# Bayesian Methods for Supervised Neural Networks

David Barber

Division of Informatics, University of Edinburgh

dbarber@anc.ed.ac.uk

Phone +44 (0) 131 650 4491 Fax: +44 (0) 131 650 6899

May 15, 2002

# Introduction

An attractive feature of artificial neural networks is their ability to model highly complex non-linear relationships in data. However, choosing an appropriate neural network model for data is compounded by the difficulty of assessing the network complexity. Since we are rarely certain about either our data measurements or model beliefs, a natural framework is to use probabilities to account for these uncertainties. How can we combine our data observations with these modelling uncertainties in a consistent and meaningful manner? The Bayesian approach provides a consistent framework for formulating a response to these difficulties, and is noteworthy for its conceptual elegance(Box and Tiao 1973; Berger 1985; MacKay 1992). The fundamental probabilistic relationship required for inference is the celebrated Bayes' rule which, for general events $A$,$B$,$C$ is

$$p(A|B,C) = \frac{p(B|A,C)p(A|C)}{p(B|C)} \tag{1}$$

It is convenient to think of different levels of uncertainty in formulating a model. At the lowest level, we may assume that we have the correct model, but are uncertain as to the parameter settings $\theta$ for this model. This assumption details how observed data is generated, $p\left(\text{data}|\theta,\text{model}\right)$. The task of inference at this level is to calculate the posterior distribution of the model parameter. Using Bayes' rule, this is

$$p(\theta|\text{data},\text{model}) = \frac{p\left(\text{data}|\theta,\text{model}\right)p\left(\theta|\text{model}\right)}{p\left(\text{data}|\text{model}\right)} \tag{2}$$

Thus, if we wish to infer model parameters from data we need two assumptions: (1) How the observed data is generated under the assumed model, the *likelihood* $p\left(\text{data}|\theta,\text{model}\right)$ and (2) Beliefs about which parameter values are appropriate, before the data has been observed, the *prior* $p(\theta|\text{model})$. (The denominator in equation (2) is the normalising constant for the posterior and plays a role in uncertainty at the higher, model level). That these two assumptions are required is an inescapable consequence of Bayes' rule, and forces the Bayesian to lay bare all necessary assumptions underlying the model.

## *Coin Tossing Example*

Let $\theta$ be the probability that a coin will land up heads. An experiment yields the data, $D = \{h,h,t,h,t,h,\ldots\}$, which contains $H$ heads and $T$ tails in $H+T$ flips of the coin. What can we infer about $\theta$ from this data? Assuming that each coin is flipped independently, the likelihood of the observed data is

$$p\left(D|\theta,\text{model}\right) = \theta^H\left(1-\theta\right)^T \tag{3}$$

A standard approach in the statistical sciences is to estimate $\theta$ by maximising the likelihood, $\theta^{ML} = \arg\max_\theta p(D|\theta,\text{model})$. This approach is non-Bayesian since it does not require the specification of a prior and, consequently, theories which deal with uncertainty in ML estimators are primarily concerned with the data likelihood, and not directly posterior

parameter uncertainty (LEARNING AND GENERALIZATION: THEORETICAL BOUNDS). In the Bayesian approach, however, we need to be explicit about our prior beliefs $p(\theta|\text{model})$. These are updated by the observed data to yield the posterior distribution

$$p\left(\theta|D,\text{model}\right) \propto \theta^H \left(1-\theta\right)^T p\left(\theta|\text{model}\right) \tag{4}$$

The Bayesian approach is more flexible than maximum likelihood since it allows (indeed, *instructs*) the user to calculate the effect that the data has in modifying prior assumptions about which parameter values are appropriate. For example, if we believe that the coin is heavily biased, we may express this using the prior distribution in fig(1a). The likelihood as a function of $\theta$ is plotted in fig(1b) for data containing 13 Tails and 12 Heads. The resulting posterior fig(1c) is bi-modal, but less extreme than the prior. It is often convenient to summarise the posterior by either the maximum a posteriori (MAP) value, or the mean, $\bar{\theta} = \int \theta p(\theta|D) d\theta$. Such a summary is not strictly required by the Bayesian framework, and the best choice of how to summarise the posterior depends on other loss criteria(Berger 1985).

### *Model Comparison and Hierarchical Models*

The above showed how we can use the Bayesian framework to assess which parameters of a model are *a posteriori* appropriate, given the data at hand. We can carry out a similar procedure at a higher, model level to asses which models are more appropriate fits to the data. In general, the model posterior is given by

$$p(M|D) = \underbrace{p(D|M)}_{\text{Model Likelihood}} \underbrace{p(M)}_{\text{Model Prior}} /p(D) \tag{5}$$

If the model is parameterised by some unknown variable $\theta$, we need to integrate this out to calculate the model likelihood,

$$p(D|M) = \int p(D|\theta, M) p(\theta|M) d\theta \tag{6}$$

Comparing two competing model hypotheses $M_1$ and $M_2$ is straightforward

$$\frac{p(M_1|D)}{p(M_2|D)} = \underbrace{\frac{p(D|M_1)}{p(D|M_2)}}_{\text{Bayes Factor}} \frac{p(M_1)}{p(M_2)} \tag{7}$$

In the coin example, we can use this to compare the biased coin hypothesis (model $M_1$ with prior given in fig(1a)) with a less unbiased hypothesis formed by using a Gaussian prior $p(\theta|M_2)$ with mean 0.5 and variance $0.1^2$ (model $M_2$). This gives a Bayes factor $p(D|M_1)/p(D|M_2) \approx 0.00018$. If we have no prior preference for either model $M_1$ or $M_2$, the data more strongly favours model $M_2$, as intuition would suggest. If we desired, we could continue in this way, forming a hierarchy of models, each less constrained than the submodels it contains.

## Bayesian Regression

Neural networks are often applied to regression in which we wish to infer an unknown input-output mapping on the basis of observed data $D = \{(\mathbf{x}^\mu, t^\mu), \mu = 1, \ldots P\}$, where $(\mathbf{x}^\mu, t^\mu)$ represents an input-output pair. For example, fit a function to the data in fig(2a). Since there is the possibility that each observed output $t^\mu$ has been corrupted by noise, we would like to recover the underlying clean input-output function. We assume that each (clean) output is generated from the model $f(\mathbf{x}; \mathbf{w})$ where the parameters $\mathbf{w}$ of the function $f$ are unknown and that the observed outputs $t^\mu$ are generated by the addition of noise $\eta$ to the clean model output,

$$t = f(\mathbf{x}; \mathbf{w}) + \eta \tag{8}$$

If the noise is Gaussian distributed, $\eta \sim N(0, \sigma^2)$, the model $M$ generates an output $t$ for input $\mathbf{x}$ with probability

$$p(t|\mathbf{w}, \mathbf{x}, M) = \exp\left\{-\frac{1}{2\sigma^2}\left(t - f(\mathbf{x}; \mathbf{w})\right)^2\right\} / \sqrt{2\pi\sigma^2} \tag{9}$$

If we assume that each data input-output pair is generated identically and independently from the others, the data likelihood is

$$p(D|\mathbf{w}, M) = \prod_{\mu=1}^{P} p(t^\mu|\mathbf{w}, \mathbf{x}^\mu, M) \tag{10}$$

(Strictly speaking, we should write $p(t^1, \ldots, t^P|\mathbf{w}, \mathbf{x}^1, \ldots, \mathbf{x}^P, M)$ on the left hand side of the above equation. However, since we assume that the training inputs are fixed and non-noisy, it is convenient and conventional to write $p(D|\mathbf{w}, M)$). The posterior distribution $p(\mathbf{w}|D, M) \propto p(D|\mathbf{w}, M)p(\mathbf{w}|M)$ is

$$\log p(\mathbf{w}|D, M) = -\frac{\beta}{2}\sum_\mu \left(t^\mu - f(\mathbf{x}^\mu; \mathbf{w})\right)^2 + \log p(\mathbf{w}|M) + \frac{P}{2}\log\beta + const. \tag{11}$$

where $\beta = 1/\sigma^2$. Note the similarity between equation (11) and the sum square regularised training error used in standard approaches to training neural networks(see GENERALIZATION AND REGULARIZATION IN NONLINEAR LEARNING SYSTEMS and Bishop 1995). In the Bayesian framework, we can motivate the choice of a sum square error measure as equivalent to the assumption of additive Gaussian noise. Typically, we wish to encourage smoother functions so that the phenomenon of overfitting is avoided. One approach to solving this problem is to use a regulariser penalty term to the training error. In the Bayesian framework, we use a prior to achieve a similar effect. In principle, however, the Bayesian should make use of the full posterior distribution, and not just a single weight value. In standard neural network training, it is good practice to use committees of networks, rather than relying on the prediction of a single network(Bishop 1995). In the Bayesian framework, the posterior automatically specifies a committee (indeed, a distribution) of networks, and the importance attached to each committee members prediction is simply the posterior probability of that network weight.

## RBFs and Generalised Linear Models

Generalised linear models have the form

$$f(\mathbf{x};\mathbf{w}) = \sum_i w_i \phi_i(\mathbf{x}) \equiv \mathbf{w}^T \Phi(\mathbf{x}) \tag{12}$$

Such models have a linear parameter dependence, but nevertheless represent a non-linear input-output mapping if the basis functions $\phi(\mathbf{x}), i = 1, \ldots, k$ are non-linear. Radial basis functions (RADIAL BASIS FUNCTION NETWORKS) are an example of such a network(Bishop 1995). A popular choice is to use Gaussian basis functions $\phi_i(\mathbf{x}) = \exp(-\left(\mathbf{x} - \boldsymbol{\mu}^i\right)^2/(2\lambda^2))$. In this discussion, we will assume that the centres $\boldsymbol{\mu}^i$ are fixed, but that the width of the basis functions $\lambda$ is a hyperparameter that can be adapted. Since the output is linearly dependent on $\mathbf{w}$, we can discourage extreme output values by penalising large weight values. A sensible weight prior is thus

$$\log p(\mathbf{w}|\alpha) = -\frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \frac{k}{2}\log\alpha + const. \tag{13}$$

Under the Gaussian noise assumption, the posterior distribution is

$$\log p(\mathbf{w}|\Gamma, D) = -\frac{\beta}{2}\sum_{\mu=1}^{P}(t^\mu - \mathbf{w}^T\Phi(x))^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + const. \tag{14}$$

where $\Gamma$ represents the hyperparameter set $\{\alpha, \beta, \lambda\}$. (We drop the fixed model dependency wherever convenient). The weight posterior is therefore a Gaussian, $p(\mathbf{w}|\Gamma, D) = N(\bar{\mathbf{w}}, \mathbf{S})$ where

$$\mathbf{S} = \left(\alpha\mathbf{I} + \beta\sum_{\mu=1}^{P}\Phi\left(\mathbf{x}^\mu\right)\Phi^T\left(\mathbf{x}^\mu\right)\right)^{-1} \qquad \bar{\mathbf{w}} = \beta\boldsymbol{\Sigma}\sum_{\mu=1}^{P} t^\mu \Phi\left(\mathbf{x}^\mu\right) \tag{15}$$

The mean predictor is straightforward to calculate, $\bar{f}(\mathbf{x}) \equiv \int f(\mathbf{x};\mathbf{w})p(\mathbf{w}|D,\Gamma)d\mathbf{w} = \bar{\mathbf{w}}^T\Phi\left(\mathbf{x}\right)$. Similarly, error bars are straightforward, $\mathrm{var}(f(\mathbf{x})) = \Phi(\mathbf{x})^T\mathbf{S}\Phi(\mathbf{x})$ (predictive standard errors are given by $\sqrt{\mathrm{var}(f) + \sigma^2}$). In fig(2b), we show the mean prediction on the data in fig(2a) using 15 Gaussian basis functions with width $\lambda = 0.03$ spread out evenly over the input space. We set the other hyperparameters to be $\beta = 100$ and $\alpha = 1$. The prediction severely overfits the data, a result of a poor choice of hyperparameters.

### *Determining Hyperparameters: ML-II*

How would the mean predictor be calculated if we were to include the hyperparameters $\Gamma$ as part of a hierarchical model? Formally, this becomes

$$\bar{f}(\mathbf{x}) = \int f(\mathbf{x};\mathbf{w})p(\mathbf{w},\Gamma|D)d\mathbf{w}d\Gamma = \int\left\{\int f(\mathbf{x};\mathbf{w})p(\mathbf{w}|\Gamma,D)d\mathbf{w}\right\}p(\Gamma|D)d\Gamma \tag{16}$$

The term in curly brackets is the mean predictor for fixed hyperparameters. We therefore weight each mean predictor by the posterior probability of the hyperparameter $p(\Gamma|D)$. Equation (16) shows how to combine different

models in an ensemble – each model prediction is weighted by the posterior probability of the model. There are other non-Bayesian approaches to model combination in which the determination of the combination coefficients is motivated heuristically (ENSEMBLE LEARNING).

Provided the hyperparameters are well determined by the data, we may instead approximate the above hyperparameter integral by finding the MAP hyperparameters $\Gamma^* = \arg\max_\Gamma p(\Gamma|D)$. Since $p(\Gamma|D) = p(D|\Gamma)p(\Gamma)/p(D)$, if the prior belief about the hyperparameters is weak ($p(\Gamma) \approx const.$), we can estimate the optimal hyperparameters by optimising the hyperparameter likelihood

$$p(D|\Gamma) = \int p(D|\Gamma, \mathbf{w})p(\mathbf{w}|\Gamma)d\mathbf{w} \tag{17}$$

This approach to setting hyperparameters is called 'ML-II' (Bishop 1995; Berger 1985) and assumes that we can calculate the integral in equation (17). In the case of GLMs, this involves only Gaussian integration, giving

$$2\log p(D|\Gamma) = -\beta\sum_{\mu=1}^{P}(t^\mu)^2 + \mathbf{d}^T\mathbf{S}^{-1}\mathbf{d} - \log|\mathbf{S}| + k\log\alpha + P\log\beta + const. \tag{18}$$

where $\mathbf{d} = \beta\sum_\mu \Phi(\mathbf{x}^\mu)t^\mu$. Using the hyperparameters $\alpha, \beta, \lambda$ that optimise the above expression gives the results in fig(2c) where we plot both the mean predictions and standard predictive error bars. This solution is more acceptable than the previous one in which the hyperparameters were not optimised, and demonstrates that overfitting is avoided automatically. A non-Bayesian approach to model fitting based on minimsing a regularised training error would typically use a procedure such as cross validation to determine the regularisation parameters (hyperparameters). Such approaches require the use of validation data(Bishop 1995). An advantage of the Bayesian approach is that hyperparameters can be set without the need for validation data, and thus all the data can be used directly for training.

### *Relation to Gaussian Processes*

The use of GLMs can be difficult in cases where the input dimension is high since the number of basis functions required to cover the input space fairly well grows exponentially with the input dimension – the so called 'curse of dimensionality'(Bishop 1995). If we specify $n$ points of interest $\mathbf{x}^i, i \in 1, \ldots n$ in the input space, the GLM specifies an $n$-dimensional Gaussian distribution on the function values $f_1, \ldots, f_n$ with mean $\bar{f}_i = \bar{\mathbf{w}}^T\Phi\left(\mathbf{x}^i\right)$ and covariance matrix with elements $c_{ij} = c(\mathbf{x}^i, \mathbf{x}^j) = \Phi\left(\mathbf{x}^i\right)^T\Sigma\Phi\left(\mathbf{x}^j\right)$ (see GAUSSIAN PROCESSES). The idea behind a GP is that we can free ourselves from the restriction to choosing a covariance function $c(\mathbf{x}^i, \mathbf{x}^j)$ of the form provided by the GLM prior – any valid covariance function can be used instead. Similarly, we are free to choose the mean function $\bar{f}_i = m(\mathbf{x}^i)$. A common choice for the covariance function is $c(\mathbf{x}^i, \mathbf{x}^j) = \exp\left(-|\mathbf{x}^i - \mathbf{x}^j|^2\right)$. The motivation is that the function space distribution will have the property that for inputs $\mathbf{x}^i$ and $\mathbf{x}^j$ which are close together, the outputs $f(\mathbf{x}^i)$ and $f(\mathbf{x}^j)$ will be highly correlated, ensuring smoothness. This is one way of obviating the curse

of dimensionality since the matrix dimensions depend on the number of training points, and not on the number of basis functions used. However, for problems with a large number of training points, computational difficulties can arise, and approximations again need to be considered.

## Multilayer Perceptrons

Consider the case of a single hidden layer neural network

$$f(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{H} v_i g\left(\mathbf{x}^T\mathbf{u}^i + b_i\right) \tag{19}$$

where $g(x)$ is a non-linear sigmoidal transfer function, for example $g(x) = 1/(1+\exp(-x))$. The set of all weights (parameters), including input-hidden weights $\mathbf{u}^i$, biases $b_i$, and hidden-output weights $v_i$, is represented by the vector $\mathbf{w}$. If the weights are small, the network function $f$ will be smooth since only the near linear regime of the transfer function $g$ will be accessed. An appropriate prior to control complexity is therefore

$$\log p(\mathbf{w}|\alpha) = -\frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \frac{k}{2}\log\alpha + const. \tag{20}$$

where $k = dim(\mathbf{w})$. For the moment, we will assume that we know the value of the parameter $\alpha$. This gives the weight posterior as

$$\log p(\mathbf{w}|\alpha,\beta,D) = -\frac{\beta}{2}\sum_{\mu=1}^{P}\left(t^\mu - f\left(\mathbf{x}^\mu;\mathbf{w}\right)\right)^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + const. \tag{21}$$

where $\beta = 1/\sigma^2$. In fig(3) we show the result of using a 6 hidden unit network to fit the training data in fig(3). With $\alpha = 0.1$ and $\beta = 1000$, we drew a number of weight vectors $\mathbf{w}^l, l = 1,\ldots..15$ from the weight posterior $p(\mathbf{w}|D)$, equation (21) and considered the corresponding functions $f(\mathbf{x};\mathbf{w}^l)$. The mean and standard error bars calculated from these samples are plotted in fig(3). How these samples are obtained is discussed below. Note how the error bars automatically increase in regions of low data density.

### *Monte Carlo Sampling*

In general, the posterior distribution $p(\mathbf{w}|\Gamma,D)$ is non-Gaussian, and the integration required over the weight space to find, for example, the mean predictor

$$\bar{f}(\mathbf{x}) = \int f\left(\mathbf{x};\mathbf{w}\right)p(\mathbf{w}|\Gamma,D)d\mathbf{w} \tag{22}$$

is difficult. An approximate solution is provided by Monte Carlo sampling(Neal 1996; Bishop 1995)

$$\int f(\mathbf{x};\mathbf{w})p(\mathbf{w}|\Gamma,D)d\mathbf{w} \approx \frac{1}{L}\sum_{i=1}^{L} f(\mathbf{x};\mathbf{w}^i) \tag{23}$$

where the sample weights $\mathbf{w}^i$ are drawn from the posterior distribution. In principle, this procedure is exact in the limit $L \to \infty$. The great difficulty, however, is in constructing a finite, representative set of samples $\{\mathbf{w}_i\}$, and it is easy to remain trapped in un-representative parts of the posterior distribution (Neal 1996).

Consider the problem of drawing samples from a general distribution $p(\mathbf{x}) \propto \psi(\mathbf{x})$ – see fig(4). Let $\mathbf{x}^{old}$ be a sample point from $p(\mathbf{x})$. We propose a new sample point $\mathbf{x}^{new} = \mathbf{x}^{old} + \boldsymbol{\eta}$ where each element $\eta_i$ is sampled from a zero mean Gaussian distribution with variance $\tau^2$. We accept $\mathbf{x}^{new}$ if $\psi(\mathbf{x}^{new}) > \psi(\mathbf{x}^{old})$, since the new candidate sample point is more likely than the old sample point. However, this does not constitute a valid sampling scheme since we only accept increasingly likely points, targeting therefore only the modes of the distribution. To correct for this, we accept a less likely candidate with probability $\psi(\mathbf{x}^{new})/\psi(\mathbf{x}^{old})$. This valid sampling scheme is called the Metropolis method and forms the basis for many generalisations(Neal 1996; Neal 1993).

In high dimensions, Metropolis sampling can be inefficient since it is unlikely that testing a new point a long way from the current sample point will result in a more likely point (if you're standing on a mountain and jump, it's more likely that you'll end up at a point lower than your current point). Thus only very small jumps will be accepted in high dimensional spaces, and many samples are required to form a good representation of the distribution. The Hybrid Monte Carlo scheme attempts to improve sampling efficiency and allow larger jumps by exploiting gradient information about the distribution and has been successfully employed in Bayesian neural networks(Neal 1996).

### *Laplace's Method*

Whilst sampling techniques can be attractive, convergence to a representative set of samples is difficult to assess and can be very slow. Laplace's method is a perturbation technique motivated by the fact that as the number $P$ of training data points is increased, the posterior distribution typically approaches a Gaussian (Walker 1969) whose variance goes to zero in the limit $P \to \infty$ (we leave aside here the issues of inherent network symmetries). In order to calculate this Gaussian approximation, we consider the posterior distribution, equation (21)

$$p(\mathbf{w}|D, \Gamma) \propto \exp\left(-\phi\left(\mathbf{w}\right)\right) \tag{24}$$

and expand $\phi$ around a mode of the distribution, $\mathbf{w}_* = \arg\min \phi\left(\mathbf{w}\right)$,

$$\phi\left(\mathbf{w}\right) \approx \phi\left(\mathbf{w}_*\right) + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}_*\right)^T \mathbf{H}\left(\mathbf{w} - \mathbf{w}_*\right), \tag{25}$$

where

$$\mathbf{H} = \left.\nabla\nabla\phi\left(\mathbf{w}\right)\right|_{\mathbf{w}_*} \tag{26}$$

is the local Hessian matrix. This local expansion defines a Gaussian approximation

$$p(\mathbf{w}|D, \Gamma) \approx \frac{|\mathbf{H}|^{1/2}}{(2\pi)^{k/2}} \exp\left\{-\frac{1}{2}\left(\mathbf{w} - \mathbf{w}_*\right)^T \mathbf{H}\left(\mathbf{w} - \mathbf{w}_*\right)\right\}. \tag{27}$$

The expected value of $f(\mathbf{x};\mathbf{w})$ as required in equation (22) can be evaluated by making a further local linearization of the function $f(\cdot,\mathbf{w})$ around the point $\mathbf{w}_*$. In a practical implementation, a standard non-linear optimization algorithm such as conjugate gradients is used to find a mode $\mathbf{w}_*$ of the log posterior distribution(Bishop 1995).

### *Determining Hyperparameters*

So far we have assumed that the hyperparameters of the MLP are fixed. In a fully Bayesian treatment we would define prior distributions of the hyperparameters and then integrate them out. Since exact integration is analytically intractable, we can use ML-II to estimate specific values for the hyperparameters by maximizing the marginal likelihood $P(D|\Gamma)$ (equation (17)) with respect to $\Gamma$. Using MLPs, the integrand in equation (17) is non-Gaussian and $p(D|\Gamma)$ needs to be approximated. This can be achieved using Laplace's method by locally expanding the integral to second order in the weights. This leads to simple re-estimation formulae for the hyperparameters expressed in terms of the eigenvalue/eigenvector decomposition of the Hessian matrix. This treatment of hyperparameters is called the *evidence* framework (MacKay 1995) and involves alternating the optimization of $\mathbf{w}$ (mode finding) for fixed hyperparameters with re-estimation of the hyperparameters by re-evaluating the Hessian matrix for the new value of $\mathbf{w}$. The various approximations involved in this approach improve as the number of data points $P \to \infty$. However, for a finite data set it can be difficult to asses the accuracy of the method. One obvious limitation is that it only takes account of the behavior of the posterior distribution at the mode.

### *The KL Variational Approach*

The Kullback Leibler divergence is a measure of the difference between two probability distributions $p(\mathbf{x})$ and $q(\mathbf{x})$(Cover and Thomas 1991)

$$KL(q,p) = \int \{q(\mathbf{x}) \log q(\mathbf{x}) - q(\mathbf{x}) \log p(\mathbf{x})\}\, d\mathbf{x} \tag{28}$$

This has the advantageous properties $KL \geq 0$ and $KL = 0$ if and only if $p \equiv q$. Consider the $KL$ divergence

$$KL(q(\mathbf{w}), p(\mathbf{w}|\Gamma, D)) \geq 0 \tag{29}$$

Finding the best distribution $q(\mathbf{w})$ in a restricted set of possible distributions by minimising $KL(q,p)$ gives the best estimate (in the $KL$ sense) to the posterior distribution. From equation (29) we immediately have the bound

$$\log p(D|\Gamma) \geq \int -q(\mathbf{w}) \log q(\mathbf{w}) d\mathbf{w} + \int q(\mathbf{w}) \log p(D|\Gamma, \mathbf{w}) p(\mathbf{w}) d\mathbf{w} \tag{30}$$

We can make use of this lower bound to carry out an approximate ML-II hyperparameter optimisation by the following two step procedure: First fix the hyperparameters $\Gamma$ and optimise the bound, equation (30), with respect to $q(\mathbf{w})$. Then, for fixed $q(\mathbf{w})$, optimise the bound with respect to $\Gamma$. This scheme is a generalisation of

the Expectation-Maximisation procedure (see the article by Neal and Hinton in Jordan 1998), and is also called 'ensemble learning'(Barber and Bishop 1997).

## Bayesian Pruning

In the previous sections, we discussed the idea of using a prior which encourages smoothness of the input-output mapping. Since neural networks are non-linear functions of a linear combination of inputs, it is reasonable to use a prior which encourages small weights, $p(\mathbf{w}) \propto \exp\left(-\mathbf{w}^T\mathbf{A}\mathbf{w}/2\right)$. Typically, only diagonal matrices $\mathbf{A}$ are considered. We can group weights into clusters containing one or more weights and associate with each cluster $c$ a common hyperparameter $\alpha_c$. The Bayesian approach results in a posterior distribution over these hyperparameters $\alpha_c$. Alternatively, we can optimise the hyperparameters using ML-II. If the posterior distribution favours large $\alpha_c$ values, then effectively the weight cluster $c$ is not contributing to the network, and may be pruned. A useful choice of clustering is to group all the weights from a single input $x_i$ into the hidden units (note that these weights are different from the weights which fan in to a hidden node). If the hyperparameter $\alpha_i$ (after ML-II optimisation) associated with the weights fanning out from input $x_i$ is large, the contribution of input $x_i$ is negligible, and can be excluded. This is called 'automatic relevance determination'(MacKay 1995).

### *The Relevance Vector Machine*

In the discussion regarding GLMs, $f(\mathbf{x}) = \sum_i w_i\phi_i(\mathbf{x})$, we fixed the centres of the basis functions $\phi_i$. Similarly, in the RVM we use fixed basis functions(Tipping 2001). By associating with each weight $w_i$ a regularising prior $p(w_i) \propto \exp(-\alpha_i w_i^2/2)$, we can perform ML-II to optimise the hyperparameters $\alpha_i$. After optimisation, typically many of the $\alpha_i$ will become very large, effectively removing the basis function $\phi_i$ from the model. This pruning procedure often results in a much sparser representation of the data in terms of only the 'relevant' basis functions; this scheme is therefore particularly useful for compression. This sparseness effect is similar –although not equivalent– to the support vector machine (SUPPORT VECTOR MACHINES), in which training points are effectively removed if they do not affect the prediction of the model.

## Classification

The previously described methods can be applied to classification, usually with only minor modification. For convenience, we consider here only problems with two classes. The dataset is $D = \{(\mathbf{x}^\mu, t^\mu), \mu = 1, \ldots, P\}$ where $t^\mu \in \{0, 1\}$. In a probabilistic framework, we use the output of the network $f(\mathbf{x}; \mathbf{w})$ to represent the probability

that the input is in class 1. In this case, the likelihood is

$$p(D|\mathbf{w}) = \prod_{\mu=1}^{P} f(\mathbf{x}^\mu; \mathbf{w})^{t^\mu} \left(1 - f(\mathbf{x}^\mu; \mathbf{w})\right)^{1-t^\mu} \tag{31}$$

For example, we could take $f(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$, where $g(x) = 1/(1+\exp(-x))$(Bishop 1995). In the Bayesian approach, we need to specify a prior belief about the weights. As before, a sensible choice is $p(\mathbf{w}) \propto \exp(-\alpha\mathbf{w}^T\mathbf{w}/2)$ since smaller weights will give less certain predictions. This results in a posterior distribution $p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w})$. For a novel input $\mathbf{x}$ the probability that it belongs to class 1 is

$$p(t = 1|\mathbf{x}, D) = \int p(t = 1|\mathbf{x}, \mathbf{w})p(\mathbf{w}|D)d\mathbf{w} = \int g(\mathbf{w}^T\mathbf{x})p(\mathbf{w}|D)d\mathbf{w} \tag{32}$$

Consider, for example, fitting the data in fig(5a). The posterior distribution is given in fig(5d). The decision boundary ($p(t = 1|\mathbf{x}, \mathbf{w}, D) = 0.5$) for the MAP solution is given in fig(5b) along with the 0.1 and 0.9 decision contours. Another decision boundary associated with the posterior weights $\mathbf{w}^A$ is plotted in fig(5b). Because the decision boundaries are linear, the predictions of these single networks away from the data remain overly confident. The Bayesian prediction, equation (32), is plotted in fig(5c), and has decision boundaries which properly account for the uncertainty in the predictions away from the training data.

Since the final integrand in equation (32) depends only on the weight vector through the 'activation' $a = \mathbf{w}^T\mathbf{x}$, we only need to know the distribution of this one dimensional quantity. A reasonable assumption is that the activation will be Gaussian distributed $p(a) = N(\bar{a}, \mathrm{var}(a))$, and the resulting one dimensional integration $p(t = 1|\mathbf{x}, D) = \int g(a)p(a)da$ can be efficiently performed using quadrature. The statistics of the activation are

$$\bar{a} = \bar{\mathbf{w}}^T\mathbf{x}, \qquad \mathrm{var}(a) = \mathbf{x}^T\mathbf{\Sigma}\mathbf{x} \tag{33}$$

where $\bar{\mathbf{w}}$ and $\mathbf{\Sigma}$ are the mean and covariance of the weight posterior $p(\mathbf{w}|D)$. It is convenient to approximate these statistics using Laplace's method.

# Discussion

The Bayesian framework deals with uncertainty in a natural, consistent manner by combining prior beliefs about which models are appropriate with how likely each model would be to have generated the data. This results in an elegant, general framework for fitting models to data which, however, may be compromised by computational difficulties in carrying out the ideal procedure. There are many approximate Bayesian implementations, using methods such as sampling, perturbation techniques and variational methods. Often these enable the successful approximate realisation of practical Bayesian schemes. An attractive, built-in effect of the Bayesian approach is an automatic procedure for combining predictions from several different models, the combination strength of a model

being given by the posterior likelihood of the model. In the case of models linear in their parameters, Bayesian neural networks are closely related to Gaussian Processes, and many of the computational difficulties of dealing with more general stochastic non-linear systems can be avoided.

Bayesian methods are readily extendable to other areas, in particular density estimation, and the benefits of dealing with uncertainty are again to be found (see the article by Bishop in Jordan 1998). Traditionally, neural networks are graphical representations of functions, in which the computations at each node are deterministic. In the classification discussion, however, the final output represents a stochastic variable. We can consider such stochastic variables elsewhere in the network, and the sigmoid belief network is an early example of a stochastic network (Neal 1992). There is a major conceptual difference between such models and conventional neural networks. Networks in which nodes represent stochastic variables are called Graphical Models (see BAYESIAN NETWORKS) and are graphical representations of *distributions* (GRAPHICAL MODELS, PROBABILISTIC INFERENCE). Such models evolve naturally from the desire of incorporating uncertainty and non-linearity in networked systems.

# References


Barber, D. and Bishop, C. (1997). Ensemble Learning in Bayesian Neural Networks. In C. Bishop (Ed.), *Neural Networks and Machine Learning*, NATO ASI Series. Springer.

* Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (Second ed.). Springer.

* Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Box, G. and Tiao, G. (1973). *Bayesian Inference in Statistical Analysis*. Reading, MA: Addison–Wesley.

Cover, M. and Thomas, J. (1991). *Elements of Information Theory*. Wiley.

Jordan, M. (Ed.) (1998). *Learning in Graphical Models*. MIT.

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

* MacKay, D. J. C. (1995). Probable Networks and plausisble predictions – a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems 6*(3).

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence 56*, 71–113.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Cananda.

* Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer. Lecture Notes in Statistics 118.

Tipping, M. E. (2001). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research* (1), 211–244.

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society, B 31*(1), 80–88.

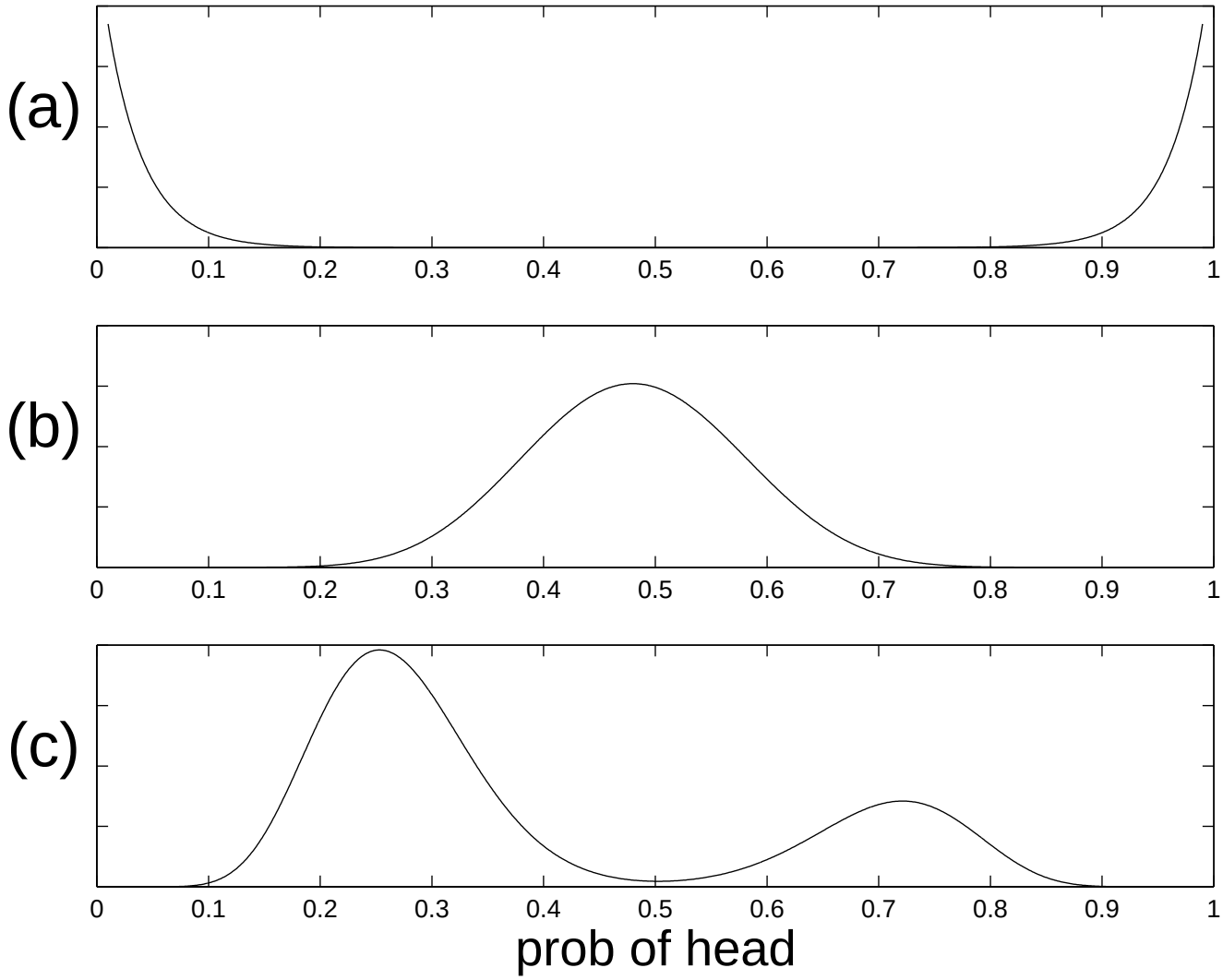
Figure 1: Coin Tossing: (a) The prior: this indicates our belief that the coin is heavily biased. (b) The likelihood after 13 Tails and 12 Heads are recorded, $\theta^{ML} = 0.48$. (c) The posterior: the data has moderated the strong prior beliefs resulting in a posterior less certain that the coin is biased. $\theta^{MAP} = 0.25$, $\bar{\theta} = 0.39$

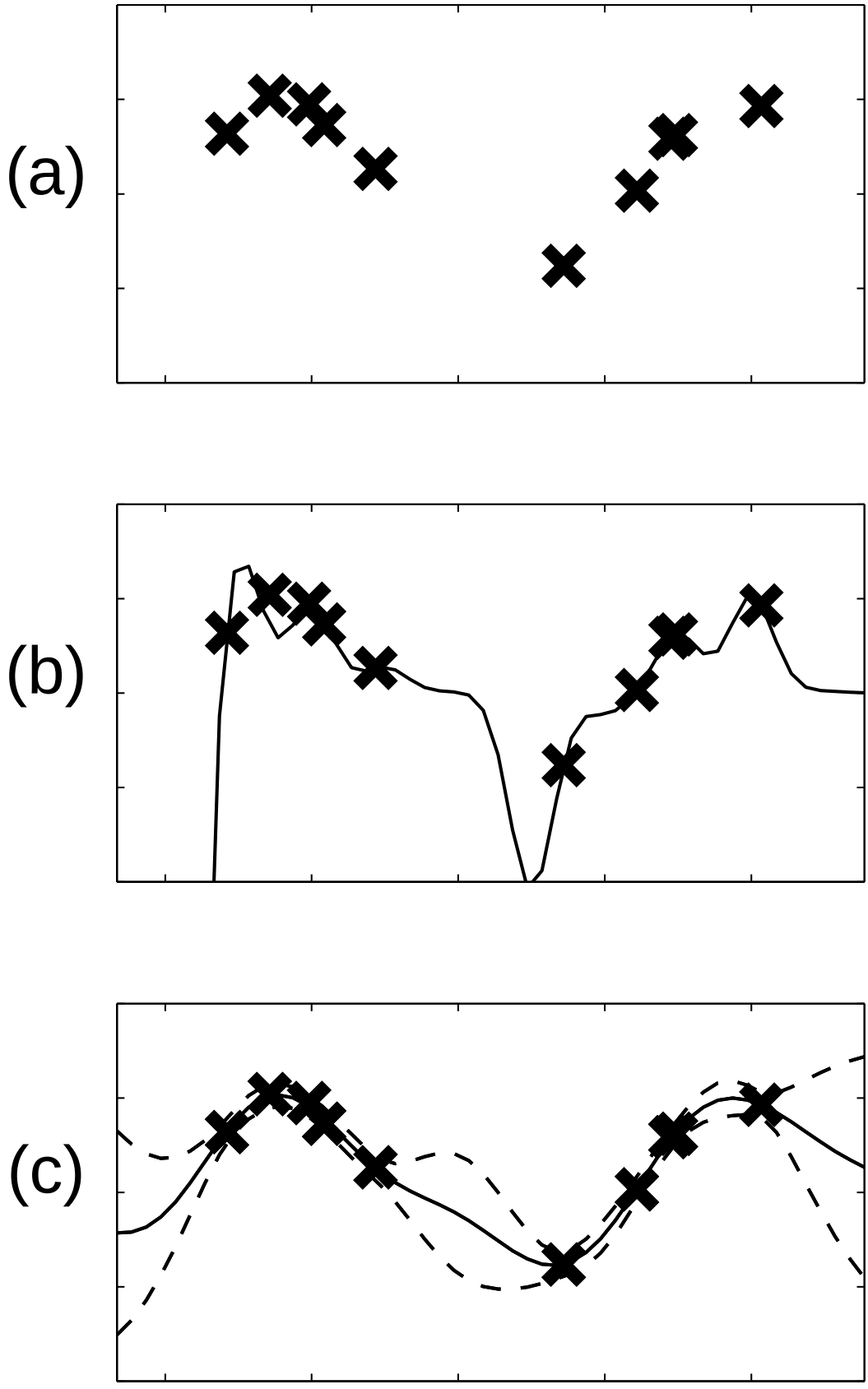
Figure 2: Along the horizontal axis we plot the input $x$ and along the vertical axis the output $t$. (a) The raw input-output training data. (b) Prediction using regularised training and fixed hyperparameters. (c) Prediction with error bars, using ML-II optimised hyperparameters.

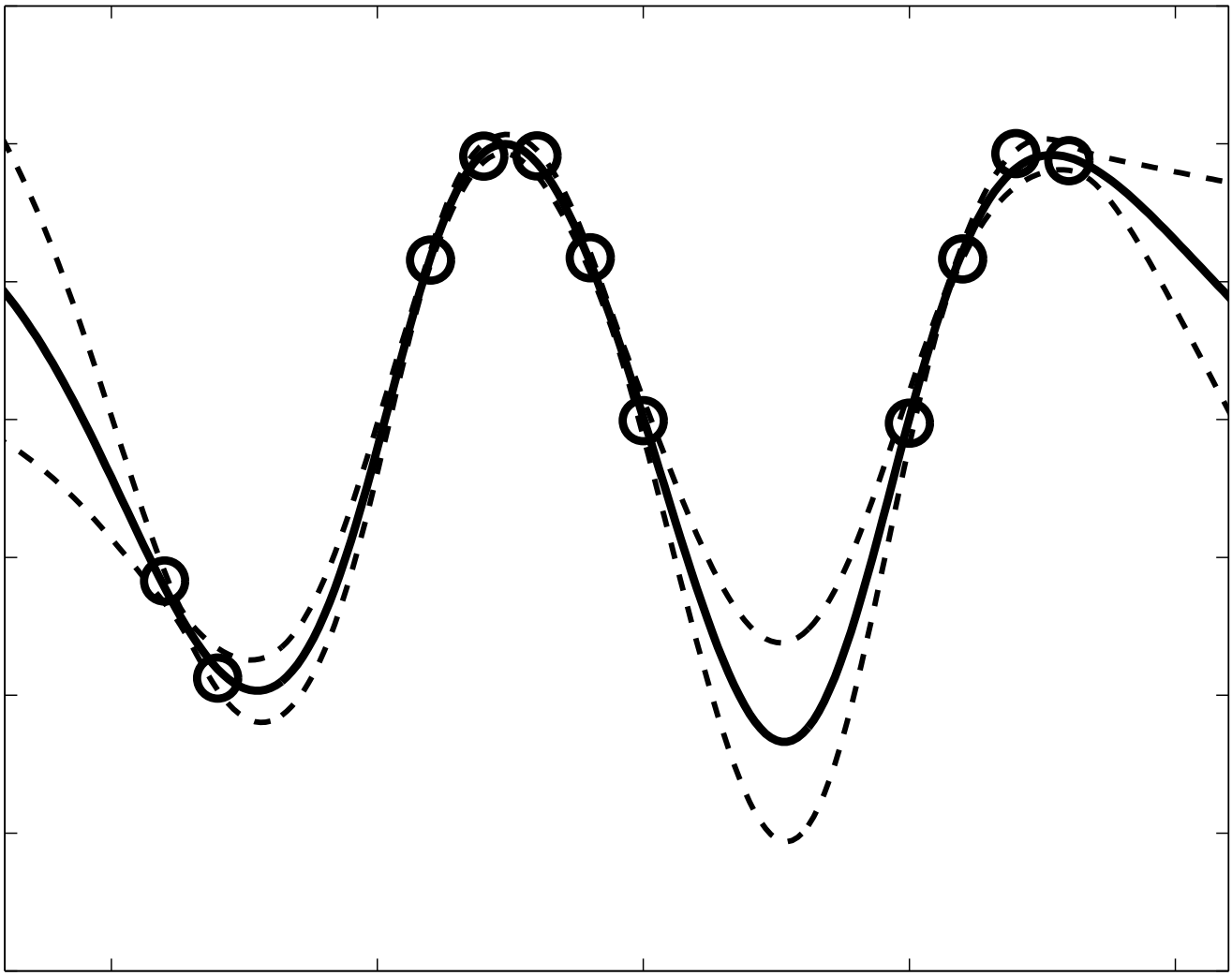
Figure 3: The raw input-output training data, with mean Bayesian MLP predictions (solid curve) and standard error bars (dashed curves). Note how the error bars increase away from the data.

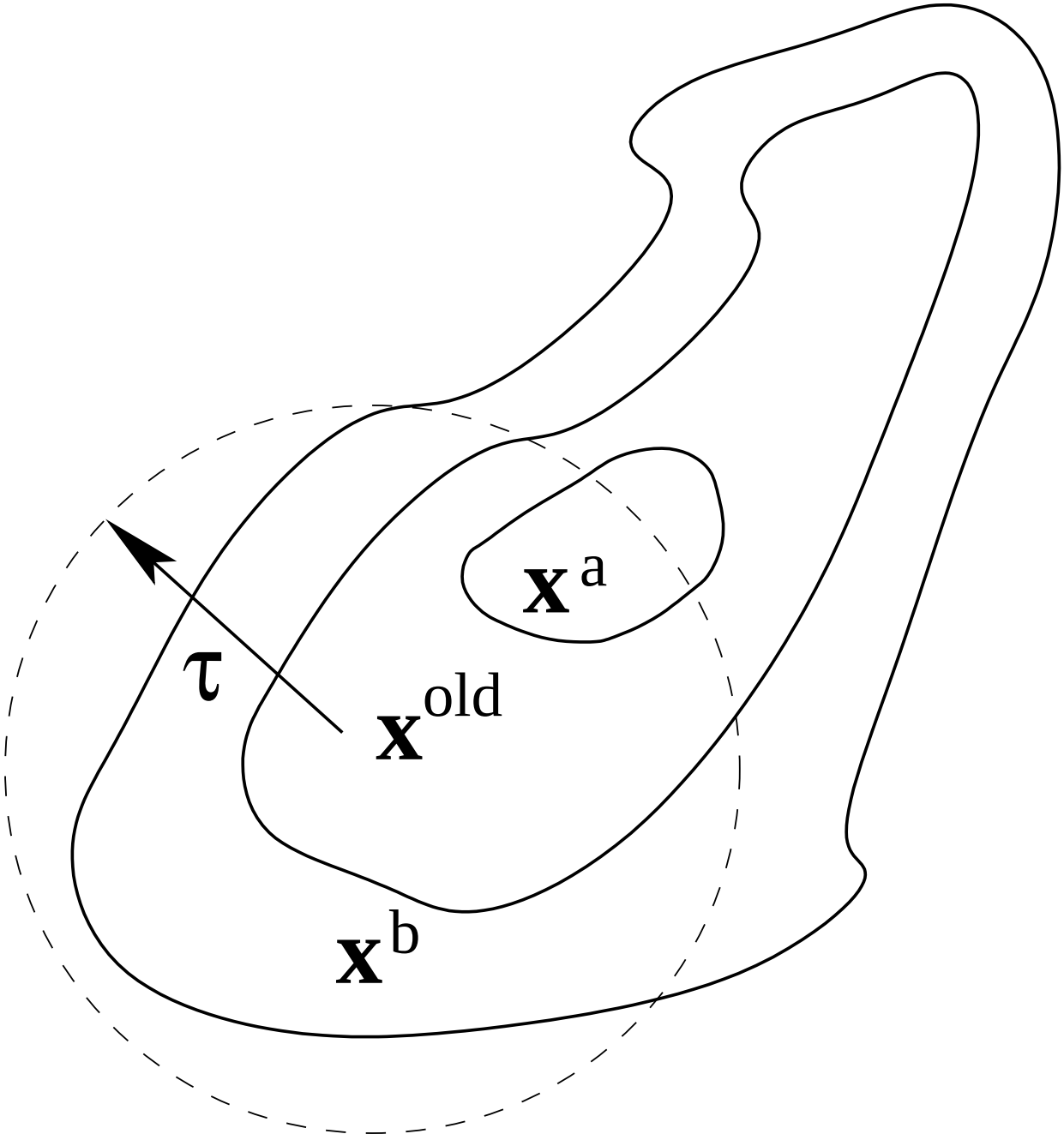
Figure 4: Metropolis Sampling from $p(\mathbf{x}) \propto \psi(\mathbf{x})$. Let $\mathbf{x}^{old}$ be a sample from the distribution $p(\mathbf{x})$. We propose a new candidate $\mathbf{x}^{new}$ by sampling from a Gaussian around $\mathbf{x}^{old}$ with width $\tau$. More likely candidates such as $\mathbf{x}^a$ are accepted. Less likely candidates such as $\mathbf{x}^b$ are accepted with probability $\psi(\mathbf{x}^b)/\psi(\mathbf{x}^{old})$.

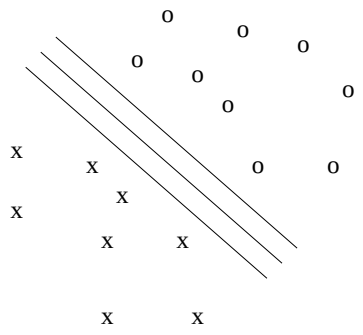
Figure 5: (a) The decision boundary and 0.1, 0.9 decision contours for the most likely predictor $\mathbf{w}^{MAP}$. (b) The predictions for $\mathbf{w}^A$. (c) The posterior averaged predictors. (d) The weight posterior distribution.


(a)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(b)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(c)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
prob of head

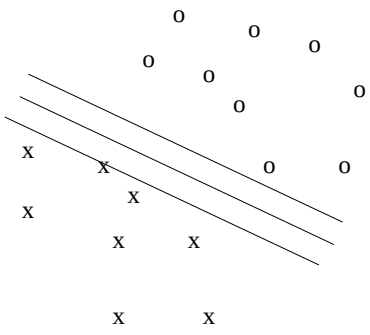

(a)
(b)
(c)



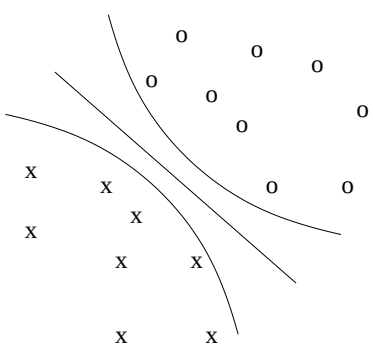

$\mathbf{x}^a$
$\tau$
$\mathbf{x}^{\text{old}}$
$\mathbf{x}^b$



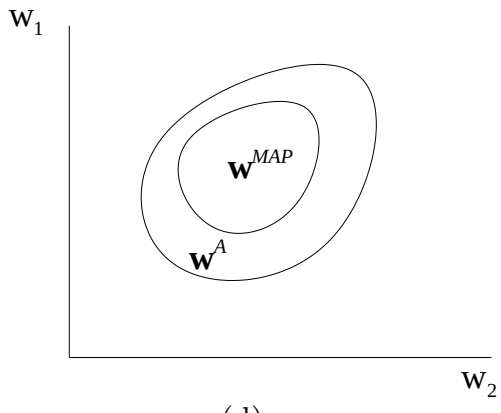
(a)



(b)



(c)




(d)