

## A Practical Bayesian Framework for Backpropagation Networks

David J. C. MacKay\*

*Computation and Neural Systems, California Institute of Technology 139-74,  
Pasadena, CA 91125 USA*

A quantitative and practical Bayesian framework is described for learning of mappings in feedforward networks. The framework makes possible (1) objective comparisons between solutions using alternative network architectures, (2) objective stopping rules for network pruning or growing procedures, (3) objective choice of magnitude and type of weight decay terms or additive regularizers (for penalizing large weights, etc.), (4) a measure of the effective number of well-determined parameters in a model, (5) quantified estimates of the error bars on network parameters and on network output, and (6) objective comparisons with alternative learning and interpolation models such as splines and radial basis functions. The Bayesian "evidence" automatically embodies "Occam's razor," penalizing overflexible and overcomplex models. The Bayesian approach helps detect poor underlying assumptions in learning models. For learning models well matched to a problem, a good correlation between generalization ability and the Bayesian evidence is obtained.

This paper makes use of the Bayesian framework for regularization and model comparison described in the companion paper "Bayesian Interpolation" (MacKay 1992a). This framework is due to Gull and Skilling (Gull 1989).

### 1 The Gaps in Backprop

---

There are many knobs on the black box of "backprop" [learning by backpropagation of errors (Rumelhart *et al.* 1986)]. Generally these knobs are set by rules of thumb, trial and error, and the use of reserved test data to assess generalization ability (or more sophisticated cross-validation). The knobs fall into two classes: (1) parameters that change the effective learning model, for example, number of hidden units, and weight decay

---

\*Present address: Darwin College, Cambridge CB3 9EU, U.K.

terms; and (2) parameters concerned with function optimization technique, for example, "momentum" terms. This paper is concerned with making objective the choice of the parameters in the first class, and with ranking alternative solutions to a learning problem in a way that makes full use of all the available data. Bayesian techniques will be described that are both theoretically well-founded and practically implementable.

Let us review the basic framework for learning in networks, then discuss the points at which objective techniques are needed. The training set for the mapping to be learned is a set of input-target pairs  $D = \{\mathbf{x}^m, \mathbf{t}^m\}$ , where  $m$  is a label running over the pairs. A neural network architecture  $\mathcal{A}$  is invented, consisting of a specification of the number of layers, the number of units in each layer, the type of activation function performed by each unit, and the available connections between the units. If a set of values  $\mathbf{w}$  is assigned to the connections in the network, the network defines a mapping  $\mathbf{y}(\mathbf{x}; \mathbf{w}, \mathcal{A})$  from the input activities  $\mathbf{x}$  to the output activities  $\mathbf{y}$ .<sup>1</sup> The distance of this mapping to the training set is measured by some error function; for example, the error for the entire data set is commonly taken to be

$$E_D(D | \mathbf{w}, \mathcal{A}) = \sum_m \frac{1}{2} [\mathbf{y}(\mathbf{x}^m; \mathbf{w}, \mathcal{A}) - \mathbf{t}^m]^2 \quad (1.1)$$

The task of "learning" is to find a set of connections  $\mathbf{w}$  that gives a mapping that fits the training set well, that is, has small error  $E_D$ ; it is also hoped that the learned connections will "generalize" well to new examples. Plain backpropagation learns by performing gradient descent on  $E_D$  in  $\mathbf{w}$ -space. Modifications include the addition of a "momentum" term, and the inclusion of noise in the descent process. More efficient optimization techniques may also be used, such as conjugate gradients or variable metric methods. This paper will not discuss computational modifications concerned only with speeding the optimization. It will address, however, those modifications to the plain backprop algorithm that implicitly or explicitly modify the objective function, with decay terms or regularizers.

It is moderately common for extra regularizing terms  $E_W(\mathbf{w})$  to be added to  $E_D$ ; for example, terms that penalize large weights may be introduced, in the hope of achieving a smoother or simpler mapping (Hinton and Sejnowski 1986; Ji *et al.* 1990; Nowlan 1991; Rumelhart 1987; Weigend *et al.* 1991). Some of the "hints" in Abu-Mostafa (1990b) also fall into the category of additive weight-dependent energies. A sample weight energy term is

$$E_W(\mathbf{w} | \mathcal{A}) = \sum_i \frac{1}{2} w_i^2 \quad (1.2)$$

---

<sup>1</sup>The framework developed in this paper will apply not only to networks composed of "neurons," but to any regression model for which we can compute the derivatives of the outputs with respect to the parameters,  $\partial \mathbf{y}(\mathbf{x}; \mathbf{w}, \mathcal{A}) / \partial \mathbf{w}$ .

The weight energy may be implicit, for example, “weight decay” (subtraction of a multiple of  $\mathbf{w}$  in the weight change rule) corresponds to the energy in equation 1.2. Gradient-based optimization is then used to minimize the combined function:

$$M = \alpha E_W(\mathbf{w} \mid \mathcal{A}) + \beta E_D(D \mid \mathbf{w}, \mathcal{A}) \quad (1.3)$$

where  $\alpha$  and  $\beta$  are “black box” parameters.

The constant  $\alpha$  should not be confused with the “momentum” parameter sometimes introduced into backprop; in the present context  $\alpha$  is a decay rate or regularizing constant. Also note that  $\alpha$  should not be viewed as causing “forgetting”;  $E_D$  is defined as the error on the entire data set, so gradient descent on  $M$  treats all data points equally irrespective of the order in which they were acquired.

**1.1 What Is Lacking.** The above procedures include a host of free parameters such as the choice of neural network architecture, and of the regularizing constant  $\alpha$ . There are not yet established ways of objectively setting these parameters, though there are many rules of thumb (see Ji *et al.* 1990; Weigend *et al.* 1991, for examples).

One popular way of comparing networks trained with different parameter values is to assess their performance by measuring the error on an unseen test set or by similar cross-validation techniques. The data are divided into two sets, a training set that is used to optimize the parameters  $\mathbf{w}$  of the network, and a test set that is used to optimize control parameters such as  $\alpha$  and the architecture  $\mathcal{A}$ . However, the utility of these techniques in determining values for the parameters  $\alpha$  and  $\beta$  or for comparing alternative network solutions, etc., is limited because a large test set may be needed to reduce the signal-to-noise ratio in the test error, and cross-validation is computationally demanding. Furthermore, if there are several parameters such as  $\alpha$  and  $\beta$ , it is out of the question to optimize such parameters by repeating the learning with all possible values of these parameters and using a test set. Such parameters must be optimized on line.

It is, therefore, interesting to study objective criteria for setting free parameters and comparing alternative solutions, that depend only on the data set used for the training. Such criteria will prove especially important in applications where the total amount of data is limited, so that one does not want to sacrifice good data for use as a test set. Rather, we wish to find a way to use *all* our data in the process of optimizing the parameters  $\mathbf{w}$  and in the process of optimizing control parameters such as  $\alpha$  and  $\mathcal{A}$ .

This paper will describe practical Bayesian methods for filling the following holes in the neural network framework just described:

1. *Objective criteria for comparing alternative neural network solutions, in particular with different architectures  $\mathcal{A}$ .* Given a single architecture

$\mathcal{A}$ , there may be more than one minimum of the objective function  $M$ . If there is a large disparity in  $M$  between the minima then it is plausible to choose the solution with smallest  $M$ . But where the difference is not so great it is desirable to be able to assign an objective preference to the alternatives. It is also desirable to be able to assign preferences to neural network solutions using different numbers of hidden units, and different activation functions. Here there is an "Occam's razor" problem: the more free parameters a model has, the smaller the data error  $E_D$  it can achieve. So we cannot simply choose the architecture with smallest data error. That would lead us to an overcomplex network that generalizes poorly. The use of weight decay does not fully alleviate this problem; networks with too many hidden units still generalize worse, even if weight decay is used (see Section 4).

2. *Objective criteria for setting the decay rate  $\alpha$ .* As in the choice of  $\mathcal{A}$  above, there is an "Occam's razor" problem: a small value of  $\alpha$  in equation 1.3 allows the weights to become large and overfit the noise in the data. This leads to a small value of the data error  $E_D$  (and a small value of  $M$ ), so we cannot base our choice of  $\alpha$  only on  $E_D$  or  $M$ . The Bayesian solution presented here can be implemented on-line, that is, it is not necessary to do multiple learning runs with different values of  $\alpha$  in order to find the best.
3. *Objective choice of regularizing function  $E_W$ .*
4. *Objective criteria for choosing between a neural network solution and a solution using a different learning or interpolation model, for example, splines or radial basis functions.*

**1.2 The Probability Connection.** Tishby *et al.* (1989) introduced a probabilistic view of learning that is an important step toward solving the problems listed above. The idea is to force a probabilistic interpretation onto the neural network technique so as to be able to make objective statements. This interpretation does not involve the addition of any new arbitrary functions or parameters, but it involves assigning a meaning to the functions and parameters that are already used.

My work is based on the same probabilistic framework, and extends it using concepts and techniques adapted from Gull and Skilling's (Gull 1989) Bayesian image reconstruction methods. This paper also adopts a shift in emphasis from Tishby *et al.*'s paper. Their work concentrated on predicting the average generalization ability of one network trained on a task drawn from a known prior ensemble of tasks. This is called *forward probability*. In this paper the emphasis will be on quantifying the relative plausibilities of many alternative solutions to an interpolation or classification task; that task is defined by a single data set produced by the real world, and we do not know the prior ensemble from which the

task comes. This is called *inverse probability*. This paper avoids using the language of statistical physics, in order to maintain wider readability, and to avoid concepts that would sound strange in that language; for example, “the probability distribution of the temperature” is unfamiliar in physics, but “the probability distribution of the noise variance” is its innocent counterpart in literal terms.

Let us now review the probabilistic interpretation of network learning.

- *Likelihood*. A network with specified architecture  $\mathcal{A}$  and connections  $\mathbf{w}$  is viewed as making predictions about the target outputs as a function of input  $\mathbf{x}$  in accordance with the probability distribution:

$$P(\mathbf{t}^m | \mathbf{x}^m, \mathbf{w}, \beta, \mathcal{A}) = \frac{\exp[-\beta E(\mathbf{t}^m | \mathbf{x}^m, \mathbf{w}, \mathcal{A})]}{Z_m(\beta)} \quad (1.4)$$

where  $Z_m(\beta) = \int d\mathbf{t} \exp(-\beta E)$ .  $E$  is the error for a single datum, and  $\beta$  is a measure of the presumed noise included in  $\mathbf{t}$ . If  $E$  is the quadratic error function then this corresponds to the assumption that  $\mathbf{t}$  includes additive gaussian noise with variance  $\sigma_v^2 = 1/\beta$ .

- *Prior*. A prior probability is assigned to alternative network connection strengths  $\mathbf{w}$ , written in the form:

$$P(\mathbf{w} | \alpha, \mathcal{A}, \mathcal{R}) = \frac{\exp[-\alpha E_W(\mathbf{w} | \mathcal{A})]}{Z_W(\alpha)} \quad (1.5)$$

where  $Z_W = \int d^k \mathbf{w} \exp(-\alpha E_W)$ . Here  $\alpha$  is a measure of the characteristic expected connection magnitude. If  $E_W$  is quadratic as specified in equation 1.2 then weights are expected to come from a gaussian with zero mean and variance  $\sigma_w^2 = 1/\alpha$ . Alternative “regularizers”  $\mathcal{R}$  (each using a different energy function  $E_W$ ) implicitly correspond to alternative hypotheses about the statistics of the environment.

- The *posterior probability* of the network connections  $\mathbf{w}$  is then

$$P(\mathbf{w} | D, \alpha, \beta, \mathcal{A}, \mathcal{R}) = \frac{\exp(-\alpha E_W - \beta E_D)}{Z_M(\alpha, \beta)} \quad (1.6)$$

where  $Z_M(\alpha, \beta) = \int d^k \mathbf{w} \exp(-\alpha E_W - \beta E_D)$ . Notice that the exponent in this expression is the same as (minus) the objective function  $M$  defined in equation 1.3.

So under this framework, minimization of  $M = \alpha E_W + \beta E_D$  is identical to finding the (locally) most probable parameters  $\mathbf{w}_{MP}$ ; minimization of  $E_D$  alone is identical to finding the maximum likelihood parameters  $\mathbf{w}_{ML}$ . Thus an interpretation has been given to backpropagation’s energy functions  $E_D$  and  $E_W$ , and to the parameters  $\alpha$  and  $\beta$ . It should be

emphasized that “the probability of the connections  $\mathbf{w}$ ” is a measure of *plausibility* that the model’s parameters should have a specified value  $\mathbf{w}$ ; this has nothing to do with the probability that a particular algorithm might converge to  $\mathbf{w}$ .

This framework offers some partial enhancements for backprop methods: The work of Levin *et al.* (1989) makes it possible to predict the average generalization ability of neural networks trained on one of a defined class of problems. However, it is not clear whether this will lead to a practical technique for choosing between alternative network architectures for real data sets.

Le Cun *et al.* (1990) have demonstrated how to estimate the “saliency” of a weight, which is the change in  $M$  when the weight is deleted. They have used this measure successfully to simplify large neural networks. However no stopping rule for weight deletion was offered other than measuring performance on a test set.

Also Denker and Le Cun (1991) demonstrated how the Hessian of  $M$  can be used to assign error bars to the parameters of a network and to its outputs. However, these error bars can be quantified only once  $\beta$  is quantified, and how to do this without prior knowledge or extra data has not been demonstrated. In fact  $\beta$  can be estimated from the training data alone.

## 2 Review of Bayesian Regularization and Model Comparison \_\_\_\_\_

In the companion paper (MacKay 1992a) it was demonstrated how the control parameters  $\alpha$  and  $\beta$  are assigned by Bayes, and how alternative interpolation models can be compared. It was noted there that it is not satisfactory to optimize  $\alpha$  and  $\beta$  by finding the joint maximum likelihood value of  $\mathbf{w}, \alpha, \beta$ ; the likelihood has a skew peak whose maximum is not located at the most probable values of the control parameters. MacKay (1992a) also reviewed how the Bayesian choice of  $\alpha$  and  $\beta$  is neatly expressed in terms of a measure of the number of well-determined parameters in a model,  $\gamma$ . However that paper assumed that  $M(\mathbf{w})$  has only one significant minimum that was well approximated as quadratic. [All the interpolation models discussed in MacKay (1992a) can be interpreted as two-layer networks with a fixed nonlinear first layer and adaptive linear second layer.] In this section I briefly review the Bayesian framework, retaining that assumption. The following section will then discuss how the framework can be modified to handle neural networks, where the landscape of  $M(\mathbf{w})$  is certainly not quadratic.

**2.1 Determination of  $\alpha$  and  $\beta$ .** By Bayes’ rule, the posterior probability for these parameters is

$$P(\alpha, \beta \mid D, \mathcal{A}, \mathcal{R}) = \frac{P(D \mid \alpha, \beta, \mathcal{A}, \mathcal{R})P(\alpha, \beta)}{P(D \mid \mathcal{A}, \mathcal{R})} \quad (2.1)$$

Now if we assign a uniform prior to  $(\alpha, \beta)$ , the quantity of interest for assigning preferences to  $(\alpha, \beta)$  is the first term on the right-hand side, the *evidence* for  $\alpha, \beta$ , which can be written as<sup>2</sup>

$$P(D \mid \alpha, \beta, \mathcal{A}, \mathcal{R}) = \frac{Z_M(\alpha, \beta)}{Z_W(\alpha)Z_D(\beta)} \quad (2.2)$$

where  $Z_M$  and  $Z_W$  were defined earlier and  $Z_D = \int d^N D e^{-\beta E_D}$ .

Let us use the simple quadratic energy functions defined in equations 1.1 and 1.2. This makes the analysis easier, but more complex cases can still in principle be handled by the same approach. Let the number of degrees of freedom in the data set, that is, the number of output units times the number of data pairs, be  $N$ , and let the number of free parameters, that is, the dimension of  $\mathbf{w}$ , be  $k$ . Then we can immediately evaluate the gaussian integrals  $Z_D$  and  $Z_W$ :  $Z_D = (2\pi/\beta)^{N/2}$ , and  $Z_W = (2\pi/\alpha)^{k/2}$ . Now we want to find  $Z_M(\alpha, \beta) = \int d^k \mathbf{w} \exp[-M(\mathbf{w}, \alpha, \beta)]$ . Supposing for now that  $M$  has a single minimum as a function of  $\mathbf{w}$ , at  $\mathbf{w}_{MP}$ , and assuming we can locally approximate  $M$  as quadratic there, the integral  $Z_M$  is approximated by

$$Z_M \simeq e^{-M(\mathbf{w}_{MP})} (2\pi)^{k/2} \det^{-1/2} \mathbf{A} \quad (2.3)$$

where  $\mathbf{A} = \nabla \nabla M$  is the Hessian of  $M$  evaluated at  $\mathbf{w}_{MP}$ .

The maximum of  $P(D \mid \alpha, \beta, \mathcal{A}, \mathcal{R})$  has the following useful properties:

$$\chi_W^2 \equiv 2\alpha E_W = \gamma \quad (2.4)$$

$$\chi_D^2 \equiv 2\beta E_D = N - \gamma \quad (2.5)$$

where  $\gamma$  is the effective number of parameters determined by the data,

$$\gamma = \sum_{a=1}^k \frac{\lambda_a}{\lambda_a + \alpha} \quad (2.6)$$

where  $\lambda_a$  are the eigenvalues of the quadratic form  $\beta E_D$  in the natural basis of  $E_W$ .

**2.2 Comparison of Different Models.** To rank alternative architectures and penalty functions  $E_W$  in the light of the data, we simply evaluate the evidence,  $P(D \mid \mathcal{A}, \mathcal{R})$ , which appeared as the normalizing constant in equation 2.1. Integrating the evidence for  $(\alpha, \beta)$ , we have:

$$P(D \mid \mathcal{A}, \mathcal{R}) = \int P(D \mid \alpha, \beta, \mathcal{A}, \mathcal{R}) P(\alpha, \beta) d\alpha d\beta \quad (2.7)$$

The evidence is the Bayesian's transportable quantity for comparing models in the light of the data.

---

<sup>2</sup>The same notation, and the same abuses thereof, will be used as in MacKay (1992a).

### 3 Adapting the Framework

---

For neural networks,  $M(\mathbf{w})$  is not quadratic. Indeed it is well known that  $M$  typically has many local minima. And if the network has a symmetry under permutation of its parameters, then we know that  $M(\mathbf{w})$  must share that symmetry, so that every single minimum belongs to a family of symmetric minima of  $M$ . For example, if there are  $H$  hidden units in a single layer then each nondegenerate minimum is in a family of size  $g = H!2^H$ . Now it may be the case that the significant minima of  $M$  are locally quadratic, so we might be able to evaluate  $Z_M$  by evaluating equation 2.3 at each significant minimum and adding up the  $Z_M$ s; but the number of those minima is unknown, and this approach to evaluating  $Z_M$  would seem dubious.

Luckily, however, we do not actually want to evaluate  $Z_M$ . We would need to evaluate  $Z_M$  in order to assign a posterior probability over  $\alpha, \beta$  for an entire model, and to evaluate the evidence for alternative entire models. This is not quite what we wish to do: when we use a neural network to perform a mapping, we typically implement only one neural network at a time, and this network will have its parameters set to a *particular solution* of the learning problem. Therefore, the alternatives we wish to rank are the different solutions of the learning problem, that is, the different minima of  $M$ . We would want the evidence as a function of the number of hidden units only if we were somehow able to simultaneously implement the entire posterior ensemble of networks for one number of hidden units. Similarly, we do not want the posterior over  $\alpha, \beta$  for the entire posterior ensemble; rather, it is reasonable to allow each solution (each minimum of  $M$ ) to choose its own optimal value for these parameters. The same method of chopping up a complex model space is used in the unsupervised classification system, AutoClass (Hanson *et al.* 1991).

Having adopted this slight shift in objective, it turns out that to set  $\alpha$  and  $\beta$  and to compare alternative solutions to a learning problem, the integral we now need to evaluate is a local version of  $Z_M$ . Assume that the posterior probability consists of well-separated islands in parameter space each centered on a minimum of  $M$ . We wish to evaluate how much posterior probability mass is in each of these islands. Consider a minimum located at  $\mathbf{w}^*$ , and define a solution  $S_{\mathbf{w}^*}$  as the ensemble of networks in the neighborhood of  $\mathbf{w}^*$ , and all symmetric permutations of that ensemble. Let us evaluate the posterior probability for alternative solutions  $S_{\mathbf{w}^*}$ , and the parameters  $\alpha$  and  $\beta$ :

$$P(S_{\mathbf{w}^*}, \alpha, \beta, \mathcal{A}, \mathcal{R} \mid D) \propto g \frac{Z_M^*(\mathbf{w}^*, \alpha, \beta)}{Z_W(\alpha)Z_D(\beta)} P(\alpha, \beta) P(\mathcal{A}, \mathcal{R}) \quad (3.1)$$

where  $g$  is the permutation factor, and

$$Z_M^*(\mathbf{w}^*, \alpha, \beta) = \int_{S_{\mathbf{w}^*}} d^k \mathbf{w} \exp[-M(\mathbf{w}, \alpha, \beta)]$$



where the integral is performed only over the neighborhood of the minimum at  $\mathbf{w}^*$ . I will refer to the quantity  $g[Z_M^*(\mathbf{w}^*, \alpha, \beta)/Z_W(\alpha)Z_D(\beta)]$  as the evidence for  $\alpha, \beta, S_{\mathbf{w}^*}$ . The parameters  $\alpha$  and  $\beta$  will be chosen to maximize this evidence. Then the quantity we want to evaluate to compare alternative solutions is the evidence<sup>3</sup> for  $S_{\mathbf{w}^*}$ ,

$$P(D, S_{\mathbf{w}^*} \mid \mathcal{A}, \mathcal{R}) = \int g \frac{Z_M^*(\mathbf{w}^*, \alpha, \beta)}{Z_W(\alpha)Z_D(\beta)} P(\alpha, \beta) d\alpha d\beta \quad (3.2)$$

This paper uses the gaussian approximation for  $Z_M^*$ :

$$Z_M^* \simeq e^{-M(\mathbf{w}^*)} (2\pi)^{k/2} \det^{-1/2} \mathbf{A} \quad (3.3)$$

where  $\mathbf{A} = \nabla \nabla M$  is the Hessian of  $M$  evaluated at  $\mathbf{w}^*$ . For general  $\alpha$  and  $\beta$  this approximation is probably unacceptable; however, we need it only to be accurate for the small range of  $\alpha$  and  $\beta$  close to their most probable value. The regime in which this approximation will definitely break down is when the number of constraints,  $N$ , is small relative to the number of free parameters,  $k$ . For large  $N/k$  the central limit theorem encourages us to use the gaussian approximation (Walker 1967). It is a matter for further research to establish how large  $N/k$  must be for this approximation to be reliable.

What obstacles remain to prevent us from evaluating the local  $Z_M^*$ ? We need to evaluate or approximate the inverse Hessian of  $M$ , and we need to evaluate or approximate its determinant and/or trace (MacKay 1992a).

Denker and Le Cun (1991) and Le Cun *et al.* (1990) have already discussed how to approximate the Hessian of  $E_D$  for the purpose of evaluating weight saliency and for assigning error bars to weights and network outputs. The Hessian can be evaluated in the same way that backpropagation evaluates  $\nabla E_D$  (see Bishop 1992 for a complete algorithm and the appendix of this paper for a useful approximation). Alternatively  $\mathbf{A}$  can be evaluated by numerical methods, for example second differences. A third option: if variable metric methods are used to minimize  $M$  instead of gradient descent, then the inverse Hessian is automatically generated during the search for the minimum. It is important, for the success of this Bayesian method, that the off-diagonal terms of the Hessian should be evaluated. Denker *et al.*'s method can do this without any additional complexity. The diagonal approximation is no good because of the strong posterior correlations in the parameters.

<sup>3</sup>Bayesian model comparison is performed by evaluating and comparing the evidence for alternative models. Gull and Skilling defined the evidence for a model  $\mathcal{H}$  to be  $P(D \mid \mathcal{H})$ . The existence of multiple minima in neural network parameter space complicates model comparison. The quantity in equation 3.2 is not  $P(D \mid S_{\mathbf{w}^*}, \mathcal{A}, \mathcal{R})$  (it includes the prior for  $S_{\mathbf{w}^*} \mid \mathcal{A}, \mathcal{R}$ ), but I have called it the evidence because it is the quantity we should evaluate to compare alternative solutions with each other and with other models.

## 4 Demonstration

This demonstration examines the evidence for various neural net solutions to a small interpolation problem, the mapping for a two joint robot arm,

$$(\theta_1, \theta_2) \rightarrow (y_a, y_b) = r_1 \cos \theta_1 + r_2 \cos(\theta_1 + \theta_2), r_1 \sin \theta_1 + r_2 \sin(\theta_1 + \theta_2)$$

For the training set I used  $r_1 = 2.0$  and  $r_2 = 1.3$ , random samples from a restricted range of  $(\theta_1, \theta_2)$  were made, and gaussian noise of magnitude 0.05 was added to the outputs. The neural nets used had one hidden layer of sigmoid units and linear output units. During optimization, the regularizer (equation 1.2) was used initially, and an alternative regularizer was introduced later;  $\beta$  was fixed to its true value (to enable demonstration of the properties of the quantity  $\gamma$ ), and  $\alpha$  was allowed to adapt to its locally most probable value.

Figure 1 illustrates the performance of a typical neural network trained in this way. Each output is accompanied by error bars evaluated using Denker *et al.*'s method, including off-diagonal Hessian terms. If  $\beta$  had not been known in advance, it could have been inferred from the data using equation 2.5. For the solution displayed, the model's estimate of  $\beta$  in fact differed negligibly from the true value, so the displayed error bars are the same as if  $\beta$  had been inferred from the data.

Figure 2 shows the data misfit versus the number of hidden units. Notice that, as expected, the data error tends to decrease monotonically with increasing number of parameters. Figure 3 shows the error of these same solutions on an unseen test set, which does not show the same trend as the data error. The data misfit cannot serve as a criterion for choosing between solutions.

Figure 4 shows the evidence for about 100 different solutions using different numbers of hidden units. Notice how the evidence maximum has the characteristic shape of an "Occam hill" — steep on the side with too few parameters, and shallow on the side with too many parameters. The quadratic approximations break down when the number of parameters becomes too big compared with the number of data points.

Figure 5 introduces the quantity  $\gamma$ , discussed in MacKay (1992a), the number of well-measured parameters. In cases where the evaluation of the evidence proves difficult, it may be that  $\gamma$  will serve as a useful tool. For example, sampling theory predicts that the addition of redundant parameters to a model should reduce  $\chi_D^2$  by one unit per well-measured parameter; a stopping criterion could detect the point at which, as parameters are deleted,  $\chi_D^2$  started to increase faster than with gradient 1 with decreasing  $\gamma$  (Figure 6).<sup>4</sup> This use of  $\gamma$  requires prior knowledge of the noise level  $\beta$ ; that is why  $\beta$  was fixed to its known value for these demonstrations.

<sup>4</sup>This suggestion is closely related to Moody's (1991) "generalized prediction error,"  $GPE = (\chi_D^2 + 2\gamma)/N$ .

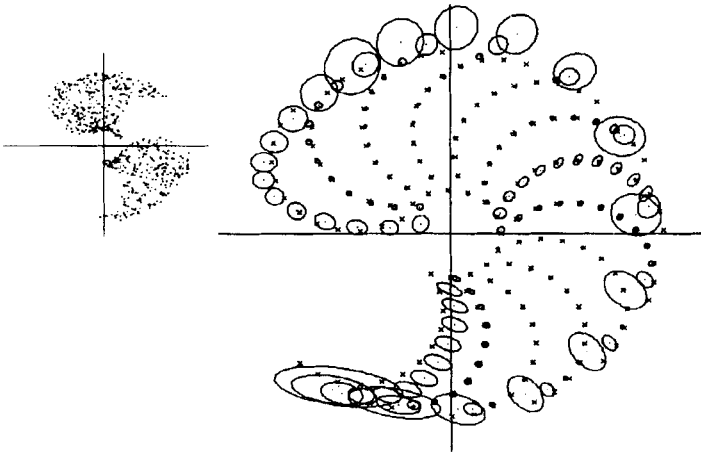


Figure 1: Typical neural network output (inset — training set). This is the output space ( $y_a, y_b$ ) of the network. The target outputs are displayed as small x's, and the output of the network with  $1\sigma$  error bars is shown as a dot surrounded by an ellipse. The network was trained on samples in two regions in the lower and upper half planes (inset). The outputs illustrated here are for inputs extending a short distance outside the training regions, and bridging the gap between them. Notice that the error bars get much larger around the perimeter. They also increase slightly in the gap between the training regions. These pleasing properties would not have been obtained had the diagonal Hessian approximation of Denker and Le Cun (1991) been used. The above solution was created by a three layer network with 19 hidden units.

Now the question is how good a predictor of network quality the evidence is. The fact that the evidence has a maximum at a reasonable number of hidden units is promising. A comparison with Figure 3 shows that the performance of the solutions on an unseen test set has similar overall structure to the evidence. However, Figure 7 shows the evidence against the performance on a test set, and it can be seen that a significant number of solutions with poor evidence actually perform well on the test set. Something is wrong! It is time for a discussion of the relationship between the evidence and generalization ability. We will return later to the failure in Figure 7 and see that it is rectified by the development of new, more probable regularizers.

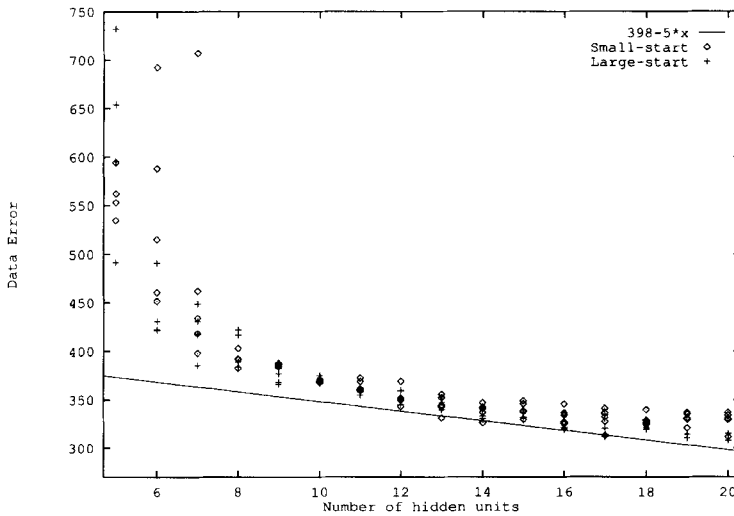


Figure 2: Data error versus number of hidden units. Each point represents one converged neural network, trained on a 200 i/o pair training set. Each neural net was initialized with different random weights and with a different initial value of  $\sigma_W^2 = 1/\alpha$ . The two point-styles correspond to small and large initial values for  $\sigma_W$ . The error is shown in dimensionless  $\chi^2$  units such that the expectation of error relative to the truth is  $400 \pm 20$ . The solid line is  $400 - k$ , where  $k$  is the number of free parameters.

**4.1 Relation to "Generalization Error".** What is the relationship between the evidence and the generalization error (or its close relative, cross-validation)? A correlation between the two is certainly expected. But the evidence is not necessarily a good predictor of generalization error (see discussion in MacKay, 1992a). First, as illustrated in Figure 8, the error on a test set is a noisy quantity, and many data have to be devoted to the test set to get an acceptable signal-to-noise ratio. Furthermore, imagine that two models have generated solutions to an interpolation problem, and that their two most probable interpolants are completely identical. In this case, the generalization error for the two solutions must be the same, but the evidence will not in general be the same: typically, the model that was a priori more complex will suffer a larger Occam factor and will have smaller evidence. Also, the evidence is a measure of plausibility of the whole ensemble of networks about the optimum, not

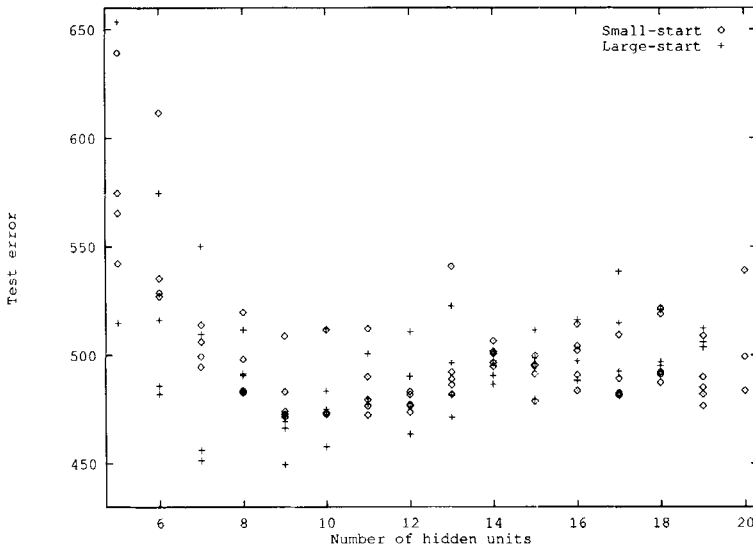


Figure 3: Test error versus number of hidden units. The training set and test set both had 200 data points. The test error for solutions found using the first regularizer is shown in dimensionless  $\chi^2$  units such that the expectation of error relative to the truth is  $400 \pm 20$ .

just the optimal network. Thus, there is more to the evidence than there is to the generalization error.

**4.2 What If the Bayesian Method Fails?** I do not want to dismiss the utility of the generalization error: it can be important for detecting failures of the model being used. For example, if we obtain a poor correlation between the evidence and the generalization error, such that Bayes fails to assign a strong preference to solutions that actually perform well on test data, then we are able to detect and attempt to correct such failures.

A failure indicates one of two things, and in either case we are able to learn and improve: either numerical inaccuracies in the evaluation of the probabilities caused the failure, or else the alternative models that were offered to Bayes were a poor selection, ill-matched to the real world (for example, using inappropriate regularizers). When such a failure is detected, it prompts us to examine our models and try to discover the implicit assumptions in the model that the data did not agree with;

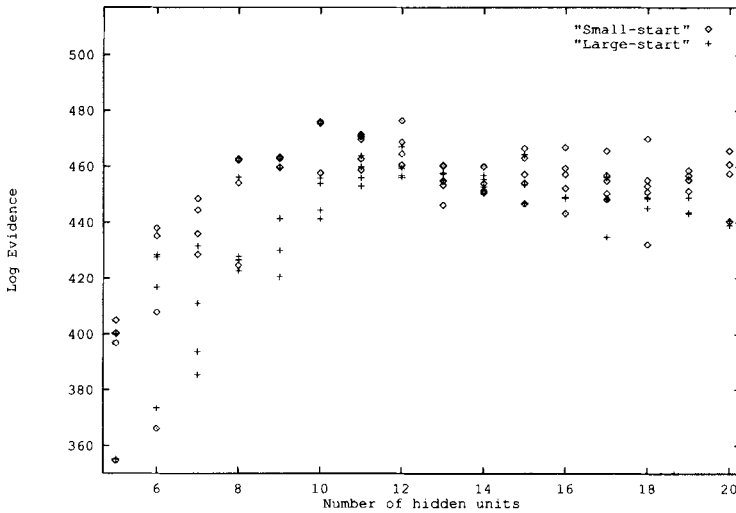


Figure 4: Log evidence for solutions using the first regularizer. For each solution, the evidence was evaluated. Notice that an evidence maximum is achieved by neural network solutions using 10, 11, and 12 hidden units. For more than  $\sim 19$  hidden units, the quadratic approximations used to evaluate the evidence are believed to break down. The number of data points  $N$  is 400 (i.e., 200 i/o pairs); cf. number of parameters in a net with 20 hidden units = 102.

alternative models can be tried until one is found that makes the data more probable.

We have just met exactly such a failure. Let us now establish what assumption in our model caused this failure and *learn* from it. Note that this mechanism for human learning is not available to those who just use the test error as their performance criterion. Going by the test error alone, there would have been no indication that there was a serious mismatch between the model and the data.

### 4.3 Back to the Demonstration: Comparing Different Regularizers.

The demonstrations thus far used the regularizer of equation 1.2. This is equivalent to a prior that expects all the weights to have the same characteristic size. This is actually an inconsistent prior: the input and output variables and hidden unit activities could all be arbitrarily rescaled; if the same mapping is to be performed (a simple consistency requirement), such transformations of the variables would imply *independent* rescaling

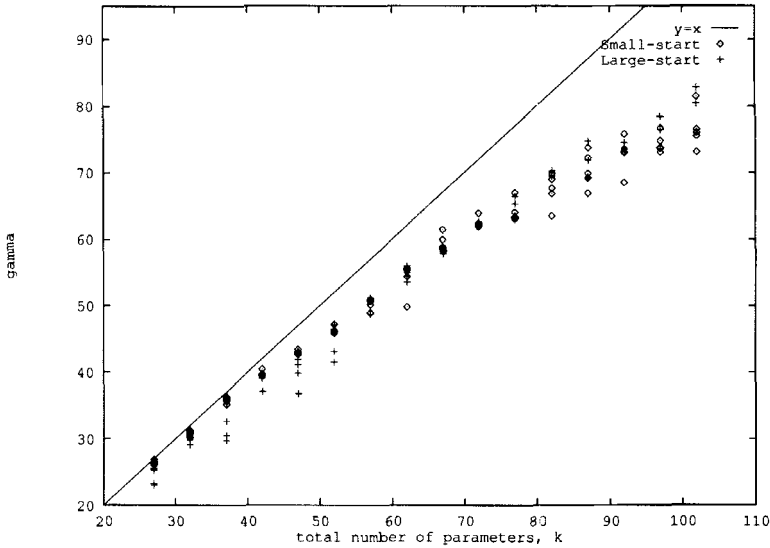


Figure 5: The number of well-determined parameters. This figure displays  $\gamma$  as a function of  $k$ , for the same network solutions as in Figure 4.

of the weights to the hidden layer and to the output layer. Thus the scales of the two layers of weights are unrelated, and it is inconsistent to force the characteristic decay rates of these different classes of weights to be the same. This inconsistency is the major cause of the failure illustrated in Figure 7. *All the networks deviating substantially from the desired trend have weights to the output layer far larger than the weights to the input layer; this poor match to the model implicit in the regularizer causes the evidence for those solutions to be small.*

This failure enables us to progress with insight to new regularizers. The alternative that I now present is a prior that is not inconsistent in the way explained above, so there are theoretical reasons to expect it to be "better." However, we will allow the data to choose, by evaluating the evidence for solutions using the new prior; we will find that the new prior is indeed *more probable*.

The second prior has three independent regularizing constants, corresponding to the characteristic magnitudes of the weights in three different classes  $c$ , namely hidden unit weights, hidden unit biases, and output weights and biases (see Fig. 9). The term  $\alpha E_W$  is replaced by  $\sum_c \alpha_c E_W^c$ , where  $E_W^c = \sum_{i \in c} w_i^2 / 2$ . Nowlan (1991) has used a similar prior modeling

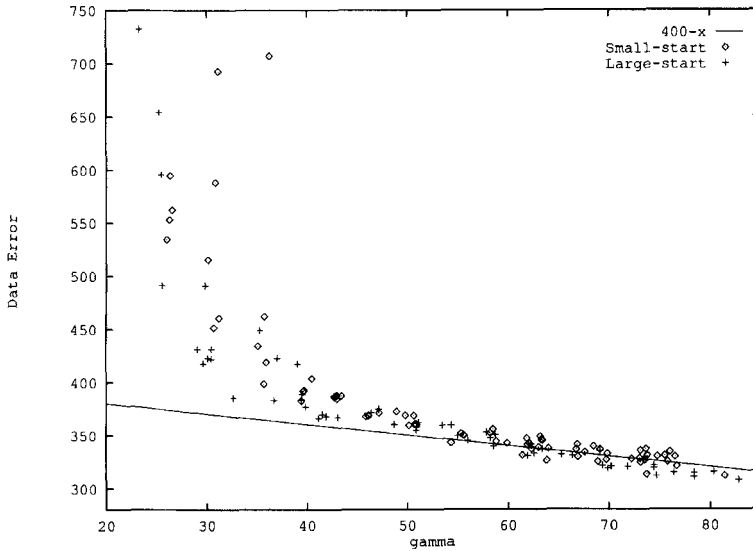


Figure 6: Data misfit versus  $\gamma$ . This figure shows  $\chi_D^2$  against  $\gamma$ , and a line of gradient  $-1$ . Toward the right, the data's misfit  $\chi_D^2$  is reduced by 1 for every well-measured parameter. When the model has too few parameters, however (toward the left), the misfit gets worse at a greater rate.

weights as coming from a gaussian mixture, and used Bayesian reestimation techniques to update the mixture parameters; he found such a model was good at discovering elegant solutions to problems with translation invariances.

Using the second prior, each regularizing constant is independently adapted to its most probable value by evaluating the number of well-measured parameters  $\gamma_c$  associated with each regularizing function, and finding the optimum where  $2\alpha_c E_W^c = \gamma_c$ . The increased complexity of this prior model is penalized by an Occam factor for each new parameter  $\alpha_c$  (see MacKay 1992a). Let me preempt questions along the lines of "why didn't you use four weight classes, or nonzero means?" — any other way of assigning weight decays is just another model, and you can try as many as you like; by evaluating the evidence you can then find out what preference the data have for the alternative decay schemes.

New solutions have been found using this second prior, and the evidence evaluated. The evidence for these new solutions with the new prior is shown in Figure 10. Notice that the evidence has increased com-



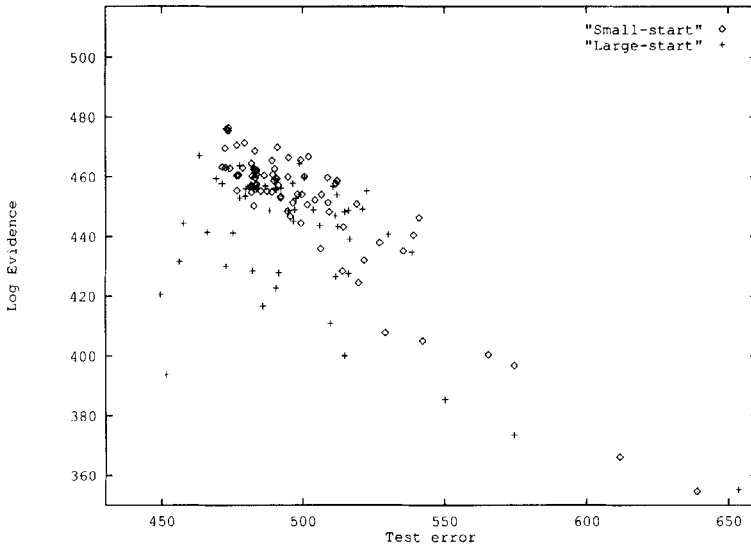


Figure 7: Log evidence versus test error for the first regularizer. The desired correlation between the evidence and the test error has negative slope. A significant number of points on the lower left violate this desired trend, so we have a failure of Bayesian prediction. The points that violate the trend are networks in which there is a significant difference in typical weight magnitude between the two layers. They are all networks whose learning was initialized with a large value of  $\sigma_W$ . The first regularizer is ill-matched to such networks, and the low evidence is a reflection of this poor prior hypothesis.

pared to the evidence for the first prior. For some solutions the new prior is more probable by a factor of  $10^{30}$ .

Now the crunch: does this more probable model make good predictions? The evidence for the second prior is shown against the test error in Figure 11. The correlation between the two is greatly improved. Notice furthermore that not only is the second prior more probable, the best test error achieved by solutions found using the second prior is slightly better than any achieved using the first prior, and the number of good solutions has increased substantially. Thus the Bayesian evidence is a good predictor of generalization ability, and the Bayesian choice of regularizers has enabled the best solutions to be found.

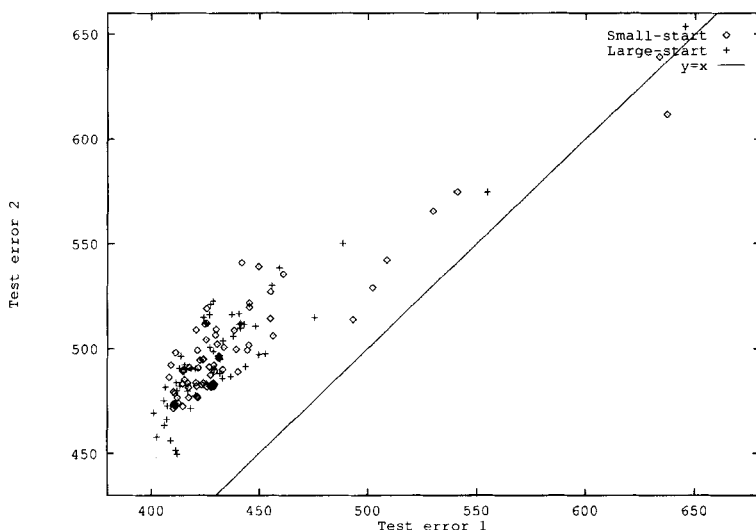


Figure 8: Comparison of two test errors. This figure illustrates how noisy a performance measure the test error is. Each point compares the error of a trained network on two different test sets. Both test sets consist of 200 data points from the same distribution as the training set.

## 5 Discussion

The Bayesian method that has been presented is well-founded theoretically, and it works practically, though it remains to be seen how this approach will scale to larger problems. For a particular data set, the evaluation of the *evidence* has led us objectively from an inconsistent regularizer to a more probable one. The evidence is maximized for a sensible number of hidden units, showing that Occam's razor has been successfully embodied with no ad hoc terms. Furthermore the solutions with greatest evidence perform better on a test set than any other solutions found. I believe there is currently no other technique that could reliably find and identify better solutions using only the training set. Essential to this success was the simultaneous Bayesian optimization of the three regularizing constants (decay terms)  $\alpha_c$ . Optimization of these parameters by any orthodox search technique such as cross-validation would be laborious; if there were many more than three regularizing constants,

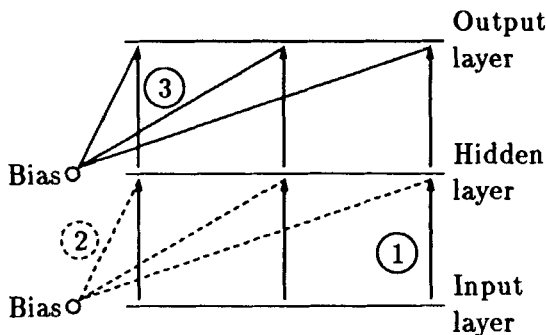


Figure 9: The three classes of weights under the second prior. (1) Hidden unit weights. (2) Hidden unit biases. (3) Output unit weights and biases. The weights in one class  $c$  share the same decay constant  $\alpha_c$ .

as could easily be the case in larger problems, it is hard to imagine any such search being possible.<sup>5</sup>

This brings up the question of how these Bayesian calculations scale with problem size. In terms of the number of parameters  $k$ , calculation of the determinant and inverse of the Hessian scales as  $k^3$ . Note that this is a computation that needs to be carried out only a small number of times compared with the immense number of derivative calculations involved in a typical learning session. However, for large problems it may be too demanding to evaluate the determinant of the Hessian. If this is the case, numerical methods are available to approximate the determinant or trace of a matrix in  $k^2$  time (Skilling, 1989).

**5.1 Application to Classification Problems.** This paper has thus far discussed the evaluation of the evidence for backprop networks trained on interpolation problems. Neural networks can also be trained to per-

<sup>5</sup>Radford Neal (personal communication) has pointed out that it is possible to evaluate the gradient of a validation error with respect to parameters such as  $\{\alpha_c\}$ , using  $\partial E_{\text{val}}/\partial \alpha_c = \partial E_{\text{val}}/\partial \mathbf{w}_{\text{MP}} \cdot \partial \mathbf{w}_{\text{MP}}/\partial \alpha_c$ . The first quantity could be evaluated by backprop, and the second term could be found within the quadratic approximation which gives  $\partial \mathbf{w}_{\text{MP}}/\partial \alpha_c = \mathbf{A}^{-1} \mathbf{I}_c \mathbf{w}_{\text{MP}}$ , where  $\mathbf{I}_c$  is the identity matrix for the weights regularized by  $\alpha_c$ , and zero elsewhere. Alternatively, Radford Neal has suggested that the gradients  $\partial E_{\text{val}}/\partial \alpha_c$  could be more efficiently calculated using "recurrent backpropagation" (Pineda 1989), viewing  $\mathbf{w}$  as the vector of activities of a recurrent network, and  $\mathbf{w}_{\text{MP}}$  as the fixed point whose error  $E_{\text{val}}$  we wish to minimize.

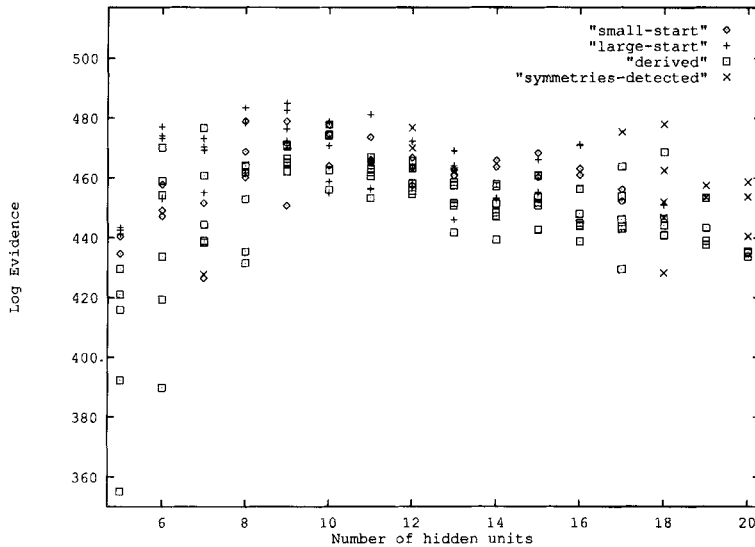


Figure 10: Log evidence versus number of hidden units for the second prior. The different point styles correspond to networks with learning initialized with small and large values of  $\sigma_W$ ; networks previously trained using the first regularizer and subsequently trained on the second regularizer; and networks in which a weight symmetry was detected (in such cases the evidence evaluation is possibly less reliable).

form classification tasks. A future publication (MacKay 1992b) will demonstrate that the Bayesian framework for model comparison can be applied to these problems too.

**5.2 Relation to V-C Dimension.** Some papers advocate the use of V-C dimension (Abu-Mostafa 1990a) as a criterion for penalizing over-complex models (Abu-Mostafa 1990b; Lee and Tenorio 1991). V-C dimension is most often applied to classification problems; the evidence, on the other hand, can be evaluated equally easily for interpolation and classification problems. V-C dimension is a worst case measure, so it yields different results from Bayesian analysis (Haussler *et al.* 1991). For example, V-C dimension is indifferent to the use of regularizers like equation 1.2, and to the value of  $\alpha$ , because the use of such regularizers does not rule out absolutely any particular network parameters. Thus V-C dimension assigns the same complexity to a model whether or not it is

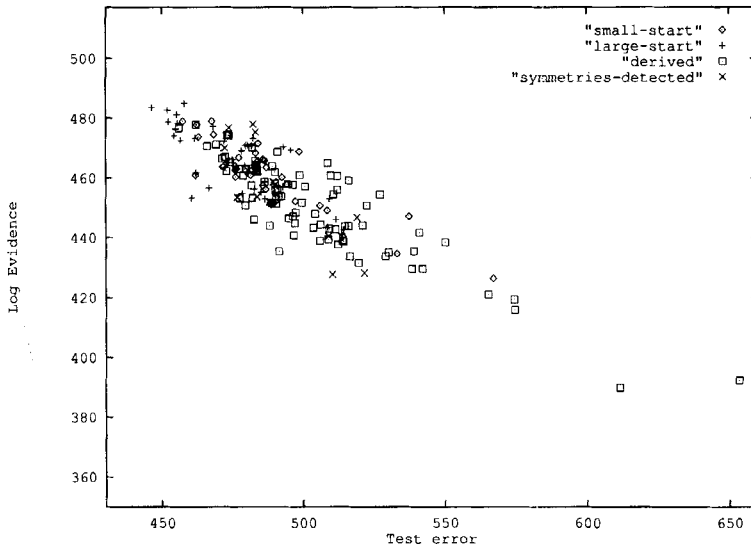


Figure 11: Log evidence for the second prior versus test error. The correlation between the evidence and the test error for the second prior is very good. Note that the largest value of evidence has increased relative to Figure 7, and the smallest test error has also decreased.

regularized.<sup>6</sup> So it cannot be used to set regularizing constants  $\alpha$  or to compare alternative regularizers. In contrast, the preceding demonstrations show that careful objective choice of regularizer and  $\alpha$  is essential for the best solutions to be obtained.

Worst case analysis has a complementary role alongside Bayesian methods. Neither can substitute for the other.

**5.3 Future Tasks.** Further work is needed to formalize the relationship of this framework to the pragmatic model comparison technique of cross-validation. Moody's (1991) work on "generalized prediction error" (GPE) is an interesting contribution in this direction. His sampling theory approach predicts that the generalization error, in  $\chi^2$  units, will be  $(\chi_D^2 + 2\gamma)/N$ . However, I have evaluated the GPE for the interpolation models in this paper's demonstration, and found the correlation

<sup>6</sup>However, E. Levin (personal communication) has mentioned that a measure of "effective V-C dimension" of a regularized model is being developed. In some cases this measure is identical to  $\gamma$ , equation 2.6.

between GPE and the actual test error was poor. More work is needed to understand this.

The gaussian approximation used to evaluate the evidence breaks down when the number of data points is small compared to the number of parameters. For the model problems I have studied so far, the gaussian approximation seemed to break down significantly for  $N/k < 3 \pm 1$ . It is a matter for further research to characterize this failure and investigate techniques for improving the evaluation of the integral  $Z_M^*$ , for example the use of random walks on  $M$  in the neighborhood of a solution.

It is expected that evaluation of the evidence should provide an objective rule for deciding whether a network pruning or growing procedure should be stopped, but a careful study of this idea has yet to be performed.

It will be interesting to see the results of evaluating the evidence for networks applied to larger real-world problems.

## 6 Appendix: Numerical Methods

**6.1 Quick and Dirty Version.** The three numerical tasks are automatic optimization of  $\alpha_c$  and  $\beta$ , calculation of error bars, and evaluation of the evidence. I will describe a cheap approximation for solving the first of these tasks without evaluating the Hessian. If we neglect the distinction between well-determined and poorly determined parameters, we obtain the following update rules for  $\alpha$  and  $\beta$ :

$$\begin{aligned}\alpha_c &:= k_c/2E_W^c \\ \beta &:= N/2E_D\end{aligned}$$

If you want an easy-to-program taste of what a Bayesian framework can offer, try using this procedure to update your decay terms.

**6.2 Hessian Evaluation.** The Hessian of  $M$ ,  $\mathbf{A}$ , is needed to evaluate  $\gamma$  (which relates to  $\text{Trace}\mathbf{A}^{-1}$ ), to evaluate the evidence (which relates to  $\det \mathbf{A}$ ), and to assign error bars to network outputs (using  $\mathbf{A}^{-1}$ ).

I used two methods for evaluating  $\mathbf{A}$ : (1) an approximate analytic method and (2) second differences. The approximate analytic method was, following Denker *et al.*, to use backprop to obtain the second derivatives, neglecting terms in  $f''$ , where  $f$  is the activation function of a neuron. The Hessian is built up as a sum of outer products of gradient vectors:

$$\nabla\nabla E_D \simeq \sum_{i,m} \mathbf{g}_i^m \mathbf{g}_i^{mT} \quad (6.1)$$

where  $\mathbf{g}_i^m = dy_i(\mathbf{x}^m)/d\mathbf{w}$ . Unlike Denker *et al.*, I did not ignore the off-diagonal terms; the diagonal approximation is not good enough! For

the evaluation of  $\gamma$  the two methods gave similar results, and either approach seemed satisfactory. However, for the evaluation of the evidence, the approximate analytic method failed to give satisfactory results. The "Occam factors" are very weak, scaling only as  $\log N$ , and the above approximation apparently introduces systematic errors greater than these. The reason that the evidence evaluation is more sensitive to errors than the  $\gamma$  evaluation is because  $\gamma$  is related to the sum of eigenvalues, whereas the evidence is related to the product; errors in small eigenvalues jeopardize the product more than the sum. I expect an exact analytic evaluation of the second derivatives (Bishop 1992) would resolve this. To save programming effort I instead used second differences, which is computationally more demanding ( $\sim kN$  backprops) than the analytic approach ( $\sim N$  backprops). There were still problems with errors in small eigenvalues, but it was possible to correct these errors, by detecting eigenvalues that were smaller than theoretically permitted.

**6.3 Demonstrations.** The demonstrations were performed as follows: Initial weight configuration: random weights drawn from a gaussian with  $\sigma_W = 0.3$ . Optimization algorithm for  $M(\mathbf{w})$ : variable metric methods, using code from Press *et al.* (1988), used several times in sequence with values of the fractional tolerance decreasing from  $10^{-4}$  to  $10^{-8}$ . Every other loop, the regularizing constants  $\alpha_c$  were allowed to adapt in accordance with the reestimation formula:

$$\alpha_c := \gamma_c / 2E_W^c \quad (6.2)$$

**6.4 Precaution.** When evaluating the evidence, care must be taken to verify that the permutation term  $g$  is appropriately set. It may be the case (probably mainly in toy problems) that the regularizer makes two or more hidden units in a network adopt identical connection values; alternatively, some hidden units might switch off, with all weights set to zero; in these cases the permutation term should be smaller. Also in these cases, it is likely that the quadratic approximation will perform badly (quartic rather than quadratic minima are likely), so it is preferable to automate the deletion of such redundant units.

## Acknowledgments

---

I thank Mike Lewicki, Nick Weir, and Haim Sompolsky for helpful conversations, and Andreas Herz for comments on the manuscript. This work was supported by a Caltech Fellowship and a Studentship from SERC, UK.

## References

---

- Abu-Mostafa, Y. S. 1990a. The Vapnik–Chervonenkis dimension: Information versus complexity in learning. *Neural Comp.* 1(3), 312–317.
- Abu-Mostafa, Y. S. 1990b. Learning from hints in neural networks. *J. Complex.* 6, 192–198.
- Bishop, C. M. 1992. Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Comp.*, in press.
- Denker, J. S., and Le Cun, Y. 1991. Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems 3*, R. P. Lippmann *et al.*, eds., pp. 853–859. Morgan Kaufmann, San Mateo, CA.
- Gull, S. F. 1989. Developments in maximum entropy data analysis. In *Maximum Entropy and Bayesian Methods*, Cambridge, 1988, J. Skilling, ed., pp. 53–71. Kluwer, Dordrecht.
- Hanson, R., Stutz, J., and Cheeseman, P. 1991. Bayesian classification theory. NASA Ames TR FIA-90-12-7-01.
- Haussler, D., Kearns, M., and Schapire, R. 1991. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. In *Proceedings of the Fourth COLT Workshop*. Morgan Kaufmann, San Mateo, CA.
- Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing*, G. E. Rumelhart *et al.*, ed., pp. 282–317. MIT Press, Cambridge, MA.
- Ji, C., Snapp, R. R., and Psaltis, D. 1990. Generalizing smoothness constraints from discrete samples. *Neural Comp.* 2(2), 188–197.
- Le Cun, Y., Denker, J. S., and Solla, S. S. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed., pp. 598–605. Morgan Kaufmann, San Mateo, CA.
- Lee, W. T., and Tenorio, M. F. 1991. On optimal adaptive classifier design criterion — How many hidden units are necessary for an optimal neural network classifier? Purdue University TR-EE-91-5.
- Levin, E., Tishby, N., and Solla, S. 1989. A statistical approach to learning and generalization in layered neural networks. In *COLT '89: 2nd Workshop on Computational Learning Theory*, pp. 245–260. Morgan Kaufmann, San Mateo, CA.
- MacKay, D. J. C. 1992a. Bayesian interpolation. *Neural Comp.* 4(3), 415–447.
- MacKay, D. J. C. 1992b. The evidence framework applied to classification networks. *Neural Comp.*, to appear.
- Moody, J. E. 1991. Note on generalization, regularization and architecture selection in nonlinear learning systems. In *First IEEE-SP Workshop on Neural Networks for Signal Processing*. IEEE Computer Society Press, New York.
- Nowlan, S. J. 1991. Soft competitive adaptation: Neural network learning algorithms based on fitting statistical mixtures. Carnegie Mellon University Doctoral thesis CS-91-126.
- Pineda, F. J. 1989. Recurrent back-propagation and the dynamical approach to adaptive neural computation. *Neural Comp.* 1, 161–172.



- Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. 1988. *Numerical recipes in C*. Cambridge University Press, Cambridge.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning representations by back propagating errors. *Nature (London)* **323**, 533–536.
- Rumelhart, D. E. 1987. Cited in Ji *et al.* 1990.
- Skilling, J. 1989. The eigenvalues of mega-dimensional matrices. In *Maximum Entropy and Bayesian Methods, Cambridge, 1988*, J. Skilling, ed., pp. 455–466. Kluwer, Dordrecht.
- Tishby, N., Levin, E., and Solla, S. A. 1989. Consistent inference of probabilities in layered networks: Predictions and generalization. In *Proc. IJCNN*, Washington.
- Walker, A. M. 1967. On the asymptotic behaviour of posterior distributions. *J. R. Stat. Soc. B* **31**, 80–88.
- Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. 1991. Generalization by weight-elimination with applications to forecasting. In *Advances in Neural Information Processing Systems 3*, R. P. Lippmann *et al.*, ed., pp. 875–882. Morgan Kaufmann, San Mateo, CA.

---

Received 21 May 1991; accepted 29 October 1991.