UNIVERSITY OF CAMBRIDGE
PROGRAMME FOR INDUSTRY

Neural Networks Summer School

# Bayesian Methods for Neural Networks: Theory and Applications

**David J.C. MacKay**
**Cavendish Laboratory,**
**Cambridge, CB3 0HE.**
`mackay@mrao.cam.ac.uk`

Monday 24 July 14.45–15.30 and 16.45–17.30;
Tuesday 25 July 14.00–14.45

Neural networks are parameterized non-linear models used for empirical regression and classification modelling. Their flexibility makes them able to discover more general relationships in data than traditional statistical models.

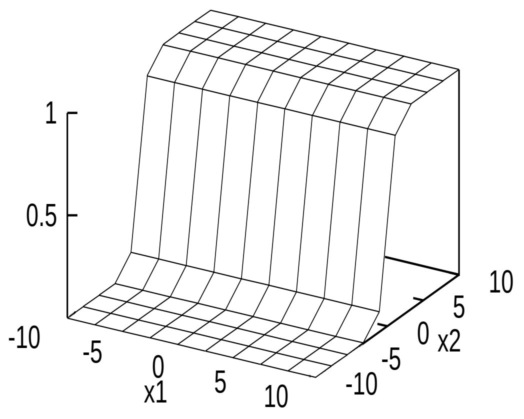
Bayesian probability theory provides a unifying framework for data modeling which offers several benefits. First, the **overfitting problem** can be solved by using Bayesian methods to control model complexity. Bayesian model comparison can be used for example to optimize weight decay rates, and to infer automatically which are the relevant input variables for a problem. Second, probabilistic modelling handles uncertainty in a natural manner. There is a unique prescription, **marginalization**, for incorporating uncertainty about parameters into predictions; this procedure yields better predictions. Third, we can define more sophisticated probabilistic models which are able to extract more information from data.

I will discuss the practical application of these methods to a problem involving the prediction of a series of building energy loads from a series of environmental variables.

I will also discuss the application of Bayesian methods to neural network classifiers.

Finally I will describe the implementation of a probabilistic regression model in BUGS. BUGS is a program that carries out Bayesian inference on statistical problems using a simulation technique known as Gibbs sampling.

## Contents



$\mathbf{w} = (0, 2)$

Figure 1: **Output of simple neural network as a function of its input**

## 1 Basic neural network concepts

A neural network implements a function $y(\mathbf{x}; \mathbf{w})$; the 'output' of the network, $y$, is a non-linear function of the 'input' $\mathbf{x}$; this function is parameterized by 'weights' $\mathbf{w}$.

A very simple neural network produces an output between 0 and 1 as the following function of $\mathbf{x}$:

$$y(\mathbf{x}; \mathbf{w}, \mathcal{H}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}} \tag{1}$$

This form of function is known to statisticians as a linear logistic, and in neural networks as a single neuron.

For convenience let us study the case where the input vector $\mathbf{x}$ and the parameter vector $\mathbf{w}$ are both two-dimensional: $\mathbf{x} = (x_1, x_2)$, $\mathbf{w} = (w_1, w_2)$. Then we can spell out the function $f$ thus:

$$y(\mathbf{x}; \mathbf{w}, \mathcal{H}) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}} \tag{2}$$

Figure 1 shows the output of the neuron as a function of the input vector, for $\mathbf{w} = (0, 2)$. The two horizontal axes of this figure are the inputs $x_1$ and $x_2$, with the output $y$ on the vertical axis. Notice that on any line perpendicular to $\mathbf{w}$, the output is constant; and along a line in the direction of $\mathbf{w}$, the output is a 'sigmoid' function.

We now introduce an important concept, namely the idea of **weight space**, that is, the parameter space of the network. In this case, there are two parameters $w_1$ and $w_2$, so the weight space is two dimensional. This weight space is shown in figure 2. For a selection of different values of the parameter vector $\mathbf{w}$, smaller inset figures show the function of $\mathbf{x}$ performed by the network when $\mathbf{w}$ is set to those given values. Each of these smaller figures is equivalent to figure 1. Thus each *point* in $\mathbf{w}$ space corresponds to a *function* of $\mathbf{x}$. Notice that the gain of the sigmoid function (the gradient of the ramp) increases as the magnitude of $\mathbf{w}$ increases.

Now, the central idea of the neural network method is this. Given **examples** of a relationship between an input vector $\mathbf{x}$, and a target $t$, we hope to make the neural network 'learn' a model of the relationship between $\mathbf{x}$ and $t$. A successfully trained network will, for any given $\mathbf{x}$, give an output $y$ that is close (in some sense) to the target value $t$. '**Training**' the network involves

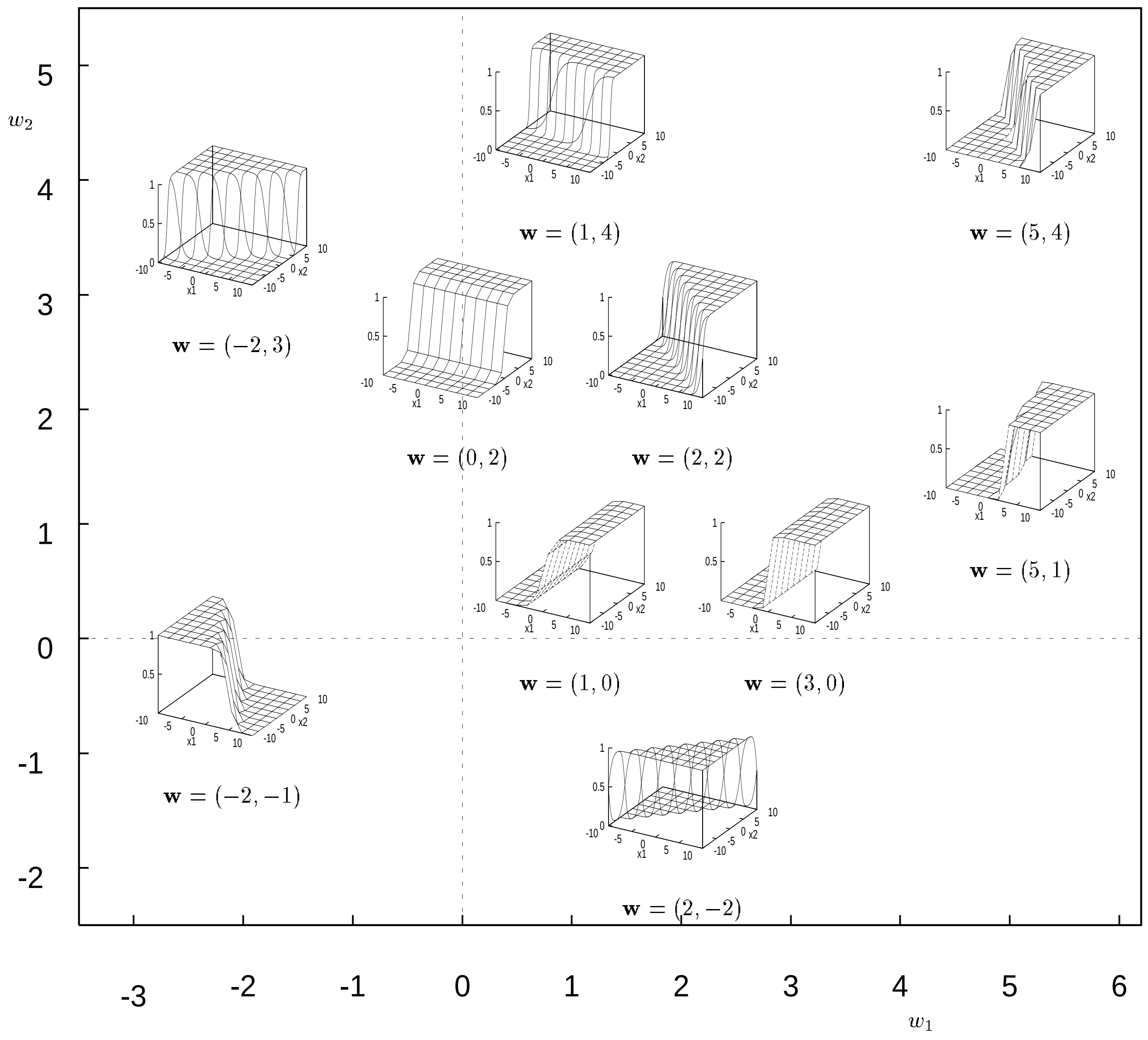


Figure 2: **Weight space.**

searching around in the weight space of the network for a value of $\mathbf{w}$ that produces a function that fits the provided training data well.

Typically an 'objective function' or 'error function' is defined as a function of $\mathbf{w}$ to measure how well the network, with its weights set to $\mathbf{w}$, solves the task. The objective function is a sum of terms, one for each input/target pair $\{\mathbf{x}, t\}$, measuring how close the output $y(\mathbf{x}; \mathbf{w})$ is to the target $t$. The training process is an exercise in *function minimization* — *i.e.*, adjusting $\mathbf{w}$ in such a way as to find a $\mathbf{w}$ that minimizes the objective function. Many function minimization algorithms make use not only of the objective function, but also its *gradient* with respect to the parameters $\mathbf{w}$. For neural networks the **backpropagation** algorithm (Rumelhart, Hinton and Williams 1986) efficiently evaluates the gradient of the output $y$ with respect to the parameters $\mathbf{w}$, and thence the gradient of the objective function with respect to $\mathbf{w}$.

### Probability theory as a framework for neural network learning

To make this discussion more concrete, let us assume we are studying a binary classification problem, and that we interpret the output $y(\mathbf{x}; \mathbf{w}, \mathcal{H})$ of the simple neuron above as defining (when its parameters $\mathbf{w}$ are specified) the probability that an input $\mathbf{x}$ belongs to class $t = 1$, rather than the alternative $t = 0$. Thus $y(\mathbf{x}; \mathbf{w}, \mathcal{H}) \equiv P(t = 1|\mathbf{x}, \mathbf{w}, \mathcal{H})$. Then each value of $\mathbf{w}$ in the picture of weight space defines a different sub–hypothesis about the probability of class 1 relative to class 0 as a function of $\mathbf{x}$.

Now imagine that we receive some data as shown in the left column of figure 3. Each data point consists of a two dimensional input vector $\mathbf{x}$ and a $t$ value indicated by $\times$ $(t = 1)$ or $\square$ $(t = 0)$.

In the traditional view of learning, a single parameter vector $\mathbf{w}$ evolves under the learning rule from an initial starting point $\mathbf{w}^0$ to a final optimum $\mathbf{w}^*$, in such a way as to minimize an objective function equal to minus the log likelihood plus a *regularizer* such as $\alpha \sum_i w_i^2/2$. The *log likelihood* measures how well parameters $\mathbf{w}$ predict the observed data; it is the log of the probability assigned to the observed $t$ values by the model with parameters set to $\mathbf{w}$, and it is shown as a function of $\mathbf{w}$ in the second column. It is a product of functions of the form (1).

The product of traditional learning is a point in $\mathbf{w}$–space, the estimator $\mathbf{w}^*$, which will be close to the maximum of the likelihood. In contrast, in the Bayesian view, the product of learning is an *ensemble* of plausible parameter values (bottom right of figure 3). We do not choose one particular sub-hypothesis $\mathbf{w}$; rather we evaluate their posterior probabilities, which by Bayes' theorem are:

$$P(\mathbf{w}|\{t\}, \{\mathbf{x}\}, \mathcal{H}) = \frac{P(\{t\}|\mathbf{w}, \{\mathbf{x}\}, \mathcal{H}) P(\mathbf{w}|\mathcal{H})}{P(\{t\}|\{\mathbf{x}\}, \mathcal{H})} \tag{3}$$

The posterior distribution is thus obtained by multiplying the likelihood by a prior distribution over $\mathbf{w}$ space (shown as a broad Gaussian at upper right of figure 3). The posterior ensemble (within a multiplicative constant) is shown in the third column of figure 3, and as a contour plot in the fourth column. As the amount of data increases (from top to bottom), the posterior ensemble becomes increasingly concentrated around the most probable value $\mathbf{w}^*$. This illustrates the Bayesian interpretation and generalization of traditional neural network learning.

The Bayesian framework for neural networks has numerous benefits that are described in this document. Before these are discussed however, perhaps we should have a tutorial on Bayesian probability theory and its application to model comparison problems.

## 2 Probability theory and Occam's razor

Bayesian probability theory provides a unifying framework for data modelling. A Bayesian data-modeller's aim is to develop probabilistic models that are well-matched to the data, and make optimal predictions using those models. The Bayesian framework has several advantages.

Probability theory forces us to make explicit all our modelling assumptions. Bayesian methods are mechanistic: once a model space has been defined, then, whatever question we wish to

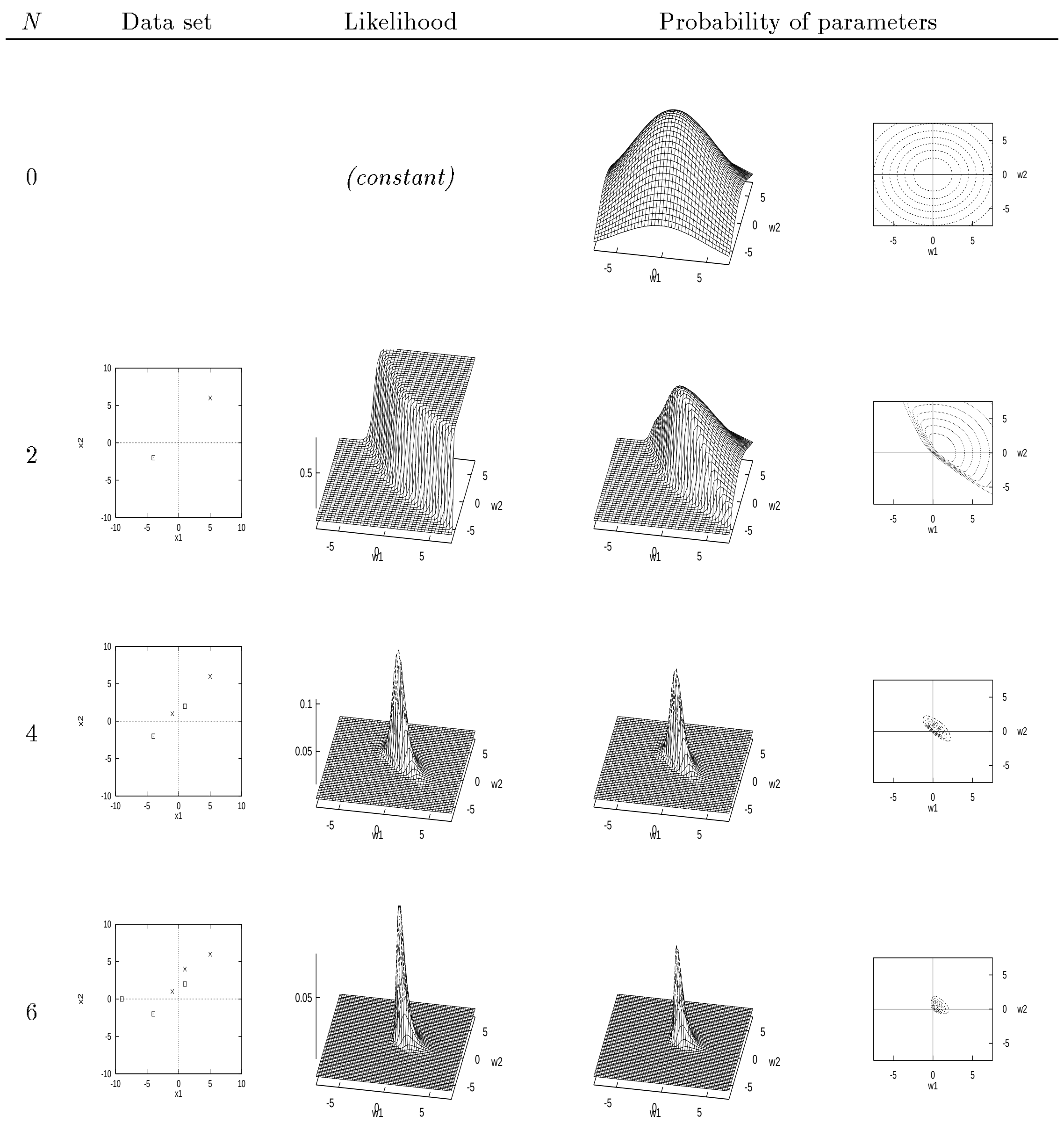


Figure 3: **Evolution of the probability distribution over parameters as data arrives.**

pose, the rules of probability theory give a unique answer which consistently takes into account all the given information. This is in contrast to non-Bayesian statistics, in which one must invent 'estimators' of quantities of interest and then choose between those estimators using some criterion measuring their sampling properties; there is no clear principle for deciding which criterion to use to measure the performance of an estimator; nor, for most criteria, is there any systematic procedure for the construction of optimal estimators.

Bayesian inference satisfies the likelihood principle (Berger 1985): our inferences depend only on the probabilities assigned to the data that were received, not on properties of other data sets which might have occurred but did not.

Probabilistic modelling handles uncertainty in a natural manner. There is a unique prescription (marginalization) for incorporating uncertainty about parameters into our predictions of other variables.

Finally, Bayesian model comparison embodies **Occam's razor**, the principle that states a preference for simple models. This point will be expanded on in a moment.

The preceding advantages of Bayesian modelling do not make all our troubles go away. The Bayesian is left with the twin tasks of defining an appropriate model space for the data, and implementing the rules of inference numerically.

## Occam's Razor

Occam's razor is the principle that states a preference for simple theories. If several explanations are compatible with a set of observations, Occam's razor advises us to buy the least complex explanation. This principle is often advocated for one of two reasons: the first is aesthetic ("A theory with mathematical beauty is more likely to be correct than an ugly one that fits some experimental data" (Paul Dirac)); the second reason is the empirical success of Occam's razor. Here I will discuss a different justification for Occam's razor, namely:

Coherent inference embodies Occam's razor automatically and quantitatively.

All coherent beliefs and predictions can be mapped onto probabilities. I will use the following notation for *conditional* probabilities: $P(A|B,\mathcal{H})$ is pronounced 'the probability of $A$, given $B$ and $\mathcal{H}$'. The statements $B$ and $\mathcal{H}$ list the conditional assumptions on which this measure of plausibility is based. For example if $A$ is "it will rain today", and $B$ is "the barometer is rising", then the quantity $P(A|B,\mathcal{H})$ is a number between 0 and 1 which expresses how probable I would think 'rain today' is, given that the barometer is rising, and given the overall assumptions $\mathcal{H}$ which define my model of the weather. This conditional probability is related to the *joint* probability of $A$ *and* $B$ by $P(A|B,\mathcal{H}) = P(A,B|\mathcal{H})/P(B|\mathcal{H})$. Note that the conditioning notation does not imply causation. $P(A|B)$ does not mean 'the probability that $A$ is caused by $B$'. Rather, it measures the plausibility of proposition $A$, assuming that the information in proposition $B$ is true. The *marginal* probability distribution of $A$ is related to the joint and conditional distributions thus: $P(A|\mathcal{H}) = \sum_B P(A,B|\mathcal{H}) = \sum_B P(A|B,\mathcal{H})P(B)$. With apologies to pure mathematicians, I shall use the same notation for probabilities of discrete variables and for probability densities over real variables.

Having enumerated a complete list of these conditional degrees of belief, we can then use the rules of probability to evaluate how our beliefs and predictions should change when we gain new information, *i.e.*, as we change the conditioning statements to the right of our "|" symbol. For example, the probability $P(B|A,\mathcal{H})$ measures how plausible it is that the barometer is rising, given that today is a rainy day; this probability can be obtained by Bayes' theorem,

$$P(A|B,\mathcal{H}) = P(B|A,\mathcal{H})P(A|\mathcal{H})/P(B|\mathcal{H}).$$

Here, my overall model of the weather, $\mathcal{H}$, is a conditioning statement on the right hand side of all the probabilities. In this sense, Bayesian inferences are 'subjective', in that it is not possible to reason about models for data without making assumptions. At the same time, Bayesian inferences are objective, in that anyone who shares the same assumptions $\mathcal{H}$ as me will draw

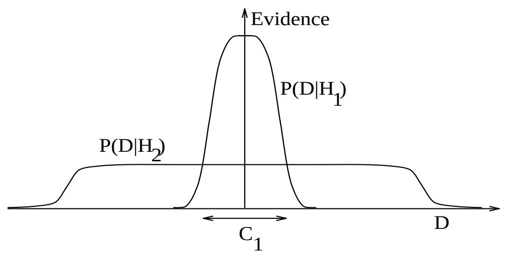


Figure 4: **Why Bayesian inference embodies Occam's razor**
This figure gives the basic intuition for why complex models are penalized. The horizontal axis represents the space of possible data sets $D$. Bayes' rule rewards models in proportion to how much they *predicted* the data that occurred. These predictions are quantified by a normalized probability distribution on $D$. In these notes, this probability of the data given model $\mathcal{H}_i$, $P(D|\mathcal{H}_i)$, is called the evidence for $\mathcal{H}_i$.
A simple model $\mathcal{H}_1$ makes only a limited range of predictions, shown by $P(D|\mathcal{H}_1)$; a more powerful model $\mathcal{H}_2$, that has, for example, more free parameters than $\mathcal{H}_1$, is able to predict a greater variety of data sets. This means, however, that $\mathcal{H}_2$ does not predict the data sets in region $\mathcal{C}_1$ as strongly as $\mathcal{H}_1$. Suppose that equal prior probabilities have been assigned to the two models. Then, if the data set falls in region $\mathcal{C}_1$, the *less powerful* model $\mathcal{H}_1$ will be the *more probable* model.

identical inferences; there is only one answer to a well-posed problem. Bayesian methods thus force us to make all tacit assumptions explicit, and then provide rules for reasoning consistently given those assumptions.

We evaluate the plausibility of two alternative theories $\mathcal{H}_1$ and $\mathcal{H}_2$ in the light of data $D$ as follows: using Bayes' rule, we relate the plausibility of model $\mathcal{H}_1$ given the data, $P(\mathcal{H}_1|D)$, to the predictions made by the model about the data, $P(D|\mathcal{H}_1)$, and the prior plausibility of $\mathcal{H}_1$, $P(\mathcal{H}_1)$. This gives the following probability ratio between theory $\mathcal{H}_1$ and theory $\mathcal{H}_2$:

$$\frac{P(\mathcal{H}_1|D)}{P(\mathcal{H}_2|D)} = \frac{P(\mathcal{H}_1)}{P(\mathcal{H}_2)} \frac{P(D|\mathcal{H}_1)}{P(D|\mathcal{H}_2)}. \tag{4}$$

The first ratio $(P(\mathcal{H}_1)/P(\mathcal{H}_2))$ on the right hand side measures how much our initial beliefs favoured $\mathcal{H}_1$ over $\mathcal{H}_2$. The second ratio expresses how well the observed data were predicted by $\mathcal{H}_1$, compared to $\mathcal{H}_2$.

How does this relate to Occam's razor, when $\mathcal{H}_1$ is a simpler model than $\mathcal{H}_2$? The first ratio $(P(\mathcal{H}_1)/P(\mathcal{H}_2))$ gives us the opportunity, if we wish, to insert a prior bias in favour of $\mathcal{H}_1$ on aesthetic grounds, or on the basis of experience. This would correspond to the motivations for Occam's razor discussed in the first paragraph. But this is not necessary: the second ratio, the data-dependent factor, embodies Occam's razor *automatically.* Simple models tend to make precise predictions. Complex models, by their nature, are capable of making a greater variety of predictions (figure 4). So if $\mathcal{H}_2$ is a more complex model, it must spread its predictive probability $P(D|\mathcal{H}_2)$ more thinly over the data space than $\mathcal{H}_1$. Thus, in the case where the data are compatible with both theories, the simpler $\mathcal{H}_1$ will turn out more probable than $\mathcal{H}_2$, without our having to express any subjective dislike for complex models. Our subjective prior just needs to assign equal prior probabilities to the possibilities of simplicity and complexity. Probability theory then allows the observed data to express their opinion.

Let us turn to a simple example. Here is a sequence of numbers:

$$-1,\ 3,\ 7,\ 11$$

The task is to predict what the next two numbers are likely to be, and infer what the underlying process probably was, that gave rise to this sequence. A popular answer to this question is the prediction "15, 19", with the explanation "add 4 to the previous number".

What about the alternative answer "$-19.9, 1043.8$" with the underlying rule being: "get the next number from the previous number, $x$, by evaluating $-x^3/11 + 9/11x^2 + 23/11$"? I assume that this prediction seems rather less plausible. But the second rule fits the data (-1, 3, 7, 11) just as well as the rule "add 4". So why should we find it less plausible? Let us give labels to the two general theories:

$\mathcal{H}_a$– the sequence is an **arithmetic** progression, 'add $n$', where $n$ is an integer.

$\mathcal{H}_c$– the sequence is generated by a **cubic** function of the form $x \to cx^3 + dx^2 + e$, where $c$, $d$ and $e$ are fractions.

One reason for finding the second explanation, $\mathcal{H}_c$, less plausible, might be that arithmetic progressions are more frequently encountered than cubic functions. This would put a bias in the prior probability ratio $P(\mathcal{H}_a)/P(\mathcal{H}_c)$ in equation (4). But let us give the two theories equal prior probabilities, and concentrate on what the data have to say. How well did each theory predict the data?

To obtain $P(D|\mathcal{H}_a)$ we must specify the probability distribution that each model assigns to its parameters. First, $\mathcal{H}_a$ depends on the added integer $n$, and the first number in the sequence. Let us say that these numbers could each have been anywhere between -50 and 50. Then since only the pair of values $\{n=4, \text{first number}=-1\}$ give rise to the observed data $D$ = (-1, 3, 7, 11), the probability of the data, given $\mathcal{H}_a$, is:

$$P(D|\mathcal{H}_a) = \frac{1}{101}\frac{1}{101} = 0.00010$$

To evaluate $P(D|\mathcal{H}_c)$, we must similarly say what values the fractions $c, d$ and $e$ might take on. [I choose to represent these numbers as fractions rather than real numbers because if we used real numbers, the model would assign, relative to $\mathcal{H}_a$, an infinitessimal probability to $D$. Real parameters are the norm however, and are assumed in the rest of these notes.] A reasonable prior might state that for each fraction the numerator could be any number between -50 and 50, and the denominator is any number between 1 and 50. As for the initial value in the sequence, let us leave its probability distribution the same as in $\mathcal{H}_a$. There are four ways of expressing the fraction $c = -1/11 = -2/22 = -3/33 = -4/44$ under this prior, and similarly there are four and two possible solutions for $d$ and $e$, respectively. So the probability of the observed data, given $\mathcal{H}_c$, is found to be:

$$\begin{aligned} P(D|\mathcal{H}_c) &= \left(\frac{1}{101}\right)\left(\frac{4}{101}\frac{1}{50}\right)\left(\frac{4}{101}\frac{1}{50}\right)\left(\frac{2}{101}\frac{1}{50}\right) \\ &= 0.0000000000025 = 2.5 \times 10^{-12} \end{aligned}$$

Thus comparing $P(D|\mathcal{H}_c)$ with $P(D|\mathcal{H}_a) = 0.00010$, even if our prior probabilities for $\mathcal{H}_a$ and $\mathcal{H}_c$ are equal, the odds, $P(D|\mathcal{H}_a) : P(D|\mathcal{H}_c)$, in favour of $\mathcal{H}_a$ over $\mathcal{H}_c$, given the sequence $D$ = (-1, 3, 7, 11), are about forty million to one.

This answer depends on several subjective assumptions; in particular, the probability assigned to the free parameters $n$, $c$, $d$, $e$ of the theories. Bayesians make no apologies for this: there is no such thing as inference or prediction without assumptions. However, the quantitative details of the prior probabilities have no effect on the qualitative Occam's razor effect; the complex theory $\mathcal{H}_c$ always suffers an 'Occam factor' because it has more parameters, and so can predict a greater variety of data sets (figure 4). This was only a small example, and there were only four data points; as we move to larger and more sophisticated problems the magnitude of the Occam factors typically increases, and the degree to which our inferences are influenced by the quantitative details of our subjective assumptions becomes smaller.

## Bayesian methods and data analysis

Let us now relate the discussion above to real problems in data analysis.

There are countless problems in science, statistics and technology which require that, given a limited data set, preferences be assigned to alternative models of differing complexities. For example, two alternative hypotheses accounting for planetary motion are Mr. Inquisition's geocentric model based on 'epicycles', and Mr. Copernicus's simpler model of the solar system. The epicyclic model fits data on planetary motion at least as well as the Copernican model, but does so using more parameters. Coincidentally for Mr. Inquisition, two of the extra epicyclic parameters for every planet are found to be identical to the period and radius of the sun's 'cycle around the earth'. Intuitively we find Mr. Copernicus's theory more probable. I will now explain in more detail how Mr. Inquisition's excess parameters are penalized automatically under probability theory.

## The mechanism of the Bayesian Occam's razor: the evidence and the Occam factor

Two levels of **inference** can often be distinguished in the task of data modelling. At the first level of inference, we assume that a particular model is true, and we fit that model to the data. Typically a model includes some free parameters; fitting the model to the data involves inferring what values those parameters should probably take, given the data. The results of this inference are often summarized by the most probable parameter values, and error bars on those parameters. This analysis is repeated for each model. The second level of inference is the task of model comparison. Here we wish to compare the models in the light of the data, and assign some sort of preference or ranking to the alternatives.[1]

Bayesian methods are able consistently and quantitatively to solve both the inference tasks. There is a popular myth that states that Bayesian methods only differ from orthodox (also known as 'frequentist' or 'sampling theoretical') statistical methods by the inclusion of subjective priors which are arbitrary and difficult to assign, and usually don't make much difference to the conclusions. It is true that, at the first level of inference, a Bayesian's results will often differ little from the outcome of an orthodox attack. What is not widely appreciated is how Bayes performs the second level of inference; this section will therefore focus on Bayesian model comparison. This emphasis should not be misconstrued as implying a belief that one ought to use the Bayesian rankings to 'choose' a single best model. What we do with the Bayesian posterior probabilities is another issue. If we wish to make predictions, for example, then we should integrate over the alternative models, weighted by their posterior probabilities (section 7).

Model comparison is a difficult task because it is not possible simply to choose the model that fits the data best: more complex models can always fit the data better, so the maximum likelihood model choice would lead us inevitably to implausible, over-parameterized models which generalize poorly. Occam's razor is needed.

Let us write down Bayes' rule for the two levels of inference described above, so as to see explicitly how Bayesian model comparison works. Each model $\mathcal{H}_i$ is assumed to have a vector of parameters $\mathbf{w}$. A model is defined by a collection of probability distributions: a 'prior' distribution $P(\mathbf{w}|\mathcal{H}_i)$ which states what values the model's parameters might plausibly take; and a set of probability distributions, one for each value of $\mathbf{w}$, defining the predictions $P(D|\mathbf{w},\mathcal{H}_i)$ that the model makes about the data $D$.

1. **Model fitting.** At the first level of inference, we assume that one model, the $i$th, say, is true, and we infer what the model's parameters $\mathbf{w}$ might be, given the data $D$. Using Bayes' rule, the **posterior probability** of the parameters $\mathbf{w}$ is:

$$P(\mathbf{w}|D,\mathcal{H}_i) = \frac{P(D|\mathbf{w},\mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)}{P(D|\mathcal{H}_i)}, \tag{5}$$

that is,

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}.$$

[1]Note that both levels of *inference* are distinct from *decision theory*. The goal of inference is, given a defined hypothesis space and a particular data set, to assign probabilities to hypotheses. Decision theory typically chooses between alternative *actions* on the basis of these probabilities so as to minimize the expectation of a 'loss function'. These notes concern inference alone and no loss functions are involved.

The normalizing constant $P(D|\mathcal{H}_i)$ is commonly ignored since it is irrelevant to the first level of inference, *i.e.*, the choice of $\mathbf{w}$; but it becomes important in the second level of inference, and we name it the **evidence** for $\mathcal{H}_i$. It is common practice to use gradient-based methods to find the maximum of the posterior, which defines the most probable value for the parameters, $\mathbf{w}_{\text{MP}}$; it is then usual to summarize the posterior distribution by the value of $\mathbf{w}_{\text{MP}}$, and error bars on these best fit parameters. The error bars are obtained from the curvature of the posterior; evaluating the Hessian at $\mathbf{w}_{\text{MP}}$, $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, \mathcal{H}_i)$, and Taylor-expanding the log posterior with $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_{\text{MP}}$:

$$P(\mathbf{w}|D,\mathcal{H}_i) \simeq P(\mathbf{w}_{\text{MP}}|D,\mathcal{H}_i) \exp\left(-\tfrac{1}{2}\Delta\mathbf{w}^{\text{T}}\mathbf{A}\Delta\mathbf{w}\right), \tag{6}$$

we see that the posterior can be locally approximated as a Gaussian with covariance matrix $\mathbf{A}^{-1}$. This matrix defines correlated error bars on the parameters $\mathbf{w}$. Whether this approximation is good or not will depend on the problem we are solving. The maximum and mean of the posterior distribution have no fundamental status in Bayesian inference — they can both be arbitrarily changed by non-linear reparameterizations. Maximization of a posterior probability is only useful if an approximation like equation (6) gives a good summary of the distribution.

2. **Model comparison.** At the second level of inference, we wish to infer which model is most plausible given the data. The posterior probability of each model is:

$$P(\mathcal{H}_i|D) \propto P(D|\mathcal{H}_i)P(\mathcal{H}_i). \tag{7}$$

Notice that the data-dependent term $P(D|\mathcal{H}_i)$ is the evidence for $\mathcal{H}_i$, which appeared as the normalizing constant in (5). The second term, $P(\mathcal{H}_i)$, is the subjective prior over our hypothesis space which expresses how plausible we thought the alternative models were before the data arrived. Assuming that we choose to assign equal priors $P(\mathcal{H}_i)$ to the alternative models, **models $\mathcal{H}_i$ are ranked by evaluating the evidence.** Equation (7) has not been normalized because in the data modelling process we may develop new models after the data have arrived, when an inadequacy of the first models is detected, for example. Inference is open ended: we continually seek more probable models to account for the data we gather.

To reiterate the key concept: to assign a preference to alternative models $\mathcal{H}_i$, a Bayesian evaluates the evidence $P(D|\mathcal{H}_i)$. This concept is very general: the evidence can be evaluated for parametric and 'non-parametric' models alike; whatever our data modelling task, a regression problem, a classification problem, or a density estimation problem, the Bayesian evidence is a transportable quantity for comparing alternative models. In all these cases the evidence naturally embodies Occam's razor.

## Evaluating the evidence

Let us now study the evidence more closely to gain insight into how the Bayesian Occam's razor works. The evidence is the normalizing constant for equation (5):

$$P(D\,|\mathcal{H}_i) = \int P(D|\mathbf{w},\mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)\,d\mathbf{w}. \tag{8}$$

For many problems, including interpolation, it is common for the posterior $P(\mathbf{w}|D,\mathcal{H}_i) \propto P(D|\mathbf{w},\mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)$ to have a strong peak at the most probable parameters $\mathbf{w}_{\text{MP}}$ (figure 5). Then, taking for simplicity the one-dimensional case, the evidence can be approximated by the height of the peak of the integrand $P(D|\mathbf{w},\mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)$ times its width, $\sigma_{w|D}$:

$$\begin{aligned} P(D\,|\mathcal{H}_i) &\simeq \underbrace{P(D\,|\mathbf{w}_{\text{MP}},\mathcal{H}_i)} \times \underbrace{P(\mathbf{w}_{\text{MP}}|\mathcal{H}_i)\,\sigma_{w|D}}. \\ \text{Evidence} &\simeq \text{Best fit likelihood} \times \text{Occam factor} \end{aligned} \tag{9}$$

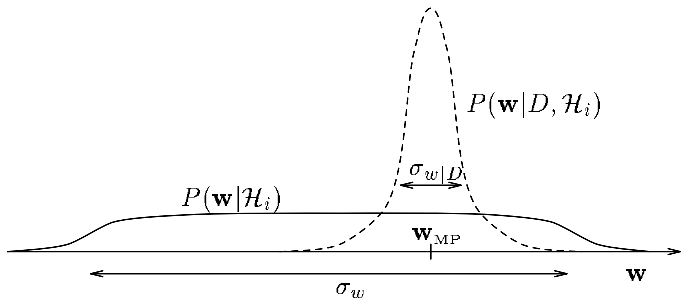

$P(\mathbf{w}|D,\mathcal{H}_i)$
$\sigma_{w|D}$
$P(\mathbf{w}|\mathcal{H}_i)$
$\mathbf{w}_{\rm MP}$
$\mathbf{w}$
$\sigma_w$

Figure 5: **The Occam factor**
This figure shows the quantities that determine the Occam factor for a hypothesis $\mathcal{H}_i$ having a single parameter $\mathbf{w}$. The prior distribution (solid line) for the parameter has width $\sigma_w$. The posterior distribution (dashed line) has a single peak at $\mathbf{w}_{\rm MP}$ with characteristic width $\sigma_{w|D}$. The Occam factor is $(\sigma_{w|D}/\sigma_w)$.

Thus the evidence is found by taking the best fit likelihood that the model can achieve and multiplying it by an 'Occam factor' (Gull 1988), which is a term with magnitude less than one that penalizes $\mathcal{H}_i$ for having the parameter $\mathbf{w}$.

## Interpretation of the Occam factor

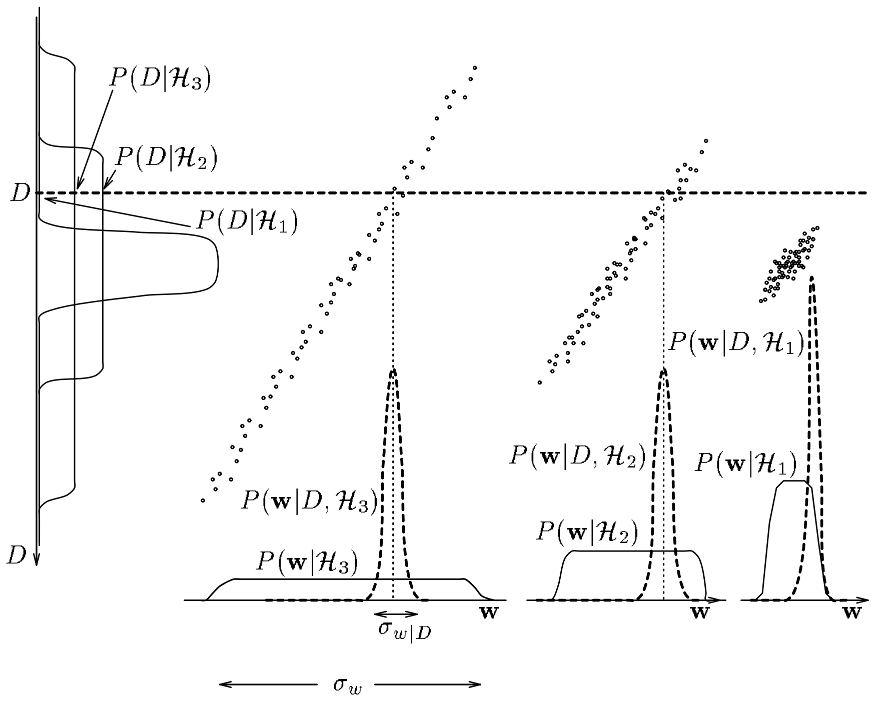
The quantity $\sigma_{w|D}$ is the posterior uncertainty in $\mathbf{w}$. Suppose for simplicity that the prior $P(\mathbf{w}|\mathcal{H}_i)$ is uniform on some large interval $\sigma_w$, representing the range of values of $\mathbf{w}$ that $\mathcal{H}_i$ thought possible before the data arrived (figure 5). Then $P(\mathbf{w}_{\rm MP}|\mathcal{H}_i) = \frac{1}{\sigma_w}$, and

$$\text{Occam factor} = \frac{\sigma_{w|D}}{\sigma_w}, \tag{10}$$

*i.e.*, **the Occam factor is equal to the ratio of the posterior accessible volume of $\mathcal{H}_i$'s parameter space to the prior accessible volume,** or the factor by which $\mathcal{H}_i$'s hypothesis space collapses when the data arrive (Gull 1988, Jeffreys 1939). The model $\mathcal{H}_i$ can be viewed as consisting of a certain number of exclusive submodels, of which only one survives when the data arrive. The Occam factor is the inverse of that number. The logarithm of the Occam factor is a measure of the amount of information we gain about the model when the data arrive.

A complex model having many parameters, each of which is free to vary over a large range $\sigma_w$, will typically be penalized by a larger Occam factor than a simpler model. The Occam factor also penalizes models which have to be finely tuned to fit the data, and favours models for which the required precision of the parameters $\sigma_{w|D}$ is coarse. The Occam factor is thus a measure of complexity of the model but, unlike the V-C dimension or algorithmic complexity, it relates to the complexity of the predictions that the model makes in data space. This depends not only on the number of parameters in the model, but also on the prior probability that the model assigns to them. Which model achieves the greatest evidence is determined by a trade-off between minimizing this natural complexity measure and minimizing the data misfit.

Figure 6 displays an entire hypothesis space so as to illustrate the various probabilities in the analysis. There are three models, $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$, which have equal prior probabilities. Each model has one parameter $\mathbf{w}$ (each shown on a horizontal axis), but assigns a different prior range $\sigma_W$ to that parameter. $\mathcal{H}_3$ is the most flexible, *i.e.*, the most complex model, assigning the broadest prior range. A one-dimensional data space is shown by the vertical axis. Each model assigns a joint probability distribution $P(\mathcal{D}, \mathbf{w}|\mathcal{H}_i)$ to the data and the parameters, illustrated by a cloud of dots. These dots represent random samples from the full probability distribution. The total number of dots in each of the three model subspaces is the same, because we assigned equal priors to the models.



$P(D|\mathcal{H}_3)$
$P(D|\mathcal{H}_2)$
$D$
$P(D|\mathcal{H}_1)$
$D$
$P(\mathbf{w}|D,\mathcal{H}_1)$
$P(\mathbf{w}|D,\mathcal{H}_2)$
$P(\mathbf{w}|\mathcal{H}_1)$
$P(\mathbf{w}|D,\mathcal{H}_3)$
$P(\mathbf{w}|\mathcal{H}_2)$
$P(\mathbf{w}|\mathcal{H}_3)$
$\mathbf{w}$
$\mathbf{w}$
$\mathbf{w}$
$\sigma_{w|D}$
$\sigma_w$

Figure 6: A hypothesis space consisting of three exclusive models, each having one parameter $\mathbf{w}$, and a one-dimensional data set $D$. The 'data set' is a single measured value which differs from the parameter $\mathbf{w}$ by a small amount of additive noise. Typical samples from the joint distribution $P(D, w, \mathcal{H})$ are shown by dots. (NB, these are not data points.) The observed 'data set' is a single particular value for $D$ shown by the dashed horizontal line. The dashed curves below show the posterior probability of $\mathbf{w}$ for each model given this data set (c.f. figure 4). The evidence for the different models is obtained by marginalizing onto the $D$ axis at the left hand side (c.f. figure 5).

When a particular data set $D$ is received (horizontal line), we infer the posterior distribution of $\mathbf{w}$ for a model ($\mathcal{H}_3$, say) by reading out the density along that horizontal line, and normalizing. The posterior probability $P(\mathbf{w}|D, \mathcal{H}_3)$ is shown by the dotted curve at the bottom. Also shown is the prior distribution $P(\mathbf{w}|\mathcal{H}_3)$ (c.f. figure 5).

We obtain figure 4 by marginalizing the joint distributions $P(D, \mathbf{w}|\mathcal{H}_i)$ onto the $D$ axis at the left hand side. This procedure gives the predictions of each model in data space. For the data set $D$ shown by the dotted horizontal line, the evidence $P(D|\mathcal{H}_3)$ for the more flexible model $\mathcal{H}_3$ has a smaller value than the evidence for $H_2$. This is because $\mathcal{H}_3$ placed less predictive probability (fewer dots) on that line. Looking back at the distributions over $\mathbf{w}$, $\mathcal{H}_3$ has smaller evidence because the Occam factor $\sigma_{w|D}/\sigma_w$ is smaller for $\mathcal{H}_3$ than for $\mathcal{H}_2$. The simplest model $\mathcal{H}_1$ has the smallest evidence of all, because the best fit that it can achieve to the data $D$ is very poor. Given this data set, the most probable model is $\mathcal{H}_2$.

## Occam factor for several parameters

If the posterior is well approximated by a Gaussian, then the Occam factor is obtained from the determinant of the corresponding covariance matrix (c.f. equation (9)):

$$\begin{array}{rcl} P(D|\mathcal{H}_i) & \simeq & \underbrace{P(D|\mathbf{w}_{\rm MP}, H_i)} \times \underbrace{P(\mathbf{w}_{\rm MP}|\mathcal{H}_i) \det^{-\frac{1}{2}}(\mathbf{A}/2\pi)}, \\ \text{Evidence} & \simeq & \text{Best fit likelihood} \times \qquad \text{Occam factor} \end{array} \tag{11}$$

where $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, \mathcal{H}_i)$, the Hessian which we evaluated when we calculated the error bars on $\mathbf{w}_{\rm MP}$ (equation (6)). As the number of data collected, $N$, increases, this Gaussian approximation is expected to become increasingly accurate.

In summary, Bayesian model selection is a simple extension of maximum likelihood model selection: **the evidence is obtained by multiplying the best fit likelihood by the Occam factor.**

To evaluate the Occam factor we need only the Hessian $\mathbf{A}$, if the Gaussian approximation is good. Thus the Bayesian method of model comparison by evaluating the evidence is no more demanding computationally than the task of finding for each model the best fit parameters and their error bars.

For background reading on Bayesian methods, the following references may be helpful. Bayesian methods are introduced and contrasted with orthodox statistics in (Jaynes 1983, Gull 1988, Loredo 1990). The Bayesian Occam's razor is demonstrated on model problems in (Gull 1988, MacKay 1992a). Useful textbooks are (Box and Tiao 1973, Berger 1985).

## Bayesian methods meet neural networks

The two ideas of neural network modelling and Bayesian statistics might at first glance seem uneasy bed-fellows. Neural networks are non-linear parallel computational devices inspired by the structure of the brain. 'Backpropagation networks' are able to learn, by example, to solve prediction and classification problems. Such a neural network is typically viewed as a black box which slaps together, by hook or by crook, an incomprehensible solution to a poorly understood problem. In contrast, Bayesian statistics are characterized by an insistence on coherent inference based on clearly defined axioms; in Bayesian circles, an 'ad hockery' is a capital offence. Thus Bayesian statistics and neural networks might seem to occupy opposite extremes of the data modelling spectrum.

However, there is a common theme uniting the two. Both fields aim to create models which are well-matched to the data. Neural networks can be viewed as more flexible versions of traditional regression techniques. Because they are more flexible (non-linear) they are able to fit the data better, and model regularities in the data that linear models cannot capture. The problem with neural networks is that an over-flexible network might be duped by stray correlations in the data into 'discovering' non-existent structure. This is where Bayesian methods play a complementary role. Using Bayesian probability theory one can automatically infer how flexible a model is warranted by the data; the Bayesian Occam's razor automatically suppresses the tendency to discover spurious structure in data. The philosophy advocated here is to use flexible models, like neural networks; then control the complexity of these models in the light of the data, using Bayesian methods.

Occam's razor is needed in neural networks for the reason illustrated in figure 7. Consider a control parameter which influences the complexity of a model, for example a regularization constant $\alpha$ (weight decay parameter). As the control parameter is varied to increase the complexity of the model (descending from figure 7*a*-*c* and going from left to right across figure 7*d*), the best fit to the **training** data that the model can achieve becomes increasingly good. However, the empirical performance of the model, the **test error**, has a minimum as a function of the control parameters. *An over-complex model overfits the data and generalizes poorly.* This problem may also complicate the choice of architecture in a multilayer perceptron, the radius of the basis functions in a radial basis function network, and the choice of the input variables themselves in

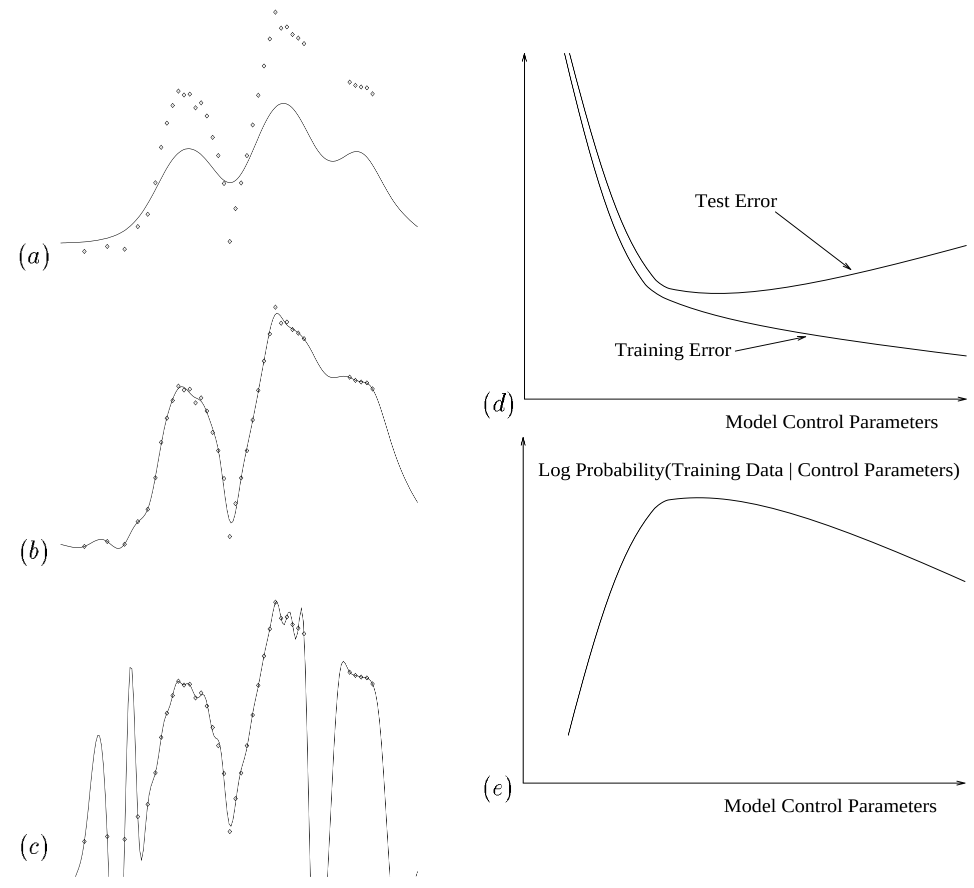


Figure 7: Optimization of model complexity. Figures *a*-*c* show a radial basis function model interpolating a simple data set with one input variable and one output variable. As the regularization constant is varied to increase the complexity of the model (from *a* to *c*), the interpolant is able to fit the training data increasingly well, but beyond a certain point the generalization ability (test error) of the model deteriorates. Probability theory allows us to optimize the control parameters without needing a test set.

any regression problem. Finding values for model control parameters that are appropriate for the data is therefore an important and non-trivial problem.

A central message is illustrated in figure 7*e*. If we give a probabilistic interpretation to the model, then we can evaluate the 'evidence' for alternative values of the control parameters. Over-complex models turn out to be less probable, and the quantity $P(\text{Data}|\text{Control Parameters})$ can be used as an objective function for optimization of model control parameters. The setting of $\alpha$ that maximizes this quantity is displayed in figure 7*b*.

Bayesian optimization of model control parameters has four important advantages. (1) No 'test set' or 'validation set' is involved, so all available training data can be devoted to both model fitting and model comparison. (2) Regularization constants can be optimized on-line, *i.e.*, simultaneously with the optimization of ordinary model parameters. (3) The Bayesian objective function is not noisy, in contrast to a cross-validation measure. (4) The gradient of the evidence with respect to the control parameters can be evaluated, making it possible to simultaneously optimize a large number of control parameters.

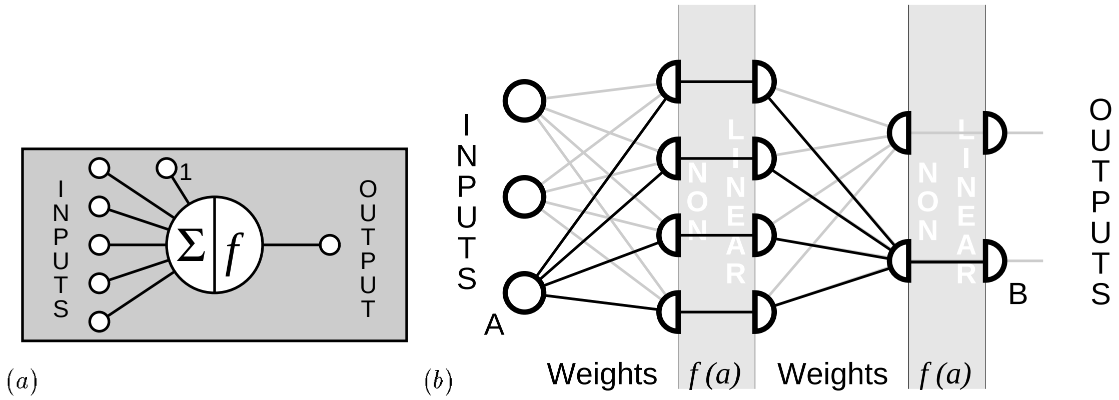


Figure 8: **Graphical representation of a neural network.**
(*a*) A single neuron. (*b*) A two layer network with 3 inputs, four hidden units and 2 outputs. The weights from input A to the hidden layer have been highlighted, and the weights from the hidden layer to output B.

## 3 Neural networks as probabilistic models

A supervised neural network is a non-linear parameterized mapping from an input $\mathbf{x}$ to an output $\mathbf{y} = \mathbf{y}(\mathbf{x}; \mathbf{w}, \mathcal{A})$. The output is a continuous function of the input and of the parameters $\mathbf{w}$; the architecture of the net, *i.e.*, the functional form of the mapping, is denoted by $\mathcal{A}$. Such networks can be 'trained' to perform regression and classification tasks.

### Regression networks

In the case of a regression problem, the mapping for a network with one hidden layer may have the form:

$$\text{Hidden layer:} \quad a_j^{(1)} = \sum_l w_{jl}^{(1)} x_l + \theta_j^{(1)}; \quad h_j = f^{(1)}(a_j^{(1)}) \tag{12}$$

$$\text{Output layer:} \quad a_i^{(2)} = \sum_j w_{ij}^{(2)} h_j + \theta_i^{(2)}; \quad y_i = f^{(2)}(a_i^{(2)}) \tag{13}$$

where, for example, $f^{(1)}(a) = \tanh(a)$, and $f^{(2)}(a) = a$. Here $l$ runs over the inputs $x_1 \ldots x_L$, $j$ runs over the hidden units, and $i$ runs over the outputs. The 'weights' $w$ and 'biases' $\theta$ together make up the parameter vector $\mathbf{w}$. The non-linear 'sigmoid' function $f^{(1)}$ at the hidden layer gives the neural network greater computational flexibility than a standard linear regression model. Graphically, we can represent the neural network as a set of layers of connected neurons. One neuron is shown in figure 8. The entire network is shown in figure 8. This is called a two layer network because it has two layers of weights. In terms of layers of neurons, it has one input layer, one hidden layer and one output layer.

### What sorts of functions can these networks implement?

Just as for the simple network we made an exploration of its weight space, examining the functions it could produce, let us make a small exploration of the weight space of a regression network. In figure 9 I take a network with one input and one output and a large number $H$ of hidden units, set the biases and weights $\theta_j^{(1)}$, $w_{jl}^{(1)}$, $\theta_i^{(2)}$ and $w_{ij}^{(2)}$ to random values, and plot the resulting function $y(x)$. I set the hidden unit biases $\theta_j^{(1)}$ to random values from a Gaussian with zero mean and standard deviation $\sigma_{\text{bias}}$; the input to hidden weights $w_{jl}^{(1)}$ to random values with

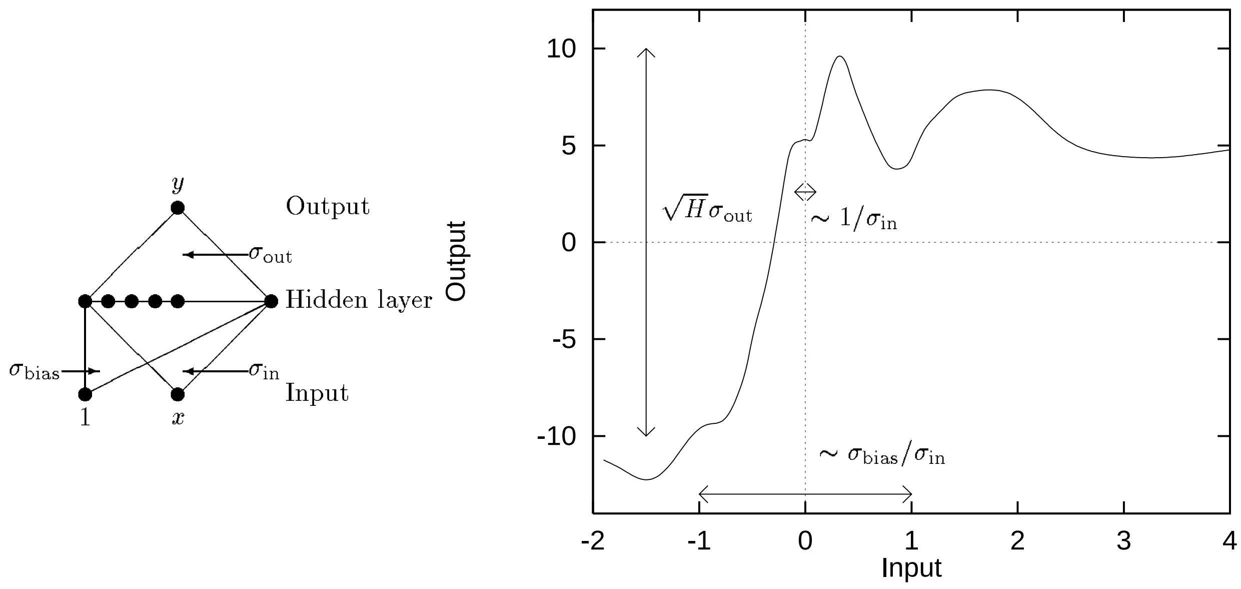


Figure 9: Properties of a function produced by a random network.
The vertical scale of a typical function produced by the network with random weights is of order $\sqrt{H}\sigma_{\text{out}}$; the horizontal range in which the function varies significantly is of order $\sigma_{\text{bias}}/\sigma_{\text{in}}$; and the shortest horizontal length scale is of order $1/\sigma_{\text{in}}$.
This network had $H = 400$, and Gaussian weights were generated with $\sigma_{\text{bias}} = 4$, $\sigma_{\text{in}} = 8$, and $\sigma_{\text{out}} = 0.5$.

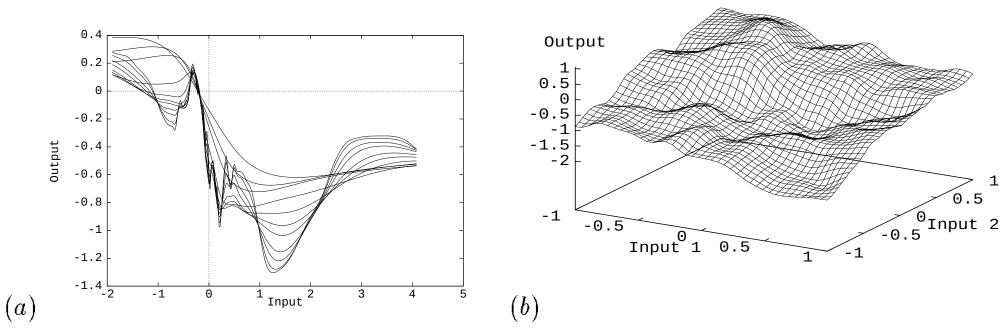


Figure 10: **Samples from the prior of** $(a)$ **a one input network; and** $(b)$ **a two input network.**
Figure 10a shows one function for each of a sequence of values of $\sigma_{\text{bias}}$ and $\sigma_{\text{in}}$. with $H = 400$, $\sigma^w_{\text{out}} = 0.05$. For each graph the parameter $\sigma^w_{\text{bias}}$ takes a different value in the sequence: 8, 6, 4, 3, 2, 1.6, 1.2, 0.8, 0.4, 0.3, 0.2. The parameter $\sigma^w_{\text{in}}$ was also varied such that $\sigma^w_{\text{in}}/\sigma^w_{\text{bias}} = 5.0$. The larger values of $\sigma^w_{\text{in}}$ and $\sigma^w_{\text{bias}}$ produce the more complex functions with more fluctuations. $(b)$ **A typical function produced by a two input network** with $\{H, \sigma^w_{\text{in}}, \sigma^w_{\text{bias}}, \sigma^w_{\text{out}}\} = \{400, 8.0, 8.0, 0.05\}$.

standard deviation $\sigma_{\text{in}}$; and the bias and output weights $\theta_i^{(2)}$ and $w_{ij}^{(2)}$ to random values with standard deviation $\sigma_{\text{out}}$.

The sort of functions that we obtain depend on the values of $\sigma_{\text{bias}}$, $\sigma_{\text{in}}$ and $\sigma_{\text{out}}$ (Neal 1994) (see labels on figure 9; see also figure 10). As the weights and biases are made bigger we obtain more complex functions with more features and a greater sensitivity to the input variable. Neal (1994) has also shown that in the limit as $H \to \infty$ the statistical properties of the functions generated by randomizing the weights are independent of the number of hidden units; so, interestingly, the complexity of the functions is independent of the number of parameters in the model. What determines the complexity of the typical functions is the characteristic magnitude of the weights. Thus we anticipate that when we fit these models to real data, an important way of controlling the complexity of the fitted function will be by controlling the characteristic magnitude of the weights.

Figure 10b shows one typical function produced by a network with two inputs and one output. This should be contrasted with the function produced by a traditional linear regression model, which is a flat plane. Clearly neural networks can create functions with much more structure than a linear regression!

## How a regression network is traditionally trained

This network is trained using a data set $D = \{\mathbf{x}^{(m)}, \mathbf{t}^{(m)}\}$ by adjusting $\mathbf{w}$ so as to minimize an error function, *e.g.*,

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_m \sum_i \left( t_i^{(m)} - y_i(\mathbf{x}^{(m)}; \mathbf{w}) \right)^2 . \tag{14}$$

This objective function is a sum of terms, one for each input/target pair $\{\mathbf{x}, t\}$, measuring how close the output $\mathbf{y}(\mathbf{x}; \mathbf{w})$ is to the target $t$.

This minimization is based on repeated evaluation of the gradient of $E_D$ using 'backpropagation' (the chain rule) (Rumelhart et al. 1986). Often, regularization (also known as 'weight decay') is included, modifying the objective function to:

$$M(\mathbf{w}) = \beta E_D + \alpha E_W \tag{15}$$

where, for example, $E_W = \frac{1}{2} \sum_i w_i^2$. This additional term favours small values of $\mathbf{w}$ and decreases the tendency of a model to 'overfit' noise in the training data.

## Neural network learning as inference

The neural network learning process above can be given the following probabilistic interpretation. The error function is interpreted as minus the log likelihood for a noise model:

$$P(D|\mathbf{w}, \beta, \mathcal{H}) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D). \tag{16}$$

Thus, the use of the sum-squared error $E_D$ (14) corresponds to an assumption of Gaussian noise on the target variables, and the parameter $\beta$ defines a noise level $\sigma_\nu^2 = 1/\beta$.

Similarly the regularizer is interpreted in terms of a log prior probability distribution over the parameters:

$$P(\mathbf{w}|\alpha, \mathcal{H}) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W). \tag{17}$$

If $E_W$ is quadratic as defined above, then the corresponding prior distribution is a Gaussian with variance $\sigma_W^2 = 1/\alpha$. The probabilistic model $\mathcal{H}$ specifies the functional form $\mathcal{A}$ of the network, the likelihood (16), and the prior (17).

The objective function $M(\mathbf{w})$ then corresponds to the **inference** of the parameters $\mathbf{w}$, given the data:

$$P(\mathbf{w}|D, \alpha, \beta, \mathcal{H}) = \frac{P(D|\mathbf{w}, \beta, \mathcal{H}) P(\mathbf{w}|\alpha, \mathcal{H})}{P(D|\alpha, \beta, \mathcal{H})} \tag{18}$$

$$= \frac{1}{Z_M} \exp(-M(\mathbf{w})). \tag{19}$$

The $\mathbf{w}$ found by (locally) minimizing $M(\mathbf{w})$ is then interpreted as the (locally) most probable parameter vector, $\mathbf{w}_{\mathrm{MP}}$.

Why is it natural to interpret the error functions as *log* probabilities? Error functions are usually additive. For example, $E_D$ is a *sum* of squared errors. Probabilities, on the other hand, are multiplicative: for independent events A and B, the joint probability is $P(A,B) = P(A)P(B)$. The logarithmic mapping maintains this correspondence.

The interpretation of $M(\mathbf{w})$ as a log probability adds little new at this stage. But new tools will emerge when we proceed to other inferences. First, though, let us establish the probabilistic interpretation of classification networks, to which the same tools apply.

## Binary classification networks

If the targets $t$ in a data set are binary classification labels (0,1), it is natural to use a neural network whose output $y(\mathbf{x};\mathbf{w},\mathcal{A})$ is bounded between 0 and 1, and is interpreted as a probability $P(t=1|\mathbf{x},\mathbf{w},\mathcal{A})$. For example, a network with one hidden layer could be described by equations (12) and (13), with $f^{(2)}(a) = 1/(1+e^{-a})$. The error function $\beta E_D$ is replaced by the log likelihood:

$$G(\mathbf{w}) = \sum_m t^{(m)} \log y(\mathbf{x}^{(m)};\mathbf{w}) + (1-t^{(m)}) \log(1-y(\mathbf{x}^{(m)};\mathbf{w})). \tag{20}$$

The total objective function is then $M = -G + \alpha E_W$. Note that this includes no parameter $\beta$.

## Multi-class classification networks

For a multi-class classification problem, we can represent the targets by a vector, $\mathbf{t}$, in which a single element is set to 1, indicating the correct class, and all other elements are set to 0. In this case it is appropriate to use a 'softmax' network (Bridle 1989) having coupled outputs which sum to one and are interpreted as class probabilities $y_i = P(t_i=1|\mathbf{x},\mathbf{w},\mathcal{A})$. The last part of equation (13) is replaced by:

$$y_i = \frac{e^{a_i}}{\sum_{i'} e^{a_{i'}}}. \tag{21}$$

The log likelihood in this case is

$$G = \sum_m \sum_i t_i \log y_i(\mathbf{x}^{(m)};\mathbf{w}). \tag{22}$$

As in the case of the regression network, the minimization of the objective function $M(\mathbf{w}) = -G + \alpha E_W$ corresponds to an inference of the form (19). Let us now study the variety of useful results that can be built on this interpretation. The results will refer to regression models; the corresponding results for classification models are obtained by replacing $\beta E_D$ by $-G$, and $Z_D(\beta)$ by 1.

## Implementation

Bayesian inference for data modelling problems may be implemented by analytical methods, by Monte Carlo sampling, or by deterministic methods employing Gaussian approximations. For neural networks there are few analytic methods. Sophisticated Monte Carlo methods which make use of gradient information have been applied to some model problems by Neal (1993a). The methods reviewed in the next few sections are based on Gaussian approximations to the posterior distribution. The methods of section 9 employ Monte Carlo methods.

## 4 Setting regularization constants $\alpha$ and $\beta$

The control parameters $\alpha$ and $\beta$ determine the complexity of the model. The term model here refers to a triple: the network architecture; the form of the prior on the parameters; and the form of the noise model. Different values for the hyperparameters $\alpha$ and $\beta$ define different sub-models. To infer $\alpha$ and $\beta$ given the data, we simply apply the rules of probability theory:

$$P(\alpha, \beta | D, \mathcal{H}) = \frac{P(D|\alpha, \beta, \mathcal{H}) P(\alpha, \beta | \mathcal{H})}{P(D|\mathcal{H})}. \tag{23}$$

The data-dependent factor $P(D|\alpha, \beta, \mathcal{H})$ is the normalizing constant from our previous inference (18); we call this factor the 'evidence' for $\alpha$ and $\beta$.

Assuming we have only weak prior knowledge about the noise level and the smoothness of the interpolant, the evidence framework optimizes the constants $\alpha$ and $\beta$ by finding the maximum of the evidence for $\alpha$ and $\beta$. If we can approximate the posterior probability distribution in equation (19) by a single Gaussian,

$$P(\mathbf{w}|D, \alpha, \beta, \mathcal{H}) \simeq \frac{1}{Z'_M} \exp\left(-M(\mathbf{w}_{\mathrm{MP}}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{\mathrm{MP}})^{\mathrm{T}} \mathbf{A} (\mathbf{w} - \mathbf{w}_{\mathrm{MP}})\right), \tag{24}$$

where $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, \mathcal{H})$, then the evidence for $\alpha$ and $\beta$ can be written down:

$$\log P(D|\alpha, \beta, \mathcal{H}) = \log \frac{Z'_M}{Z_W(\alpha) Z_D(\beta)} \tag{25}$$

$$= -M(\mathbf{w}_{\mathrm{MP}}) - \frac{1}{2} \log \det \mathbf{A} - \log Z_W(\alpha) - \log Z_D(\beta) + \frac{k}{2} \log 2\pi, \tag{26}$$

where $k$ is the number of parameters in $\mathbf{w}$. The terms $-\frac{1}{2} \log \det \mathbf{A} - \log Z_W(\alpha)$ constitute the log of a volume factor which penalizes small values of $\alpha$: the ratio $(2\pi)^{k/2} \det^{-\frac{1}{2}} \mathbf{A} / Z_W(\alpha)$ is the ratio of the posterior accessible volume in parameter space to the prior accessible volume. The maximum of the evidence has some elegant properties which allow it to be located efficiently by on-line re-estimation techniques. Technically, there may be multiple evidence maxima, but this is not common when the model space is well matched to the data. As shown in (Gull 1989, MacKay 1992a), the maximum evidence $\alpha = \alpha_{\mathrm{MP}}$ satisfies the following self-consistent equation:

$$1/\alpha_{\mathrm{MP}} = \sum_i w_i^{\mathrm{MP}\,2} / \gamma \tag{27}$$

where $\mathbf{w}^{\mathrm{MP}}$ is the parameter vector which minimizes the objective function $M = \beta E_D + \alpha E_W$, and $\gamma$ is the 'number of well-determined parameters', given by

$$\gamma = k - \alpha \mathrm{Trace}(\mathbf{A}^{-1}). \tag{28}$$

Here $k$ is the total number of parameters, and the matrix $\mathbf{A}^{-1}$ measures the size of the error bars on the parameters $\mathbf{w}$ (equation (6)). Thus $\gamma \to k$ when the parameters are all well-determined in relation to their prior range, which is defined by $\alpha$. The quantity $\gamma$ always lies between 0 and $k$. Recalling that $\alpha$ corresponds to the variance $\sigma_W^2 = 1/\alpha$ of the assumed distribution for $\{w_i\}$, equation (27) specifies an intuitive condition for matching the prior to the data: the variance is estimated by $\sigma_W^2 = \langle w^2 \rangle$, where the average is over the $\gamma$ effective well determined parameters; the other $k - \gamma$ effective parameters having been set to zero by the prior.

Similarly, in a regression problem with a Gaussian noise model, the maximum evidence value of $\beta$ satisfies:

$$1/\beta_{\mathrm{MP}} = 2E_D / (N - \gamma). \tag{29}$$

Since $2E_D$ is the sum of squared residuals, this expression can be recognized as a variance estimator with the number of degrees of freedom set to $\gamma$.

Equations (27) and (29) can be used as re-estimation formulae for $\alpha$ and $\beta$. The computational overhead for these Bayesian calculations is not severe: it is only necessary to evaluate properties of the error bar matrix, $\mathbf{A}^{-1}$. This matrix may be evaluated explicitly (MacKay 1992c, Bishop 1992), which does not take significant time when the number of parameters is small (a few hundred). For large problems these calculations can be performed more efficiently using algorithms which evaluate products $\mathbf{A}\mathbf{v}$ without explicitly evaluating $\mathbf{A}$ (Skilling 1993, Pearlmutter 1994).

## Relationship to ideal hierarchical Bayesian modelling

Bayesian probability theory has been used above to *optimize* the hyperparameters $\alpha$ and $\beta$. This procedure is known in some circles as 'generalized maximum likelihood'. Ideally we would *integrate over* these nuisance parameters in order to obtain the posterior distribution over the parameters $P(\mathbf{w}|D,\mathcal{H})$ and the predictive distributions $P(\mathbf{t}^{(N+1)}|D,\mathcal{H})$; however, if a hyperparameter is well determined by the data, integrating over it is very much like estimating the hyperparameter from the data and then using that estimate in our equations (Bretthorst 1988, Gull 1988, MacKay 1995b). The intuition is that if, in the predictive distribution

$$P(\mathbf{t}^{(N+1)}|D,\mathcal{H}) = \int d\alpha\, P(\mathbf{t}^{(N+1)}|D,\alpha,\mathcal{H})P(\alpha|D,\mathcal{H}), \tag{30}$$

the posterior $P(\alpha|D,\mathcal{H})$ is sharply peaked at $\alpha = \alpha_{\rm MP}$ with width $\sigma_{\log\alpha|D}$, and if the distribution $P(\mathbf{t}^{(N+1)}|D,\alpha,\mathcal{H})$ varies slowly with $\log\alpha$ on a scale of $\sigma_{\log\alpha|D}$, then $P(\alpha|D,\mathcal{H})$ is effectively a delta function, so that:

$$P(\mathbf{t}^{(N+1)}|D,\mathcal{H}) \simeq P(\mathbf{t}^{(N+1)}|D,\alpha_{\rm MP},\mathcal{H}). \tag{31}$$

Now the error bars on $\log\alpha$ and $\log\beta$, found by differentiating $\log P(D|\alpha,\beta,\mathcal{H})$ twice, are (MacKay 1992a):

$$\sigma^2_{\log\alpha|D} \simeq 2/\gamma; \quad \sigma^2_{\log\beta|D} \simeq 2/(N-\gamma). \tag{32}$$

Thus the error introduced by optimizing $\alpha$ and $\beta$ is expected to be small for $\gamma \gg 1$ and $N-\gamma \gg 1$. How large $\gamma$ needs to be depends on the problem, but for many neural network problems a value of $\gamma$ greater than about 3 may suffice, since the predictions of an optimized network are often insensitive to an $e$-fold change in $\alpha$.

It is often possible to integrate over $\alpha$ and $\beta$ early in the calculation, obtaining a true prior and a true likelihood (Bretthorst 1988, Gull 1988). Some authors have recommended this procedure (Buntine and Weigend 1991, Wolpert 1993), but it is counterproductive as far as practical manipulation is concerned (Gull 1988, MacKay 1995b): the resulting true posterior is a skew-peaked distribution, and apart from Monte Carlo methods there are currently no computational techniques which can cope directly with such distributions.

Later a correction term will be given which approximates the integration over $\alpha$ and $\beta$ when predictions are made, *i.e.* as a final step in the calculations.

## Multiple regularization constants

For simplicity, it has so far been assumed that there is only a single class of weights, which are modelled as coming from a single Gaussian prior with $\sigma_W^2 = 1/\alpha$. However, in dimensional terms, weights usually fall into three or more distinct groups, which for consistency should not be modelled as coming from a single prior. It is therefore desirable to divide the parameters into several classes $c$, with independent scales $\alpha_c$. Assuming a Gaussian prior for each class, we can define $E_{W(c)} = \sum_{i\in c} w_i^2/2$, and assign a Gaussian prior:

$$P(\{w_i\}|\alpha_c,\mathcal{H}) = \frac{1}{\prod Z_{W(c)}} \exp\left(-\sum_c \alpha_c E_{W(c)}\right), \tag{33}$$

This gives a weight decay scheme with a different decay rate $\alpha_c$ for each class. It is often found that network performance can be enhanced by this division of weights into different classes. The automatic relevance determination model (section 6) uses this prior.

The evidence framework optimizes the decay constants by finding their most probable value, *i.e.*, the maximum over $\{\alpha_c\}$ of $P(D|\{\alpha_c\},\mathcal{H})$. And, as before, the maximum evidence $\{\alpha_c\}$ satisfy the following self-consistent equations:

$$1/\alpha_c^{\mathrm{MP}} = \sum_{i\in c} w_i^{\mathrm{MP}\,2}/\gamma_c \tag{34}$$

where $\mathbf{w}^{\mathrm{MP}}$ is the parameter vector which minimizes the objective function $M = \beta E_D + \sum_c \alpha_c E_{W(c)}$, and $\gamma_c$ is the number of well-determined parameters in class $c$, $\gamma_c = k_c - \alpha_c \mathrm{Trace}_c(\mathbf{A}^{-1})$; $k_c$ is the number of parameters in class $c$, and the trace is over those parameters only.

For simplicity, the following discussion will assume once more that there is only a single parameter $\alpha$.

## 5 Model comparison

The evidence framework divides our inferences into distinct 'levels of inference', of which we have now completed the first two.

- **Level 1:** Infer the parameters $\mathbf{w}$ for given values of $\alpha, \beta$:

$$P(\mathbf{w}|D,\alpha,\beta,\mathcal{H}) = \frac{P(D|\mathbf{w},\alpha,\beta,\mathcal{H})P(\mathbf{w}|\alpha,\beta,\mathcal{H})}{P(D|\alpha,\beta,\mathcal{H})}. \tag{35}$$

- **Level 2a:** Infer $\alpha, \beta$:

$$P(\alpha,\beta|D,\mathcal{H}) = \frac{P(D|\alpha,\beta,\mathcal{H})P(\alpha,\beta|\mathcal{H})}{P(D|\mathcal{H})}. \tag{36}$$

- **Level 2b:** Compare models:

$$P(\mathcal{H}|D) \propto P(D|\mathcal{H})P(\mathcal{H}). \tag{37}$$

There is a pattern in these three applications of Bayes' rule: at each of the higher levels 2a and 2b, the data-dependent factor (*e.g.* in level 2a, $P(D|\alpha,\beta,\mathcal{H})$) is precisely the normalizing constant (the 'evidence') from the preceding level of inference. This pattern of inference continues when we compare different models $\mathcal{H}$, which might use different architectures, preprocessings, regularizers or noise models. Alternative models are ranked by evaluating $P(D|\mathcal{H})$, the normalizing constant of inference (36).

In the preceding section we reached level 2a by using a Gaussian approximation to $P(\mathbf{w}|D,\alpha,\beta,\mathcal{H})$. We now evaluate the evidence for $\mathcal{H}$. Using a Gaussian approximation for $P(\log\alpha,\log\beta|D,\mathcal{H})$, and neglecting the slight correlations in this posterior, we obtain the estimate

$$P(D|\mathcal{H}) \simeq P(D|\alpha_{\mathrm{MP}},\beta_{\mathrm{MP}},\mathcal{H})P(\log\alpha_{\mathrm{MP}},\log\beta_{\mathrm{MP}}|\mathcal{H})\;(2\pi)^{\frac{1}{2}}\sigma_{\log\alpha|D}(2\pi)^{\frac{1}{2}}\sigma_{\log\beta|D}, \tag{38}$$

where $P(D|\alpha_{\mathrm{MP}},\beta_{\mathrm{MP}},\mathcal{H})$ is obtained from equation (26), and the error bars on $\log\alpha$ and $\log\beta$ are as given in equation (32). This Gaussian approximation over $\alpha$ and $\beta$ holds good for $\gamma \gg 1$ and $N-\gamma \gg 1$ (MacKay 1995b).

## Multi-modal distributions

The preceding exposition falls into difficulty if the posterior distribution $P(\mathbf{w}|D, \alpha, \beta, \mathcal{H})$ is significantly multi-modal; this is usually the case for multi-layer neural networks. However we can persist with the use of Gaussian approximations if we introduce two modifications.

First, we recognize that a typical optimum $\mathbf{w}_{\mathrm{MP}}$ will be related to a number of equivalent optima by symmetry operations, such as interchange of hidden units and inversion of signs of weights. When evaluating the evidence using a local Gaussian approximation, a symmetry factor should be included in equation (25) to take into account these equivalent islands of probability mass. In the case of a net with one hidden layer of $H$ units, the appropriate permutation factor is $H!2^H$, for general $\mathbf{w}_{\mathrm{MP}}$.

Second, there are multiple optima which are not related to each other by model symmetries. We modify the above framework by changing our goals; specifically, we view each of the local probability peaks as a distinct model. Instead of inferring the posterior over $\alpha, \beta$ for the entire model $\mathcal{H}$, we allow each local peak of the posterior to choose its own optimal value for these parameters. Similarly, instead of evaluating the evidence for the entire model $\mathcal{H}$, we aim to calculate the posterior probability mass in each local optimum. This seems natural, since a typical implementation of the model will involve setting the parameter vector to a particular value, or a small set of values. Thus we do not care about the probability of an entire model; what matters is the probability of the local solutions we find. The same method of chopping up a complex model space is used in the unsupervised classification system, AutoClass (Hanson, Stutz and Cheeseman 1991).

Henceforth, the term 'model' will refer to a pair $\{\mathcal{H}, S_{\mathbf{w}^*}\}$, where $\mathcal{H}$ denotes the model specification and $S_{\mathbf{w}^*}$ specifies a solution neighbourhood around an optimum $\mathbf{w}^*$. Adopting this shift in objective, the Gaussian integrals above can be used without alteration to set $\alpha$ and $\beta$ and to compare alternative solutions, assuming that the posterior probability consists of well separated islands in parameter space that are roughly Gaussian.

For general $\alpha$ and $\beta$ the Gaussian approximation over $\mathbf{w}$ will not be accurate; however we only need it to be accurate for the small range of $\alpha$ and $\beta$ close to their most probable value. For sufficiently large amounts of data compared to the number of parameters, this approximation is expected to hold. Practical experience indicates that this is a useful approximation for many real problems.

## Practical recommendation regarding model comparison

When designing a model space, one may have the choice between a discrete model space and a continuous model space. Examples of discrete model spaces are: (1) a set of interpolation models that employ different sets of input variables; (2) two interpolation models, which respectively assume the noise distribution is Gaussian ($\propto \exp -\beta r^2/2$, where $r$ is a residual), and exponential ($\propto \exp -\beta r$). One could compare these models by evaluating the evidence for each discrete model. Alternatively, one could imagine continuous model spaces, where there is a single mega-model with continuous hyperparameters: (1) a single interpolation model with a very large number of inputs, and with one hyperparameter per input controlling the participation of that input in the regression; (2) an interpolation model with noise distribution $\propto \exp -\beta r^p/p$, where $p \in (1, 2)$ is an unknown hyperparameter.

The numerical evaluation of the evidence for discrete models, using the Gaussian approximation, involves computation of the log determinant of $\mathbf{A}^{-1}$. In contrast, in a continuous model space, we can evaluate the *derivative* of the evidence with respect to the hyperparameters. Such derivatives typically involve something to do with the trace of $\mathbf{A}^{-1}$. There are two reasons for preferring the latter. First, the trace is a numerically better-conditioned object to compute. Second, the gradient of the evidence can be evaluated with respect to multiple hyperparameters, giving lots of information about where to go in the hypothesis space; one would have to make many discrete evidence evaluations to get the same information, which for numerical reasons would probably be more noisy.

I therefore recommend the deliberate design of continuous model spaces where possible. An illustration of this attitude is given by the Automatic Relevance Determination model.

## 6 Automatic Relevance Determination

The automatic relevance determination (ARD) model (MacKay and Neal 1994) can be implemented with the methods described in the previous sections.

Suppose that in a regression problem there are many input variables, of which some are irrelevant to the prediction of the output variable. Because a finite data set will show random correlations between the irrelevant inputs and the output, any conventional neural network (even with weight decay) will fail to set the coefficients for these junk inputs to zero. Thus the irrelevant variables will hurt the model's performance, particularly when the variables are many and the data are few.

What is needed is a model whose prior over the regression parameters embodies the concept of relevance, so that the model is effectively able to infer which variables are relevant and then switch the others off. A simple and 'soft' way of doing this is to introduce multiple weight decay constants, one '$\alpha$' associated with each input. The decay rates for junk inputs will automatically be inferred to be large, preventing those inputs from causing significant overfitting.

The ARD model uses the prior of equation (33). For a network having one hidden layer, the weight classes are: one class for each input, consisting of the weights from that input to the hidden layer; one class for the biases to the hidden units; and one class for each output, consisting of its bias and all the weights from the hidden layer. Control of the ARD model can be implemented using equation (34).

## 7 Error bars and predictions

Having progressed up the three levels of modelling, the next inference task is to make predictions with our adapted model. It is common practice simply to use the most probable values of $\mathcal{H}$, $\mathbf{w}$, etc., when making predictions, but this is not optimal. Bayesian prediction of a new datum $\mathbf{t}^{(N+1)}$ involves *marginalizing* over all these levels of uncertainty:

$$P(\mathbf{t}^{(N+1)}|D) = \sum_{\mathcal{H}} \int d\alpha\, d\beta \int d^k\mathbf{w}\, P(\mathbf{t}^{(N+1)}|\mathbf{w},\alpha,\beta,\mathcal{H}) P(\mathbf{w},\alpha,\beta,\mathcal{H}|D). \tag{39}$$

The evaluation of the distribution $P(\mathbf{t}^{(N+1)}|\mathbf{w},\alpha,\beta,\mathcal{H})$ for specified model parameters $\mathbf{w}$ is generally straightforward, requiring a single forward pass through the network. Typically, marginalization over $\mathbf{w}$ and $\mathcal{H}$ affects the predictive distribution significantly, but integration over $\alpha$ and $\beta$ has a lesser effect.

### Implementation

Marginalization can rarely be done analytically. The alternatives are the Gaussian approximations described here, and Monte Carlo methods (Neal 1993a).

### Error bars in regression

Integrating first over $\mathbf{w}$ for fixed $\alpha$ and $\beta$, the predictive distribution is:

$$P(\mathbf{t}^{(N+1)}|D,\alpha,\beta,\mathcal{H}) = \int d^k\mathbf{w}\, P(\mathbf{t}^{(N+1)}|\mathbf{w},\beta,\mathcal{H}) P(\mathbf{w}|D,\alpha,\beta,\mathcal{H}). \tag{40}$$

If a Gaussian approximation is made for the posterior $P(\mathbf{w}|D,\alpha,\beta,\mathcal{H})$; if the noise model is Gaussian; and if a local linearization of the output is made as a function of the parameters,

$$y(\mathbf{x}^{N+1},\mathbf{w}) \simeq y(\mathbf{x}^{N+1};\mathbf{w}_{\mathrm{MP}}) + \mathbf{g}\cdot(\mathbf{w}-\mathbf{w}_{\mathrm{MP}}), \tag{41}$$

with $\mathbf{g} = \frac{\partial y}{\partial \mathbf{w}}$; then the predictive distribution (40) is a straightforward Gaussian integral. This distribution has mean $y(\mathbf{x}^{N+1}, \mathbf{w}_{\text{MP}})$, and variance $\sigma^2_{t|\alpha,\beta} = \mathbf{g}^{\text{T}}\mathbf{A}^{-1}\mathbf{g} + \sigma^2_{\nu}$, where $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, \alpha, \beta, \mathcal{H})$.

Integration over the regularization constants $\alpha$ and $\beta$ contributes an additional variance in only one direction; to leading order in $\gamma^{-1}$, $P(\mathbf{t}^{(N+1)}|D, \mathcal{H})$ is normal, with variance (MacKay 1995b):

$$\sigma_t^2 = \mathbf{g}^{\text{T}} \left( \mathbf{A}^{-1} + (\sigma^2_{\log\alpha|D} + \sigma^2_{\log\beta|D}) \mathbf{w}'_{\text{MP}} {\mathbf{w}'_{\text{MP}}}^{\text{T}} \right) \mathbf{g} + \sigma^2_{\nu}, \tag{42}$$

where $\mathbf{w}'_{\text{MP}} \equiv \partial \mathbf{w}_{\text{MP}|\alpha} / \partial(\log \alpha) = \alpha \mathbf{A}^{-1}\mathbf{w}_{\text{MP}}$, and $\sigma^2_{\log\alpha|D} = \frac{2}{\gamma}$ and $\sigma^2_{\log\beta|D} = \frac{2}{N-\gamma}$.

## Integrating over models: committees

If we have multiple regression models $\mathcal{H}$, then our predictive distribution is obtained by summing together the predictive distribution of each model, weighted by its posterior probability. If a single prediction is required and the loss function is quadratic, the optimal prediction is a weighted mean of the models' predictions $y(\mathbf{x}^{N+1}; \mathbf{w}_{\text{MP}}, \mathcal{H})$. The weighting coefficients are the posterior probabilities, which are obtained from the evidences $P(D|\mathcal{H})$. If we cannot evaluate these accurately then alternative pragmatic prescriptions for the weighting coefficients exist (Breiman 1992).

## Error bars in classification

In the case of linearized regression discussed above, the mean of the predictive distribution (40) was identical to the prediction of the mean, $\mathbf{w}_{\text{MP}}$. This is not the case in classification problems. The best fit parameters give over-confident predictions. A non-Bayesian approach to this problem is to down-weight all predictions uniformly, by an empirically determined factor (Copas 1983). But a Bayesian viewpoint helps us to understand the cause of the problem, and provides a straightforward solution that is demonstrably superior to this ad hoc procedure.

This issue is illustrated for a simple two-class problem in figure 11. Figure 11a shows a binary data set, which, in figure 11b is modelled with a linear logistic function. The best fit parameter values give predictions which are shown by three contours. Are these reasonable predictions? Consider new data arriving at points A and B. The best-fit model assigns both of these examples probability 0.9 of being in class 1. But intuitively we might be inclined to assign a less confident probability (closer to 0.5) at B than at A, since point B is far from the training data.

Precisely this result is obtained by marginalizing over parameters, whose posterior probability distribution is depicted in figure 11c. Two random samples from the posterior define two different classification surfaces, which are illustrated in figures 11d, e. The point B is classified differently by these different plausible classifiers, whereas the classification of A is relatively stable. We obtain the Bayesian predictions (figure 11f) by averaging together the predictions of the plausible classifiers. The resulting 0.5 contour remains similar to that for the best-fit parameters. However, the width of the decision boundary increases as we move away from the data, in full accordance with intuition.

The Bayesian approach is superior because the best-fit model's predictions are *selectively* downweighted, to a different degree for each test case. The consequence is that a Bayesian classifier is better able to identify the points where the classification is uncertain. This pleasing behaviour results simply from a mechanical application of the rules of probability.

For a binary classifier, a numerical approximation to the integral over a Gaussian posterior distribution is given in (MacKay 1992b). An equivalent approximation for a multi-class classifier has not yet been implemented.

This marginalization can also be done by Monte Carlo methods. A disadvantage of a straightforward Monte Carlo approach would be that it is a poor way of estimating the probability of an improbable event, *i.e.* a $P(t|D, \mathcal{H})$ that is very close to zero, if the improbable event is most likely to occur in conjunction with improbable parameter values. In such cases one might instead temporarily add the event in question to the data set, and evaluate the evidence $P(D, t^{(N+1)}|\mathcal{H})$.

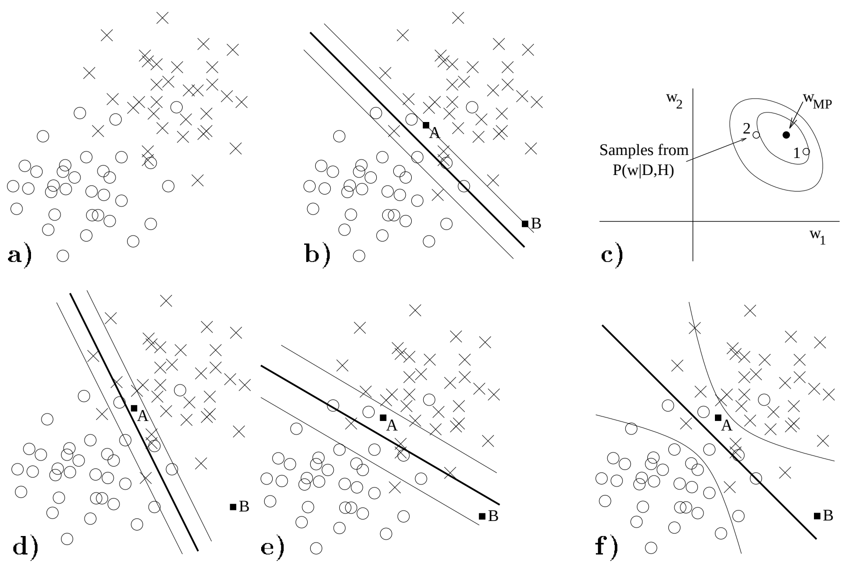


Figure 11: **Integrating over error bars in a classifier**
**a) A binary data set.** The two classes are denoted by $\times$=1, $\circ$=0. **b)** The data are modelled with a linear logistic function. Here the **best-fit** model is shown by its 0.1, 0.5 and 0.9 predictive contours. The best fit model assigns probability 0.9 of being in class 1 to both inputs A and B. **c)** The posterior probability distribution of the model parameters, $P(\mathbf{w}|D,\mathcal{H})$ (schematic; the third parameter, the bias, is not shown). The parameters are not perfectly determined by the data. Two typical samples from the posterior are indicated by the points labeled 1 and 2. The following two panels show the corresponding classification contours. **d)** Sample 1. **e)** Sample 2. Notice how the point B is classified differently by these different plausible classifiers, whereas the classification of A is relatively stable. **f)** We obtain the Bayesian predictions by integrating over the posterior distribution of **w**. The width of the decision boundary increases as we move away from the data (point B). See text for further discussion.

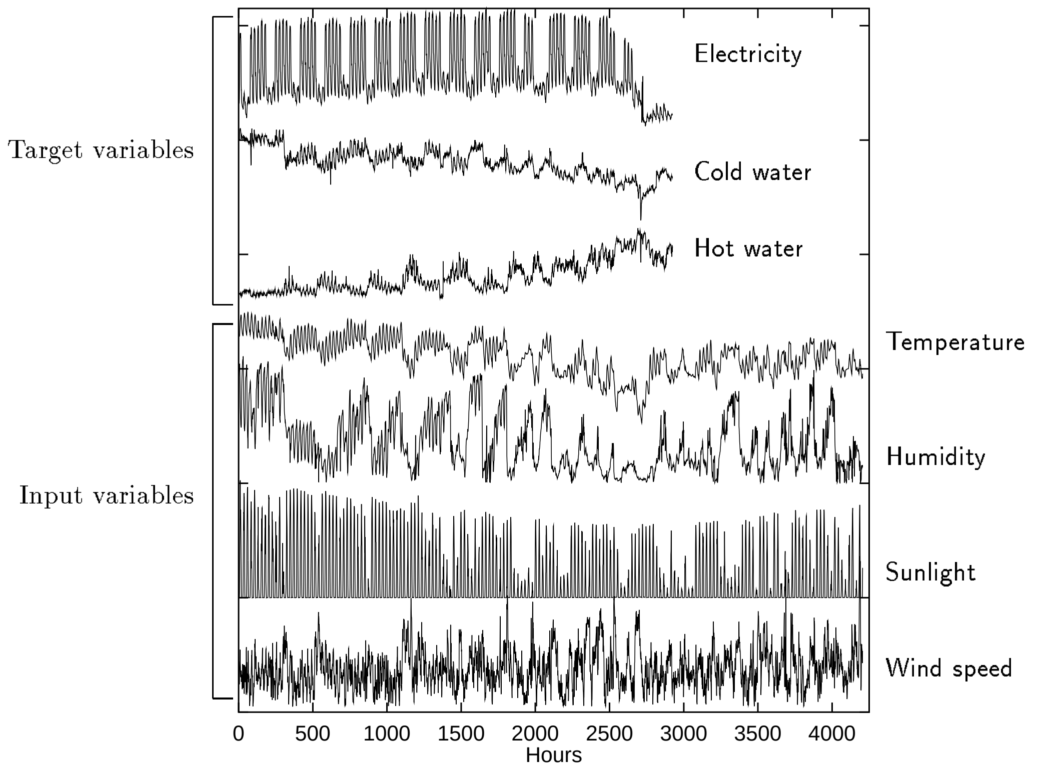
The desired probability is obtained by comparing this with either the previous evidence $P(D|\mathcal{H})$, or the evidence for the complementary virtual data set $P(D, \overline{t^{(N+1)}}|\mathcal{H})$.

## 8 Prediction competition

The American Society of Heating, Refrigeration and Air Conditioning Engineers organised a prediction competition in 1993. The competition had two parts, both of which involved creating an empirical model based on training data (as distinct from a physical model), and making predictions for a test set. Part A, which I describe here, involved three target variables, and the test set came from a different time period from the training set, so that extrapolation was involved.

### The task

The training set (figure 12) consisted of hourly measurements from September 1 1989 to December 31 1989 of four input variables (temperature, humidity, solar flux and wind), and three target variables (electricity, cooling water and heating water) — 2926 data points for each target. The testing set consisted of the input variables for the next 54 days — 1282 points. The organizers requested predictions for the test set; no error bars on these predictions were requested. The performance measures for predictions were the Coefficient of Variation ('CV', a sum squared error measure normalized by the data mean), and the mean bias error ('MBE', the average residual normalized by the data mean).



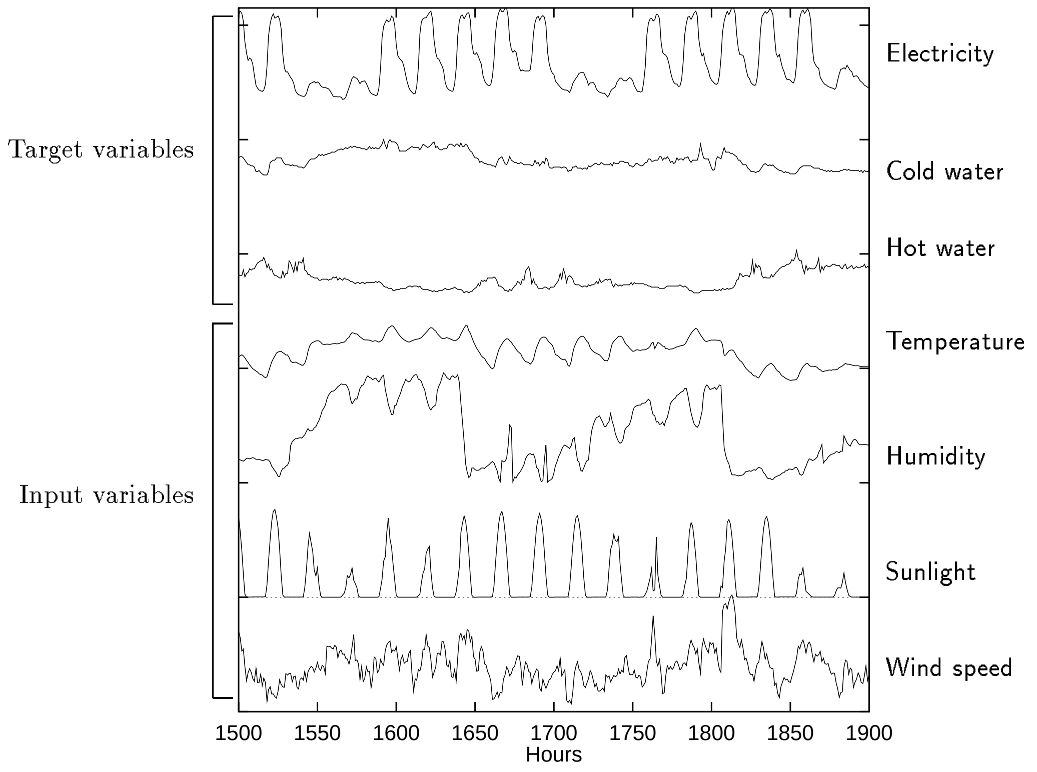


Figure 12: **The Prediction Competition Data** a) The entire training set. b) Detail.

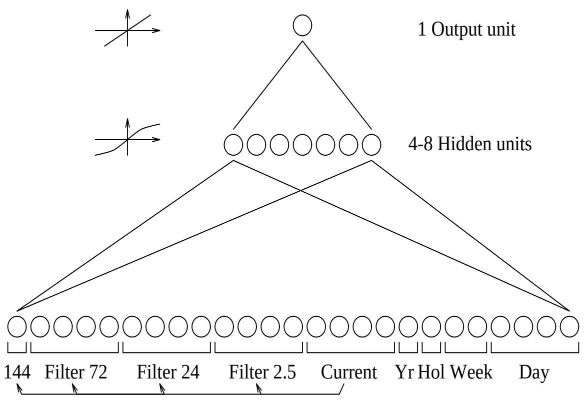


Figure 13: **A typical network used for problem A.**
The filters produced moving averages of the four environmental inputs on three time-scales: 2.5, 24 and 72 hours. The temperature variable was also given a 144 hour filter. Time was represented using the cos of the year angle, a holiday indicator, and the cos and sin of: the week angle, the day angle, and twice the day angle. All hidden and output units also had a connection to a bias unit (not shown).

## Method

I trained a large number of neural nets using the ARD model, for each of the prediction problems. The data seemed to include some substantial glitches. Because I had not yet developed an automatic Bayesian noise model that anticipates outliers (though this could be done (Box and Tiao 1973)), I omitted by hand those data points which gave large residuals relative to the first models that were trained. These omitted periods are indicated on some of the graphs in these notes. 25% of the data was selected at random as training data, the remainder being left out (a) to speed the optimizations, and (b) for use as a validation set. All the networks had a single hidden layer of tanh units, and a single linear output (figure 13). It was found that models with between 4 and 8 hidden units were appropriate for these problems.

A large number of inputs were included: different temporal preprocessings of the environmental inputs, and different representations of time and holidays. All these inputs were controlled by the ARD model. ARD proved a moderately useful guide for decisions concerning preprocessing of the data, in particular, how much time history to include. Moving averages of the environmental variables were created using filters with a variety of exponential time constants. This was thought to be a more appropriate representation than time delays, because (a) filters suppress noise in the input variables, allowing one to use fewer filtered inputs with long time constant; (b) with exponentially filtered inputs it is easy to create (what I believe to be) a natural model, giving equal status to filters having timescales 1, 2, 4, 8, 16, etc..

The on–line optimization of regularization constants was very successful. For problem A, 28 such control constants were simultaneously optimized in every model. Most models did not show 'overtraining' as the optimization proceeded, so 'early stopping' was not generally used. The numerical evaluation of the 'evidence' for the models proved problematic, so validation errors were used to rank the models for prediction. For each task, a committee of models was assembled, and their predictions were averaged together; this procedure was intended to mimic the Bayesian predictions $P(\mathbf{t}|D) = \int P(\mathbf{t}|D,\mathcal{H})P(\mathcal{H}|D)\, d\mathcal{H}$. The size of the committee was chosen so as to minimize the validation error of the mean predictions. This method of selecting committee size has also been described under the name 'stacked generalization' (Breiman 1992). In all cases, a committee was found that performed significantly better on the validation set than any individual model.

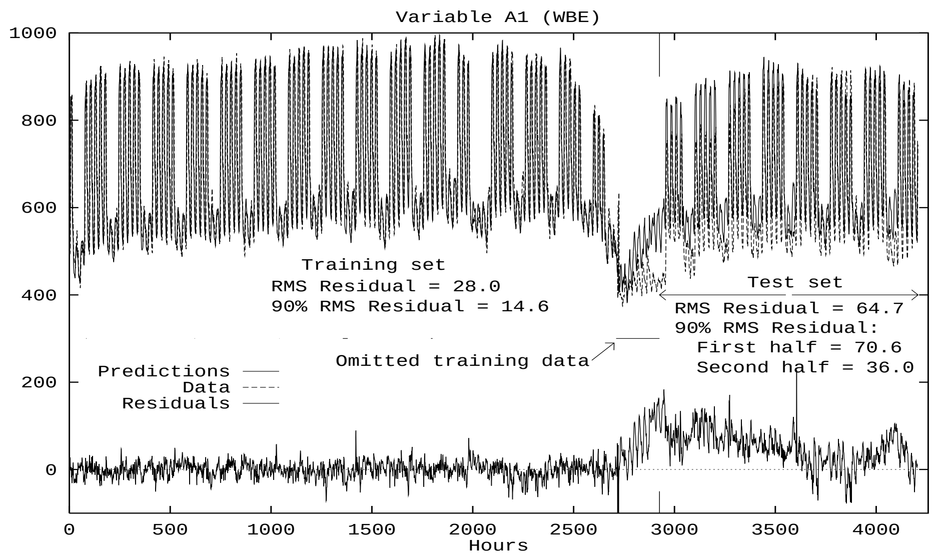


Figure 14: **Target A1 — Electricity**

## Results

The three target variables are displayed in their entirity, along with my models' final predictions and residuals, in figures 14–16.

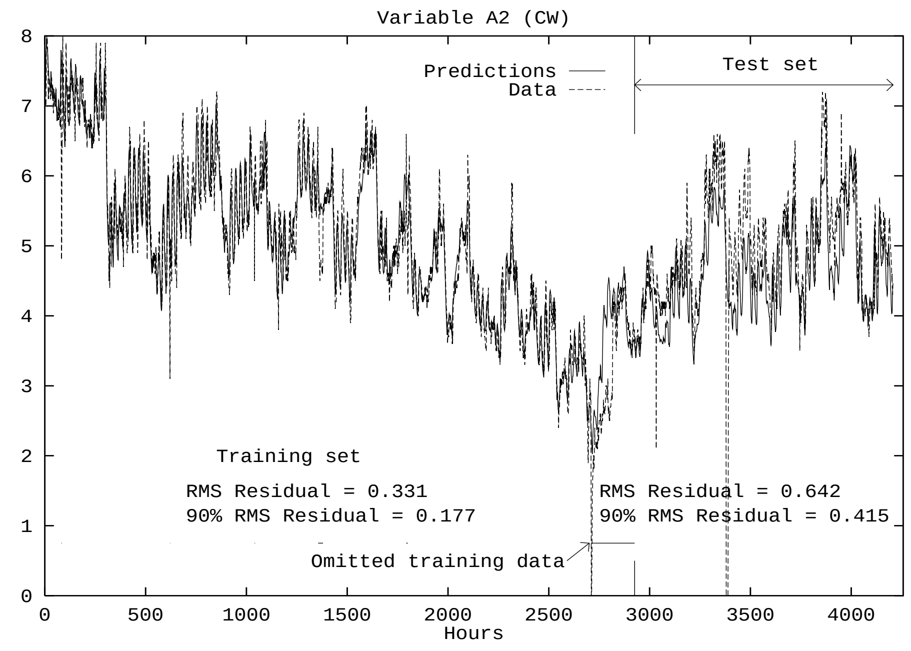
There are local trends in the testing data which the models were unable to predict. Such trends were presumably 'overfitted' in the training set. Clearly a model incorporating local correlations among residuals is called for. Such a model would not perform much better by the competition criteria, but its on-line predictive performance would be greatly enhanced.

In the competition rules, it was suggested that scatter plots of the model predictions versus temperature should be made. The scatter plot for problem A3 is particularly interesting. Target A3 showed a strong correlation with temperature in the training set (dots in figures 17b). When I examined my models' predictions for the testing set, I was surprised to find that, for target A3, a significantly offset correlation was predicted ('+'s in figure 17a). This change in correlation turned out to be correct ('+'s in figure 17b). Most other entrants' predictions for target A3 showed a large bias; presumably none of their models extracted the same structure from the data.

In the models used for problem A3, I have examined the values of the parameters $\{\alpha_c, \gamma_c\}$, which give a qualitative indication of the inferred 'relevance' of the inputs. For prediction of the hot water consumption, the time of year and the current temperature were the most relevant variables. Also highly relevant were the holiday indicator, the time of day, the current solar and wind speed, and the moving average of the temperature over the last 144 hours. The current humidity was not relevant, but the moving average of the humidity over 72 hours was. The solar was relevant on a timescale of 24 hours. None of the 2.5 hour filtered inputs seemed especially relevant.

## How much did ARD help?

An indication of the utility of the ARD prior was obtained by taking the *final* weights of the networks in the optimal committees as a starting point, and training them further using the standard model's regularizer (*i.e.*, just three regularization constants). The dotted lines in figure 18 show the validation error of these networks before and after adaptation. As a control, the



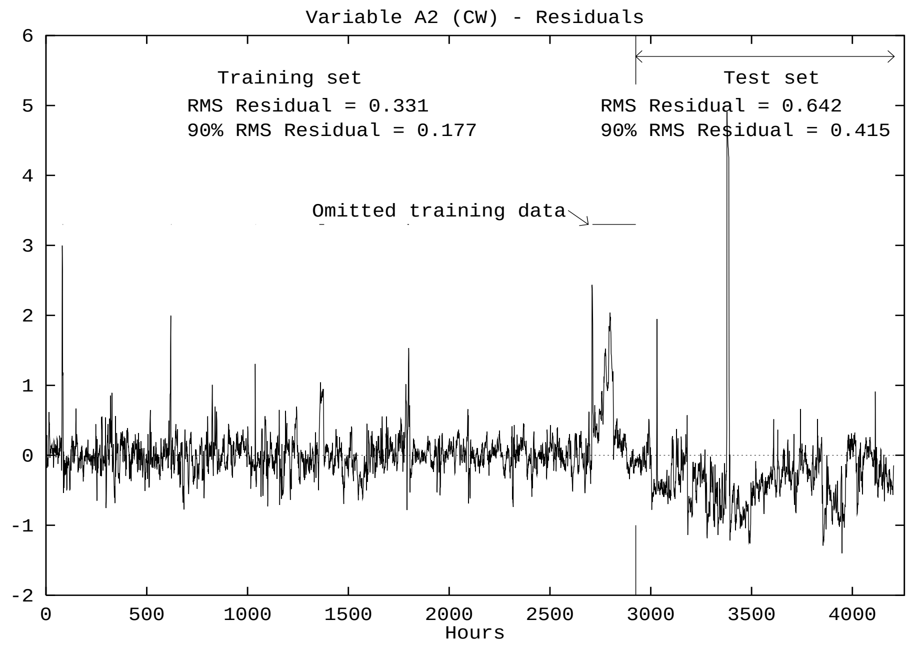
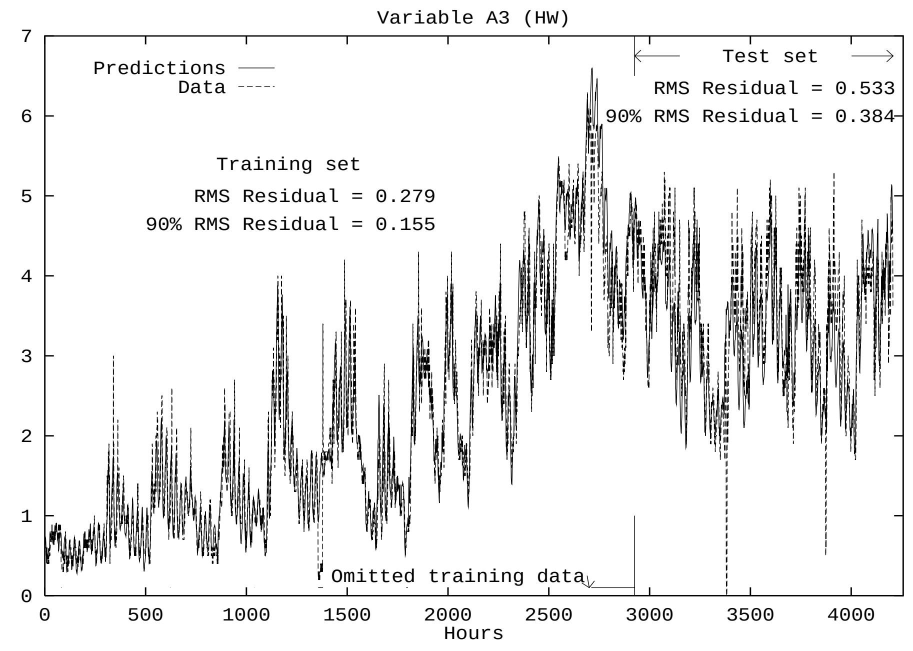


Figure 15: **Target A2 — Cooling water**



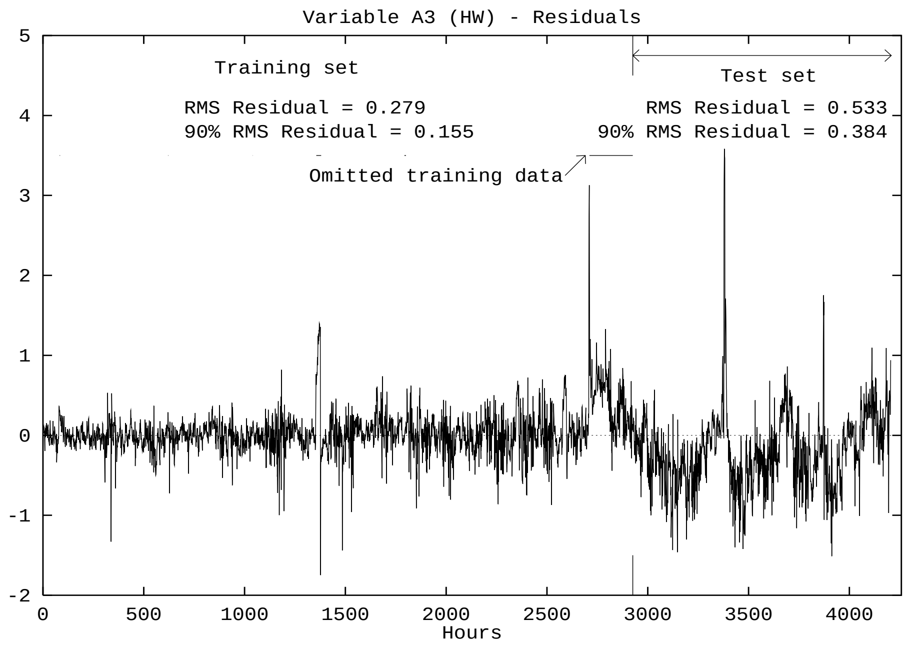


Figure 16: **Target A3 — Heating water**

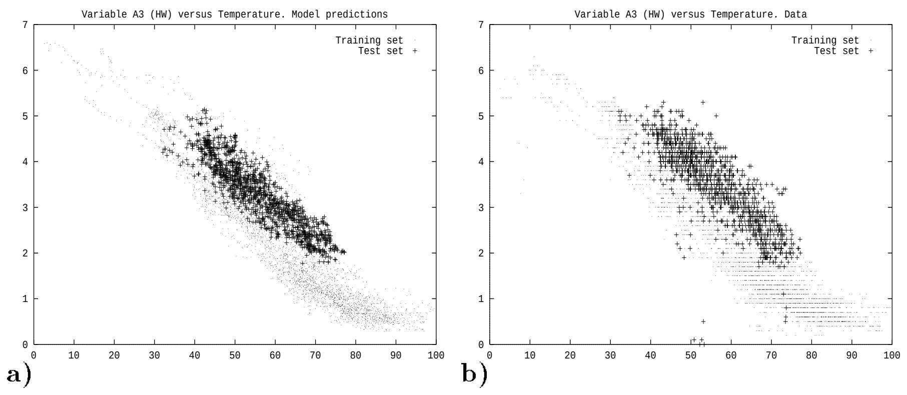


Figure 17: **Predictions for target A3 (HW) versus temperature**
**a)** Model predictions. This graph shows that my model predicted a substantially different correlation between target A3 and temperature (+) from that shown in the training set (·). **b)** Data. This predicted offset was correct.

solid lines show what happened to the validation error when the same networks were used as a starting point for continued optimization under the ARD model. The validation error is a noisy performance measure, but the trend is clear: the standard models suffer between 5% and 30% increase in error because of overfitting by the parameters of the less relevant inputs; the ARD models, on the other hand, do not overfit with continued training. The validation errors for the ARD model in some cases change with continued training, because my restarting procedure set the $\alpha_c$ to default values, which displaced the model parameters into a new optimum.

On the competition test data, the performance difference between these two sets of models is not so pronounced, because the residuals are dominated by other effects. To assess properly how much ARD contributed to the success of these predictions, a comparison should be made with independently derived models.

After the competition, it was revealed that the building in this study was a large university engineering center in Texas. Some of the glitches in the data were caused by the bursting of cold water pipes during a frost — a rare event apparently not anticipated by Texan architects!

The holiday period for staff ended on January 1st, but the student population did not return to the building for a couple of weeks. This may account for the significant bias error in the predictions of electricity usage (figure 14). Another factor which changed between the training period and the test period is that the Computer Science department moved to another building. This too will have caused a reduction in electricity usage. The reduction in electricity consumption may also account for some fraction of the biases in the cold and/or hot water supplies: one might expect less cooling water to be used, or more heating water, to make up the missing energy. The observed average electrical power deficit (according to my model) of 50kW corresponds to an expected decrease in CW or increase in HW consumption of 0.17 $\times 10^6$Btu (assuming that the CW and HW figures measure the actual energy delivered to the building). This is a small fraction of the overall shift in correlation between HW and temperature shown in figure 17b. In fact, relative to my models, both CW and HW showed an increase of this order.

## Discussion

The ARD prior was a success because it made it possible to include a large number of inputs without fear of overfitting. Further work could be well spent on improving the noise model, which assumes the residuals are Gaussian and uncorrelated from frame to frame. A better predictive model for the residuals shown in figures 14–16 might represent the data as the sum of the neural net prediction and an unpredictable, but auto-correlated, additional disturbance.

| **Problem A1** | RMS | Mean | **CV** | MBE | $\mathrm{RMS}_{90\%}$ | $\mathrm{Mean}_{90\%}$ | **RCV** |
|---|---|---|---|---|---|---|---|
| ARD | 64.7 | 50.3 | **10.3** | 8.1 | 54.1 | 42.2 | **11.1** |
| ARD off | 71.2 | 56.2 | **11.4** | 9.0 | 59.3 | 47.3 | **12.2** |
| Entrant 6 | | | **11.8** | 10.5 | | | |
| Median | | | **16.9** | -10.4 | | | |
| **Problem A2** | RMS | Mean | **CV** | MBE | $\mathrm{RMS}_{90\%}$ | $\mathrm{Mean}_{90\%}$ | **RCV** |
| ARD | .642 | -.314 | **13.0** | -6.4 | .415 | -.296 | **11.2** |
| ARD off | .668 | -.367 | **13.5** | -7.4 | .451 | -.349 | **12.2** |
| Entrant 6 | | | **13.0** | -5.9 | | | |
| Median | | | **14.8** | -7.6 | | | |
| **Problem A3** | RMS | Mean | **CV** | MBE | $\mathrm{RMS}_{90\%}$ | $\mathrm{Mean}_{90\%}$ | **RCV** |
| ARD | .532 | -.204 | **15.2** | -5.8 | .384 | -.167 | **9.15** |
| ARD off | .495 | -.121 | **14.2** | -3.5 | .339 | -.094 | **8.08** |
| Entrant 6 | | | **30.6** | -27.3 | | | |
| Median | | | **31.0** | -27.0 | | | |

**Key:**

My models:

| | |
|---|---|
| ARD | The predictions entered in the competition using the ARD model. |
| ARD off | Predictions obtained using derived models with the standard regularizer. |

Other entries:

| | |
|---|---|
| Entrant 6 | The entry which came 2nd by the competition's average CV score. |
| Median | Median (by magnitude) of scores of all entries in competition. |

Raw Performance measures:

| | |
|---|---|
| RMS | Root mean square residual. |
| Mean | Mean residual. |
| **CV** | Coefficient of variation (percentage). The competition performance measure. |
| MBE | Mean Bias Error (percentage). |

Robust Performance measures:

| | |
|---|---|
| $\mathrm{RMS}_{90\%}$ | Root mean square of the smallest 90% of the residuals. |
| $\mathrm{Mean}_{90\%}$ | Mean of those residuals. |
| **RCV** | $\mathrm{RMS}_{90\%}$/( 90% data range). |

Normalizing constants:

| Problem | Mean of test data | 90% data range |
|---|---|---|
| A1 | 624.77 | 486.79 |
| A2 | 4.933 | 3.7 |
| A3 | 3.495 | 4.2 |

Table 1: **Performances of different methods on test sets**

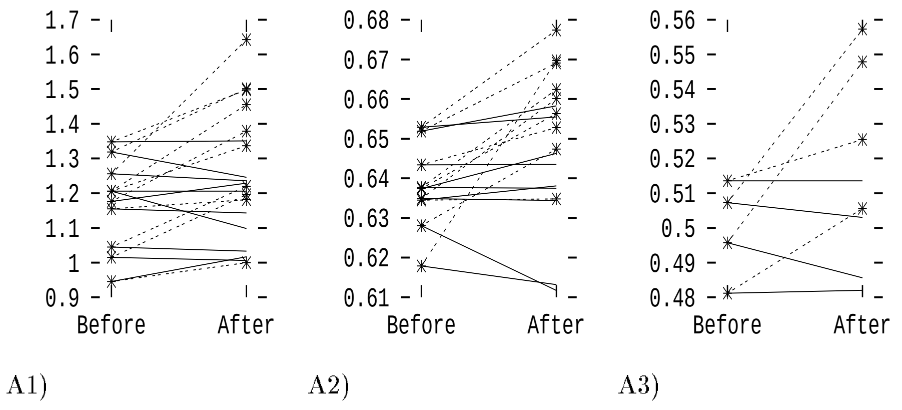


Figure 18: **Change in validation error when the ARD prior is suspended**
The solid lines without stars show the performance of ARD models. The dotted lines with stars show the models with ARD suspended. In most cases, these standard ('ARD off') models get significantly worse.

Also, a robust Bayesian noise model is needed which captures the concept of outliers.

It should be emphasised that this was not a fully automated modelling method. Human interaction was essential, and common sense was needed to check whether the assumptions were appropriate and intervene when not (see appendices A and B). Having said this, the Bayesian framework allowed for far more automation than is possible in a traditional neural network approach. Specifically, because appropriate priors (regularizers / weight decay) are used, there is no need to worry about 'early stopping'. And the problem of variable selection can be solved by using the ARD prior.

In conclusion, the winning entry in this competition was created using the following data modelling philosophy: use huge flexible models, including all possibilities that you can imagine might be appropriate; control the flexibility of these models using sophisticated priors; and use Bayes as a helmsman to guide the search in this model space.

## 9 Implementing complex regression models in BUGS

In this section, I describe the implementation of a probabilistic regression model in BUGS. BUGS is a program that carries out Bayesian inference on statistical problems using a simulation technique known as Gibbs sampling. It is possible to implement surprisingly complex regression models in this environment. I demonstrate the simultaneous inference of an interpolant and an input-dependent noise level.

### Traditional regression models and their Bayesian interpretation

In traditional regression methods, the interpolant $y(x)$ is represented as a parameterized function $y(x; \mathbf{w})$, and, given data $\{x_n, t_n\}_{n=1}^N$, the parameters $\mathbf{w}$ are optimized to minimize the weighted sum of an error function $E_D(\mathbf{w}) = \sum_{n=1}^N (t_n - y_n(x_n; \mathbf{w}))^2/2$, and a regularizer $E_W(\mathbf{w})$:

$$M(\mathbf{w}) = \beta \sum_{n=1}^{N} (t_n - y_n(x_n; \mathbf{w}))^2/2 + \alpha E_W(\mathbf{w}) \tag{43}$$

The coefficients $\alpha$ and $\beta$ are known as hyperparameters, and the regularizer $E_W$ is a function which is smallest for parameter vectors $\mathbf{w}$ that correspond to smooth functions.

In the Bayesian interpretation (reviewed in (MacKay 1992a)), this optimization maps onto a probabilistic model $\mathcal{H}$ where the function $-\beta E_D$ is the log of a likelihood function, $P(\{t_n\}|\mathbf{w}, \beta, \mathcal{H}) =$

$\exp(-\beta E_D)/Z_D(\beta)$, and $-\alpha E_W$ is the log of a prior distribution on the parameters, $P(\mathbf{w}|\alpha,\mathcal{H}) = \exp(-\alpha E_W)/Z_W(\alpha)$, so that the minimization of $M(\mathbf{w})$ corresponds to the maximization of the posterior probability:

$$P(\mathbf{w}|\{t_n\},\alpha,\beta,\mathcal{H}) = \frac{P(\{t_n\}|\mathbf{w},\beta,\mathcal{H})P(\mathbf{w}|\alpha,\mathcal{H})}{P(\{t_n\}|\alpha,\beta,\mathcal{H})}. \tag{44}$$

The hyperparameter $\beta$ defines a noise variance $\sigma_\nu^2 = 1/\beta$, and the model $\mathcal{H}$ assumes that the residuals $(t-y)$ between the data and the unknown function are independent and identically distributed Gaussian variables. If $E_W$ is a quadratic function of $\mathbf{w}$ then the hyperparameter $\alpha$ defines a variance for the parameters $w$. An important problem in regression modelling is to control the relative strength of these two hyperparameters. If $\alpha/\beta$ is too large then the model cannot fit the data well, but if the ratio is too small then overfitting occurs. As reviewed in (MacKay 1992a), we can extend this probabilistic model by including a prior distribution over the hyperparameters $P(\alpha,\beta|\mathcal{H})$ that expresses our lack of knowledge about the noise variance and the variance of the parameters $\mathbf{w}$, and then we can use Bayesian methods to infer the most plausible values for the hyperparameters from the data.

This Bayesian approach to regression modelling has the following advantages. (1) The hyperparameters $\alpha, \beta$ are controlled using the same data as are used to optimize the parameters $\mathbf{w}$; there is no need for cross-validation. This is true even for models with multiple hyperparameters $\{\alpha\}$ (MacKay 1994, MacKay and Takeuchi 1995). (2) One obtains quantified error bars on model parameters and predictions. (3) One obtains a measure of the effective number of well-determined parameters in a model. (4) Objective model comparison is possible, comparing regression models with different basis functions, for example, or different regularizers (MacKay 1992a). (5) Generalizations to better probabilistic models are easy to formulate.

### Modelling an input-dependent noise level

One improvement that we might wish to make is a modification of the noise model, which, in the above model, assumes that all the residuals are independent and Gaussian with identical variance. We might believe that in fact the noise level is a parameterized function $\beta(x;\mathbf{b})$ of the input variable and we might wish to infer it from the data. This inference is not straightforward; a maximum likelihood approach, in which $\mathbf{w}$ and $\mathbf{b}$ are simultaneously varied to maximize the likelihood, leads to a biased estimate of $\beta(x)$—because the optimized function $y(x;\hat{\mathbf{w}})$ unavoidably fits some of the noise so that the noise variance is systematically underestimated. The maximum likelihood solution may even have singularities with the local value of $\beta(x)$ going to infinity in regions where the data points are interpolated perfectly. In (MacKay 1991, Chapter 6) an approximate Bayesian approach to the inference of an input-dependent noise level is described which solves this problem using Gaussian approximations. Here an alternative approach is described using a program that implements Bayesian inference using Gibbs sampling.

## Markov Chain Monte Carlo

In Bayesian inference, the two key tasks are to write down a probabilistic model for the domain, and to implement inferences given the observed data. Thus in the model with an input-dependent noise level discussed above, the probability of everything is:

$$\begin{aligned} P(\{t_n\},\mathbf{w},\mathbf{b},\alpha_w,\alpha_b|\mathcal{H}) &= \left[\prod_n P\left(t_n|y(x_n;\mathbf{w}),\beta(x_n;\mathbf{b}),\mathcal{H}\right)\right] \times \\ &\quad P(\mathbf{w}|\alpha_w,\mathcal{H})P(\mathbf{b}|\alpha_b,\mathcal{H})P(\alpha_w,\alpha_b|\mathcal{H}). \end{aligned}$$

Here an extra hyperparameter $\alpha_b$ has been introduced that defines the expected smoothness of the function $\beta(x;\mathbf{b})$. The input variables $\{x_n\}$ are not being modelled, *i.e.*, they are assumed given under $\mathcal{H}$.

We are interested in the inference, given the data $\{t_n\}_{n=1}^N$, of the functions $y(x)$ and $\beta(x)$, and the prediction of new target variables $t_{N+1}$ at locations $x_{N+1}$. These inferences are jointly described by a single equation:

$$P(\mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b, t_{N+1} | \{t_n\}_{n=1}^N, \mathcal{H}) = \frac{P(\{t_n\}, \mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b | \mathcal{H})}{P(\{t_n\}_{n=1}^N | \mathcal{H})}. \tag{45}$$

The task is to make a computational implementation of this complicated posterior distribution. There are three principal approaches to implementing Bayesian inference for this sort of problem: approximations based on Gaussians, as for example in (MacKay 1992c); an 'ensemble learning' approach, in which an approximating distribution is optimized by variational free energy minimization, introduced by Hinton and van Camp (1993); and Markov chain Monte Carlo techniques, as reviewed and developed by Neal (1993b).

In a Markov chain Monte Carlo (MCMC) approach, the posterior distribution (45) is not represented directly; rather, a procedure is used iteratively to take a state vector $\theta = (\mathbf{w}, \mathbf{b}, \alpha_w, \alpha_b, t_{N+1})$ and generate a new random state vector $\theta'$ from a probability distribution $q(\theta'; \theta)$. Then we obtain the next state $\theta''$ by sampling from the distribution $q(\theta''; \theta')$, and so forth. The transition probability $q$ is constructed in such a way that an ergodic Markov process is defined with stationary distribution equal to the desired posterior distribution (45). Thus assymptotically the MCMC sampling procedure produces a sequence of states $\theta$ each of which is a sample from the posterior distribution (though consecutive state vectors are not in general *independent* samples from that distribution). Then properties of interest, *e.g.*, moments of predictive distributions, can be estimated from large numbers of samples $\{\theta\}$.

There are two principal ways of constructing transition probabilities $q$ that converge to desired distributions $p(\theta)$.

**Metropolis methods.** The Metropolis algorithm makes use of a a *proposal density* $g(\theta'|\theta)$, which in the simplest case might be a simple random distribution centred on the current $\theta$. A tentative new state $\theta'$ is generated from this distribution, and is accepted with probability:

$$P(\text{accept}) = \min\left(1, \frac{p(\theta|x)}{g(\theta|\theta')} \frac{g(\theta'|\theta)}{p(\theta'|x)}\right)$$

If the step is rejected, then we set $\theta' = \theta$. To compute the acceptance probability we need to be able to compute the probability ratios $p(\theta|x)/p(\theta'|x)$ and $g(\theta|\theta')/g(\theta'|\theta)$. Simple Metropolis algorithms perform poorly in high dimensional problems because they explore the space by a slow random walk. More sophisticated Metropolis algorithms such as hybrid Monte Carlo (see Neal (1993b)) make use of proposal densities that give faster movement through the state space.

**Gibbs sampling.** In Gibbs sampling, each iteration $\theta \to \theta'$ involves a separate sampling of each variable in turn from its distribution *conditional* on the current values of all the other variables in the model. For many models (though not for general neural networks) these conditional distributions are straightforward to sample from. Conditional distributions that are not of standard form may still be sampled from by 'rejection sampling' if the conditional distribution satisfies certain convexity properties.

Gibbs sampling suffers from the problem of simple Metropolis algorithms that the state space is explored by random walk. However it is a relatively parameter-free method and so is attractive as an implementation strategy.

## BUGS

A new tool, BUGS, makes simple the implementation of complex Bayesian models by Gibbs sampling. BUGS (Thomas, Spiegelhalter and Gilks 1992) is copyright by the MRC Biostatistics Unit, Robinson Way, Cambridge CB2 2SR, and is available by ftp from `ftp.mrc-bsu.cam.ac.uk`.

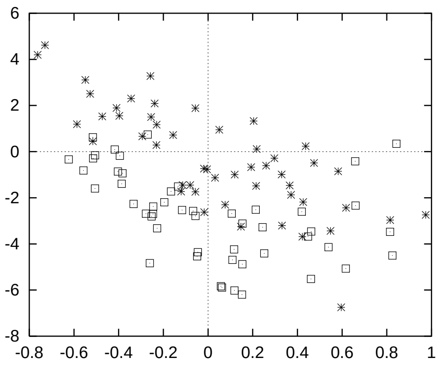


Figure 19: Toy data from two populations differing by an offset `dy`.

In BUGS, a statistical model is expressed using the BUGS language; a compiler processes the model and the available data; and a sampler generates appropriate values of the unknown quantities. BUGS is intended for the analysis of complex models in which there may be many unknown quantities but for which substantial conditional independence assumptions are appropriate.

## Inferring an interpolant and an input-dependent noise level with BUGS

A slight generalization of the model given above is made in this work. Data in the form of $(x, t)$ pairs are assumed to be obtained from two populations (figure 19). Both populations depend on the same function $y(x)$ and have the same noise level $\beta(x)$, but there is an offset `dy` added to all the points in one population. We have fifty labelled points from each population. I make the assumption that we are particularly interested in inferring the offset between the populations. This task is motivated by the problem of inferring the difference between the distances to two populations of Cepheid stars (Freedman, Madore, Mould, Hill et al. 1994).

The underlying function is described by a linear combination of basis functions $y(x; \mathbf{w}) = \sum_h w_{h=1}^{K_W} \phi_h(x)$. The basis functions $\phi$ are Legendre polynomials. In neural network terms, this is a two-layer network with a fixed non-linear hidden layer and adaptive linear output connections. Similarly, the varying noise level is written as $\beta(x; \mathbf{b}) = \exp\left(\sum_{h=1}^{K_B} b_h \phi_h(x)\right)$. The parameters of the model are $\mathbf{w} = \{w_h\}_1^{K_W}$ and $\mathbf{b} = \{b_h\}_1^{K_B}$. Gaussian priors on these parameters are specified with hyperparameters $\alpha_w$ and $\alpha_b$ that have broad gamma priors. For convenience I introduce $K = \max(K_W, K_B)$. We obtain the special case of uniform noise by setting $K_B = 1$ and the special case of a straight line relationship by setting $K_W = 2$.

### Defining the model in the BUGS language

BUGS uses a language similar to that of **`S-plus`**, but access to the latter is not required. Expressing the model in the BUGS language is almost as simple as writing the model on paper. The symbol `<-` defines a deterministic relationship, giving the quantity on the left as a function of the quantity on the right. The symbol `~` specifies a conditional distribution. This model uses normal distributions **`dnorm`** and gamma distributions **`dgamma`**. The notation **`dnorm(m,beta)`** defines a normal distribution of mean $m$ and variance $1/\beta$. The function **`inprod(w[],p[])`** denotes the inner product of the vectors `w` and **`p`**.

Here is the file **`astro10.bug`**, which specifies a model in which the interpolant and the noise level are both described by Legendre polynomials of degree 10. The data from the two populations are stored in vectors **`x`** and **`t`** such that population 'a' gave rise to $\{(x_n, t_n)\}_{n=1}^{N_a}$, and population 'b' gave rise to $\{(x_n, t_n)\}_{n=N_a+1}^{N}$.

```
model astro10;

const             # ------- constants ------------------
  N=100,          # Number of data
  Na=50,          # Number of data in first population
  KW=10,          # Number of basis functions for y(x)
  KB=10,          # Number of basis functions for beta(x)
  K=10,           # K=MAX(KW,KB)
  x0=0.0,dx=1.0;  # characteristic range of the x-axis

var                                     # -- variables --
      x[N], dy , y[N], t[N], w[K], b[K],
      phi[K,N],
      alphab, alphaw, beta[N];

data  in "astro.dat" ;
inits in "astro1.in"  ;

{
  dy ~ dnorm( 0.0, 0.0001 ) ;           # noninformative prior for
                                        #    offset between populations

  for (m in 1:N) {                      # recurrence relation to
      phi[1,m] <- 1.0 ;                 #    define Legendre polynomials
      phi[2,m] <- (x[m]-x0)/dx ;
      for (h in 3:K) {
            phi[h,m] <- ( ( 2 * h - 3 ) * (x[m]-x0)/dx * phi[h-1,m]
                          - ( h - 2 ) * phi[h-2,m] ) / ( h - 1 ) ;
      }
  }
  for (h in 1:KW) {
      w[h] ~ dnorm( 0.0 , alphaw ) ;    #  Gaussian prior for w
  }
  for (h in 1:KB) {
      b[h] ~ dnorm( 0.0 , alphab ) ;    #  Gaussian prior for b
  }
  for (h in KW+1:K) {
      w[h] <- 0.0 ;                     #  Excess parameters
  }
  for (h in KB+1:K) {
      b[h] <- 0.0 ;                     #  Excess parameters
  }

  for (m in 1:N) {
      t[m]     ~  dnorm( y[m] , beta[m] ) ;        #  Gaussian residuals
      beta[m]  <- exp( inprod( b[] , phi[,m] ) ) ; #  Noise level
  }
  for (m in 1:Na) {
      y[m]     <- inprod( w[] , phi[,m] ) ;        #  Interpolant
  }
  for (m in Na+1:N) {
      y[m]     <- dy + inprod( w[] , phi[,m] ) ;   #  Other interpolant
  }
  alphab  ~ dgamma( 1.0E-3 , 1.0E-3 ) ; # noninformative priors
  alphaw  ~ dgamma( 1.0E-3 , 1.0E-3 ) ; #    for hyperparameters
}
```

And here is the file `astro1.in`, which specifies the initial conditions for the sampler:

```
list ( alphab=1.0 , alphaw=1.0 , dy=0 )
```

The data file `astro.dat` contains an ASCII list of the values $\{x_n, t_n\}$. The toy data were generated using true values $K_B = K_W = 3$ and `dy` $= 3.0$. The study here examines the inference of the offset `dy` when interpolation models of various orders, with and without the input-dependent noise level, are used.

For each model a burn–in period of 500 iterations was followed by 1000 iterations during which statistics on `dy` were recorded.

### Results

I present, for each of the four models $(K_W, K_B) = (2,1), (2,10), (10,1), (10,10)$, the inferred value of `dy`, in terms of its posterior mean and standard deviation (as estimated by BUGS), and a 95% confidence interval.

| `dy` $K_B$ | 1 | | | 10 | | |
|---|---|---|---|---|---|---|
| $K_W$ | mean | sd | 2.5%:97.5% | mean | sd | 2.5%:97.5% |
| 2 | 2.44 | 0.30 | 1.86 : 3.02 | 2.76 | 0.26 | 2.25 : 3.24 |
| 10 | 2.54 | 0.31 | 1.98 : 3.16 | 2.71 | 0.27 | 2.18 : 3.23 |

The over–simple model $K_B = 1, K_W = 2$, which assumes a linear interpolant and constant noise level, gives a 95% confidence interval for `dy` that only just includes the true value. As is often the case with over-simple models, this model is not only wrong, it gives over-confident predictions too. Changing from the over–simple model to the model with more parameters in the interpolant (moving down from the top left corner) produces a slight increase in the uncertainty of `dy`. The increase is only slight because there are two opposing effects: first, for any particular value of noise level $1/\beta$, the more flexible interpolant is less well determined and the uncertainty in `dy` increases; but second, the greater flexibility of the interpolant allows it to fit the curving shape of the data and makes smaller noise levels $1/\beta$ probable. Small noise levels give more accurate inferences. A similar effect occurs as we increase the number of terms in the representation of $\beta(x)$ (going from left to right). The estimation of `dy` can become more precise, in intuitive terms, because the model is able to discover that some values of $x$ give more reliable measurements than others, so that the inference of `dy` can be based on them, ignoring the more noisy measurements. The net effect is that when we change from the over–simple model to the most flexible model (bottom right), the confidence interval becomes smaller and more accurate.

## Conclusion

Complex regression models that would take considerable effort to implement by Bayesian methods based on Gaussian approximations can be simulated in BUGS with great ease. BUGS implicitly infers hyperparameters from the data and automatically marginalizes over all the parameters to give the desired predictions. The Gibbs sampling method is not the most efficient of Markov chain Monte Carlo methods, but there may be problems of interest where the convenience of this tool outweighs this drawback. Not all probabilistic models can be implemented in BUGS; the conditional distributions that are sampled from must be log concave. But this example has demonstrated that interesting new models can be implemented.

# Appendices

## A Summary of neural network modelling suggestions

1. When data modelling is in the research stages, *always* divide the available data into three sets—a training set, a validation set, and a testing set—unless data are really too scarce.

   The training set is used to optimize the parameters of each model. The validation set is used to compare alternative models and to compare alternative modelling strategies. It is a surrogate for the testing set which is set aside and reserved as the final once-only 'real world test'.

   In a non-Bayesian approach, the validation set may be used to set hyperparameters such as weight decay rates. In the Bayesian approach, the training set alone can be used both to infer parameters and hyperparameters.

   The training, validation and test sets should be designed in a way that reflects the real world problem to be solved. This might in some cases mean randomly selecting inputs to put them in each set; or in other scenarios one might wish to systematically arrange that the training set, say, contains a uniform sampling of the input space. Or one might know that in the real world one would have to extrapolate, in which case it would be appropriate to construct a training set having a different input distribution from the other sets.

   Given scarce data it might be worth systematically repeating an experiment, say, ten times, with ten different divisions of the data into training and validation sets.

2. Eyeball the data to check for glitches and outliers.

3. Choose the measure of validation and test performance carefully. It is popular to use the test error (sum squared error) as the default performance measure, but in fact this may be a misleading criterion (MacKay and Oldfield 1995). In many applications there will be an opportunity not to simply make a scalar prediction, but rather to make a prediction with error bars, or maybe an even more complicated predictive distribution. It is then reasonable to compare models in terms of their predictive performance as measured by the log predictive probability of the test data or validation data. Under the log predictive error, as contrasted with the test error, the penalty for making a wild prediction is much less if that wild prediction is accompanied by appropriately large error bars. The log predictive error (LPE), assuming that for each example $m$ the model gives a prediction $\text{Normal}(y^{(m)}, {\sigma_y^{(m)}}^2)$ is:

$$\begin{aligned} \text{LPE} &= \sum_m -\log P(t^{(m)}|D,\mathcal{H}) \\ &= \sum_m \left[\frac{1}{2}\left(t^{(m)} - y^{(m)}\right)^2 / {\sigma_y^{(m)}}^2 + \log(\sqrt{2\pi}\sigma_y^{(m)})\right] \end{aligned}$$

   Also, going back to scalar predictions: consider using more robust error measures than the mean-square error. After all, is it usually the case that if an error of size 2.0 costs 3 pounds more than an error of size 1.0, then an error of size 3.0 costs 5 pounds more than the size 2.0 error, and an error of size 4.0 costs 7 pounds more than the size 3.0 error? Often a more robust measure such as the mean absolute deviation, or the mean squared error of the smallest 90% of the residuals, may be a more realistic loss function.

4. Optimize the model using a high quality optimizer. Don't use a plain steepest descent optimizer. Use conjugate gradients or other methods that keep track of higher order information about the optimized function (see Press, Flannery, Teukolsky and Vetterling (1988) and Jervis and Fitzgerald (1993), but don't use the code from Press et al. (1988), because it is suboptimal in many ways).

5. Use regularizers, also known as priors, also known as weight decay. If possible, control the regularization constants either using your prior knowledge of the problem, or using Bayesian hyperparameter optimization methods. If you have other ways of controlling the hyperparameters, try those too; and compare them with the Bayesian approach on appropriate validation or test data.

6. Run the optimizations multiple times using different seeds, different initial conditions, different optimization schedules, and examine the validation scores to check that you have got the optimizers running with the knobs set at their best values. Bear in mind that too much regularization early on in the learning may prevent a network from learning interesting structure. A possible rule of thumb is to hold the regularization at a weak value for a considerable fraction of the optimization, and only then allow the hyperparameter control to kick in.

7. During optimization of a model's parameters, keep an eye on the validation performance. If the performance reaches an optimum and then becomes significantly worse, then something is probably wrong with the model. A traditional response to the symptom of 'over-training' is 'early stopping'—stopping the training at the point where the validation performance is best. This seems to me to be a terrible response to a clear model sickness. When a child learns, it is not thought important to whisk them away from the teacher before they 'over-learn'! Over-learning is often an indication that a probabilistic assumption in the noise model or the regularizer is inappropriate.

   The following diagnostic procedures may help track down the problem, and should be followed even if there is not an overtraining problem.

   (a) Look at the residuals, *i.e.*, the differences between the target and the model's output. Are there some huge outliers? You may now be able to detect glitches in the data that were not obvious at the start. (If so, and if they are caused by severely non-Gaussian noise, then maybe they should be left out of the training data.)

   (b) Look at the weights. Are there some parameters that are far bigger than others? Are there some parameters that are all going to zero? Are some hidden units in the network only exploring their linear regime?

   (c) Look at the weights in each regularization class. Do they all have similar magnitudes to each other, or are some weights much bigger than others? If the latter, then this may be a hint that they should not be in the same class as each other.

8. Investigate alternative representations of the problem, *e.g.*, alternative input and output representations. If you think you have a good idea for a preprocessing of the inputs, try it out; but also try feeding a net the raw unprocessed inputs, so it can figure out its own preprocessing (maybe giving such a net an extra hidden layer). Consider constructing a net such that the human solution is a special case of the mappings the net can perform, and initialize the net to that special case. Then when the net is optimized, things can only get better. The worst possible outcome is that you end up with a net that is as good as the preexisting human solution!

9. When making predictions:

   (a) use parameter error bars to compute predictive error bars.

   (b) don't just use one best model. Use multiple good models and combine their predictions. This is called 'forming a committee'. The size of the committee can be controlled for example by looking at the validation error of the committee's predictions, by whatever error function is to be used at the end of the day. Typically there will be an optimum committee of size $L > 1$.

   The following formula (derived by computing the mean and variance of a mixture of Gaussian distributions) gives the error bars for the predictions of a committee consisting of $L$ equally weighted members whose predictions are $y^{(l)} \pm \sigma_y^{(l)}$:

$$\begin{aligned}\text{Committee prediction } \bar{y} &= \frac{1}{L}\sum_k y^{(l)} \\ \text{Variance } \sigma^2 &= \frac{1}{L}\sum_l {\sigma_y^{(l)}}^2 + \frac{1}{L}\sum_l (y^{(l)} - \bar{y})^2.\end{aligned}$$

Thus the variance of the committee's predictive distribution is the variance of the means plus the mean of the variances.

10. Try to ensure that prior knowledge of the problem is hard wired into the structure of the network.

    If the function is expected to have a certain continuity property as a function of the input variables, then use an input representation that respects this continuity. For example, if an input variable is an angle, do not represent it by a number between -180 and 180; rather, since 180 and -180 are in fact the same as each other, use a representation that makes this explicit. For example, represent the angle as its cosine and sine.

    If an output variable is known to be between 0 and 1, use an output function $f$ that respects these bounds.

11. Use continuous model spaces controlled by hyperparameters, rather than discrete model spaces.

## B Summary of assumptions

It is important to be aware of the assumptions implicit in the above modelling procedure. If any of these assumptions are expected to give trouble then (with additional effort) new probabilistic models can be designed which implement more appropriate or more general assumptions. The above approach implicitly assumes:

1. That the target is a noisy version of the true output.
2. That the noise is Gaussian with a variance that is the same for all examples.
3. That the underlying mapping from input to output is the same for all time.
4. That the input variables are noise-free.[2]
5. That the true function has the character of typical samples from the distribution illustrated in figure 10. This means in general that:
   (a) The true function is a continuous function of the inputs.
   (b) The typical values of the input variables should be centred on zero. Rescaling of input variables so that they are in the interval [-0.5,0.5], is often a good idea, though not obligatory.
6. That there are no missing inputs. If there are in fact missing inputs then caution is needed. It is not valid to set missing inputs to zero. Probabilistic models that can handle missing inputs (MacKay 1995a) are hard work to implement if they are based on neural networks. An idea worth looking into might be to represent each input by two numbers defining a confidence interval; missing inputs would have the broades interval. An alternative approach is to turn to more tractable graphical probabilistic models (Spiegelhalter and Lauritzen 1990), which can handle missing inputs with no problems, but don't have the same non-linear capabilities as neural networks. The `BUGS` program developed by Spiegelhalter *et al* is an excellent environment for implementing graphical models.

[2]This assumption could be removed if we made a more sophisticated output noise model.

## References

Berger, J. (1985). *Statistical Decision theory and Bayesian Analysis*, Springer.

Bishop, C. M. (1992). Exact calculation of the Hessian matrix for the multilayer perceptron, *Neural Computation* **4**(4): 494–501.

Box, G. E. P. and Tiao, G. C. (1973). *Bayesian inference in statistical analysis*, Addison–Wesley.

Breiman, L. (1992). Stacked regressions, *Technical Report 367*, Dept. of Stat., Univ. of Cal. Berkeley.

Bretthorst, G. (1988). *Bayesian spectrum analysis and parameter estimation*, Springer.

Bridle, J. S. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, *in* F. Fougelman-Soulie and J. Hérault (eds), *Neuro-computing: algorithms, architectures and applications*, Springer–Verlag.

Buntine, W. and Weigend, A. (1991). Bayesian back–propagation, *Complex Systems* **5**: 603–643.

Copas, J. B. (1983). Regression, prediction and shrinkage (with discussion), *J. R. Statist.Soc B* **45**(3): 311–354.

Freedman, W. L., Madore, B. F., Mould, J. R., Hill, R. et al. (1994). Distance to the Virgo cluster galaxy M100 from Hubble space telescope observations of Cepheids, *Nature* **371**: 757–762.

Gull, S. F. (1988). Bayesian inductive inference and maximum entropy, *in* G. Erickson and C. Smith (eds), *Maximum Entropy and Bayesian Methods in Science and Engineering, vol. 1: Foundations*, Kluwer, Dordrecht, pp. 53–74.

Gull, S. F. (1989). Developments in maximum entropy data analysis, *in* J. Skilling (ed.), *Maximum Entropy and Bayesian Methods, Cambridge 1988*, Kluwer, Dordrecht, pp. 53–71.

Hanson, R., Stutz, J. and Cheeseman, P. (1991). Bayesian classification with correlation and inheritance, *Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia.*

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights, In: *Proceedings of COLT-93.*

Jaynes, E. T. (1983). Bayesian intervals versus confidence intervals, *in* R. D. Rosencrantz (ed.), *E.T. Jaynes. Papers on Probability, Statistics and Statistical Physics*, Kluwer, p. 151.

Jeffreys, H. (1939). *Theory of Probability*, Oxford Univ. Press.

Jervis, T. T. and Fitzgerald, W. J. (1993). Optimization schemes for neural networks, *Technical Report CUED/F-INFENG/TR 144*, Cambridge University Engineering Department, Trumpington Street, Cambridge, England.

Loredo, T. J. (1990). From Laplace to supernova SN 1987A: Bayesian inference in astrophysics, *in* P. Fougere (ed.), *Maximum Entropy and Bayesian Methods, Dartmouth, U.S.A., 1989*, Kluwer, pp. 81–142.

MacKay, D. J. C. (1991). *Bayesian Methods for Adaptive Models*, PhD thesis, California Institute of Technology.

MacKay, D. J. C. (1992a). Bayesian interpolation, *Neural Computation* **4**(3): 415–447.

MacKay, D. J. C. (1992b). The evidence framework applied to classification networks, *Neural Computation* **4**(5): 698–714.

MacKay, D. J. C. (1992c). A practical Bayesian framework for backpropagation networks, *Neural Computation* **4**(3): 448–472.

MacKay, D. J. C. (1994). Bayesian non-linear modelling for the prediction competition, *ASHRAE Transactions, V.100, Pt.2*, ASHRAE, Atlanta Georgia, pp. 1053–1062.

MacKay, D. J. C. (1995a). Bayesian neural networks and density networks, *Nuclear Instruments and Methods in Physics Research, Section A* **354**(1): 73–80.

MacKay, D. J. C. (1995b). Hyperparameters: Optimize, or integrate out?, *in* G. Heidbreder (ed.), *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, Kluwer, Dordrecht.

MacKay, D. J. C. and Neal, R. M. (1994). Automatic relevance determination for neural networks, *Technical Report In preparation*, Cambridge University.

MacKay, D. J. C. and Oldfield, M. J. (1995). Generalization error and the number of hidden units in a multilayer perceptron, In preparation.

MacKay, D. J. C. and Takeuchi, R. (1995). Interpolation models with multiple hyperparameters, *in* J. Skilling and S. Sibisi (eds), *Maximum Entropy and Bayesian Methods, Cambridge 1994*, Kluwer, Dordrecht.

Neal, R. M. (1993a). Bayesian learning via stochastic dynamics, *in* C. L. Giles, S. J. Hanson and J. D. Cowan (eds), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, California, pp. 475–482.

Neal, R. M. (1993b). Probabilistic inference using Markov chain Monte Carlo methods, *Technical Report CRG–TR–93–1*, Dept. of Computer Science, University of Toronto.

Neal, R. M. (1994). Priors for infinite networks, *Technical Report In preparation*, Univ. of Toronto.

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

Press, W., Flannery, B., Teukolsky, S. A. and Vetterling, W. T. (1988). *Numerical Recipes in C*, Cambridge.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning representations by back–propagating errors, *Nature* **323**: 533–536.

Skilling, J. (1993). Bayesian numerical analysis, *in* W. T. Grandy, Jr. and P. Milonni (eds), *Physics and Probability*, C.U.P., Cambridge.

Spiegelhalter, D. J. and Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures, *Networks* **20**: 579–605.

Thomas, A., Spiegelhalter, D. J. and Gilks, W. R. (1992). BUGS: A program to perform Bayesian inference using Gibbs sampling, *in* J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith (eds), *Bayesian Statistics 4*, Clarendon Press, Oxford, pp. 837–842.

Wolpert, D. H. (1993). On the use of evidence in neural networks, *in* C. L. Giles, S. J. Hanson and J. D. Cowan (eds), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, California, pp. 539–546.