (o^k(w_{ij} + \delta w_{ij}) - o^k(w_{ij}))^2 \approx \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \delta w_{ij} \right)^2 ,$$

where $o^k(w_{ij})$ has the obvious meaning, and δw_{ij} is the i, jth component of δw. Linear approximation is justified by the choice of ϵ in inequality 4.1. As will be seen in the MDL justification to be presented below, flatness condition 2 favors the box that minimizes the mean perturbation error within the box. This corresponds to minimizing the variance of the net functions induced by weights within the box (recall that $ED(w, \delta w)$ is quadratic).

4.3 Deriving the Algorithm from Flatness Conditions 1 and 2. We first solve equation 4.2 for $|\delta w_{ij}|$:

$$|\delta w_{ij}| = |\delta w_{uv}| \sqrt{\frac{\sum_k \left(\frac{\partial o^k}{\partial w_{uv}}\right)^2}{\sum_k \left(\frac{\partial o^k}{\partial w_{ij}}\right)^2}} \text{ (fixing } u, v \text{ for all } i, j).$$

Then we insert the $|\delta w_{ij}|$ (with fixed u, v) into inequality 4.1 (replacing the second "$\leq$" in 4.1 with "$=$", because we search for the box with maximal volume). This gives us an equation for the $|\delta w_{uv}|$ (which depend on w, but this is notationally suppressed):

$$|\delta w_{uv}| = \frac{\sqrt{\epsilon}}{\sqrt{\sum_k \left(\frac{\partial o^k}{\partial w_{uv}}\right)^2} \sqrt{\sum_k \left(\sum_{i,j} \frac{\left|\frac{\partial o^k}{\partial w_{ij}}\right|}{\sqrt{\sum_k \left(\frac{\partial o^k}{\partial w_{ij}}\right)^2}} \right)^2}}. \tag{4.3}$$

The $|\delta w_{ij}|$ (u, v is replaced by i, j) approximate the Δw_{ij} from Section 2. The box M_w is approximated by AM_w, the box with center w and edge lengths $2\delta w_{ij}$. M_w's volume $V(\Delta w)$ is approximated by AM_w's box volume $V(\delta w) := 2^L \prod_{ij} |\delta w_{ij}|$. Thus, $\tilde{B}(w, x_p)$ (see Section 3) can be approximated by $B(w, x_p) := -\log \frac{1}{2^L} V(\delta w) = \sum_{i,j} -\log |\delta w_{ij}|$. This immediately leads to the algorithm given by equation 3.1.

4.4 How Can the Above Approximations Be Justified? The learning process itself enforces their validity (see Section A.2). Initially, the conditions above are valid only in a very small environment of an "initial" acceptable minimum. But during the search for new acceptable minima with more associated box volume, the corresponding environments are enlarged. Section A.2 will prove this for feedforward nets (experiments indicate that this appears to be true for recurrent nets as well).

4.5 Comments. Flatness condition 2 influences the algorithm as follows: (1) The algorithm prefers to increase the δw_{ij}'s of weights whose current contributions are not important to compute the target output; (2) The algorithm enforces equal sensitivity of all output units with respect to weights of connections to hidden units. Hence, output units tend to share hidden units; that is, different hidden units tend to contribute equally to the computation of the target. The contributions of a particular hidden unit to different output unit activations tend to be equal too.

Flatness condition 2 is essential. Flatness condition 1 by itself corresponds to nothing more than first-order derivative reduction (ordinary sensitivity

reduction). However, what we really want is to minimize the variance of the net functions induced by weights near the actual weight vector.

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

4.6 MDL Justification of Flatness Condition 2. Let us assume a sender wants to send a description of the function induced by w to a receiver who knows the inputs x_p but not the targets y_p, where $(x_p, y_p) \in D_0$. The MDL principle suggests that the sender wants to minimize the expected description length of the net function. Let $ED_{mean}(w, X_0)$ denote the mean value of ED on the box. Expected description length is approximated by $\mu ED_{mean}(w, X_0) + B(w, X_0) + c$, where c, μ are positive constants. One way of seeing this is to apply Hinton and van Camp's "bits back" argument to a uniform weight prior (ED_{mean} corresponds to the 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 two 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 $\tilde{B}(w, X_0) + c$, 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 by selecting an initialization weight vector within the bit box and using gradient descent to decrease $B(w_a, X_0)$ until $ED(w_a, w_b - w_a) = ED(w, w_b - w)$, 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. 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 $\mu ED(w, w_b - w)$ (Shannon 1948). w_b, the center of the bit box, cannot be known before training. However, we do know the expected description length of the net function,

which is $\mu ED_{mean} + \tilde{B}(w, X_0) + c$ (c is a constant independent of w). Let us approximate ED_{mean}:

$$
\begin{aligned}
ED_{l,mean}(w, \delta w) &:= \frac{1}{V(\delta w)} \int_{AM_w} ED_l(w, \delta v) d\delta v \\
&= \frac{1}{V(\delta w)} 2^L \frac{1}{3} \sum_{i,j} \left((\delta w_{ij})^3 \sum_k \right. \\
&\quad \left. \times \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2 \prod_{u,v \text{ with } u,v \neq i,j} \delta w_{uv} \right) \\
&= \frac{1}{3} \sum_{i,j} (\delta w_{ij})^2 \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2 .
\end{aligned}
$$

Among those w that lead to equal $B(w, X_0)$ (the negative logarithm of the box volume plus $L \log 2$), we want to find those with minimal description length of the function induced by w. Using Lagrange multipliers (viewing the δw_{ij} as variables), it can be shown that $ED_{l,mean}$ is minimal under the condition $B(w, X_0) = constant$ iff flatness condition 2 holds. To conclude: With given box volume, we need flatness condition 2 to minimize the expected description length of the function induced by w.

5 Experimental Results

5.1 Experiment 1: Noisy Classification.

5.1.1 Task. The first task is taken from Pearlmutter and Rosenfeld (1991). The task is to decide whether the x-coordinate of a point in two-dimensional space exceeds zero (class 1) or does not (class 2). Noisy training–test examples are generated as follows: data points are obtained from a gaussian with zero mean and standard deviation 1.0, bounded in the interval $[-3.0, 3.0]$. The data points are misclassified with probability 0.05. Final input data are obtained by adding a zero mean gaussian with stdev 0.15 to the data points. In a test with 2 million data points, it was found that the procedure above leads to 9.27% misclassified data. No method will misclassify less than 9.27%, due to the inherent noise in the data (including the test data). The training set is based on 200 fixed data points (see Fig. 3). The test set is based on 120,000 data points.

5.1.2 Results. Ten conventional backprop (BP) nets were tested against ten equally initialized networks trained by flat minimum search (FMS). After 1000 epochs, the weights of our nets essentially stopped changing

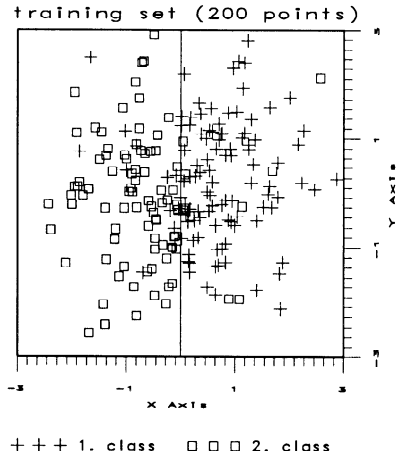

Figure 3: The 200 input examples of the training set. Crosses represent data points from class 1. Squares represent data points from class 2.

(automatic "early stopping"), while BP 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 BP, 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 5.6 for parameters common to all experiments.

5.2 Experiment 2: Recurrent Nets.

5.2.1 Time-varying inputs. 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

Table 1: Ten Comparisons of Conventional Backpropagation (BP) and Flat Minimum Search (FMS).

	Backpropagation		FMS			Backpropagation		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

Note: The MSE column shows mean squared error on the test set. The dto column shows the difference between the percentage of misclassifications and the optimal percentage (9.27). The remaining rows provide the analogous information for FMS, which clearly outperforms backpropagation.

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.

5.2.2 Non-weight-decay-like results. With conventional recurrent net algorithms, after training, both hidden units were used to store the input vector. This was not so with our new approach. We trained 20 networks. All of them learned perfect solutions. As 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 5.6 for parameters common to all experiments.

5.3 Experiment 3: Stock Market Prediction 1.

5.3.1 Task. We predict the DAX[1] (the German stock market index) using fundamental indicators. Following Rehkugler and Poddig (1990), the net sees the following indicators: (1) German interest rate (Umlaufsrendite), (2) industrial production divided by money supply, and (3) 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.

5.3.2 Details. 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. FMS is compared against (1) conventional backpropagation (BP8) with 8 hidden units, (2) BP 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; Hassibi and Stork 1993), with a few improvements (see Section 5.6), and (4) weight decay (WD) according to Weigend *et al.* (1991) (WD and OBS were chosen because they are well known and widely used).

5.3.3 Performance Measure. Since wrong predictions lead to loss of money, performance is measured as follows. The sum of incorrectly predicted DAX changes is subtracted from the sum of correctly predicted DAX changes. The result is divided by the sum of absolute DAX changes.

5.3.4 Results. Table 2 shows the results. Our method outperforms the other methods. Note that MSE is not a reasonable performance measure for this task. For instance, although FMS typically makes more correct classifications than WD, FMS's MSE often exceeds WD's. This is because WD's wrong classifications tend to be close to 0, while FMS often prefers large weights yielding strong output activations. FMS's few false classifications tend to contribute a lot to MSE.

Parameters

Learning rate: 0.01.

Architecture: (3-8-1), except BP4 with (3-4-1).

Number of training examples: 20 million.

[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.

Table 2: Comparisons of Conventional Backpropagation (BP4, BP8), Optimal Brain Surgeon (OBS), Weight Decay (WD), and Flat Minimum Search (FMS).

Method	Train MSE	Test MSE	Removed w	Removed u	Performance Max	Performance Min	Performance Mean
BP8	0.003	0.945			47.33	25.74	37.76
BP4	0.043	1.066			42.02	42.02	42.02
OBS	0.089	1.088	14	3	48.89	27.17	41.73
WD	0.096	1.102	22	4	44.47	36.47	43.49
FMS	0.040	1.162	24	4	47.74	39.70	43.62

Note: All nets except BP4 start out with eight hidden units. Each value is a mean of seven trials. Column MSE shows mean squared error. Column w shows the number of pruned weights. Column u shows the number of pruned units. The final three rows (max, min, mean) list maximal, minimal, and mean performance (see text) over seven 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). Our method outperforms all other methods.

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 5.6 for parameters common to all experiments.

5.4 Experiment 4: Stock Market Prediction 2.

5.4.1 Task. We predict the DAX again, using the basic setup of the experiment in Section 5.3. However, the following modifications are introduced:

- There are two additional inputs: dividend rate and 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).
- Performance of WD and FMS is also tested on networks "spoiled" by conventional BP ("WDR" and "FMSR"—the "R" stands for Retraining).

Table 3: Comparisons of Conventional Backpropagation (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), and Flat Minimum Search (FMS).

Method	Train MSE	Test MSE	Removed w	Removed u	Performance Max	Performance Min	Performance Mean
BP	0.181	0.535			57.33	20.69	41.61
OBS	0.219	0.502	15	1	50.78	32.20	40.43
WDR	0.180	0.538	0	0	62.54	13.64	41.17
FMSR	0.180	0.542	0	0	64.07	24.58	41.57
WD	0.235	0.452	17	3	54.04	32.03	40.75
FMS	0.240	0.472	19	3	54.11	31.12	44.40

Note: All nets start out with eight hidden units. Each value is a mean of ten trials. Column MSE shows mean squared error. Column w shows the number of pruned weights, column u shows the number of pruned units, the final three rows (max, min, mean) list maximal, minimal and mean performance (see text) over ten trials (note again that MSE is an irrelevant performance measure for this task). Flat minimum search outperforms all other methods.

5.4.2 Results. Table 3 shows the results. The average performance of our method exceeds the ones of weight decay, OBS, and conventional BP. Table 3 also shows the 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.

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 5.6 for parameters common to all experiments.

5.5 Experiment 5: Stock Market Prediction 3.

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

5.5.2 Data Analysis. To analyze the data, we computed (1) the pairwise correlation coefficients of the 35 technical indicators and (2) the maximal pairwise correlation coefficients of all indicators and all linear combinations of two indicators. This analysis revealed that only four indicators are not highly correlated. For such reasons, our nets see only the eight 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, and (d) MACD (smoothened difference of exponentially weighted 6-week and 24-week DAX).

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

5.5.4 Comparison. The following methods are applied to the training set: (1) conventional BP, (2) optimal brain surgeon / optimal brain damage (OBS / OBD), (3) weight decay (WD) according to Weigend *et al.* (1991), and (4) flat minimum search (FMS). The resulting nets are evaluated on a test set consisting of 100 data points (August 1991–July 1993). Performance is measured as in Section 5.3.

5.5.5 Results. Table 4 shows the results. Again, our method outperforms the other methods.

Parameters

Learning rate: 0.01.

Architecture: (12-9-1).

Training time: 10 million examples.

Method-specific parameters:

OBS / OBD: $E_{tol} = 0.34$.

FMS: $E_{tol} = 0.34$; $\Delta\lambda = 0.003$. If $E_{\text{average}} < E_{tol}$ then $\Delta\lambda$ is set to 0.03.

WD: like with FMS, but $w_0 = 0.2$.

See Section 5.6 for parameters common to all experiments.

Table 4: Comparisons of Conventional Backpropagation (BP), Optimal Brain Surgeon (OBS), Weight Decay (WD), and Flat Minimum Search (FMS).

Method	Train MSE	Test MSE	Removed w	Removed u	Performance Max	Performance Min	Performance Mean
BP	0.13	1.08			28.45	−16.7	8.08
OBS	0.38	0.912	55	1	27.37	−6.08	10.70
WD	0.51	0.334	110	8	26.84	−6.88	12.97
FMS	0.46	0.348	103	7	29.72	18.09	21.26

Note: All nets start out with nine hidden units. Each value is a mean of ten trials. Column MSE shows mean squared error. Column w shows the number of pruned weights. Column u shows the number of pruned units. The final three rows (max, min, mean) list maximal, minimal, and mean performance (see text) over ten trials (note again that MSE is an irrelevant performance measure for this task). Flat minimum search outperforms all other methods.

5.6 Details and Parameters. With the exception of the experiment in Section 5.2, 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 barely punish weights below $w_0 = 0.2$. E_{average} is the average error on the training set, approximated using exponential decay: $E_{\text{average}} \leftarrow \gamma E_{\text{average}} + (1-\gamma)E(net(w), D_0)$, where $\gamma = 0.85$.

5.6.1 FMS Details. To control $B(w, D_0)$'s influence during learning, its gradient is normalized and multiplied by the length of $E(net(w), D_0)$'s gradient (same for weight decay; see below). λ is computed as in Weigend *et al.* (1991) 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 4.5) exceeds the length of the weight range. However, the unknown scaling factor ϵ (see inequality 4.3 and equation 4.5) 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, which differ from each other by a factor ranging from 10^2 to 10^5).

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 reinitialized 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.

5.6.2 Weight Decay Details. We used Weigend *et al.*'s weight decay term: $D(w) = \sum_{i,j} \frac{w_{ij}^2/w_0}{1+w_{ij}^2/w_0}$. As with FMS, $D(w, w_0)$'s gradient was normalized and multiplied by the length of $E(net(w), D_0)$'s gradient. λ was adjusted as with FMS. Lost units were brought back as with FMS.

5.6.3 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 1993):

- 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 and Stork) 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).

- As with OBD (LeCun *et al.* 1990), to prevent numerical errors due to small eigenvalues of H, we do: if $L_q < 0.00001$ or $T_q < 0.00001$ or $\|I - H^{-1}H\| > 10.0$ (bad approximation of H^{-1}), we delete only the weight detected in the previous step. The other weights remain the same. Here $\| \cdot \|$ denotes the sum of the absolute values of all components of a matrix.

- If OBS's 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} barely influences the result.

- OBS is stopped if $E_{\text{average}} > E_{tol}$ after retraining. The most recent weight deletion is countermanded.

6 Relation to Previous Work

Most previous algorithms for finding low-complexity networks with high generalization capability are based on different prior assumptions. They can be broadly classified into two categories (see Schmidhuber 1994a for an exception):

1. Assumptions about the prior weight distribution. Hinton and van Camp (1993) and Williams (1994) assume that pushing the posterior weight distribution close to the weight prior leads to "good" generalization (see more details below). Weight decay (e.g., Hanson and Pratt 1989; Krogh and Hertz 1992) can be derived, for example, from gaussian or Laplace weight priors. Nowlan and Hinton (1992) assume that a distribution of networks with many similar weights generated by gaussian mixtures is "better" a priori. MacKay's weight priors (1992b) are implicit in additional penalty terms, which embody the assumptions made. The problem with these approaches is that there may not be a "good" weight prior for all possible architectures and training sets. With FMS, however, we do not have to select a "good" weight prior but instead choose a prior over input-output functions. This automatically takes the net architecture and the training set into account.

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 (Mosteller and Tukey 1968; Stone 1974; Eubank 1988; Hastie and Tibshirani 1990), for example, generalized cross-validation (Craven and Wahba 1979; Golub *et al.* 1979), final prediction error (Akaike 1970), and generalized prediction error (Moody and Utans 1992; Moody 1992). See also Holden (1994), Wang *et al.* (1994), Amari and Murata (1993), and Vapnik's structural risk minimization (Guyon *et al.* 1992; Vapnik 1992).

6.1 Constructive Algorithms and Pruning Algorithms. Other architecture selection methods are less flexible in the sense that they can be used only before or after weight adjustments. Examples are sequential network construction (Fahlman and Lebiere 1990; Ash 1989; Moody 1989), input pruning (Moody 1992; Refenes *et al.* 1994), unit pruning (White 1989; Mozer and Smolensky 1989; Levin *et al.* 1994), and weight pruning, for example, optimal brain damage (LeCun *et al.* 1990), and optimal brain surgeon (Hassibi and Stork 1993).

6.2 Hinton and van Camp (1993). They minimize the sum of two terms. The first is conventional error plus variance; the other is the distance $\int p(w \mid D_0) \log \frac{p(w|D_0)}{p(w)} dw$ between posterior $p(w \mid D_0)$ and weight prior $p(w)$. They

have to choose a "good" weight prior. But perhaps there is no "good" weight prior for all possible architectures and training sets. With FMS, however, we do not depend on a "good" weight prior; instead we have a prior over input-output functions, thus taking into account net architecture and training set. Furthermore, Hinton and van Camp have to compute variances of weights and unit activations, which (in general) cannot be done using linear approximation. Intuitively, their weight variances are related to our Δw_{ij}. Our approach, however, does justify linear approximation, as seen in Section A.2.

6.3 Wolpert (1994a). 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 the 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 neighboring 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.

6.4 Murray and Edwards (1993). 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).

7 Limitations, Final Remarks, and Future Research

7.1 How to Adjust λ. Given recent trends in neural computing (see, e.g., MacKay 1992a, 1992b), it may seem like a step backward that λ is adapted using an ad hoc heuristic from Weigend *et al.* (1991). However, for determining λ in MacKay's style, one would have to compute the Hessian of the cost function. Since our term $B(w, X_0)$ includes first-order derivatives, adjusting λ would require the computation of third-order derivatives. This is impracticable. Also, to optimize the regularizing parameter λ (see MacKay 1992b), we need to compute the function $\int d^L w \exp(-\lambda B(w, X_0))$, but it is not obvious how; the "quick and dirty version" (MacKay 1992a) cannot deal with the unknown constant ϵ in $B(w, X_0)$.

Future work will investigate how to adjust λ without too much computational effort. In fact, as will be seen in Section A.1, the choices of λ and E_{tol} are correlated. The optimal choice of E_{tol} may indeed correspond to the optimal choice of λ.

7.2 Generalized Boxes. The boxes found by the current version of FMS are axis aligned. This may cause an underestimate of flat minimum volume. Although our experiments indicate that box search works very well, it

will be interesting to compare alternative approximations of flat minimum volumes.

7.3 Multiple Initializations. First, consider this FMS alternative: Run conventional BP starting with several random initial guesses and pick the flattest minimum with the largest volume. This does not work. Conventional BP changes the weights according to the steepest descent; it runs away from flat ranges in weight space. Using an "FMS committee" (multiple runs with different initializations), however, would lead to a better approximation of the posterior. This is left for future work.

7.4 Notes on Generalization Error. If the prior distribution of targets $p(f)$ (see Section A.1) is uniform (or if the distribution of prior distributions is uniform), no algorithm can obtain a lower expected generalization error than training error reducing algorithms (see, e.g., Wolpert 1994b). Typical target distributions in the real world are not uniform, however; the real world appears to favor problem solutions with low algorithmic complexity (see, e.g., Schmidhuber 1994a). MacKay (1992a) suggests searching for alternative priors if the generalization error indicates a "poor regularizer." He also points out that with a "good" approximation of the nonuniform prior, more probable posterior hypotheses do not necessarily have a lower generalization error. For instance, there may be noise on the test set, or two hypotheses representing the same function may have different posterior values, and the expected generalization error ought to be computed over the whole posterior, not for a single solution. Schmidhuber (1994b) proposes a general "self-improving" system whose entire life is viewed as a single training sequence and continually attempts to modify its priors incrementally based on experience with previous problems (see also Schmidhuber 1996).

7.5 Ongoing Work on Low-Complexity Coding. FMS can also be useful for unsupervised learning. In recent work, we postulate that a generally useful code of given input data fulfills three MDL-inspired criteria: (1) It conveys information about the input data, (2) it can be computed from the data by a low-complexity mapping, and (3) the data can be computed from the code by a low-complexity mapping. To obtain such codes, we simply train an auto-associator with FMS (after training, codes are represented across the hidden units). In initial experiments, depending on data and architecture, this always led to well-known kinds of codes considered useful in previous work by numerous researchers. We sometimes obtained factorial codes, sometimes local codes, and sometimes sparse codes. In most cases, the codes were of the low-redundancy, binary kind. Initial experiments with a speech data benchmark problem (vowel recognition) already showed the true usefulness of codes obtained by FMS: Feeding the codes into standard,

supervised, overfitting backpropagation classifiers, we obtained much better generalization performance than competing approaches.

Appendix – Theoretical Justification

An alternative version of this Appendix (but with some minor errors) can be found in Hochreiter and Schmidhuber (1994). An expanded version of this Appendix (with a detailed description of the algorithm) is available on the World-Wide Web (see our home pages).

A.1. Flat Nets: The Most Probabale Hypotheses—Outline. We introduce a novel kind of generalization error that can be split into an overfitting error and an underfitting error. To find hypotheses causing low generalization error, we first select a subset of hypotheses causing low underfitting error. We are interested in those of its elements causing low overfitting error.

After listing relevant definitions we will introduce a somewhat unconventional variant of the Gibbs algorithm, designed to take into account that FMS uses only the training data D_0 to determine $G(\cdot \mid D_0)$, a distribution over the set of hypotheses expressing our prior belief in hypotheses (here we do not care where the data came from; this will be treated later).

This variant of the Gibbs algorithm will help us to introduce the concept of expected extended generalization error, which can be split into an overfitting error (relevant for measuring whether the learning algorithm focuses too much on the training set) and an underfitting error (relevant for measuring whether the algorithm sufficiently approximates the training set). To obtain these errors, we measure the Kullback-Leibler distance between posterior $p(\cdot \mid D_0)$ after training on the training set and posterior $p_{D_0}(\cdot \mid D)$ after (hypothetical) training on all data (here the subscript D_0 indicates that for learning D, $G(\cdot \mid D_0)$ is used as prior belief in hypotheses too). The overfitting error measures the information conveyed by $p(\cdot \mid D_0)$ but not by $p_{D_0}(\cdot \mid D)$. The underfitting error measures the information conveyed by $p_{D_0}(\cdot \mid D)$, but not by $p(\cdot \mid D_0)$.

We then introduce the tolerable error level and the set of acceptable minima. The latter contains hypotheses with low underfitting error, assuming that D_0 indeed conveys information about the test set (every training-set-error-reducing algorithm makes this assumption). In the remainder of the Appendix, we will focus only on hypotheses within the set of acceptable minima.

We introduce the relative overfitting error, which is the relative contribution of a hypothesis to the mean overfitting error on the set of acceptable minima. The relative overfitting error measures the overfitting error of hypotheses with low underfitting error. The goal is to find a hypothesis with low overfitting error and, consequently, low generalization error.

The relative overfitting error is approximated based on the trade-off between low training set error and large values of $G(\cdot \mid D_0)$. The distribution

$G(\cdot \mid D_0)$ is restricted to the set of acceptable minima, to obtain the distribution $G_{M(D_0)}(\cdot \mid D_0)$.

We then assume the data are obtained from a target chosen according to a given prior distribution. Using previously introduced distributions, we derive the expected test set error and the expected relative overfitting error. We want to reduce the latter by choosing a certain $G_{M(D_0)}(\cdot \mid D_0)$ and $G(\cdot \mid D_0)$.

The special case of noise-free data is considered.

To be able to minimize the expected relative overfitting error, we need to adopt a certain prior belief $p(f)$. The only unknown distributions required to determine $G_{M(D_0)}(\cdot \mid D_0)$ are $p(D_0 \mid f)$ and $p(D \mid f)$. They describe how (noisy) data are obtained from the target. We have to make the following assumptions: the choice of prior belief is "appropriate," the noise on data drawn from the target has mean 0, and small noise is more probable than large noise (the noise assumptions ensure that reducing the training error—by choosing some h from $M(D_0)$—reduces the expected underfitting error). We do not need gaussian assumptions, though.

We show that FMS approximates our special variant of the Gibbs algorithm. The prior is approximated locally in weight space, and flat $net(w)$ are approximated by flat $net(w')$ with w' near w in weight space.

A.1.1. Definitions. Let $A = \{(x, y) \mid x \in X, y \in Y\}$ be the set of all possible input-output pairs (pairs of vectors). Let NET be the set of functions that can be implemented by the network. For every net function $g \in NET$ we have $g \subset A$. Elements of NET are parameterized with a parameter vector w from the set of possible parameters W. $net(w)$ is a function that maps a parameter vector w onto a net function g (net is surjective). Let T be the set of target functions f, where $T \subset NET$. Let H be the set of hypothesis functions h, where $H \subset T$. For simplicity, take all sets to be finite, and let all functions map each $x \in X$ to some $y \in Y$. Values of functions with argument x are denoted by $g(x)$, $net(w)(x)$, $f(x)$, $h(x)$. We have $(x, g(x)) \in g$; $(x, net(w)(x)) \in net(w)$; $(x, f(x)) \in f$; $(x, h(x)) \in h$.

Let $D = \{(x_p, y_p) \mid 1 \le p \le m\}$ be the data, where $D \subset A$. D is divided into a training set $D_0 = \{(x_p, y_p) \mid 1 \le p \le n\}$ and a test set $D \setminus D_0 = \{(x_p, y_p) \mid n < p \le m\}$. For the moment, we are not interested in how D was obtained.

We use squared error $E(D, h) := \sum_{p=1}^{m} \|y_p - h(x_p)\|^2$, where $\|\cdot\|$ is the Euclidean norm. $E(D_0, h) := \sum_{p=1}^{n} \|y_p - h(x_p)\|^2$. $E(D \setminus D_0, h) := \sum_{p=n+1}^{m} \|y_p - h(x_p)\|^2$. $E(D, h) = E(D_0, h) + E(D \setminus D_0, h)$ holds.

A.1.2. Learning. We use a variant of the Gibbs formalism (see Opper and Haussler 1991, or Levin *et al.* 1990). Consider a stochastic learning algorithm (random weight initialization, random learning rate). The learning algorithm attempts to reduce training set error by randomly selecting a hypothesis with low $E(D_0, h)$, according to some conditional distribution

$G(\cdot \mid D_0)$ over H. $G(\cdot \mid D_0)$ is chosen in advance, but in contrast to traditional Gibbs (which deals with unconditional distributions on H), we may take a look at the training set before selecting G. For instance, one training set may suggest linear functions as being more probable than others, another one splines, and so forth. The unconventional Gibbs variant is appropriate because FMS uses only X_0 (the set of first components of D_0's elements; see Section 3) to compute the flatness of $net(w')$. The trade-off between the desire for low $E(D_0, h)$ and the a priori belief in a hypothesis according to $G(\cdot \mid D_0)$ is governed by a positive constant β (interpretable as the inverse temperature from statistical mechanics, or the amount of stochasticity in the training algorithm).

We obtain $p(h \mid D_0)$, the learning algorithm applied to data D_0:

$$p(h \mid D_0) = \frac{G(h \mid D_0) \exp(-\beta E(D_0, h))}{Z(D_0, \beta)}, \tag{A.1}$$

where

$$Z(D_0, \beta) = \sum_{h \in H} G(h \mid D_0) \exp(-\beta E(D_0, h)) . \tag{A.2}$$

$Z(D_0, \beta)$ is the error momentum generating function or the weighted accessible volume in configuration space or the partition function (from statistical mechanics).

For theoretical purposes, assume we know D and may use it for learning. To learn, we use the same distribution $G(h \mid D_0)$ as above (prior belief in some hypotheses h is based exclusively on the training set). There is a reason that we do not use $G(h \mid D)$ instead: $G(h \mid D)$ does not allow for making a distinction between a better prior belief in hypotheses and a better approximation of the test set data. However, we are interested in how $G(h \mid D_0)$ performs on the test set data $D \setminus D_0$. We obtain

$$p_{D_0}(h \mid D) = \frac{G(h \mid D_0) \exp(-\beta E(D, h))}{Z_{D_0}(D, \beta)}, \tag{A.3}$$

where

$$Z_{D_0}(D, \beta) = \sum_{h \in H} G(h \mid D_0) \exp(-\beta E(D, h)). \tag{A.4}$$

The subscript D_0 indicates that the prior belief is chosen based on D_0 only.

A.1.3. Expected Extended Generalization Error. We define the expected extended generalization error $E_G(D, D_0)$ on the unseen test exemplars $D \setminus D_0$:

$$E_G(D, D_0) := \sum_{h \in H} p(h \mid D_0) E(D \setminus D_0, h) - \sum_{h \in H} p_{D_0}(h \mid D) E(D \setminus D_0, h). \tag{A.5}$$

Here $E_G(D, D_0)$ is the mean error on $D \setminus D_0$ after learning with D_0, minus the mean error on $D \setminus D_0$ after learning with D. The second (negative) term is a lower bound (due to nonzero temperature) for the error on $D \setminus D_0$ after learning the training set D_0. For the zero temperature limit $\beta \to \infty$ we get (summation convention explained at the end of this paragraph)

$$E_G(D, D_0) = \sum_{h \in H, D_0 \subset h} \frac{G(h \mid D_0)}{Z(D_0)} E(D \setminus D_0, h),$$

where

$$Z(D_0) = \sum_{h \in H, D_0 \subset h} G(h \mid D_0).$$

In this case, the generalization error depends on $G(h \mid D_0)$, restricted to those hypotheses h compatible with D_0 ($D_0 \subset h$). For $\beta \to 0$ (full stochasticity), we get $E_G(D, D_0) = 0$.

Summation convention: In general, $\sum_{h \in H, D_0 \subset h}$ denotes summation over those h satisfying $h \in H$ and $D_0 \subset h$. In what follows, we will keep an analogous convention: the first symbol is the running index, for which additional expressions specify conditions.

A.1.4. Overfitting and Underfitting Error. Let us separate the generalization error into an overfitting error E_o and an underfitting error E_u (in analogy to Wang *et al.* 1994 and Guyon *et al.* 1992). We will see that overfitting and underfitting error correspond to the two different error terms in our algorithm: decreasing one term is equivalent to decreasing E_o, and decreasing the other is equivalent to decreasing E_u. Using the Kullback-Leibler distance (Kullback 1959), we measure the information conveyed by $p(\cdot \mid D_0)$, but not by $p_{D_0}(\cdot \mid D)$ (see Fig. 4). We may view this as information about $G(\cdot \mid D_0)$: since there are more h that are compatible with D_0 than there are h compatible with D, $G(\cdot \mid D_0)$'s influence on $p(h \mid D_0)$ is stronger than its influence on $p_{D_0}(h \mid D)$. To get the nonstochastic bias (see definition of E_G), we divide this information by β and obtain the overfitting error:

$$\begin{aligned} E_o(D, D_0) &:= \frac{1}{\beta} \sum_{h \in H} p(h \mid D_0) \ln \frac{p(h \mid D_0)}{p_{D_0}(h \mid D)} \qquad \text{(A.6)} \\ &= \sum_{h \in H} p(h \mid D_0) E(D \setminus D_0, h) + \frac{1}{\beta} \ln \frac{Z_{D_0}(D, \beta)}{Z(D_0, \beta)} . \end{aligned}$$

Analogously, we measure the information conveyed by $p_{D_0}(\cdot \mid D)$, but not by $p(\cdot \mid D_0)$ (see Fig. 5). This information is about $D \setminus D_0$. To get the nonstochastic bias (see the definition of E_G), we divide this information by

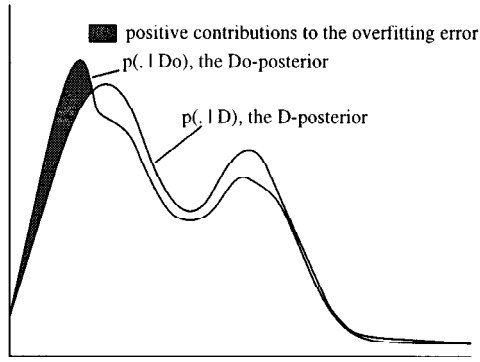

Figure 4: Positive contributions to the overfitting error $E_o(D, D_0)$, after learning the training set with a large β.

β and obtain the underfitting error:

$$E_u(D, D_0) := \frac{1}{\beta} \sum_{h \in H} p_{D_0}(h \mid D) \ln \frac{p_{D_0}(h \mid D)}{p(h \mid D_0)} \tag{A.7}$$

$$= -\sum_{h \in H} p_{D_0}(h \mid D) E(D \setminus D_0, h) + \frac{1}{\beta} \ln \frac{Z(D_0, \beta)}{Z_{D_0}(D, \beta)} .$$

Peaks in $G(\cdot \mid D_0)$ that do not match peaks of $p_{D_0}(\cdot \mid D)$ produced by $D \setminus D_0$ lead to overfitting error. Peaks of $p_{D_0}(\cdot \mid D)$ produced by $D \setminus D_0$ that do not match peaks of $G(\cdot \mid D_0)$ lead to underfitting error. Overfitting and underfitting error tell us something about the shape of $G(\cdot \mid D_0)$ with respect to $D \setminus D_0$, that is, to what degree the prior belief in h is compatible with $D \setminus D_0$.

A.1.5. Why Are They Called "Overfitting" and "Underfitting" Error? Positive contributions to the overfitting error are obtained where peaks of $p(\cdot \mid D_0)$ do not match (or are higher than) peaks of $p_{D_0}(\cdot \mid D)$: there some h will have large probability after training on D_0 but lower probability after training on all data D. This is either because D_0 has been approximated too closely or because of sharp peaks in $G(\cdot \mid D_0)$; the learning algorithm specializes either on D_0 or on $G(\cdot \mid D_0)$ ("overfitting"). The specialization on D_0 will become even worse if D_0 is corrupted by noise (the case of noisy D_0 will be treated later). Positive contributions to the underfitting error are obtained where peaks of $p_{D_0}(\cdot \mid D)$ do not match (or are higher than) peaks of $p(\cdot \mid D_0)$: there some h will have large probability after training on all data

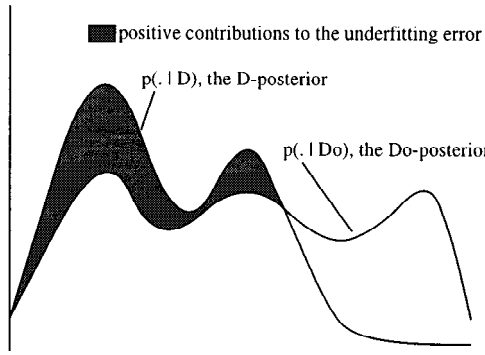

Figure 5: Positive contributions to the underfitting error $E_u(D_0, D)$, after learning the training set with a small β. Again, we use the D-posterior from Figure 4, assuming it is almost fully determined by $E(D, h)$ (even if β is smaller than in Figure 4).

D but will have lower probability after training on D_0. This is due to either a poor D_0 approximation (note that $p(\cdot \mid D_0)$ is almost fully determined by $G(\cdot \mid D_0)$), or to insufficient information about D conveyed by D_0 ("underfitting"). Either the algorithm did not learn "enough" of D_0, or D_0 does not tell us anything about D. In the latter case, there is nothing we can do; we have to focus on the case where we did not learn enough about D_0.

A.1.6. Analysis of Overfitting and Underfitting Error. $E_G(D, D_0) = E_o(D, D_0) + E_u(D, D_0)$ holds. For zero temperature limit $\beta \to \infty$ we obtain $Z_{D_0}(D) = \sum_{h \in H, D \subset h} G(h \mid D_0)$ and $Z(D_0) = \sum_{h \in H, D_0 \subset h} G(h \mid D_0)$. $E_o(D, D_0) = \sum_{h \in H, D_0 \subset h} \frac{G(h|D_0)}{Z(D_0)} E(D \setminus D_0, h) = E_G(D, D_0)$. $E_u(D, D_0) = 0$; that is, there is no underfitting error. For $\beta \to 0$ (full stochasticity) we get $E_u(D, D_0) = 0$ and $E_o(D, D_0) = 0$ (recall that E_G is not the conventional but the extended expected generalization error).

Since $D_0 \subset D$, $Z_{D_0}(D, \beta) < Z(D_0, \beta)$ holds. In what follows, averages after learning on D_0 are denoted by $\langle_{D_0} \cdot \rangle$, and averages after learning on D are denoted by $\langle_D \cdot \rangle$. Since

$$Z_{D_0}(D, \beta) = \sum_{h \in H} G(h \mid D_0) \exp(-\beta E(D_0, h)) \exp(-\beta E(D \setminus D_0, h)),$$

we have

$$\frac{Z_{D_0}(D,\beta)}{Z(D_0,\beta)} = \sum_{h\in H} p(h \mid D_0)\exp(-\beta E(D \setminus D_0, h)) = \langle_{D_0}\exp(-\beta E(D \setminus D_0,\cdot))\rangle\,.$$

Analogously, we have $\frac{Z(D_0,\beta)}{Z_{D_0}(D,\beta)} = \langle_D \exp(\beta E(D \setminus D_0,\cdot))\rangle$. Thus, $E_o(D,D_0) = \langle_{D_0} E(D \setminus D_0,\cdot)\rangle + \frac{1}{\beta}\ln\langle_{D_0}\exp(-\beta E(D \setminus D_0,\cdot))\rangle$, and $E_u(D,D_0) = -\langle_D E(D \setminus D_0,\cdot)\rangle + \frac{1}{\beta}\ln\langle_D \exp(\beta E(D \setminus D_0,\cdot))\rangle$.[2] With large β, after learning on D_0, E_o measures the difference between average test set error and a minimal test set error. With large β, after learning on D, E_u measures the difference between average test set error and a maximal test set error. Assume we do have a large β (large enough to exceed the minimum of $\frac{1}{\beta}\ln\frac{Z_{D_0}(D,\beta)}{Z(D_0,\beta)}$). We have to assume that D_0 indeed conveys information about the test set. Preferring hypotheses h with small $E(D_0,h)$ by using a larger β leads to smaller test set error (without this assumption, no error-decreasing algorithm would make sense). E_u can be decreased by enforcing less stochasticity (by further increasing β), but this will increase E_o. Similarly, decreasing β (enforcing more stochasticity) will decrease E_o but increase E_u. Increasing β decreases the maximal test set error after learning D more than it decreases the average test set error, thus decreasing E_u, and vice versa. Decreasing β increases the minimal test set error after learning D_0 more than it increases the average test set error, thus decreasing E_o, and vice versa. This is the trade-off between stochasticity and fitting the training set, governed by β.

A.1.7. Tolerable Error Level and Set of Acceptable Minima. Let us implicitly define a tolerable error level $E_{tol}(\alpha,\beta)$ that, with confidence $1-\alpha$, is the upper bound of the training set error after learning:

$$p(E(D_0,h) \le E_{tol}(\alpha,\beta)) = \sum_{h\in H, E(D_0,h)\le E_{tol}(\alpha,\beta)} p(h \mid D_0) = 1-\alpha\,. \quad \text{(A.8)}$$

With $(1-\alpha)$-confidence, we have $E(D_0,h) \le E_{tol}(\alpha,\beta)$ after learning. $E_{tol}(\alpha,\beta)$ decreases with increasing β,α. Now we define $M(D_0) := \{h \in H \mid E(D_0,h) \le E_{tol}(\alpha,\beta)\}$, which is the set of acceptable minima (see Section 2). The set of acceptable minima is a set of hypotheses with low underfitting error.

[2] We have $-\frac{\partial \ln Z(D_0,\beta)}{\partial\beta} = \langle_{D_0} E(D_0,\cdot)\rangle$ and $-\frac{\partial \ln Z_{D_0}(D,\beta)}{\partial\beta} = \langle_D E(D,\cdot)\rangle$. Furthermore, $\frac{\partial^2 \ln Z(D_0,\beta)}{\partial\beta^2} = \langle_{D_0}(E(D_0,\cdot) - \langle_{D_0} E(D_0,\cdot)\rangle)^2\rangle$ and $\frac{\partial^2 \ln Z_{D_0}(D,\beta)}{\partial\beta^2} = \langle_D(E(D,\cdot) - \langle_D E(D,\cdot)\rangle)^2\rangle$. See also Levin *et al.* (1990). Using these expressions, it can be shown: by increasing β (starting from $\beta = 0$), we will find a β that minimizes $\frac{1}{\beta}\ln\frac{Z_{D_0}(D,\beta)}{Z(D_0,\beta)} < 0$. Increasing β further makes this expression go to 0.

With probability $1 - \alpha$, the learning algorithm selects a hypothesis from $M(D_0) \subset H$. Note that for the zero temperature limit $\beta \to \infty$, we have $E_{tol}(\alpha) = 0$ and $M(D_0) = \{h \in H \mid D_0 \subset h\}$. By fixing a small E_{tol} (or a large β), E_u will be forced to be low.

We would like to have an algorithm decreasing (1) training set error (this corresponds to decreasing underfitting error) and (2) an additional error term, which should be designed to ensure low overfitting error, given a fixed small E_{tol}. The remainder of Section A.1 will lead to an answer as to how to design this additional error term. Since low underfitting is obtained by selecting a hypothesis from $M(D_0)$, in what follows we will focus on $M(D_0)$ only. Using an appropriate choice of prior belief, at the end of this section, we will finally see that the overfitting error can be reduced by an error term expressing preference for flat nets.

A.1.8. Relative Overfitting Error. Let us formally define the relative overfitting error E_{ro}, which is the relative contribution of some $h \in M(D_0)$ to the mean overfitting error of hypotheses set $M(D_0)$:

$$\text{where} \quad E_{ro}(D, D_0, M(D_0), h) = p_{M(D_0)}(h \mid D_0) E(D \setminus D_0, h), \tag{A.9}$$

$$p_{M(D_0)}(h \mid D_0) := \frac{p(h \mid D_0)}{\sum_{h \in M(D_0)} p(h \mid D_0)} \tag{A.10}$$

for $h \in M(D_0)$, and zero otherwise.

For $h \in M(D_0)$, we approximate $p(h \mid D_0)$ as follows. We assume that $G(h \mid D_0)$ is large where $E(D_0, h)$ is large (trade-off between low $E(D_0, h)$ and $G(h \mid D_0)$). Then $p(h \mid D_0)$ has large values (due to large $G(h \mid D_0)$) where $E(D_0, h) \approx E_{tol}(\alpha, \beta)$ (assuming $E_{tol}(\alpha, \beta)$ is small). We get

$$p(h \mid D_0) \approx \frac{G(h \mid D_0) \exp(-\beta E_{tol}(\alpha, \beta))}{Z(D_0, \beta)}$$

The relative overfitting error can now be approximated by

$$E_{ro}(D, D_0, M(D_0), h) \approx \frac{G(h \mid D_0)}{\sum_{h \in M(D_0)} G(h \mid D_0)} E(D \setminus D_0, h) \,. \tag{A.11}$$

To obtain a distribution over $M(D_0)$, we introduce $G_{M(D_0)}(\cdot \mid D_0)$, the normalized distribution $G(\cdot \mid D_0)$ restricted to $M(D_0)$. For approximation A.11 we have

$$E_{ro}(D, D_0, M(D_0), h) \approx G_{M(D_0)}(h \mid D_0) E(D \setminus D_0, h) \,. \tag{A.12}$$

A.1.9. Prior Belief in f and D. Assume D was obtained from a target function f. Let $p(f)$ be the prior on targets and $p(D \mid f)$ the probability of

obtaining D with a given f. We have

$$p(f \mid D_0) = \frac{p(D_0 \mid f)p(f)}{p(D_0)}, \tag{A.13}$$

where $p(D_0) = \sum_{f \in T} p(D_0 \mid f)p(f)$.

The data are drawn from a target function with added noise (the noise-free case is treated below). We do not make any assumptions about the nature of the noise; it does not have to be gaussian (as in MacKay's work, 1992b).

We want to select a $G(\cdot \mid D_0)$ that makes E_{ro} small; that is, those $h \in M(D_0)$ with small $E(D \setminus D_0, h)$ should have high probabilities $G(h \mid D_0)$.

We do not know $D \setminus D_0$ during learning. D is assumed to be drawn from a target f. We compute the expectation of E_{ro}, given D_0. The probability of the test set $D \setminus D_0$, given D_0, is

$$p(D \setminus D_0 \mid D_0) = \sum_{f \in T} p(D \setminus D_0 \mid f)p(f \mid D_0), \tag{A.14}$$

where we assume $p(D \setminus D_0 \mid f, D_0) = p(D \setminus D_0 \mid f)$ (we do not remember which exemplars were already drawn). The expected test set error $E(\cdot, h)$ for some h, given D_0, is

$$\begin{aligned} &\sum_{D \setminus D_0} p(D \setminus D_0 \mid D_0)E(D \setminus D_0, h) \qquad (A.15) \\ &= \sum_{f \in T} p(f \mid D_0) \sum_{D \setminus D_0} p(D \setminus D_0 \mid f)E(D \setminus D_0, h)\,. \end{aligned}$$

The expected relative overfitting error $E_{ro}(\cdot, D_0, M(D_0), h)$ is obtained by inserting equation A.15 into equation A.12:

$$\begin{aligned} &E_{ro}(\cdot, D_0, M(D_0), h) \qquad (A.16) \\ &\approx G_{M(D_0)}(h \mid D_0) \sum_{f \in T} p(f \mid D_0) \sum_{D \setminus D_0} p(D \setminus D_0 \mid f)E(D \setminus D_0, h). \end{aligned}$$

A.1.10. Minimizing Expected Relative Overfitting Error. We define a $G_{M(D_0)}(\cdot \mid D_0)$ such that $G_{M(D_0)}(\cdot \mid D_0)$ has its largest value near small expected test set error $E(\cdot, \cdot)$ (see equations A.12 and A.15). This definition leads to a low expectation of $E_{ro}(\cdot, D_0, M(D_0), \cdot)$ (see equation A.16). Define

$$G_{M(D_0)}(h \mid D_0) := \delta(\mathrm{argmin}_{h' \in M(D_0)}(E(\cdot, h')) - h), \tag{A.17}$$

where δ is the Dirac delta function, which we will use with loose formalism; the context will make clear how the delta function is used.

Using equation A.15 we get

$$G_{M(D_0)}(h \mid D_0) \tag{A.18}$$
$$= \delta \left(\text{argmin}_{h' \in M(D_0)} \left(\sum_{f \in T} p(f \mid D_0) \sum_{D \setminus D_0} p(D \setminus D_0 \mid f) E(D \setminus D_0, h') \right) - h \right).$$

$G_{M(D_0)}(\cdot \mid D_0)$ determines the hypothesis h from $M(D_0)$ that leads to the lowest expected test set error. Consequently, we achieve the lowest expected relative overfitting error.

$G_{M(D_0)}$ helps us define G:

$$G(h \mid D_0) := \frac{\zeta + G_{M(D_0)}(h \mid D_0)}{\sum_{h \in H} (\zeta + G_{M(D_0)}(h \mid D_0))}, \tag{A.19}$$

where $G_{M(D_0)}(h \mid D_0) = 0$ for $h \notin M(D_0)$, and where ζ is a small constant ensuring positive probability $G(h \mid D_0)$ for all hypotheses h.

To appreciate the importance of the prior $p(f)$ in the definition of $G_{M(D_0)}$ (see also equation A.24), in what follows, we will focus on the noise-free case.

A.1.11. The Special Case of Noise-Free Data. Let $p(D_0 \mid f)$ be equal to $\delta(D_0 \subset f)$ (up to a normalizing constant):

$$p(f \mid D_0) = \frac{\delta(D_0 \subset f) p(f)}{\sum_{f \in T, D_0 \subset f} p(f)}. \tag{A.20}$$

Assume $p(D \setminus D_0 \mid f) = \frac{\delta(D \setminus D_0 \subset f)}{\sum_{D \setminus D_0} \delta(D \setminus D_0 \subset f)}$. Let F be the number of elements in X. $p(D \setminus D_0 \mid f) = \frac{\delta(D \setminus D_0 \subset f)}{2^{F-n}}$. We expand $\sum_{D \setminus D_0} p(D \setminus D_0 \mid f) E(D \setminus D_0, h)$ from equation A.15:

$$\frac{1}{2^{F-n}} \sum_{D \setminus D_0 \subset f} E(D \setminus D_0, h) = \frac{1}{2^{F-n}} \sum_{D \setminus D_0 \subset f} \sum_{(x,y) \in D \setminus D_0} E((x,y), h) \tag{A.21}$$
$$= \frac{1}{2^{F-n}} \sum_{(x,y) \in f \setminus D_0} E((x,y), h) \sum_{i=1}^{F-n} \binom{F-n-1}{i-1}$$
$$= \frac{1}{2} E(f \setminus D_0, h).$$

Here $E((x,y), h) = \|y - h(x)\|^2$, $E(f \setminus D_0, h) = \sum_{(x,y) \in f \setminus D_0} \|y - h(x)\|^2$, and $\sum_{i=1}^{F-n} \binom{F-n-1}{i-1} = 2^{F-n-1}$. The factor $1/2$ results from considering the mean

test set error (where the test set is drawn from f), whereas $E(f \setminus D_0, h)$ is the maximal test set error (obtained by using a maximal test set). From equations A.15 and A.21, we obtain the expected test set error $E(\cdot, h)$ for some h, given D_0:

$$\sum_{D \setminus D_0} p(D \setminus D_0 \mid D_0) E(D \setminus D_0, h) = \frac{1}{2} \sum_{f \in T} p(f \mid D_0) E(f \setminus D_0, h) \,. \tag{A.22}$$

From equations A.22 and A.12, we obtain the expected $E_{ro}(\cdot, D_0, M(D_0), h)$:

$$\begin{aligned} & E_{ro}(\cdot, D_0, M(D_0), h) \\ & \quad \approx \tfrac{1}{2} G_{M(D_0)}(h \mid D_0) \sum_{f \in T} p(f \mid D_0) E(f \setminus D_0, h) \,. \end{aligned} \tag{A.23}$$

For $G_{M(D_0)}(h \mid D_0)$ we obtain in this noise-free case

$$\begin{aligned} & G_{M(D_0)}(h \mid D_0) \\ & = \quad \delta \left(\text{argmin}_{h' \in M(D_0)} \left(\sum_{f \in T} p(f \mid D_0) E(f \setminus D_0, h') \right) - h \right) . \end{aligned} \tag{A.24}$$

The lowest expected test set error is measured by $\frac{1}{2} \sum_{f \in T} p(f \mid D_0) E(f \setminus D_0, h)$. See equation A.22.

A.1.12. Noisy Data and Noise-Free Data: Conclusion. For both the noise-free and the noisy case, equation A.14 shows that given D_0 and h, the expected test set error depends on prior target probability $p(f)$.

A.1.13. Choice of Prior Belief. Now we select some $p(f)$, our prior belief in target f. We introduce a formalism similar to Wolpert's (1994a). $p(f)$ is defined as the probability of obtaining $f = net(w)$ by choosing a w randomly according to $p(w)$.

Let us first look at Wolpert's formalism: $p(f) = \int dw p(w) \delta(net(w) - f)$. By restricting W to W_{inj}, he obtains an injective function $net_{inj} : W_{inj} \to NET$: $net_{inj}(w) = net(w)$, which is *net* restricted to W_{inj}. net_{inj} is surjective (because *net* is surjective):

$$\begin{aligned} p(f) &= \int_W p(w) \frac{\delta(net_{inj}(w) - f)}{|\det net'_{inj}(w)|} |\det net'_{inj}(w)| dw \\ &= \int_{NET} p(net_{inj}^{-1}(g)) \frac{\delta(g - f)}{|\det net'_{inj}(net_{inj}^{-1}(g))|} dg \\ &= \frac{p(net_{inj}^{-1}(f))}{|\det net'_{inj}(net_{inj}^{-1}(f))|} \,, \end{aligned} \tag{A.25}$$

where $|\det net'_{inj}(w)|$ is the absolute Jacobian determinant of net_{inj}, evaluated at w. If there is a locally flat $net(w) = f$ (flat around w), then $p(f)$ is high.

However, we prefer to follow another path. Our algorithm (flat minimum search) tends to prune a weight w_i if $net(w)$ is very flat in w_i's direction. It prefers regions where $\det net'(w) = 0$ (where many weights lead to the same net function). Unlike Wolpert's approach, ours distinguishes the probabilities of targets $f = net(w)$ with $\det net'(w) = 0$. The advantage is that we do not only search for $net(w)$ that are flat in one direction but for $net(w)$ that are flat in many directions (this corresponds to a higher probability of the corresponding targets). Define

$$net^{-1}(g) := \{w \in W \mid net(w) = g\} \tag{A.26}$$

and

$$p(net^{-1}(g)) := \sum_{w \in net^{-1}(g)} p(w)\,. \tag{A.27}$$

We have

$$p(f) = \frac{\sum_{g \in NET} p(net^{-1}(g))\delta(g-f)}{\sum_{f \in T}\sum_{g \in NET} p(net^{-1}(g))\delta(g-f)} = \frac{p(net^{-1}(f))}{\sum_{f \in T} p(net^{-1}(f))}. \tag{A.28}$$

net partitions W into equivalence classes. To obtain $p(f)$, we compute the probability of w being in the equivalence class $\{w \mid net(w) = f\}$, if randomly chosen according to $p(w)$. An equivalence class corresponds to a net function; *net* maps all w of an equivalence class to the same net function.

A.1.14. Relation to FMS Algorithm. FMS (from Section 3) works locally in weight space W. Let w' be the actual weight vector found by FMS (with $h = net(w')$). Recall the definition of $G_{M(D_0)}(h \mid D_0)$ (see equations A.17 and A.18); we want to find a hypothesis h that best approximates those f with large $p(f)$ (the test data have a high probability of being drawn from such targets). We will see that those $f = net(w)$ with flat $net(w)$ locally have high probability $p(f)$. Furthermore we will see that a w' close to w with flat $net(w)$ has flat $net(w')$ too. To approximate such targets f, the only thing we can do is find a w' close to many w with $net(w) = f$ and large $p(f)$. To justify this approximation (see definition of $p(f \mid D_0)$ while recalling that $h \in G_{M(D_0)}$), we assume that the noise has mean 0 and that small noise is more likely than large noise (e.g., gaussian, Laplace, Cauchy distributions).

To restrict $p(f) = p(net(w))$ to a local range in W, we define regions of equal net functions $F(w) = \{\bar{w} \mid \forall_\tau 0 \le \tau \le 1, w + \tau(\bar{w} - w) \in W : net(w) = net(w + \tau(\bar{w} - w))\}$. Note that $F(w) \subset net^{-1}(net(w))$. If $net(w)$ is flat along long distances in many directions $\bar{w} - w$, then $F(w)$ has many elements. Locally in weight space, at w' with $h = net(w')$, for $\gamma > 0$ we define: if the

minimum $w = \text{argmin}_{\bar{w}}\{\|\bar{w} - w'\| \mid \|\bar{w} - w'\| < \gamma, net(\bar{w}) = f\}$ exists, then $p_{w',\gamma}(f) = c\, p(F(w))$, where c is a constant. If this minimum does not exist, then $p_{w',\gamma}(f) = 0$. $p_{w',\gamma}(f)$ locally approximates $p(f)$. During search for w' (corresponding to a hypothesis $h = net(w')$), to decrease locally the expected test set error (see equation A.15), we want to enter areas where many large $F(w)$ are near w' in weight space. We wish to decrease the test set error, which is caused by drawing data from highly probable targets f (those with large $p_{w',\gamma}(f)$). We do not know, however, which w's are mapped to target's f by $net(\cdot)$. Therefore, we focus on $F(w)$ (w near w' in weight space), instead of $p_{w',\gamma}(f)$. Assume $\|w' - w\|$ is small enough to allow for a Taylor expansion, and that $net(w')$ is flat in direction $(\bar{w} - w')$:

$$\begin{aligned} net(w) &= net(w' + (w - w')) \\ &= net(w') + \nabla net(w')(w - w') \\ &\quad + \frac{1}{2}(w - w')H(net(w'))(w - w') + \cdots, \end{aligned}$$

where $H(net(w'))$ is the Hessian of $net(\cdot)$ evaluated at w', $\nabla net(w)(\bar{w} - w') = \nabla net(w')(\bar{w} - w') + O(w - w')$, and $(\bar{w} - w')H(net(w))(\bar{w} - w') = (\bar{w} - w')H(net(w'))(\bar{w} - w') + O(w - w')$ (analogously for higher-order derivatives). We see that in a small environment of w', there is flatness in direction $(\bar{w} - w')$ too. And, if $net(w')$ is not flat in any direction, this property also holds within a small environment of w'. Only near w' with flat $net(w')$, there may exist w with large $F(w)$. Therefore, it is reasonable to search for a w' with $h = net(w')$, where $net(w')$ is flat within a large region. This means searching for the h determined by $G_{M(D_0)}(\cdot \mid D_0)$ of equation A.17. Since $h \in M(D_0)$, $E(D_0, net(w')) \leq E_{tol}$ holds, we search for a w' living within a large connected region, where for all w within this region $E(net(w'), net(w), X) = \sum_{x \in X} \|net(w')(x) - net(w)(x)\|^2 \leq \epsilon$, where ϵ is defined in Section 2. To conclude, we decrease the relative overfitting error and the underfitting error by searching for a flat minimum (see the definition of flat minima in Section 2).

A.1.15. Practical Realization of the Gibbs Variant.

1. Select α and $E_{tol}(\alpha, \beta)$, thus implicitly choosing β.
2. Compute the set $M(D_0)$.
3. Assume we know how data are obtained from target f; that is, we know $p(D_0 \mid f)$, $p(D \backslash D_0 \mid f)$, and the prior $p(f)$. Then we can compute $G_{M(D_0)}(\cdot \mid D_0)$ and $G(\cdot \mid D_0)$.
4. Start with $\beta = 0$ and increase β until equation A.8 holds. Now we know the β from the implicit choice above.

5. Since we know all we need to compute $p(h \mid D_0)$, select some h according to this distribution.

A.1.16. Three Comments on Certain FMS Limitations.

1. *FMS only approximates* the Gibbs variant given by the definition of $G_{M(D_0)}(h \mid D_0)$ (see equations A.17 and A.18). We only locally approximate $p(f)$ in weight space. If $f = net(w)$ is locally flat around w, then there exist units or weights that can be given with low precision (or can be removed). If there are other weights w_i with $net(w_i) = f$, then one may assume that there are also points in weight space near such w_i where weights can be given with low precision (think of, for example, symmetrical exchange of weights and units). We assume the local approximation of $p(f)$ is good. The most probable targets represented by flat $net(w)$ are approximated by a hypothesis h, which is also represented by a flat $net(w')$ (where w' is near w in weight space). To allow for approximation of $net(w)$ by $net(w')$, we have to assume that the hypothesis set H is dense in the target set T. If $net(w')$ is flat in many directions, then there are many $net(w) = f$ that share this flatness and are well approximated by $net(w')$. The only reasonable thing FMS can do is to make $net(w')$ as flat as possible in a large region around w', to approximate the $net(w)$ with large prior probability (recall that flat regions are approximated by axis-aligned boxes, as discussed in Section 7.2). This approximation is fine if $net(w')$ is smooth enough in "unflat" directions (small changes in w' should not result in very different net functions).

2. *Concerning point 3 above.* $p(f \mid D_0)$ depends on $p(D_0 \mid f)$ (how the training data are drawn from the target; see equation A.13). $G_{M(D_0)}(h \mid D_0)$ depends on $p(f \mid D_0)$ and $p(D \backslash D_0 \mid f)$ (how the test data are drawn from the target). Since we do not know how the data are obtained, the quality of the approximation of the Gibbs algorithm may suffer from noise that has not mean 0, or from large noise being more probable than small noise.
   Of course, if the choice of prior belief does not match the true target distribution, the quality of $G_{M(D_0)}(h \mid D_0)$'s approximation will suffer as well.

3. *Concerning point 5 above.* FMS outputs only a single h instead of $p(h \mid D_0)$. This issue is discussed in Section 7.3.

A.1.17. Conclusion. Our FMS algorithm from Section 3 only approximates the Gibbs algorithm variant. Two important assumptions are made. The first is that an appropriate choice of prior belief has been made. The second is that the noise on the data is not too "weird" (mean 0, small noise more likely). The two assumptions are necessary for any algorithm based

on an additional error term besides the training error. The approximations are that $p(f)$ is approximated locally in weight space, and flat $net(w)$ are approximated by flat $net(w')$ with w' near w's. Our Gibbs variant takes into account that FMS uses only X_0 for computing flatness.

A.2. Why Does the Hessian Decrease? This section shows that second-order derivatives of the output function vanish during flat minimum search. This justifies the linear approximations in Section 4.

A.2.1. Intuition. We show that the algorithm tends to suppress the following values: unit activations, first-order activation derivatives, and the sum of all contributions of an arbitrary 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.

A.2.2. Formal Details. We consider a strictly layered feedforward network with K output units and g layers. We use the same activation function f for all units. For simplicity, in what follows we focus on a single input vector x_p. x_p (and occasionally w itself) will be notationally suppressed. We have

$$\frac{\partial y^l}{\partial w_{ij}} = f'(s_l) \left\{ \begin{array}{ll} y^j & \text{for } i = l \\ \sum_m w_{lm} \frac{\partial y^m}{w_{ij}} & \text{for } i \neq l \end{array} \right\}, \tag{A.29}$$

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

The last term of equation 3.1 (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 equation 3.1. We are left with the second term. With equation A.29, we obtain:

$$\begin{aligned} &\sum_{i,j} \log \sum_k \left(\frac{\partial o^k}{\partial w_{ij}} \right)^2 \\ &= 2 \sum_{\text{unit } k \text{ in the } g\text{th layer}} (\text{fan-in of unit } k) \log |f'(s_k)| \\ &\quad + 2 \sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-in of unit } j) \log \sum_k \left(f'(s_k) w_{kj}\right)^2 \\
&+2 \sum_{\text{unit } j \text{ in the } (g-1)\text{th layer}} (\text{fan-in of unit } j) \log |f'(s_j)| \\
&+2 \sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| \\
&+ \sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-in of unit } j) \log \sum_k \\
&\times \left(f'(s_k) \sum_l f'(s_l) w_{kl} w_{lj} \right)^2 \\
&+2 \sum_{\text{unit } j \text{ in the } (g-2)\text{th layer}} (\text{fan-in of unit } j) \log |f'(s_j)| \\
&+2 \sum_{\text{unit } j \text{ in the } (g-3)\text{th layer}} (\text{fan-out of unit } j) \log |y^j| \\
&+ \sum_{i,j, \text{ where unit } i \text{ in a layer } <(g-2)} \log \sum_k \\
&\times \left(f'(s_k) \sum_{l_1} f'(s_{l_1}) w_{kl_1} \sum_{l_2} w_{l_1 l_2} \frac{\partial y^{l_2}}{\partial w_{ij}} \right)^2 .
\end{aligned}
\tag{A.30}
$$

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 $\sum_k (f'(s_k) \sum_{l_1} f'(s_{l_1}) w_{kl_1} \sum_{l_2} \cdots \sum_{l_r} f'(s_{l_r}) w_{l_{r-1} l_r} w_{l_r j})^2$ 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 $\sum_k (f'(s_k) w_{kj})^2$). These terms also decrease in proportion to unit j's fan-in. Analogously, equation A.30 can be extended to the case of additional layers.

A.2.3. 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 toward 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 that push $f(s_j)$ or $f'(s_j)$ toward 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 5.2.

A.2.4. How Does This Influence the Hessian? The entries in the Hessian corresponding to output o^k can be written as follows:

$$\frac{\partial^2 o^k}{\partial w_{ij}\partial w_{uv}} = \frac{f''(s_k)}{(f'(s_k))^2}\frac{\partial o^k}{\partial w_{ij}}\frac{\partial o^k}{\partial w_{uv}} + f'(s_k)\left(\sum_l w_{kl}\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}} + \bar{\delta}_{ik}\frac{\partial y^j}{\partial w_{uv}} + \bar{\delta}_{uk}\frac{\partial y^v}{\partial w_{ij}}\right), \tag{A.31}$$

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 A.31 decreases during learning.

According to the analysis above, the first-order derivatives in the second term of equation A.31 are pushed toward zero. So are the w_{kl} of the sum in the second term of equation A.31.

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 the weights, the activation functions, their first- and second-order derivatives, and the inputs are bounded. This is indeed the case, as will be shown for networks with one or two hidden layers:

In case 1, for unit l in a single hidden layer ($g = 3$), we obtain

$$\left|\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}}\right| = |\bar{\delta}_{li}\bar{\delta}_{lu} f''(s_l) y^j y^v| < C_1, \tag{A.32}$$

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

In case 2, for unit l in the third layer of a net with two hidden layers ($g = 4$), we obtain

$$\begin{aligned}\left|\frac{\partial^2 y^l}{\partial w_{ij}\partial w_{uv}}\right| &= |f''(s_l)(w_{li}y^j + \bar{\delta}_{il}y^j)(w_{lu}y^v + \bar{\delta}_{ul}y^v) \\ &\quad + f'(s_l)(w_{li}\bar{\delta}_{iu} f''(s_i) y^j y^v + \bar{\delta}_{il}\bar{\delta}_{uj} f'(s_j) y^v + \bar{\delta}_{ul}\bar{\delta}_{iv} f'(s_v) y^j)| \\ &< C_2,\end{aligned} \tag{A.33}$$

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

A.2.5. Conclusion. As desired, our algorithm makes the $H^k_{ij,uv}$ decrease where $|\delta w_{ij}|$ or $|\delta w_{uv}|$ increase.

A.3. Explicit Derivative of Equation 3.1. We explicitly compute the derivatives of equation 3.1. For simplicity, in what follows we focus on a single input vector x_p. Again, x_p (and occasionally w itself) will be notationally suppressed.

The derivative of the right-hand side of equation 3.1 is:

$$\frac{\partial B(w,x_p)}{\partial w_{uv}} = \sum_{i,j} \frac{\sum_k \frac{\partial o^k}{\partial w_{ij}} \frac{\partial^2 o^k}{\partial w_{ij} \partial w_{uv}}}{\sum_m \left(\frac{\partial o^m}{\partial w_{ij}} \right)^2}$$

$$+ L \frac{\sum_k \left(\sum_{i,j} \frac{\left| \frac{\partial o^k}{\partial w_{ij}} \right|}{\sqrt{\sum_m (\frac{\partial o^m}{\partial w_{ij}})^2}} \sum_{i,j} \left(\operatorname{sign}\left(\frac{\partial o^k}{\partial w_{ij}} \right) \frac{\frac{\partial^2 o^k}{\partial w_{ij} \partial w_{uv}} \sum_m \left(\frac{\partial o^m}{\partial w_{ij}} \right)^2 - \frac{\partial o^k}{\partial w_{ij}} \sum_m \frac{\partial o^m}{\partial w_{ij}} \frac{\partial^2 o^m}{\partial w_{ij} \partial w_{uv}}}{\left(\sum_m \left(\frac{\partial o^m}{\partial w_{ij}} \right)^2 \right)^{\frac{3}{2}}} \right) \right)}{\sum_k \left(\sum_{i,j} \frac{\left| \frac{\partial o^k}{\partial w_{ij}} \right|}{\sqrt{\sum_m \left(\frac{\partial o^m}{\partial w_{ij}} \right)^2}} \right)^2}. \tag{A.34}$$

To compute equation 3.2, we need

$$\frac{\partial B(w,x_p)}{\partial \left(\frac{\partial o^k}{\partial w_{ij}} \right)} = \frac{\frac{\partial o^k}{\partial w_{ij}}}{\sum_m \left(\frac{\partial o^m}{\partial w_{ij}} \right)^2}$$

$$+ L \frac{\sum_m \left(\sum_{l,r} \left(\frac{\left| \frac{\partial o^m}{\partial w_{lr}} \right|}{\sqrt{\sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{lr}})^2}} \right) \operatorname{sign}\left(\frac{\partial o^m}{\partial w_{ij}} \right) \frac{\bar{\delta}_{mk} \sum_{\bar{m}} (\frac{\partial o^{\bar{m}}}{\partial w_{ij}})^2 - \frac{\partial o^m}{\partial w_{ij}} \frac{\partial o^k}{\partial w_{ij}}}{\left(\sum_{\bar{m}} \left(\frac{\partial o^{\bar{m}}}{\partial w_{ij}} \right)^2 \right)^{\frac{3}{2}}} \right)}{\sum_m \left(\sum_{l,r} \frac{|\frac{\partial o^m}{\partial w_{lr}}|}{\sqrt{\sum_{\bar{m}} \left(\frac{\partial o^{\bar{m}}}{\partial w_{lr}} \right)^2}} \right)^2}, \tag{A.35}$$

where $\bar{\delta}$ is the Kronecker delta. Using the nabla operator and equation A.35, we can compress equation A.34:

$$\nabla_{uv} B(w, x_p) = \sum_k H^k \left(\nabla_{\frac{\partial o^k}{\partial w_{ij}}} B(w, x_p) \right), \tag{A.36}$$

where H^k is the Hessian of the output o^k. Since the sums over l, r in equation A.35 need to be computed only once (the results are reusable for all i, j), $\nabla_{\frac{\partial o^k}{\partial w_{ij}}} B(w, x_p)$ can be computed in $O(L)$ time. The product of the Hessian and a vector can be computed in $O(L)$ time. With constant number of output units, the computational complexity of our algorithm is $O(L)$.

The long version of this paper (available on the World-Wide Web; see our home pages) contains pseudo-code of an efficient implementation. It is based on fast multiplication of the Hessian and a vector due to Pearlmutter (1994) and Møller (1993).

Acknowledgments

We thank David Wolpert and several anonymous referees for numerous comments that helped to improve previous drafts of this article. This work was supported by DFG grant SCHM 942/3-1 from Deutsche Forschungsgemeinschaft.

References

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

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

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

Bishop, C. M. 1993. Curvature-driven smoothing: A learning algorithm for feed-forward networks. *IEEE Transactions on Neural Networks* **4**(5), 882–884.

Buntine, W. L., and Weigend, A. S. 1991. Bayesian back-propagation. *Complex Systems* **5**, 603–643.

Carter, M. J., Rudolph, F. J., and Nucci, A. xJ. 1990. Operational fault tolerance of CMAC networks. In *Advances in Neural Information Processing Systems 2*, pp. 340–347. Morgan Kaufmann, San Mateo, CA.

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

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

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

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

Guyon, I., Vapnik, V., Boser, B., Bottou, L., and Solla, S. A. 1992. Structural risk minimization for character recognition. In *Advances in Neural Information*

Processing Systems 4, J. E. Moody, S. J. Hanson, and R. P. Lippman, eds., pp. 471–479. Morgan Kaufmann, San Mateo, CA.

Hanson, S. J., and Pratt, L. Y. 1989. Comparing biases for minimal network construction with backpropagation. In *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, ed., pp. 177–185. Morgan Kaufmann, San Mateo, CA.

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

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

Hinton, G. E., and van Camp, D. 1993. Keeping neural networks simple. In *Proceedings of the International Conference on Artificial Neural Networks, Amsterdam*, pp. 11–18. Springer-Verlag, Berlin.

Hochreiter, S. and Schmidhuber, J. 1994. *Flat minimum search finds simple nets.* Tech. Rep. FKI-200-94, Fakultät für Informatik, Technische Universität München.

Hochreiter, S., and Schmidhuber, J. 1995. Simplifying nets by discovering flat minima. In *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, eds., pp. 529–536. MIT Press, Cambridge MA.

Holden, S. B. 1994. *On the theory of generalization and self-structuring in linearly weighted connectionist networks.* Ph.D. thesis, Cambridge University.

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

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

Kullback, S. 1959. *Statistics and Information Theory*. John Wiley, New York.

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

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

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

MacKay, D. J. C. 1992a. Bayesian interpolation. *Neural Computation* **4**, 415–447.

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

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

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

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

Moody, J. E. 1989. Fast learning in multi-resolution hierarchies. In *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, ed., pp. 29–39. Morgan Kaufmann, San Mateo, CA.

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

Moody, J. E., and Utans, J. 1992. Principled architecture selection strategies for neural networks: Application to corporate bond rating prediction. In *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippman, eds., pp. 847–854. Morgan Kaufmann, San Mateo, CA.

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

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

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

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

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

Opper, M., and Haussler, D. 1991. Calculation of the learning curve of Bayes optimal classification algorithm for learning a perceptron with noise. In *Computational Learning Theory: Proceedings of the Fourth Annual Workshop*. Morgan Kaufmann, San Mateo, CA.

Pearlmutter, B. A. 1994. Fast exact multiplication by the Hessian. *Neural Computation* **6**(1), 147–160.

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

Refenes, A. N., Francis, G., and Zapranis, A. D. 1994. Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks* **7**, 375–388.

Rehkugler, H., and Poddig, T. 1990. *Statistische Methoden versus Künstliche Neuronale Netzwerke zur Aktienkursprognose*. Tech. Rep. 73, University Bamberg, Fakultät Sozial- und Wirtschaftswissenschaften.

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

Schmidhuber, J. H. 1994a. *Discovering problem solutions with low Kolmogorov complexity and high generalization capability.* Tech. Rep. FKI-194-94, Fakultät für Informatik, Technische Universität München. Short version in *Machine Learning: Proceedings of the Twelfth International Conference*, A. Prieditis and S. Russell, eds., pp. 488-496, Morgan Kaufmann, San Mateo, 1995.

Schmidhuber, J. H. 1994b. *On learning how to learn learning strategies.* Tech. Rep. FKI-198-94, Fakultät für Informatik, Technische Universität München.

Schmidhuber, J. 1996. A general method for multi-agent learning and incremental self-improvement in unrestricted environments. In *Evolutionary Computation: Theory and Applications*, X. Yao, ed., Scientific Publ. Co., Singapore.

Shannon, C. E. 1948. A mathematical theory of communication (parts I and II). *Bell System Technical Journal* **27**, 379–423.

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

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

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

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

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

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

Williams, P. M. 1994. *Bayesian regularisation and pruning using a Laplace prior.* Tech. Rep., School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton.

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

Wolpert, D. H. 1994b. *The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework.* Tech. Rep. SFI-TR-03-123, Santa Fe Institute, NM.

Received January 5, 1995; accepted March 19, 1996.