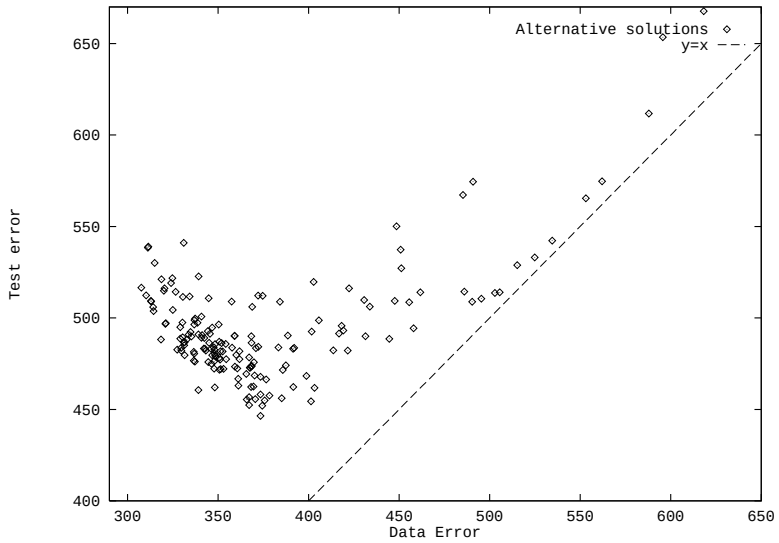

Figure 3.4: **Test error vs. data error.**
Each point represents the performance of a single trained neural network on the training set and on the test set. The horizontal axis displays the data error, that is, the network's performance on the training data. A small value of data error corresponds to a network that has learnt the training data well. The vertical axis displays the test error, that is, how well each network generalises to new examples. The smaller the test error, the better the generalisation ability of the network.

This graph illustrates the 'Occam problem' that the best generalisation is not achieved by the models which fit the training data best.

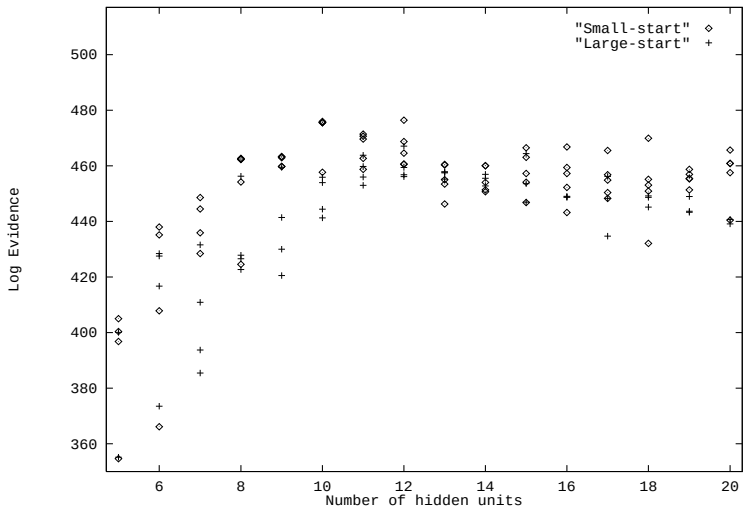

Figure 3.5: **Log Evidence for solutions using the first regulariser.**
For each solution, the evidence was evaluated. Notice that an evidence maximum is achieved by neural network solutions using 10, 11 and 12 hidden units. For more than $\sim$ 19 hidden units, the quadratic approximations used to evaluate the evidence are believed to break down. The number of data points N is 400 (*i.e.* 200 i/o pairs); *c.f.* number of parameters in a net with 20 hidden units = 102.

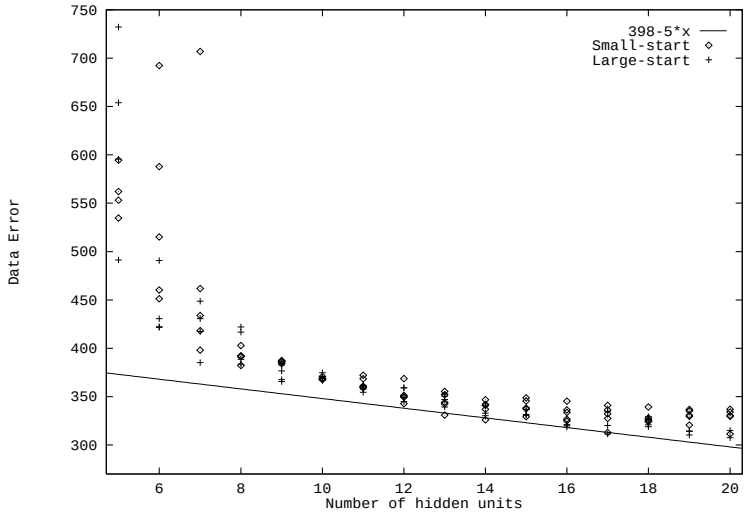

Figure 3.2: **Data error versus number of hidden units.**
Each point represents one converged neural network, trained on a 200 i/o pair training set. Each neural net was initialised with different random weights and with different values for the initial value of $\sigma_W^2 = 1/\alpha$. The two point–styles correspond to small and large initial values for σ_W. The error is shown in dimensionless χ^2 units such that the expectation of error relative to the truth is 400 ± 20. The solid line is $400 - k$, where k is the number of free parameters.

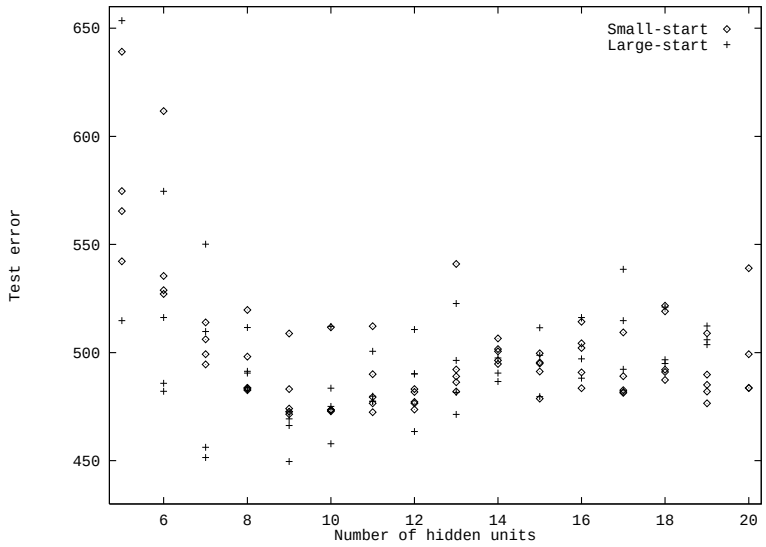

Figure 3.3: **Test error versus number of hidden units**
The training set and test set both had 200 data points. The test error for solutions found using the first regulariser is shown in dimensionless χ^2 units such that the expectation of error relative to the truth is 400 ± 20.

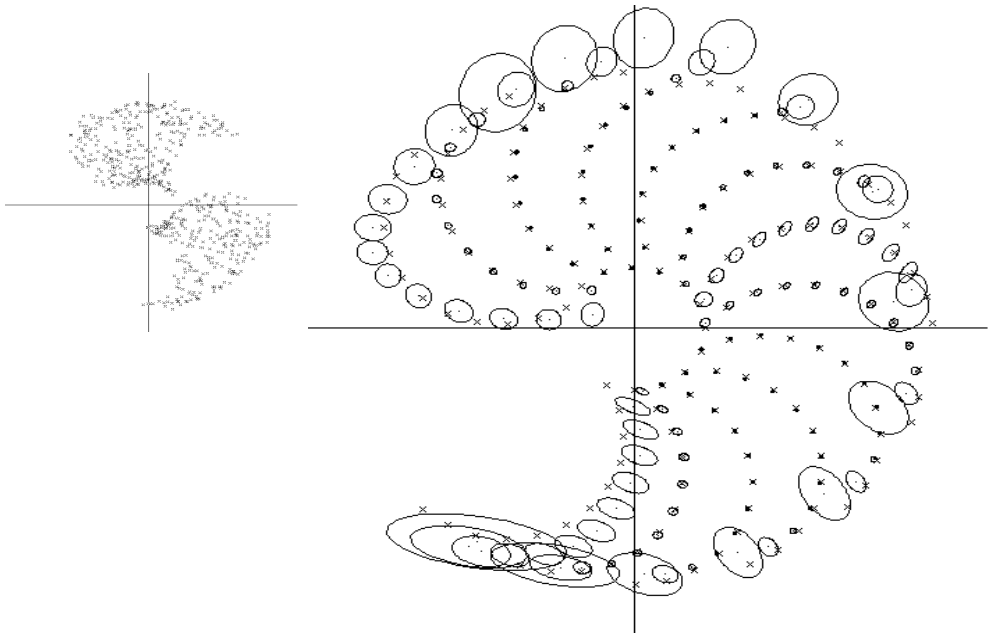

Figure 3.1: **Typical neural network output.** (Inset – training set)
This is the output space (y_a, y_b) of the network. The target outputs are displayed as small x's, and the output of the network with 1σ error bars is shown as a a dot surrounded by an ellipse. The network was trained on samples in two regions in the lower and upper half planes (inset). The outputs illustrated here are for inputs extending a short distance outside the two 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 are not obtained if the diagonal Hessian approximation of [14] is used. The above solution was created by a three layer network with 19 hidden units.

Denker *et. al.* 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 [14, 34]. The Hessian can be evaluated in the same way that backpropagation evaluates ∇E_D (see [8] for a complete algorithm and the appendix of this chapter 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 minimise 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.

3.4 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 (θ_1, θ_2) were made, and gaussian noise of magnitude 0.05 was added to the outputs. The

we use a neural network to perform a mapping, we typically only implement 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, *i.e.* the different minima of M. We would only want the evidence as a function of the number of hidden units 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 α, β 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 [25].

Having adopted this slight shift in objective, it turns out that to set α and β 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 centred 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 neighbourhood 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 α and β:

$$P(S_{\mathbf{w}^*}, \alpha, \beta, \mathcal{H}|D) \propto g \frac{Z_M^*(\mathbf{w}^*, \alpha, \beta)}{Z_W(\alpha) Z_D(\beta)} P(\alpha, \beta|\mathcal{H}) P(\mathcal{H}) \tag{3.14}$$

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 neighbourhood of the minimum at $\mathbf{w}^*$. I will refer to the quantity $g \frac{Z_M^*(\mathbf{w}^*, \alpha, \beta)}{Z_W(\alpha) Z_D(\beta)}$ as the evidence for $\alpha, \beta, S_{\mathbf{w}^*}$. The parameters α and β will be chosen to maximise this evidence. Then the quantity we want to evaluate to compare alternative solutions is the evidence[3] for $S_{\mathbf{w}^*}$,

$$P(D, S_{\mathbf{w}^*}|\mathcal{H}) = \int g \frac{Z_M^*(\mathbf{w}^*, \alpha, \beta)}{Z_W(\alpha) Z_D(\beta)} P(\alpha, \beta|\mathcal{H}) \, d\alpha \, d\beta. \tag{3.15}$$

This thesis uses the gaussian approximation for Z_M^*:

$$Z_M^* \simeq e^{-M(\mathbf{w}^*)} (2\pi)^{k/2} \det{}^{-\frac{1}{2}} \mathbf{A} \tag{3.16}$$

where $\mathbf{A} = \nabla\nabla M$ is the Hessian of M evaluated at $\mathbf{w}^*$. For general α and β this approximation is probably unacceptable; however we only need it to be accurate for the small range of α and β 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 [73]. 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 [40].

[3] 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 $P(D|\mathcal{H})$. The existence of multiple minima in neural network parameter space complicates model comparison. The quantity in (3.15) is not $P(D|S_{\mathbf{w}^*}, \mathcal{H})$ (it includes the prior for $S_{\mathbf{w}^*}|\mathcal{H}$), 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.

Let us use the simple quadratic energy functions defined in equations (3.1,3.2). 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, *i.e.* the number of output units times the number of data pairs, be N, and let the number of free parameters, *i.e.* 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 $\mathbf{w}_{\rm MP}$, and assuming we can locally approximate M as quadratic there, the integral Z_M is approximated by:

$$Z_M \simeq e^{-M(\mathbf{w}_{\rm MP})}(2\pi)^{k/2}\det{}^{-\frac{1}{2}}\mathbf{A} \tag{3.9}$$

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

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

$$\chi^2_W \equiv 2\alpha E_W = \gamma \tag{3.10}$$

$$\chi^2_D \equiv 2\beta E_D = N - \gamma \tag{3.11}$$

where γ is the effective number of parameters determined by the data,

$$\gamma = \sum_{a=1}^{k} \frac{\lambda_a}{\lambda_a + \alpha} \tag{3.12}$$

where λ_a are the eigenvalues of the quadratic form βE_D in the natural basis of E_W.

Comparison of different models

To rank alternative architectures, noise models, and penalty functions E_W in the light of the data, we simply evaluate the evidence for $\mathcal{H} = \{\mathcal{A},\mathcal{N},\mathcal{R}\}$, $P(D\,|\mathcal{H})$, which appeared as the normalising constant in (3.7). Integrating the evidence for (α,β), we have:

$$P(D|\mathcal{H}) = \int P(D|\alpha,\beta,\mathcal{H})P(\alpha,\beta|\mathcal{H})\,d\alpha\,d\beta \tag{3.13}$$

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

3.3 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 non–degenerate 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 (3.9) at each significant minimum and adding up the Z_Ms; 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 α,β for an entire model, and to evaluate the evidence for alternative entire models. This is not quite what we wish to do: when

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.* [36] makes it possible to predict the average generalisation 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.* have demonstrated how to estimate the 'saliency' of a weight, which is the change in M when the weight is deleted [34]. 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 demonstrated how the Hessian of M can be used to assign error bars to the parameters of a network and to its outputs [14]. However, these error bars can only be quantified once β is quantified, and how to do this without prior knowledge or extra data has not been demonstrated. In fact β can be estimated from the training data alone.

3.2 Review of Bayesian regularisation and model comparison

In chapter 2 it was demonstrated how the control parameters α and β are assigned by Bayes, and how alternative interpolation models $\mathcal{H} = \{\mathcal{A}, \mathcal{R}, \mathcal{N}\}$ can be compared. It was noted there that it is not satisfactory to optimise α and β 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. Chapter 2 also reviewed how the Bayesian choice of α and β is neatly expressed in terms of a measure of the number of well–determined parameters in a model, γ. However that chapter assumed that $M(\mathbf{w})$ only has one significant minimum which was well approximated as quadratic. (All the interpolation models discussed in chapter 2 can be interpreted as two–layer networks with a fixed non–linear 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.

Determination of α and β

By Bayes' rule, the posterior probability for these parameters is:

$$P(\alpha, \beta \,|D, \mathcal{H}) = \frac{P(D|\alpha, \beta, \mathcal{H}) P(\alpha, \beta|\mathcal{H})}{P(D|\mathcal{H})} \tag{3.7}$$

Now if we assign a uniform prior to (α, β), the quantity of interest for assigning preferences to (α, β) is the first term on the right hand side, the **evidence** for α, β, which can be written as[2]

$$P(D|\alpha, \beta, \mathcal{H}) = \frac{Z_M(\alpha, \beta)}{Z_W(\alpha) Z_D(\beta)} \tag{3.8}$$

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

[2]The same notation, and the same abuses thereof, will be used as in chapter 2.

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 Bayesian image reconstruction methods [22]. This chapter also adopts a shift in emphasis from Tishby *et. al.*'s paper. Their work concentrated on predicting the average generalisation ability of one network trained on a task drawn from a known prior ensemble of tasks. This is called *forward probability*. In this thesis 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 thesis avoids using the language of statistical physics, partly so as 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:

$$P(\mathbf{t}^m|\mathbf{x}^m,\mathbf{w},\beta,\mathcal{A},\mathcal{N}) = \frac{\exp(-\beta E(\mathbf{t}^m|\mathbf{x}^m,\mathbf{w},\mathcal{A}))}{Z_m(\beta)}, \tag{3.4}$$

  where $Z_m(\beta) = \int d\mathbf{t}\, \exp(-\beta E)$. E is the error for a single datum, and β 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 $\sigma_\nu^2 = 1/\beta$. The symbol $\mathcal{N}$ denotes the implicit noise model.

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

$$P(\mathbf{w}|\alpha,\mathcal{H}) = \frac{\exp(-\alpha E_W(\mathbf{w}|\mathcal{A}))}{Z_W(\alpha)} \tag{3.5}$$

  where $Z_W = \int d^k\mathbf{w}\, \exp(-\alpha E_W)$. Here α is a measure of the characteristic expected connection magnitude. If E_W is quadratic as specified in equation (3.2) then weights are expected to come from a gaussian with zero mean and variance $\sigma_W^2 = 1/\alpha$. Alternative 'regularisers' $\mathcal{R}$ (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:

$$P(\mathbf{w}|D,\alpha,\beta,\mathcal{A},\mathcal{N},\mathcal{R}) = \frac{\exp(-\alpha E_W - \beta E_D)}{Z_M(\alpha,\beta)} \tag{3.6}$$

  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 (3.3).

So under this framework, minimisation of $M = \alpha E_W + \beta E_D$ is identical to finding the (locally) most probable parameters $\mathbf{w}_{\rm MP}$; minimisation of E_D alone is identical to finding the maximum likelihood parameters $\mathbf{w}_{\rm ML}$. Thus an interpretation has been given to backpropagation's energy functions E_D and E_W, and to the parameters α and β. It should be emphasised that 'the probability of the connections $\mathbf{w}$' is a measure of *plausibility* that

optimise the parameters $\mathbf{w}$ of the network, and a test set, which is used to optimise control parameters such as α and the architecture $\mathcal{A}$. However the utility of these techniques in determining values for the parameters α and β 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 like α and β, it is out of the question to optimise such parameters by repeating the learning with all possible values of these parameters and using a test set. Such parameters must be optimised on line.

It is therefore interesting to study objective criteria for setting free parameters and comparing alternative solutions which 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 doesn't 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 optimising the parameters $\mathbf{w}$ *and* in the process of optimising control parameters like α and $\mathcal{A}$.

This chapter 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 over–complex network which generalises poorly. The use of weight decay does not fully alleviate this problem; networks with too many hidden units still generalise worse, even if weight decay is used (see section 3.4).

2. **Objective criteria for setting the decay rate α.** As in the choice of $\mathcal{A}$ above, there is an 'Occam's razor' problem: a small value of α in equation (3.3) 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 α only on E_D or M. The Bayesian solution presented here can be implemented on–line, *i.e.* it is not necessary to do multiple learning runs with different values of α in order to find the best.

3. **Objective choice of regularising 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.

The probability connection

Tishby *et. al.* [71] introduced a probabilistic view of learning which is an important step towards 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

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 $\mathbf{y}$.[1] 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

$$E_D(D\,|\mathbf{w},\mathcal{A}) = \sum_m \frac{1}{2}\Big(\mathbf{y}(\mathbf{x}^m;\mathbf{w},\mathcal{A}) - \mathbf{t}^m\Big)^2 \tag{3.1}$$

The task of 'learning' is to find a set of connections $\mathbf{w}$ which gives a mapping which fits the training set well, *i.e.* has small error E_D; it is also hoped that the learned connections will 'generalise' 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 optimisation techniques may also be used, such as conjugate gradients or variable metric methods. This chapter will not discuss computational modifications concerned only with speeding the optimisation. It will address however those modifications to the plain backprop algorithm which implicitly or explicitly modify the objective function, with decay terms or regularisers.

It is moderately common for extra regularising terms $E_W(\mathbf{w})$ to be added to E_D; for example terms which penalise large weights may be introduced, in the hope of achieving a smoother or simpler mapping [27, 32, 47, 57, 77]. Some of the 'hints' in [2] also fall into the category of additive weight–dependent energies. A sample weight energy term is:

$$E_W(\mathbf{w}|\mathcal{A}) = \sum_i \frac{1}{2} w_i^2 \tag{3.2}$$

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 (3.2). Gradient–based optimisation is then used to minimise the combined function:

$$M = \alpha E_W(\mathbf{w}|\mathcal{A}) + \beta E_D(D\,|\mathbf{w},\mathcal{A}) \tag{3.3}$$

where α and β are 'black box' parameters.

The constant α should not be confused with the 'momentum' parameter sometimes introduced into backprop; in the present context α is a decay rate or regularising constant. Also note that α 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.

What is lacking

The above procedures include a host of free parameters such as the choice of neural network architecture, and of the regularising constant α. There are not yet established ways of objectively setting these parameters, though there are many rules of thumb (see [32, 77] 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 which is used to

[1] The framework developed in this chapter 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, $\partial\mathbf{y}(\mathbf{x};\mathbf{w},\mathcal{A})/\partial\mathbf{w}$.

Chapter 3

A Practical Bayesian Framework for Backprop Networks

Abstract

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 regularisers (for penalising 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; (6) objective comparisons with alternative learning and interpolation models such as splines and radial basis functions. The Bayesian 'evidence' automatically embodies 'Occam's razor,' penalising over–flexible and over–complex models. The Bayesian approach helps detect poor underlying assumptions in learning models. For learning models well matched to a problem, a good correlation between generalisation ability and the Bayesian evidence is obtained.

3.1 The gaps in backprop

There are many knobs on the black box of 'backprop' (learning by back–propagation of errors [56]). Generally these knobs are set by rules of thumb, trial and error, and the use of reserved test data to assess generalisation ability (or more sophisticated cross–validation). The knobs fall into two classes: (1) parameters which change the effective learning model, for example, number of hidden units, and weight decay terms; and (2) parameters concerned with function optimisation technique, for example, 'momentum' terms. This chapter 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 which makes full use of all the available data. Bayesian techniques will be described which 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

[0]Chapter 3 of PhD thesis 'Bayesian Methods for Adaptive Models' by David MacKay, California Institute of Technology, submitted December 10 1991.