An Empirical Evaluation of Bayesian Sampling with Hybrid Monte Carlo for Training Neural Network Classifiers

Dirk Husmeier, William D. Penny, Stephen J. Roberts
Neural Systems Research Group
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road, London SW7 2BT
United Kingdom

Email: d.husmeier@ic.ac.uk

Abstract

This article gives a concise overview of Bayesian sampling for neural networks, and then presents an extensive evaluation on a set of various benchmark classification problems. The main objective is to study the sensitivity of this scheme to changes in the prior distribution of the parameters and hyperparameters, and to evaluate the efficiency of the so-called automatic relevance determination (ARD) method. The paper concludes with a comparison of the achieved classification results with those obtained with (i) the evidence scheme and (ii) with non-Bayesian methods.

Keywords

Bayesian statistics, prior and posterior distribution, parameters and hyperparameters, Gibbs sampling, hybrid Monte Carlo, automatic relevance determination (ARD), evidence approximation, classification problems, benchmarking.

1 Theory: Sampling of network weights and hyperparameters from the posterior distribution

The objective of this section is to give a concise yet self-contained overview of the Bayesian approach to learning with neural networks. The focus is put on the discussion of the choice and functional form of the prior and hyperprior, since this played a central role in our benchmark simulations. The article will only discuss classification problems, but the results equally apply to regression tasks. The reader who is interested in a more detailed exposition of this topic is advised to read the book by Neal (Neal, 1996), who introduced the method of Bayesian sampling to the neural network community.

1.1 Introduction

Consider a K-fold classification problem, where an m-dimensional feature vector $\mathbf{x}_t$ is assigned to one of K classes $\{C_1, \ldots, C_K\}$ indicated by a label vector $\mathbf{y}_t = (y_t^1, \ldots, y_t^K)$, $y_t^k = 1$ if $\mathbf{x}_t \in C_k$, $y_t^k = 0$ if $\mathbf{x}_t \notin C_k$, $\|\mathbf{y}_t\| = 1$. For a neural network with K softmax units in the final layer, the network outputs

$$f_k(\mathbf{x}_t; \mathbf{w}) \;=\; \frac{\exp[I_k(\mathbf{x}_t; \mathbf{w})]}{\sum_{i=1}^K \exp[I_i(\mathbf{x}_t; \mathbf{w})]} \in [0, 1] \tag{1}$$

can be interpreted as predictors for the class-conditional probabilities $P(y_t^k = 1|\mathbf{x}_t)$:

$$P(y_t^k = 1|\mathbf{x}_t, \mathbf{w}) \quad := \quad f_k(\mathbf{x}_t; \mathbf{w}) \tag{2}$$

where $\mathbf{w}$ denotes the vector of all network weights (including biases), and $I_k(\mathbf{x}_t; \mathbf{w})$ in (1) the input to the kth softmax node. The probability of the entire target vector $\mathbf{y}_t$ is equal to the probability of the indicated class and can be written as

$$P(\mathbf{y}_t|\mathbf{x}_t,\mathbf{w}) \quad = \quad \prod_{k=1}^{K} f_k(\mathbf{x}_t;\mathbf{w})^{y_t^k} \tag{3}$$

The likelihood for the entire training set $\mathbf{D} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{t=N}$ is given by

$$P(\mathbf{D}|\mathbf{w}) \quad = \quad \prod_{t=1}^{N}\prod_{k=1}^{K} f_k(\mathbf{x}_t;\mathbf{w})^{y_t^k} \tag{4}$$

and conventional network training schemes aim at finding the (local) maximum of this function or, equivalently, the (local) minimum of the energy or cost function

$$E(\mathbf{w}) \;:=\; -\ln P(\mathbf{D}|\mathbf{w}) \;=\; -\sum_{t=1}^{N}\sum_{k=1}^{K} y_t^k \ln f_k(\mathbf{x}_t;\mathbf{w}). \tag{5}$$

1.2 Sampling the weights from the posterior distribution

A disadvantage of the above maximum likelihood estimator is that the ensuing model may have a poor generalisation performance when the data is sparse. A more robust approach is to follow a Bayesian scheme and sample the weights from the posterior distribution $P(\mathbf{w}|\mathbf{D})$. Formally, the best prediction that can be obtained on the basis of the observed training set $\mathbf{D}$ is the probability of $\mathbf{y}$ conditional on the input vector $\mathbf{x}$ and the training data $\mathbf{D}$, which is obtained by integrating over the network weights $\mathbf{w}$,

$$\begin{aligned} P(\mathbf{y}|\mathbf{x},\mathbf{D}) &= \int P(\mathbf{y},\mathbf{w}|\mathbf{x},\mathbf{D})d\mathbf{w} \;=\; \int P(\mathbf{y}|\mathbf{w},\mathbf{x},\mathbf{D})P(\mathbf{w}|\mathbf{x},\mathbf{D})d\mathbf{w} \\ &= \int P(\mathbf{y}|\mathbf{x},\mathbf{w})P(\mathbf{w}|\mathbf{D})d\mathbf{w}. \end{aligned} \tag{6}$$

Note that the last step follows from the fact that the prediction model is solely determined by the network weights $\mathbf{w}$, $P(\mathbf{y}|\mathbf{w},\mathbf{x},\mathbf{D}) = P(\mathbf{y}|\mathbf{x},\mathbf{w})$, and that for supervised learning the distribution of the input vector $\mathbf{x}$ is not modelled by the network: $P(\mathbf{w}|\mathbf{x},\mathbf{D}) = P(\mathbf{w}|\mathbf{D})$. Let us define a regularisation or penalty term[1]

$$R(\mathbf{w}) \quad = \quad -\ln P(\mathbf{w}), \tag{7}$$

where $P(\mathbf{w})$ denotes the prior distribution of the weights *before* any data has been observed. Applying Bayes' rule and making use of (5) and (7), we obtain

$$\ln P(\mathbf{w}|\mathbf{D}) \;=\; \ln P(\mathbf{D}|\mathbf{w}) + \ln P(\mathbf{w}) + \text{const} \;=\; -E(\mathbf{w}) - R(\mathbf{w}) + \text{const} \tag{8}$$

With the definition of the total error function

$$U(\mathbf{w}) \quad := \quad E(\mathbf{w}) + R(\mathbf{w}) \tag{9}$$

we thus obtain from (8):

$$P(\mathbf{w}|\mathbf{D}) \quad = \quad \frac{1}{Z}\exp[-U(\mathbf{w})], \tag{10}$$

where Z is a normalisation constant. Inserting (10) into (6) yields

$$P(\mathbf{y}|\mathbf{x},\mathbf{D}) \quad = \quad \int P(\mathbf{y}|\mathbf{x},\mathbf{w})\frac{1}{Z}\exp[-U(\mathbf{w})]d\mathbf{w}, \tag{11}$$

which in general, however, cannot be solved analytically. A straightforward numerical approximation is the Metropolis algorithm (Metroplis et al., 1953): Given a weight vector $\mathbf{w}_t$ at time t, draw a new candidate state $\tilde{\mathbf{w}}$ from a symmetric distribution about $\mathbf{w}_t$. If $U(\tilde{\mathbf{w}}) < U(\mathbf{w}_t)$, then accept the new state: $\mathbf{w}_{t+1} = \tilde{\mathbf{w}}$, otherwise accept it only with the probability $P(\mathbf{w}_{t+1} = \tilde{\mathbf{w}}|\mathbf{w}_t) = \exp[U(\mathbf{w}_t) - U(\tilde{\mathbf{w}})]$. The resulting time series of configurations $\{\mathbf{w}_t\}$ constitutes a Markov chain which, assuming the condition of ergodicity is satisfied, converges towards a sample drawn from the equi-

[1] This term actually depends on some further, so-called *hyperparameters*, as will be discussed shortly.

librium distribution (10). The integral in (11) can then be approximated by

$$P(\mathbf{y}|\mathbf{x},\mathbf{D}) \;=\; \frac{1}{n_s}\sum_{t=n_{eq}+1}^{n_{eq}+n_s} P(\mathbf{y}|\mathbf{x},\mathbf{w}_t) \tag{12}$$

in which n_{eq} is the number of configurations discarded during equilibration, and n_s is the size of the sample drawn from the equilibrium posterior distribution.

1.2.1 Hybrid Monte Carlo

The disadvantage of the simple Metropolis algorithm is the exploration of the configuration space in a slow random walk: consecutive configurations are highly correlated and equilibration as well as sampling times can become exorbitantly long. This difficulty can be avoided with the hybrid Monte Carlo scheme, originally developed in quantum chromodynamics (Duane et al., 1987) and introduced to the neural network community by Neal (Neal, 1996). The idea is to introduce conjugate variables p_i and define the Hamiltonian

$$H(\mathbf{w},\mathbf{p}) \;:=\; U(\mathbf{w}) + K(\mathbf{p}) \;:=\; U(\mathbf{w}) + \sum_i \frac{p_i^2}{2m_i} \tag{13}$$

The parameters m_i correspond to masses of particles in a ficticious physical system, the weights w_i to their coordinates, and the conjugate variables p_i to their momenta. Solving the Hamiltonian equations of motion (e.g. (Balian, 1982)),

$$\frac{\partial w_i}{\partial t} \;=\; \frac{\partial H}{\partial p_i}, \qquad \frac{\partial p_i}{\partial t} \;=\; -\frac{\partial H}{\partial w_i}, \tag{14}$$

leads to a trajectory on a hypershell of constant energy, $H(\mathbf{w},\mathbf{p}) = C$. Assuming ergodicity, sampling along this trajectory is thus equivalent to sampling from the *microcanonical ensemble* (e.g. (Balian, 1982)),

$$P(\mathbf{w}, \mathbf{p}|\mathbf{D}) \quad = \quad \delta[H(\mathbf{w}, \mathbf{p}) - C]. \tag{15}$$

If, however, the momenta are resampled in regular intervals from the Maxwell distribution

$$P(\mathbf{p}|\mathbf{D}) \; = \; P(\mathbf{p}) \; = \; \frac{1}{Z_p} \exp[-K(\mathbf{p})] \; = \; \frac{1}{Z_p} \exp\left[-\sum_i \frac{p_i^2}{2m_i}\right], \tag{16}$$

which corresponds to bringing the system into contact with a heatbath of temperature $T = 1$ and thus allows transitions between different energy shells, then the resulting equilibrium distribution is that of the *canonical* ensemble (e.g. (Balian, 1982))

$$P(\mathbf{w}, \mathbf{p}|\mathbf{D}) \; = \; \frac{1}{Z_p Z} \exp[-H(\mathbf{w}, \mathbf{p})] \; = \; \frac{1}{Z_p} \exp[-K(\mathbf{p})] \frac{1}{Z} \exp[-U(\mathbf{w})]. \tag{17}$$

A comparison with (10) shows that the posterior distribution $P(\mathbf{w}|\mathbf{D})$ is given by the marginal distribution of $P(\mathbf{w}, \mathbf{p}|\mathbf{D})$ obtained after integrating out the momenta. Since, as seen from (17), this joint distribution factorises into the product of its marginals, the integration need not be carried out explicitly, but is simply effected by discarding the momentum term (16). The algorithm for sampling from (10) thus reduces to a numerical integration of the Hamiltonian equations of motion (14) and a regular resampling of the momenta from (16). However, inaccuracies in the numerical integration can lead to systematic errors in the sampling process. This can be corrected by introducing a Monte-Carlo acceptance criterion (therefore the name *'hybrid Monte Carlo'*, where the word *'hybrid'* is due to the fact that a new state is found by means of simulating a deterministic dynamical system): When simulating the movement on the energy hypershell (that is, *between* two consecutive resamplings of the momenta), the final configuration $\mathbf{w}_2$ is compared with the initial configuration $\mathbf{w}_1$ and accepted only with the probability

$$P_{accept}(\mathbf{w}_2|\mathbf{w}_1) \quad = \quad \min\{1, \exp[H(\mathbf{w}_1) - H(\mathbf{w}_2)]\}. \tag{18}$$

It is seen that in this way unstable algorithms with $H(\mathbf{w}_2) \gg H(\mathbf{w}_1)$ lead to high rejection rates. The resulting set of configurations $\{\mathbf{w}_t\}$ is applied as before to approximate the class-conditional probability (11) by (12). Note that the masses m_i are free parameters that can, in principle, be chosen arbitrarily. In practice, however, they determine the (local!) step-widths of the numerical integration scheme and therefore play a crucial role in preventing numerical instabilities. A heuristic algorithm for setting their values is discussed at length in (Neal, 1996).

1.3 Gibbs sampling of the hyperparameters

Let us return to the prior distribution $P(\mathbf{w})$, which itself depends on a set of so-called hyperparameters. A common choice is a Gaussian distribution

$$P(\mathbf{w}|\alpha) \quad = \quad \left(\frac{\alpha}{2\pi}\right)^{W/2} \exp\left(-\frac{\alpha}{2}\sum_{i=1}^{W} w_i^2\right), \tag{19}$$

which, as a result of (7), corresponds to a uniform weight-decay regulariser:

$$R(\mathbf{w};\alpha) \;=\; -\ln P(\mathbf{w}|\alpha) \;=\; \frac{\alpha}{2}\sum_{i=1}^{W} w_i^2 + C \tag{20}$$

(W denotes the total number of weights, and C is an expression independent of $\mathbf{w}$). This simple function, however, is not a particularly good choice for a prior since it violates basic scaling laws of the network parameters (see, e.g., (Bishop, 1995), chapter 9). A more general choice is a product of different Gaussian distributions for different weight groups $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_G)$, each having their own hyperparameter α_g:

$$P(\mathbf{w}|\boldsymbol{\alpha}) \;=\; \prod_{g=1}^{G} P(\mathbf{w}_g|\alpha_g) \;=\; \prod_{g=1}^{G} \left(\frac{\alpha_g}{2\pi}\right)^{W_g/2} \exp\left(-\frac{\alpha_g}{2}\sum_{i_g=1}^{W_g} w_{i_g}^2\right), \tag{21}$$

in which $W_g = \dim \mathbf{w}_g$ denotes the total number of parameters in the gth weight group. The corresponding regularisation term is now given by

$$R(\mathbf{w};\boldsymbol{\alpha}) \;=\; -\ln P(\mathbf{w}|\boldsymbol{\alpha}) \;=\; \sum_{g=1}^{G} \frac{\alpha_g}{2} \sum_{i_g=1}^{W_g} w_{i_g}^2 + C \tag{22}$$

The network prediction is given by marginalisation,

$$\begin{aligned} P(y|\mathbf{x},\mathbf{D}) \;&=\; \int P(y,\mathbf{w},\boldsymbol{\alpha}|\mathbf{x},\mathbf{D}) d\mathbf{w} d\boldsymbol{\alpha} \;=\; \int P(y|\mathbf{w},\boldsymbol{\alpha},\mathbf{x},\mathbf{D}) P(\mathbf{w},\boldsymbol{\alpha}|\mathbf{x},\mathbf{D}) d\mathbf{w} d\boldsymbol{\alpha} \\ &=\; \int P(y|\mathbf{w},\mathbf{x}) P(\mathbf{w},\boldsymbol{\alpha}|\mathbf{D}) d\mathbf{w} d\boldsymbol{\alpha}. \end{aligned} \tag{23}$$

The last step follows, again, from the fact that (i) the prediction model is solely determined by the network weights $\mathbf{w}$ and that (ii) learning is supervised. Sampling from the posterior distribution $P(\mathbf{w},\boldsymbol{\alpha}|\mathbf{D})$ now follows an iterative process known as *Gibbs sampling.* In a first step, the hyperparameters $\boldsymbol{\alpha}$ are kept constant and $\mathbf{w}$ is sampled[2] from $P(\mathbf{w}|\boldsymbol{\alpha},\mathbf{D})$. This follows exactly the way described above, with the regularisation term (7) determined by the current hyperparameters $\boldsymbol{\alpha}$ according to (22). In a second step, the parameters $\mathbf{w}$ are kept constant and the hyperparameters $\boldsymbol{\alpha}$ are sampled from the distribution

$$P(\boldsymbol{\alpha}|\mathbf{w},\mathbf{D}) \;=\; \frac{P(\mathbf{D},\mathbf{w},\boldsymbol{\alpha})}{P(\mathbf{D},\mathbf{w})} \;=\; \frac{P(\mathbf{D}|\mathbf{w})P(\mathbf{w}|\boldsymbol{\alpha})P(\boldsymbol{\alpha})}{P(\mathbf{D}|\mathbf{w})P(\mathbf{w})} \;=\; \frac{P(\mathbf{w}|\boldsymbol{\alpha})P(\boldsymbol{\alpha})}{P(\mathbf{w})} \tag{24}$$

This can be done efficiently if the regulariser is chosen such that a conjugate prior $P(\boldsymbol{\alpha})$ to the posterior distribution $P(\boldsymbol{\alpha}|\mathbf{w})$ exists and is of a sufficiently

[2] We here use the word *'sampling'* with the generalised meaning that the parameter vector $\mathbf{w}$ is updated according to the hybrid Monte-Carlo procedure, which implies that the correct distribution is left invariant.

simple form. This will be discussed in more detail in the following section.

1.3.1 A conjugate hyperprior

Assume that we have chosen the simple regulariser of equation (20), which corresponds to a common Gaussian distribution with precision α (19). The prior distribution on the weights is then given by

$$P(\mathbf{w}|\alpha) \;\propto\; \alpha^{W/2}\exp[-R(\mathbf{w})] \;=\; \alpha^{W/2}\exp[-\alpha r(\mathbf{w})] \qquad (25)$$

in which we have defined

$$r(\mathbf{w}) \;=\; \frac{R(\mathbf{w})}{\alpha} \;=\; \frac{1}{2}\sum_{i=1}^{W} w_i^2 \qquad (26)$$

The natural conjugate prior is a gamma distribution[3],

$$P(\alpha|\nu,\mu) \;=\; \mathcal{G}(\alpha|\nu,\mu) \;:=\; \frac{(\nu/2\mu)^{\nu/2}}{\Gamma(\nu/2)}\alpha^{\nu/2-1}\exp\left(-\frac{\nu}{2\mu}\alpha\right) \;\propto\; \alpha^{\nu/2-1}\exp\left(-\frac{\nu}{2\mu}\alpha\right) \qquad (27)$$

which is easily verified by inserting (25) and (27) into (24):

$$\begin{aligned}
P(\alpha|\mathbf{w},\mathbf{D}) \;&\propto\; P(\mathbf{w}|\alpha)P(\alpha) \;\propto\; \alpha^{W/2}\exp[-\alpha r(\mathbf{w})]\,\alpha^{\nu/2-1}\exp\left(-\frac{\nu}{2\mu}\alpha\right) \\
&=\; \alpha^{(W+\nu)/2-1}\exp\left(-\left[\frac{\nu}{2\mu}+r(\mathbf{w})\right]\alpha\right) \\
&=\; \alpha^{(W+\nu)/2-1}\exp\left(-\left[\frac{\nu+2\mu r(\mathbf{w})}{2\mu}\right]\alpha\right) \\
&=\; \alpha^{(W+\nu)/2-1}\exp\left(-\left[\frac{[\nu+2\mu r(\mathbf{w})][\nu+W]}{2(\nu+W)\mu}\right]\alpha\right) \\
&=\; \alpha^{\tilde{\nu}/2-1}\exp\left(-\frac{\tilde{\nu}}{2\tilde{\mu}}\alpha\right) \qquad (28)
\end{aligned}$$

[3]This parameterisation is not universal, but chosen because it allows an easy interpretation of the parameters μ and ν. A more commonly used functional form is $P(\alpha) \propto \alpha^{(b-1)}\exp(-\alpha/c)$, as in (Hoel, 1984), page 89. Obviously, $c = 2\mu/\nu$ and $b = \nu/2$.

In the last step the definitions

$$\tilde{\nu} = \nu + W \tag{29}$$

$$\tilde{\mu} = \frac{[\nu + W]\mu}{\nu + 2\mu r(\mathbf{w})} \tag{30}$$

have been introduced. It is seen that the resulting posterior distribution for α has again the form of a gamma density, that is, on normalisation one obtains:

$$P(\alpha|\mathbf{w}, \mathbf{D}) = \mathcal{G}(\alpha|\tilde{\nu}, \tilde{\mu}) \tag{31}$$

The mean of the gamma distribution $\mathcal{G}(\alpha|\nu, \mu)$ is equal to μ (e.g. (Hoel, 1984), p.88),

$$\langle\alpha\rangle = \mu \tag{32}$$

and the variance is given by (e.g. (Hoel, 1984), p.88)

$$\mathrm{var}(\alpha) = \frac{2\mu^2}{\nu}. \tag{33}$$

Thus small values of the so-called *shape parameter* ν correspond to vague prior distributions, whereas for larger values the distribution becomes more localised and thus more informative. A function plot of $\mathcal{G}(\alpha|\nu, \mu)$ for different combinations of ν and μ is shown in Figure 1 (Note that this plot is on a logarithmic scale, which implies that the standard transformation rule for probability densities, $P(y) = |\frac{dx}{dy}|P(x)$, has been applied).

1.3.2 Weight groups, automatic relevance determination, and scaling properties

In the previous section, a common prior for all network weights has been assumed. As mentioned before, this is a suboptimal choice since basic scaling laws are violated ((Bishop, 1995), chapter 9). A better choice is to break up the whole parameter set into different weight groups, each having their own hyperparameter α_g. The prior distributions for these hyperparameters are again chosen to be of the gamma form,

$$P(\alpha_g|\nu_g, \mu_g) \quad = \quad \mathcal{G}(\alpha_g|\nu_g, \mu_g) \tag{34}$$

The posterior distributions are then, in generalisation of equations (29), (30), and (31), given by

$$\begin{aligned} P(\alpha_g|\mathbf{w}, \mathbf{D}) &= \mathcal{G}(\alpha_g|\tilde{\nu}_g, \tilde{\mu}_g) && (35) \\ \tilde{\nu}_g &:= \nu_g + W_g && (36) \\ \tilde{\mu}_g &:= \frac{[\mu_g + 2\nu_g r_g(\mathbf{w})][\nu_g + W_g]}{\nu_g} && (37) \end{aligned}$$

where

$$r_g(\mathbf{w}) \quad := \quad \frac{1}{2}\sum_{i=1}^{W_g} w_{gi}^2 \tag{38}$$

and W_g denotes the total number of weights in the gth weight group. Especially, all the weights exiting the same input node are binned into a seperate weight group with a common weight decay constant α_g. The average size of α_g after equilibration is a measure of the relevance of the corresponding input, therefore this scheme is usually referred to as the method of *automatic relevance determination* (ARD) (Neal, 1996). If a given input x_g is irrelevant, the mean of the respective posterior distribution $P(\alpha_g|\mathbf{w}, \mathbf{D})$ can be assumed to be considerably increased. In this way the sampling process will tend to select large weight-decay constants α_g, which forces the associated weights to vanish. Hence the irrelevant input will be effectively discarded.

We conclude this subsection with a brief comment on the proper scaling of the hyper-hyperparameters ν_g and μ_g. Recall that an output node of the network has a sigmoidal squashing function to ensure that $f(\mathbf{x}; \mathbf{w}) \in [0, 1]$. Let us assume for simplicity that the network contains only one hidden layer with H nodes, and that there are no direct connections between the input and the output layers. Moreover, let I_k denote the total input to the kth output node. If the average size of the individual weights is fixed, then I_k scales like $\sqrt{H}$, that is, for large values of H the input I_k will most likely be somewhere in the saturation range of the squashing function. This clearly is an undesirable behaviour, so an intuitively plausible remedy of this deficiency is a scaling of the hyper-hyperparameters μ_{out} for the hidden-to-output weights according to

$$\mu_{out} \propto H. \tag{39}$$

Since μ_{out} determines the mean of the prior distribution $P(\alpha_{out})$, the sampling process will tend to select larger values of the weight decay hyperparameter α_{out}. Note that α_{out} is the inverse of the prior standard deviation of the weights, which thus scales like $1/\sqrt{H}$. Consequently, the average size of the hidden-to-output weights will be appropriately decreased, thus avoiding the scenario described above. In fact, a theoretical study ((Neal, 1996), chapter 2) shows that the scaling rule of equation (39) leads to a reasonable prior over functions in the limit of large hidden layers, $H \to \infty$. Scaling laws for the more complex cases of several hidden layers and direct weights between the input and hidden layers are derived in (Neal, 1996).

1.3.3 Hierarchical hyperpriors

The previous section has introduced the concept of independent, distinct weight groups. This scheme can be generalised to a hierarchical order in which a soft coupling between conceptually similar weight groups is introduced. Take, for example, the hyperparameters α_g for different groups of input-to-hidden weights (ARD scheme). If several of these hyperparameters are driven to small values during the sampling process, this might indicate that the means μ_k of the hyperpriors $P(\alpha_k|\nu_k, \mu_k)$ were chosen too large. Rather than correcting these means for each weight group seperately, a soft coupling scheme would allow a global correction of this deficiency for all weight groups simultaneously. This can be effected by introducing a hierarchical hyperprior of the following form:

$$P(\alpha_g|\nu_g, \mu_g) = \mathcal{G}(\alpha_g|\nu_g, \mu_g) \tag{40}$$

$$P(\mu_g|\nu_0, \mu_0) = \mathcal{G}(\mu_g|\nu_0, \mu_0) \tag{41}$$

Equation (40) is identical to (27) except that the hyper-hyperparameter μ_g, which defines the mean of the distribution in (40), is itself given a higher-order hyperprior (41) with higher-order hyper-hyperparameters ν_0 and μ_0. The sampling process from the respective posterior distribution can be effected with a modified version of rejection sampling, as described in the appendix of (Neal, 1996).

1.4 Bayesian sampling for neural networks: A summary of the complete scheme

The complete sampling scheme can be summarised as follows. Given the values for the momenta and the hyperparameters, the Hamiltonian equations of motion (14) are numerically integrated over L discrete time steps using, for instance, the Leapfrog algorithm (Berendsen and van Gunsteren, 1986). The final configuration is accepted according to equation (18) on the basis of the difference between the final and the initial energy, $H(\mathbf{w}_2) - H(\mathbf{w}_1)$. Given the new set of network weights, the hyperparameters are sampled from the posterior distribution (35) and the momenta p_i from the Maxwell distribution (16). Note that a change of the hyperparameters leads to a change of the error surface $U(\mathbf{w})$ via (9) and (22). This requires, in practice, a re-evaluation of the discretisation stepsizes, that is, the ficticious masses m_i. A critical parameter of the algorithm is the trajectory length L. If L is chosen too short, consecutive configurations are highly correlated and the configuration space will be explored in a slow random walk. If, conversely, L is too large, the hyperparameters are re-sampled too rarely, which again slows down the overall convergence rate. A detailed investigation of this aspect can be found in (Neal, 1996), chapter 3. We will here only state a general rule of thumb: In the equilibration phase, L should be chosen as small as $L \approx 50 - 100$ to allow a fast dissipation of energy by frequently resampling the momenta. In the sampling phase, however, much larger values of about $L > 1000$ are required to effectively avoid a random walk.

2 Benchmark problems

We tested the Bayesian sampling scheme on a set of five benchmark classification problems, which will be described briefly in the following section. An overview can be found in Table 1.

2.1 Neal's synthetic three-way classification problem

This problem is taken from Neal (Neal, 1997), who created a synthetic data set for a three-way classification problem as follows. Data items are generated by first drawing quantities x_1, x_2, x_3, and x_4 independently from distributions uniform over (0,1). The class of the item, represented by '0', '1', or '2', is then selected as follows: if the two-dimensional Euclidean distance of (x_1,x_2) from the point (0.4,0.5) is less than 0.35, the class is set to '0'; otherwise, if $0.8x_1 + 1.8x_2$ is less than 0.6, the class is set to '1'; and if neither of these conditions holds, the class is set to '2'. Note that x_3 and x_4 have no effect on the class. The class selected by this procedure is the target value for the network; the inputs available for use in predicting the class are not x_1, x_2, x_3, and x_4 themselves, however, but rather these four values plus Gaussian noise of standard deviation 0.1. The training set contained $N_{train} = 400$ cases, and $N_{gen} = 600$ exemplars were used for testing. The class labels as a function of the relevant variables x_1 and x_2 are plotted in Figure 2, bottom.

2.2 Ripley's synthetic data

This problem is taken from (Ripley, 1994). There are two features and two classes, where each class has a bimodal distribution, as seen from Figure 2, top left. The class distributions are given by equal mixtures of two normal distributions, whose overlap is chosen to allow a best-possible error rate (Bayes limit) of about 8%. As a further difficulty, two further irrelevant inputs, drawn from a uniform distribution over the interval $[0, 1]$, are included (this is different from the original problem studied in (Ripley, 1994)). A data set of size $N_{train} = 250$ exemplars was used for training, and the classification performance was tested on an independent test set of size $N_{gen} = 1000$.

2.3 Tremor data

The aim of this classification task is to identify Parkinson's disease on the basis of muscle tremor. The data set was collected by Spyers-Ashby (Spyers-Ashby, 1996) and consists of two input features derived from measurements of arm muscle tremor and a class label representing patient or non-patient (see Figure 2, top right). As in the previous examples, the input vectors were augmented by two irrelevant inputs drawn from the uniform distribution over $[0, 1]$. We used $N_{train} = 179$ exemplars for training, and tested the classification performance on an independent test set of size $N_{gen} = 178$.

2.4 Ionosphere data

This radar data was collected by a system in Goose Bay, Labrador, which consists of a phased array of 16 high-frequency antennas; see (Sigillito et al., 1989) for details. The targets are free electrons in the ionosphere. 'Good' radar returns are those showing evidence of some type of structure in the ionosphere. 'Bad' returns are those that do not; their signals pass through the ionosphere. From pre-processing a complex signal with 17 pulse numbers was obtained, where each pulse number is described by 2 attributes. The total number of class attributes thus amounts to $m = 34$, with a binary target variable representing 'good' and 'bad' signals. The network was trained on a training set of size $N_{train} = 250$, and $N_{gen} = 150$ independent exemplars were used for assessing the classification performance after training.

2.5 Vowel recognition

This problem was taken from (Robinson, 1989) and concerns speaker-independent recognition of the eleven steady state vowels of British English. Eleven words, in which the eleven vowel sounds were recorded, were uttered once by each of fifteen speakers. Four male and four female speakers were used to train the networks, and the other four male and three female speakers were used for testing the performance. The speech signals were low pass filtered and digitised, and a 10-dimensional feature vector was extracted on the basis of the reflection coefficients as described in (Rabiner and Schafer, 1978). Each speaker was given six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the networks and 462 frames from the seven speakers used to test the networks.

3 Objective of the study and methodology

3.1 Prior distributions

The performance of the Bayesian MCMC scheme hinges on two decisions the experimenter has to take. The first concerns the sampling procedure and includes the setting of parameters like the numerical integration stepsize, the length of MCMC trajectories between successive hyperparameter updates etc. A poor choice of these parameters may leave the method extremely inefficient, but does not invalidate the approach as such. If, for example, the discretisation stepsize has been chosen too large, the numerical integration scheme will become unstable and trajectories may diverge to high-energy hypershells. This flaw, however, does not lead to any systematic errors in the procedure as it will be corrected by an increasing rejection rate. Hence given a sufficiently long sampling time, one can still expect to get reasonable results; it is just that in practice the computational resources for such inefficient sampling are usually not available. The second decision, however, is more fundamental and concerns the prior distributions for the network parameters and hyperparameters (henceforth referred to as the *prior* and the *hyperprior*, respectively). This inclusion of subjective prior knowledge is inherent to the whole Bayesian approach and has, at least for sparse data, a significant influence on the inference scheme. Yet while in conventional statistical applications the incorporation of prior knowledge can usually be done explicitly because the parameters of interest have *per se* some physical meaning, the neural network approach is troubled with the problem that the prior distribution is defined in the 'wrong' space. Given that the network ultimately implements an input-output mapping trying to fit a limited set of observations or measurements, one would like to specify a prior distribution in the space of functions itself, rendering, for instance, smooth mappings more likely than those with a large curvature. What one needs to do, though, is to decide on a prior in the *parameter space*, which usually defies the incorporation of prior domain-specific knowledge. Neal has studied the relationship between priors over parameters and priors over functions in the limit of infinitely large networks (Neal, 1996), and some more recent approaches do away with neural networks altogether and apply related models that allow a more straightforward specification of the prior over functions (Barber and Williams, 1998), (Neal, 1997). However, given that neural networks are increasingly being applied to a wide range of real-world pattern recognition and classification tasks and that in practice these networks are finite, it is important to understand the influence of the prior on the outcome of the Bayesian sampling scheme. For this reason all simulations were carried out with three different functional forms of the prior:

Different priors

- *No ARD:* All weights on connections between input and hidden-layer nodes have the same prior distribution governed by a common hyperparameter.
- *One-level ARD (ARD1):* Weights on connections between input and hidden-layer nodes are divided into weight groups, with weights on connections exiting the same input node belonging to the same group. The distributions of the weights in different weight groups are independent, each determined by their own hyperparameter.
- *Two-level ARD (ARD2):* Similar to the one-level scheme, the weights between the input and hidden layers are divided into separate weight groups with different hyperparameters. Different to the previous case, however, a coupling between the hyperparameters is introduced in that they are drawn from distributions whose hyper-hyperparameters are distributed according to a common hyper-hyperparameter distribution (whereas in the one-level scheme these hyper-hyperparameters are fixed). See equations (40) and (41).

For the first two schemes, no ARD and one-level ARD, at least three different hyperpriors for the weight-decay hyperparameters α_g were applied:

Different hyperpriors for ARD1

- A vague hyperprior, corresponding to the lack of prior knowledge of what α_g should be like. This is similar to the constant hyperprior usually applied in the evidence scheme (MacKay, 1992a).

- A narrow (informative) hyperprior supporting large values of the weight decay hyperparameter α_g.

- A narrow (informative) hyperprior supporting small values of the weight decay hyperparameter α_g.

For the two-level scheme, two different forms of the hyperprior were investigated:

Different hyperpriors for ARD2

- As a first alternative, the hyperprior for the high-level hyperparameter was chosen very vague, whereas the hyperpriors for the low-level hyperparameters, that is, the actual weight-decay constants α_g, was chosen rather informative.

- As a second alternative, the hyperpriors for both levels were chosen identically, with a medium informativeness between the extremes of the previous scheme.

The dependence of the hyperprior on the choice of the hyper-hyperparameters is shown in Figure 1 (note the logarithmic scale, which implies that the density has been duly transformed). The specification is given in terms of two hyper-hyperparameters: μ/n defining the mean of α/n, and ν determining the width of the distribution. Here n represents the total number of connections feeding into a target node, and α/n defines the weight-decay parameter per weight[4]. In what follows, we will refer to the hyperpriors by triplets of the form $(\mu/n, \nu_{\text{high}}, \nu_{\text{low}})$, where ν_{high} represents the vagueness hyper-hyperparameter for the top-level and ν_{low} the vagueness hyper-hyperparameter for the bottom-level hyperprior. When an entry is skipped, the respective hyper-hyperparameter is taken to be infinite, which effectively removes one level from the hierarchy. For example, $(100, 1.0, -)$ represents a simulation without ARD, where the hyperparameters in all weight groups are drawn from the same hyperprior $P(\alpha_g) = \mathcal{G}(\alpha_g|100n, 1.0)$. For $(100, -, 1.0)$ we have a one-level ARD scheme, with independent hyperpriors of the same functional form, $P(\alpha_g) = \mathcal{G}(\alpha_g|100n, 1.0)$. A two-level ARD scheme is specified, for instance, by a triplet of the form $(100, 0.1, 1.0)$, where the hyper-hyperparameter μ has the prior distribution $P(\mu) = \mathcal{G}(\mu|100n, 0.1)$, and the hyperparameters α_g have independent prior distributions of the form $P(\alpha_g) = \mathcal{G}(\alpha_g|\mu, 1.0)$.

3.2 Details of the simulations

Sampling of the parameters and hyperparameters from the posterior distribution followed the scheme outlined above, that is, a numerical integration of the Hamiltonian equations of motion (14) in parameter space between consecutive Gibbs-sampling updates of the hyperparameters. The discretisation stepsize was adapted heuristically as in (Neal, 1996), and the simulation was divided into three parts:

Equilibration and sampling scheme

[4]Recall from Section 1.3.2 that the intuitive idea behind this scaling scheme is that by increasing the number of connections feeding into a node, the average absolute input to that node is increased and thus pushed further towards the saturation range of the nonlinear transfer function. In order to keep the degree of nonlinear complexity the same, the weight decay constants should therefore be increased with increasing number of weights or, put in another way, the degree of nonlinear complexity is defined by α/n rather than by α. A more rigorous discussion of these scaling laws can be found in (Neal, 1996).

1. *Equilibration with fixed hyperparameters.* A total of n_0 trajectories over T_0 numerical integration steps are computed, while the hyperparameters are fixed at $\alpha_i = 0.5$ to prevent them from taking on unrealistic and unrepresentative values during this early phase of the equilibration process. This corresponds to small fixed weight-decay parameters in the evidence scheme (see (MacKay, 1992a)) at an early stage of the optimisation process.

2. *Equilibration with adapted hyperparameters.* Further n_1 trajectories over T_1 numerical integration steps each are simulated. At the end of each trajectory the hyperparameters are updated by Gibbs sampling, and the momenta are re-sampled from the Maxwell distribution (16). The value of T_1 should be chosen rather small to ensure a fast dissipation of energy by frequent re-sampling of the momenta.

3. *Main trajectory.* The above simulations are repeated with n_2 and T_2, where $T_2 \gg T_1$ to ensure that consecutive configurations are sufficiently uncorrelated. The final n_s configurations are stored and used for sampling, obtaining predictions by application of equation (12).

The values used in the present study were as follows: $n_0 = 10$, $T_0 = 100$, $n_1 = 100$, $T_1 = 100$, $n_2 = 400$, $T_2 = 1000$, except for the ionosphere data, for which $n_2 = 600$, and the vowel data, for which $n_2 = 1000$. We chose two different values for n_s: $n_s = 100, 200$ ($n_s = 100, 500$ for the vowel data). This allows testing how much the predictions vary with respect to changes in the length and time location of the sampling interval. An inspection of the parameter and hyperparameter trajectories[5], Figures 3-9, suggests that these settings of the simulation parameters were, in most cases, sufficient for equilibrating the system and for obtaining a sample representative of the posterior distribution; see, however, the discussion in section 4.4.

[5]These trajectories show only the evolution during the third phase of the above sampling scheme.

The computing times required for the different data sets are listed in Table 2. It is noted that a certain acceleration of the sampling scheme can be achieved by applying the method of partial gradients or by resampling the momenta with persistence (Neal, 1996). These issues, however, are not addressed in the present study as our objective was to investigate the dependence of the prediction performance on the choice of prior rather than looking more deeply into different ways of how to optimise the sampling procedure.

3.3 Further details

In all simulations, we applied a feedforward network with tanh units in a single hidden layer and all-to-all connections between adjacent layers. For multiclass problems with $K > 2$, the number of output nodes was chosen equal to the number of classes, and the softmax transfer function of equation (1) was employed. For binary classification problems, the network structure was simplified by employing only one output node estimating the conditional probability for class $\mathcal{C}_1$:

$$p(y_t = 1|\mathbf{x}_t, \mathbf{w}) \quad = \quad f(\mathbf{x}_t; \mathbf{w}). \tag{42}$$

The total probability is thus given by

$$p(y_t|\mathbf{x}_t, \mathbf{w}) \quad = \quad f(\mathbf{x}_t; \mathbf{w})^{y_t}[1 - f(\mathbf{x}_t; \mathbf{w})]^{1-y_t}. \tag{43}$$

This implies that the energy function of equation (5) has to be modified slightly. With the likelihood for the entire training set now given by

$$P(\mathbf{D}|\mathbf{w}) \quad = \quad \prod_{t=1}^{N} f(\mathbf{x}_t; \mathbf{w})^{y_t}[1 - f(\mathbf{x}_t; \mathbf{w})]^{1-y_t}, \tag{44}$$

where the target y_t is a binary scalar variable, the energy becomes

$$E(\mathbf{w}) \ := \ -\ln P(\mathbf{D}|\mathbf{w}) \ := \ -\sum_t y_t \ln f(\mathbf{x}_t;\mathbf{w}) - \sum_t (1-y_t)\ln[1-f(\mathbf{x}_t;\mathbf{w})]. \tag{45}$$

While one of the main objectives of this study was to investigate the dependence of the classification results on the prior for the weight-decay hyperparameters α_g of the input-to-hidden layer weights, we did not vary the prior distribution for the hyperparameters α_i of the remaining weight groups, that is, the (i) hidden-to-output weights, (ii) the hidden-unit biases, and (iii) the output-unit biases. For the hyperparameters corresponding to the biases (groups (ii) and (iii)), we chose a rather vague prior of the form

$$P(\alpha_{bias}) \quad = \quad \mathcal{G}(\alpha_{bias}|400, 0.5). \tag{46}$$

For the hyperparameters pertaining to the hidden-to-output weights (group (i)), we also chose a vague prior, but scaled the mean of this distribution with the number of hidden units H according to the scaling laws discussed in (Neal, 1996):

$$P(\alpha_{out}) \quad = \quad \mathcal{G}(\alpha_{out}|400H, 0.5). \tag{47}$$

Finally, it is mentioned that the data was subjected to the standard preprocessing technique of separate normalisation, that is, all explanatory variables were transformed to zero mean and unit variance.

3.4 Performance measure

The generalisation performance was estimated on the basis of the classification results obtained for the test set. Two different performance measures were employed. The first was based on a hard classification, which assigns an exemplar to the class labelled by the mode of the predicted class-conditional probability:

$$\mathbf{x}_t \in C_k \quad \text{iff} \quad P(k|\mathbf{x}_t) > P(i|\mathbf{x}_t) \quad \forall i \neq k \tag{48}$$

This lends itself to the percent misclassification error, which is a rather illustrative performance measure and useful for a comparison with alternative methods. The disadvantage, however, is that relevant information about the predicted class-conditional probability distribution is discarded. For a binary classification task, for example, an exemplar $\mathbf{x}_t$ is assigned to class C_1 no matter whether the predicted class-conditional probability is $P(C_1|\mathbf{x}_t) = 0.51$ or $P(C_1|\mathbf{x}_t) = 0.99$. In most safety-critical applications, to the contrary, the class-conditional probability is to be taken into account explicitly, and exemplars should only be classified if the predicted class-conditional probability exceeds a certain threshold (rejection option). We therefore chose as a second performance measure the negative log likelihood of the correct class, averaged over the test set $\mathbf{D}_{gen}$ (having size N_{gen}):

$$E_{gen}(\mathbf{w}) \;:=\; -\frac{1}{N_{gen}} \ln P(\mathbf{D}_{gen}|\mathbf{w}) \;=\; -\frac{1}{N_{gen}} \sum_{t=1}^{N_{gen}} \sum_{k=1}^{K} y_t^k \ln f_k(\mathbf{x}_t;\mathbf{w}). \tag{49}$$

This expression is similar to (5) except that the sum is now taken over the test set $\mathbf{D}_{gen}$ and that it is normalised by the factor $1/N_{gen}$. For the binary case the expression is slightly modified, as in (45):

$$E_{gen}(\mathbf{w}) \;:=\; -\frac{1}{N_{gen}} \left[\sum_t y_t \ln f(\mathbf{x}_t;\mathbf{w}) + \sum_t (1-y_t) \ln[1 - f(\mathbf{x}_t;\mathbf{w})] \right]. \tag{50}$$

This performance measure captures more information about the actual class-conditional distribution predicted by the network. However, the resulting numbers are less illustrative than the percentage classification error and usually do not allow a comparison with other models studied in the literature (since most authors only state the percent misclassification error).

4 Results

4.1 Model complexity

A central problem of conventional approaches to neural computation is that of model selection, that is, of finding the most parsimonious network structure that can sufficiently account for the observed data. This requirement for penalising model complexity results from the fact that conventional training schemes only aim at finding a local minimum of the (penalised) cost function, which is equivalent to finding the mode of the posterior distribution of the parameters conditional on the observed data. If the number of parameters is large and the available data limited, the mode is usually not representative of the distribution as a whole. This fact is known as overfitting. Note that the same problem applies to the evidence approach to Bayesian modelling (MacKay, 1992a), (MacKay, 1992b), which approximates the posterior distribution by a Gaussian bump centred on the mode.

In contrast, the proper Bayesian approach to neural computation investigated in the present study does not suffer from this deficiency as it explores the *whole* configuration space, and samples the parameters from the *entire* posterior distribution. For this reason over-complex models do not lead to any significant overfitting and, as pointed out by Neal (Neal, 1996), should be chosen in lack of any prior knowledge to ensure sufficient universal approximation capability of the network.

In the present study we tested this conjecture empirically by applying networks with different numbers of hidden units to Ripley's synthetic classification problem, the tremor data and the ionosphere data. The results are shown in Table 3. It can be seen that the dependence of the classification results on the network complexity is rather weak, and the deviations are not significant. Note especially that the largest networks with 8 hidden units do not give worse results than than their less complex counterparts. This is to be compared with a model selection study based on the evidence (Penny and Roberts, 1998) (obtained for the same data sets), which suggests that the optimal model size for conventional backpropagation training schemes is much smaller, namely about 3 hidden units.

For the following studies, networks with 8 hidden units were employed throughout.

4.2 Simulation results

Figure 3 shows, for Neal's synthetic problem, the evolution of the weight-decay hyperparameters α_g for the four weight groups $g \in \{1, 2, 3, 4\}$ of the input-to-hidden weights. The actual value plotted is the square root of the inverse of α_g, $\sqrt{\frac{1}{\alpha_g}}$, which represents the standard deviation of the (zero-mean Gaussian) prior distribution of the weights in the gth weight group. Figure 4 shows the square root of the average square weight obtained from the trajectories, that is, empirically sampled from the posterior distribution:

$$\sqrt{\langle w^2 \rangle_g} \quad \approx \quad \sqrt{\frac{1}{W_g} \sum_{i=1}^{W_g} w_{g,i}^2} \tag{51}$$

Note the similarity between corresponding graphs in the two figures. The top row in Figure 3 shows the evolution of $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$ when no ARD scheme is applied and all hyperparameters are identical. However, when we do apply ARD, the trajectories of the hyperparameters split up into two groups. Hyperparameters pertaining to relevant inputs, shown by the solid lines, take on rather large values, whereas the hyperparameters for irrelevant inputs, represented by dotted lines, are supressed by between one and two orders of magnitude. A similar behaviour can be found in the graphs for $\sqrt{\langle w^2 \rangle_g}$, Figure 4. When we do not apply ARD, the weights belonging to connections out of irrelevant inputs decrease only slightly (top graphs), whereas the application of ARD leads to a considerable suppression of their contributions (bottom graphs). It is interesting to inspect the graphs in the right column of Figures 3 and 4 more closely, which were obtained with a narrow, informative hyperprior $P(\alpha_g)$ with a very small value of $\mu = 10^{-4}$. This corresponds to the rather unrealistic prior assumption that the weight decay constants α_g should be very small, $\alpha_g \approx 10^{-4}$, and that we are fairly certain about this size. As a consequence, the average weight magnitude increases (right column of Figure 4), and overfitting results (as seen from Figure 5).

Figures 6 and 7 show the evolution of the square root of the average square weight $\sqrt{\langle w^2 \rangle_g}$ for Ripley's synthetic classification problem and for the tremor data, respectively. It can be seen that without ARD there is no clear differentiation between relevant and irrelevant weight groups, and all the input-to-hidden weights are of a similar size. When ARD is applied, weights exiting irrelevant inputs are noticeably suppressed. Moreover, there is a clear differentiation with respect to the importance of the two relevant inputs, where especially for the tremor data (Figure 7) one of the two relevant inputs is significantly suppressed. This can be understood from an inspection of Figure 10, which shows the tremor data and the classification boundaries implemented in a network trained with the Bayesian evidence method. Three regimes in the data space can be distinguished: (i) a cluster of '+'-exemplars *above* the upper decision boundary, (ii) a mixed cluster dominated by 'o'-exemplars *between* the two decision boundaries (shown in black in Figure 10), (iii) a mixed cluster dominated by '+'-exemplars *below* the lower decision boundary.

The upper cluster can easily be seperated from the rest by a straight line parallel to the abscissa axis, that is, on the basis of x_2 alone. It is only for the few '+'-exemplars in the third cluster that a nonlinear decision boundary in the $x_1 - x_2$ plane is required. Taking x_1 into account allows the additional correct classification of about 10 further data points. Given that the total size of the tremor data is $N = 357$, the maximum improvement in the classification performance by including x_1 is rather small, about $\frac{10}{357}100 = 2.8\%$. This suggests that the first coordinate, x_1, is indeed considerably less important than x_2, which is captured by the network and reflected in the evolution of the trajectories.

Figure 8 shows a similar plot for the ionosphere data, where each graph shows the square root of the average square weight $\sqrt{\langle w^2 \rangle_g}$, plotted against time, for each of the 34 input-to-hidden weight groups. Again, we observe that without ARD (upper row) the weights are of similar size, whereas the application of ARD results in a considerable broadening of the weight-magnitude spectrum. Interestingly, a comparison of the sub-figures in the bottom row suggests that a decrease of the vagueness of the hyperprior $P(\alpha)$ reduces the broadness of the posterior weight-magnitude spectrum, which is intuitively plausible.

Finally, Figure 9 compares the influence of a vague and an informative hyperprior $P(\alpha_g)$ on the weight-magnitude trajectories for the vowel data. Again it is seen that the vague hyperprior increases the broadness of the weight spectrum. Moreover, an inspection of the graphs on the left of Figure 9 suggests that the system has not yet reached equilibrium, which points to a possible 'danger' of vague hyperpriors. This will be discussed in more detail in the following subsection.

Comment: A detailed overview of all the simulation results obtained in the present study is given in Tables 4-13. The main results of interest are extracted and displayed in Figures 11,12, and 13.

4.3 The influence of the hyperprior: vague versus informative

The scatter plots in Figure 11 compare the results obtained with a vague hyperprior ($\nu = 0.1$) with those of different informative hyperpiors ($\nu = 1.0$). Values above the dotted line indicate a performance improvement, values below the dotted line point to a deterioration as a consequence of using a vague hyperprior. The star (*) stems from simulations on Neal's three-way classification problem, where a hyperprior supporting unrealistically small values for the weight-decay parameters was chosen ($\mu/n = 0.0001$). It is not surprising that in the latter case the results are worse than those obtained with a vague hyperprior - see also the discussion above in subsection 4.2. However, four values in the right plot of Figure 11 lie noticeably below the dashed line and thus indicate a significant performance deterioration caused by the vague hyperprior. The corresponding sampling trajectories are shown in the graphs on the left of Figure 9 (vowel data), which exhibit a broadened weight magnitude spectrum and oscillations. This points to a possible problem for vague hyperpriors. By enlarging the high-probability region in hyperparameter space, that is, the effectively accessible domain therein, equilibration as well as convergence times may increase considerably. This is especially a problem if low-frequency oscillations as a consequence of multimodality occur, which can partly be observed in Figure 9.

Most other entries in Figure 11, however, can be found near the dashed line, which indicates equal performance irrespective of the vagueness of the hyperprior. Moreover, the influence of the mode in the informative case, that is, whether it is situated at a large (circles) or a small (x-marks) weight-decay parameter, seems to be rather irrelevant. This suggests that, for most of the data sets studied here, the particular choice of hyperprior is a rather uncritical factor in the methodology, as long as the mode in the informative case is not completely misplaced and insufficient convergence can be ruled out.

4.4 The influence of the prior: ARD versus no ARD

Figures 12 and 13 show scatter plots comparing the generalisation performance obtained with and without ARD, both in terms of percent misclassification (Figure 12) and negative log likelihood of the correct class (Figure 13). It is seen that for Ripley's synthetic problem, the application of an ARD prior leads to a clear reduction of the generalisation error. This also holds, though to a lesser extent, for Neal's synthetic problem and for the vowel data, where ARD gives improved classification results in terms of one of the two performance measures. However, for the tremor and the ionosphere data, ARD clearly fails, showing a considerable misclassification increase in terms of both performance measures. This suggests that ARD is no panacea for improving the prediction performance and should not be applied recklessly.

4.4.1 Statistical significance

One may wonder whether the deterioration of the ARD results on the tremor and ionosphere data, as suggested by the scatter plots in Figures 12 and 13, is statistically significant. Moreover, all experiments were done on only a single randomly chosen data set, and different results might have been obtained if a different data set had been used. These questions are addressed in Appendix 2. A *matched-pair t test* suggests that the results obtained on the given data set are in fact statistically significant. The question of whether different results would have been obtained on other data sets is more difficult to answer, since the deterioration of ARD was observed on real-world data, and no further data than those used in our study are available. In order to get an approximate idea of the influence of resampling the data, we have applied the method of bootstrapping, which again suggests that the results obtained are significant.

4.4.2 Failure of ARD on the tremor data

Recall that for the tremor data most of the exemplars can be correctly classified on the basis of the second input x_2 alone, whereas the additional inclusion of x_1 achieves only a maximum further improvement of about 2.8% (see Figure 10). This difference in the relative importance of the two relevant inputs is reflected in the trajectories of Figure 7, which show that the system changes repeatedly between supporting and suppressing x_1. A comparison with the simulations not using ARD reveals that as a consequence of employing ARD, low-frequency oscillations and a broadening of the weight spectrum result. This requires much larger equilibration and sampling times, and the deterioration observed in the simulations seems to result from the fact that the set of weights obtained from the last 100 or 200 sampling steps is not representative of the equilibrium posterior distribution.

4.4.3 Failure of ARD on the ionosphere data

Figures 12, 13 and Tables 7, 12 show that the results obtained with ARD on the ionosphere data are considerably worse than the performance achieved without ARD. We therefore repeated the simulations with decorrelated (whitened) inputs. The idea behind this is as follows. For the prior distribution, different weight groups $\mathbf{w}_i$, $\mathbf{w}_j$ are independent, $P(\mathbf{w}_i, \mathbf{w}_j) = P(\mathbf{w}_i)P(\mathbf{w}_j)$. Now assume that the outputs are linear in the weights, $\mathbf{w}$, and that the data are Gaussian distributed. Then the posterior distribution of the weights, $P(\mathbf{w}|\mathbf{D})$, is also a Gaussian, whose inverse covariance matrix (the 'Hessian') is proportional to the expectation value of the outer product of the input vectors, $\langle \mathbf{x}\mathbf{x}^\dagger \rangle$, plus a diagonal matrix (resulting from the prior) (see, for instance, (Bishop, 1995), chapter 3). If the inputs x_i are uncorrelated, $\langle x_i x_j \rangle \propto \delta_{ij}$, this matrix is diagonal and there are no correlations between weights exiting different inputs. Hence[6] $P(\mathbf{w}_1, \ldots, \mathbf{w}_m|\mathbf{D}) = \prod_i P(\mathbf{w}_i|\mathbf{D})$, that is, in the posterior distribution different weight groups are also independent. This implies that rather than sampling from an mW dimensional space (where m is the number of inputs, and W the number of weights exiting a given input), we simultaneously sample from m independent W-dimensional spaces. For the ionosphere data this implies that as a result of decorrelating the inputs we reduce the dimensionality of the space from $mW = 34 \cdot 8 = 272$ down to only $m = 8$, which alleviates the sampling problem condsiderably and should increase the speed of convergence to the equilibrium distribution significantly.

For the nonlinear case the above argument no longer holds since the effect of the nonlinearities is to introduce correlations between the different weight groups. However, we may assume that these correlations are rather weak and can be treated as perturbations. Sampling then takes place in

[6]Note that uncorrelation implies independence due to the Gaussian distribution.

m weakly-perturbed 'semi-independent' W-dimensional subspaces, so that the convergence towards the equilibrium distribution might still be faster than when the original inputs are used, which consequently should lead to improved prediction results.

This conjecture was in fact borne out in the simulations. Figures 12, 13 and Tables 7, 12 suggest that as a consequence of decorrelating the inputs, a significant improvement in the classification performance over the results obtained with the original inputs was achieved. Note, however, that in terms of percent misclassification this performance is still worse than that obtained without ARD (see Appendix 2 for a significance test).

4.4.4 Increased simulation times

It was mentioned before that for the ARD prior applied to the tremor data, the trajectories of the weight magnitudes (Figure 7) show low-frequency oscillations. This suggests that the observed deterioration in the classification performance may result from too short a sampling period. Similarly, the discussion of the previous subsection suggests that the same applies to the ionosphere data, although this is not indicated explicitly in the respective trajectories (Figure 8, bottom). We therefore repeated the ARD simulations on the tremor and ionosphere data with considerably longer equilibration and sampling times. For the tremor data, the value of n_2 (defined in Section 3.2) was increased by a factor of 25 from $n_2 = 400$ to $n_2 = 10,000$ (that is, $10,000$ hybrid Monte-Carlo trajectories over 1000 leapfrog steps each were computed). For the ionosphere data, we increased the value of n_2 by a factor of 50 from $n_2 = 600$ to $n_2 = 30,000$ (thus $30,000$ hybrid Monte-Carlo trajectories over 1000 leapfrog steps each were computed). Note that in the latter case this required a computing time of about 50 hours on a

SUN ULTRA 10.[7] The results obtained from the second half of the configurations thus sampled are shown in Table 14. It is seen that increasing the equilibration and sampling times consistently improves the generalisation performance. This suggests that the worse results obtained with ARD are the consequence of an insufficient equilibration period and can be ultimately improved upon by increasing the simulation times. However, even with the longer simulation times chosen in this study, the results obtained *with ARD* are still worse (for the ionosphere data considerably worse!) than the results obtained *without ARD*. It thus seems that as a result of employing ARD, the convergence towards the equilibrium distribution has slowed down markedly, and extremely high computational resources are required to achieve results equivalent to those obtained without ARD.

4.5 Comparison with alternative methods

The classification results obtained with the Bayesian sampling scheme were compared with (i) the evidence approximation, outlined in Appendix 1 (see (MacKay, 1992a) for details), and (ii) alternative non-Bayesian methods. The simulations for the evidence scheme followed those in (Penny and Roberts, 1998). All networks contained a single hidden layer of tanh units, with all-to-all connections between adjacent layers, and a bias unit of constant activity 1 connected to all nodes in the hidden and output layers. A linear weight-decay regulariser was employed, with different regularisation parameters for the following weight groups: weights between the hidden and output layers, and weights exiting the same input node (ARD). Weights connected to the bias unit were left unregularised, in accordance with scaling laws discussed in (Bishop, 1995). The networks differed with respect to the

[7]Repeating these simulations on a SUN ULTRA 1, as used in our earlier simulations, would have required more than double the time (that is, more than 100 hours).

number of hidden units[8], and for each configuration, the simulations were repeated ten times, starting from different (random) initialisations. Each network was trained with conjugate gradients till convergence, while the hyperparameters were recalculated in regular intervals according to (57); see (MacKay, 1992a) for explicit formulas. The networks were ranked according to their model evidence, (65), where the explicit formula can be found in (MacKay, 1992a) again. The 5th column in Table 15 shows the classification performance of the network with the largest evidence. The 6th column is the error from an evidence-ranked committee, whose size was chosen after eyeballing a histogram of evidence values. A comparison of the computational costs with those required for Bayesian sampling is shown in Table 2.

The last column in Table 15 indicates the classification error obtained with other, non-Bayesian methods. This covers a broad spectrum of different approaches. The result on the tremor data was obtained with a committee of radial basis function classifiers (Roberts and Penny, 1997). The performance on Ripley's data set is that of a three-hidden unit multilayer perceptron trained with a single weight-decay regulariser (Ripley, 1994). For the ionosphere data, a nearest-neighbour classifier was employed. In their original paper, Sigillito et al. (Sigillito et al., 1989) quote a test error of 4%. We did not include this in the table, however, as minimum test error was used as a stopping criterion in an 'early-stopping' training scheme. This therefore constitutes a biased result. Finally, the result on the vowel data was obtained with an 88-hidden unit multilayer perceptron trained with early stopping (Robinson, 1989). We note that all these results were obtained after a good deal of experimentation by the respective authors, and generally speaking, the best results were reported. Moreover, the input vectors for

[8] For the data sets 'Ripley', 'Tremor', and 'Ionosphere', networks with 2,3,4, and 5 hidden units were employed. The networks applied to Neal's data set contained 3,4,5,6, and 8 nodes in the hidden layer. Finally, a plethora of networks with 3, 5, 7, 9, 11, 13, 15, and 22 hidden units was applied to the vowel data.

Ripley's data and the tremor data were not augmented by two irrelevant noisy inputs, which to some extent alleviates the classification problem.

Columns 2-4 of Table 15 show, for each of the three priors *no ARD*, *one-level ARD*, and *two-level ARD*, the median over all hyperpriors. It is seen that except for the tremor data (where the comparison with 'other' methods is not fair since in the latter case the input vectors did not include any noisy inputs), Bayesian sampling always achieves the best results. However, the performance varies largely with the functional form of the prior. For the vowel data, the best results are obtained with a two-level ARD scheme, for Ripley's synthetic problem, a one-level ARD prior gives the best results, and in the case of the ionosphere data, all methods are outperformed by Bayesian sampling *without* ARD. Comparing the alternative methods with these best Bayesian results is methodologically incorrect, though, as the choice of prior would be made *after* the data and results have been seen. Whether the choice of the optimal prior can be made on the basis of other (fair) first principles is not clear. The analysis of trajectories such as those in Figure 7, for example, might suggest rejecting an ARD prior on the grounds that the observed oscillations indicate inherent problems with the sampling scheme. However, to ensure a fair comparison, all alternatives results should be compared with those obtained with a two-level ARD scheme, which is the most general one and, in lack of any domain-specific knowledge, the common method of choice. Applying this idea to Table 15 still gives a very favourable result for Bayesian sampling, and suggests that its performance equals or exceeds that of the evidence scheme and the alternative non-Bayesian methods (except for the tremor data, where the comparison is not fair for reasons mentioned before).

5 Conclusions

The main results of this study can be summarised as follows:

- Given that the network remains sufficiently complex to be capable of modelling the data-generating function, the generalisation performance is robust with respect to changes in the number of hidden units. Note that this is fundamentally different to standard optimisation schemes which aim at finding the minimum of the (penalised) negative log likelihood of the data.
- If the prior and hyperprior are chosen appropriately, the training and test set performance do not differ significantly, that is, the generalisation performance can be estimated by the training error (see Figure 5).
- The details of the hyperprior are rather uncritical, unless extreme and unrealistic values for its mode are chosen. However, vague priors may slow down the convergence and sampling process, which for limited sampling times can deteriorate the classification results.
- The application of ARD is no panacea for improving the generalisation performance of a network. Whereas for three problems the error decreased as a result of applying ARD, the application of ARD to the tremor and ionosphere data led to a significant deterioration of the classification results. This could be redeemed slightly by increasing the equilibration and sampling times, which suggests that the poor ARD performance was caused by sampling from a non-equilibrium distribution and therefore could ultimately be improved upon by increasing the simulation times. The required computational costs, however, might become huge. Also, note that for the ionosphere problem, the non-equilibrium state is not indicated by the weight-size trajectories.

- In spite of these difficulties, Bayesian sampling tends to achieve equal or better results than the evidence scheme or alternative, non-Bayesian methods.

6 Acknowledgements

We would like to thank Radford M. Neal for helpful communication and for providing us with the programs for carrying out the MCMC simulations. William D. Penny is a research fellow funded by the UK Engineering and Physical Science Research Council (EPSRC). Dirk Husmeier is a research fellow supported by the Jefferiss Research Trust.

References

Balian, R. (1982). *From Microphysics to Macrophysics. Methods and Applications of Statistical Physics.* Springer-Verlag.

Barber, D. and Williams, C. K. I. (1998). Gaussian processes for bayesian classification via hybrid monte carlo. *Advances in Neural Information Processing Systems*, 9:340–346.

Berendsen, H. J. C. and van Gunsteren, W. F. (1986). Practical algorithms for dynamic simulations. In Ciccotti, G. and Hoover, W., editors, *Molecular-Dynamics Simulation of Statistical-Mechanical Systems.* North-Holland, Amsterdam.

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

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics Letters B*, 195:216–222.

Efron, B. and Gong, G. (1983). A leisurely look at the bootstrap, the jacknife, and cross-validation. *The American Statistician*, 37(1):36–47.

Hoel, P. G. (1984). *Introduction to Mathematical Statistics.* John Wiley and Sons, Singapore.

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

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

MacKay, D. J. C. (1993). Hyperparameters: optimize, or integrate out. In Heidbreder, G., editor, *Maximum Entropy and Bayesian Methods*, pages 43–59. Kluwer Academic Publisher, Santa Barbara.

Metroplis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York. ISBN 0-387-94724-8.

Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical report, Department of Statistics, University of Toronto.

Penny, W. D. and Roberts, S. J. (1998). Bayesian neural networks for classification: how useful is the evidence? Neural Networks. to appear.

Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals*. Prentice Hall, Englewood Cliffs, New Jersey.

Ripley, B. D. (1994). Neural networks and related methods for classification. *Journal of the Royal Statistical Society B*, 56(3):409–456.

Roberts, S. J. and Penny, W. D. (1997). Novelty, confidence and errors in connectionist systems. Technical report, Department of Electrical and Electronic Engineering, Imperial College, London.

Robinson, A. J. (1989). *Dynamic Error Propagation Network*. PhD thesis, Cambridge University Engineering Department.

Sigillito, V. G., Wing, S. P., Hutton, L. V., and Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266.

Spyers-Ashby, J. M. (1996). *The recording and analysis of tremor in neurological disorders*. PhD thesis, Imperial College, London.

Appendix 1: The evidence approximation to Bayesian modelling

This section shows how the evidence scheme, introduced to the neural network community by MacKay (MacKay, 1992a), (MacKay, 1992b), can be viewed as an approximation to proper Bayesian sampling. The main idea of the evidence approach is to seek an analytic solution to equation (23),

$$P(y|\mathbf{x},\mathbf{D}) \quad = \quad \int P(y|\mathbf{w},\mathbf{x})P(\mathbf{w}|\boldsymbol{\alpha},\mathbf{D})P(\boldsymbol{\alpha}|\mathbf{D})d\mathbf{w}d\boldsymbol{\alpha}, \tag{52}$$

by introducing the following two approximations. First, the density $P(\boldsymbol{\alpha}|\mathbf{D})$ is assumed to be unimodal and sharply peaked about its mode $\hat{\boldsymbol{\alpha}}$, whereas the term $P(y|\mathbf{w},\mathbf{x})P(\mathbf{w}|\boldsymbol{\alpha},\mathbf{D})$ is assumed to vary smoothly. This is equivalent to the approximation of $P(\boldsymbol{\alpha}|\mathbf{D})$ by a delta function

$$P(\boldsymbol{\alpha}|\mathbf{D}) \quad = \quad \delta(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}) \tag{53}$$

and a collapse of the integral in (52):

$$P(y|\mathbf{x},\mathbf{D}) \quad = \quad \int P(y|\mathbf{w},\mathbf{x})P(\mathbf{w}|\hat{\boldsymbol{\alpha}},\mathbf{D})d\mathbf{w} \tag{54}$$

For a discussion of the validity of this step, the reader is referred to (MacKay, 1993). In the second simplification, the density $P(\mathbf{w}|\hat{\boldsymbol{\alpha}},\mathbf{D})$ is approximated by a multivariate Gaussian

$$P(\mathbf{w}|\hat{\boldsymbol{\alpha}},\mathbf{D}) \quad = \quad \sqrt{\frac{\det\mathbf{A}}{(2\pi)^W}}\exp\left[-\frac{1}{2}(\mathbf{w}-\hat{\mathbf{w}})^{\dagger}\mathbf{A}(\mathbf{w}-\hat{\mathbf{w}})\right]\exp\left[-U(\hat{\mathbf{w}})\right] \tag{55}$$

which, as a consequence of (8) and (9), is equivalent to a quadratic approximation of the total error function U,

$$U(\mathbf{w}) \;=\; U(\hat{\mathbf{w}}) + \frac{1}{2}(\mathbf{w}-\hat{\mathbf{w}})^{\dagger}\mathbf{A}(\mathbf{w}-\hat{\mathbf{w}}); \qquad \mathbf{A} = [\nabla\nabla^{\dagger}U]_{\hat{\mathbf{w}}}. \tag{56}$$

Together with a further simplification of the expression $P(y|\mathbf{w}, \mathbf{x})$, this allows an analytic solution of the integral in (54); see (MacKay, 1992b) for details. For a constant prior over $\boldsymbol{\alpha}$, the mode $\hat{\boldsymbol{\alpha}}$ of the posterior distribution $P(\boldsymbol{\alpha}|\mathbf{D})$ is equivalent to the maximum likelihood estimate

$$\hat{\boldsymbol{\alpha}} = \mathrm{argmax}_{\boldsymbol{\alpha}}\{P(\mathbf{D}|\boldsymbol{\alpha})\} \tag{57}$$

Note that this implies an integration over the weights $\mathbf{w}$, which as a result of the quadratic approximation (56) can be carried out analytically:

$$\begin{aligned} P(\mathbf{D}|\boldsymbol{\alpha}) &= \int P(\mathbf{D}|\mathbf{w})P(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w} \;=\; \int \exp\left[-U(\mathbf{w})\right] d\mathbf{w} \\ &= \exp\left[-U(\hat{\mathbf{w}})\right] \int \exp\left[\frac{1}{2}(\mathbf{w}-\hat{\mathbf{w}})^{\dagger}\mathbf{A}(\mathbf{w}-\hat{\mathbf{w}})\right] d\mathbf{w} \\ &= \sqrt{\frac{(2\pi)^W}{\det \mathbf{A}}}\exp\left[-U(\hat{\mathbf{w}})\right] \end{aligned} \tag{58}$$

The expression after the second equal sign results from equations (5), (7), and (9). The complete algorithm then consists of the following iteration scheme:

- Given $\boldsymbol{\alpha}$, find the weight vector $\hat{\mathbf{w}}$ which minimises the total error function $U(\mathbf{w})$.
- Given $\mathbf{w}$, find the hyperparameters $\hat{\boldsymbol{\alpha}}$ that maximise the so-called evidence
  $P(\mathbf{D}|\boldsymbol{\alpha})$, given by equation (58).

The scheme is iterated until a self-consistent solution $(\hat{\mathbf{w}}, \hat{\boldsymbol{\alpha}})$ has been found.

In a further step, the dependence of the prediction on the particular model can be removed. The conditional probability $P(y|\mathbf{x}, \mathbf{D})$ is formally obtained by integrating over the model space

$$\begin{aligned} P(y|\mathbf{x},\mathbf{D}) &= \int P(y,\mathcal{M}|\mathbf{x},\mathbf{D})d\mathcal{M} \;=\; \int P(y|\mathbf{x},\mathcal{M})P(\mathcal{M}|\mathbf{x},\mathbf{D})d\mathcal{M} \\ &= \int P(y|\mathbf{x},\mathcal{M})P(\mathcal{M}|\mathbf{D})d\mathcal{M}, \end{aligned} \tag{59}$$

where the last step results from the fact that the distribution of the input data $\mathbf{x}$ is not modelled by the network, and where we have made use of the fact that $P(y|\mathbf{x},\mathbf{D},\mathcal{M}) = P(y|\mathbf{x},\mathcal{M})$. In practice the integral in (59) is approximated by a sum over a network committee:

$$P(y|\mathbf{x},\mathbf{D}) \quad = \quad \sum_{i=1}^{N_{Com}} P(\mathcal{M}_i|\mathbf{D})P(y|\mathbf{x},\mathcal{M}_i). \tag{60}$$

Applying Bayes' rule and assuming a constant prior on the models, $P(\mathcal{M}) =$ const, gives

$$P(\mathcal{M}_i|\mathbf{D}) \;=\; \frac{P(\mathbf{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathbf{D})} \;=\; \frac{P(\mathbf{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{\sum_j P(\mathbf{D}|\mathcal{M}_j)P(\mathcal{M}_j)} \;=\; \frac{P(\mathbf{D}|\mathcal{M}_i)}{\sum_j P(\mathbf{D}|\mathcal{M}_j)}. \tag{61}$$

Define

$$\mathrm{Ev}_i \quad := \quad \ln P(\mathbf{D}|\mathcal{M}_i). \tag{62}$$

The sum in (60) can then be written as

$$P(y|\mathbf{x},\mathbf{D}) \quad = \quad \sum_{i=1}^{N_{Com}} c_i P(y|\mathbf{x},\mathcal{M}_i), \tag{63}$$

in which the weighting factors c_i are given by

$$c_i \quad = \quad \frac{\exp(\mathrm{Ev}_i)}{\sum_j \exp(\mathrm{Ev}_j)}. \tag{64}$$

The term in (62) is the so-called model evidence, which is formally given by integrating out the hyperparameters,

$$P(\mathbf{D}|\mathcal{M}_i) \;=\; \int P(\mathbf{D},\boldsymbol{\alpha}|\mathcal{M}_i)d\boldsymbol{\alpha} \;=\; \int P(\mathbf{D}|\boldsymbol{\alpha},\mathcal{M}_i)P(\boldsymbol{\alpha}|\mathcal{M}_i)d\boldsymbol{\alpha} \tag{65}$$

The first term in the integral on the right-hand side, $P(\mathbf{D}|\boldsymbol{\alpha}, \mathcal{M}_i)$, was already calculated above in equation (58), except that in the earlier notation the dependence on the prediction model $\mathcal{M}_i$ was not stated explicitly. The second term, $P(\boldsymbol{\alpha}|\mathcal{M}_i)$, is usually chosen constant. An analytic solution of (65) can be obtained when approximating $\ln P(\mathbf{D}|\boldsymbol{\alpha}, \mathcal{M}_i)$ by a quadratic function of $\boldsymbol{\alpha}$. Note that the overall approximation consists of two quadratic approximations, since the calculation of $P(\mathbf{D}|\boldsymbol{\alpha}, \mathcal{M}_i)$ in (58) was based on a quadratic approximation for the parameters $\mathbf{w}$. Explicit expressions for the model evidence can be found in (MacKay, 1992a), (MacKay, 1992b), and (Bishop, 1995), chapter 10.

Appendix 2: Statistical significance of the comparison between the simulations with and without ARD

6.1 Matched-pair t test

The comparison between the generalisation performance with and without ARD is based on $n = 8$ simulations, which differ with respect to the functional form of the hyperprior (determined by the hyper-hyperparameters μ and ν) and the length of the sampling time (100 or 200 configurations). In order to test the statistical significance of the results, we follow the approach of *matched pairs*, as described in (Hoel, 1984), section 5.11. Let u_i represent the result of a simulation without, and v_i the result of a simulation with ARD. The tupel (u_i, v_i) denotes a pair of results obtained under identical experimental conditions, that is, for an equivalent functional form of the hyperprior and the same sampling conditions. We then calculate the differences

$$d_i \quad := \quad v_i - u_i \tag{66}$$

as well as the empirical mean and standard deviation

$$\overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{67}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \overline{d})^2} \tag{68}$$

Denote the actual mean of d_i by μ. Under the assumption[9] that the d_i are independently and identically Gaussian distributed, the random variable

$$t := \frac{\overline{d} - \mu}{s/\sqrt{n}} \tag{69}$$

has a t *distribution* with $n-1$ degrees of freedom. Now define the hypothesis H_0 (the so-called *null hypothesis*) as

$$H_0 \quad : \quad \mu = 0 \tag{70}$$

that is, H_0 expresses the assumption that simulations with and without ARD lead to equivalent results. Under this hypothesis we have

$$t = \frac{\overline{d}}{s/\sqrt{n}} \tag{71}$$

The critical region for H_0 (that is, the values of t for which H_0 is to be rejected) is given by

$$\{t : \; |t| > t_c\}, \tag{72}$$

[9]This assumption does not hold exactly. First, it is not assured that the distribution of the d_i's is in fact Gaussian. Second, because the sampling intervals were not chosen distinct, the d_i's are not independent. However, the considerable difference in the lengths of the sampling intervals suggests that the stochastic dependence between different d_i's is only weak. In fact, for the tremor data we also calculated the t *statistic* for distinct sampling intervals – see section 6.2 – and obtained a very similar result. Moreover, it is known that moderate deviations from normality do not affect the distribution in (71) appreciably (see, e.g., (Hoel, 1984), section 5.10). This suggests that the application of the t test is justified.

where, for a 95% significance level, the so-called critical value t_c is defined by[10]

$$P(|t| < t_c) \quad = \quad 0.95 \tag{73}$$

and can be found in standard tables like (Hoel, 1984). The t *statistics* and the critical values for the various experiments carried out in this study are listed in Table 16. It is seen that most of the results are significant, that is, the *null hypothesis* H_0 is to be rejected. In the few cases where the results are *not* significant, we find that this only applies to one of the two performance measures: for Neal's data set, the difference in the performance between *ARD* and *no ARD* is not significant in terms of percent misclassification, but it is significant in terms of the negative log likelihood. For the whitened ionosphere data and the vowel data, this is the other way round. Hence the improvement or deterioration observed as as result of applying ARD is always significant with respect to at least one performance measure.

6.2 Bootstrapping

All the experiments performed in the present study were based on one particular randomly chosen set of training and test data. It is possible that different results would have been obtained with a different random choice of data. For the synthetic problems this can, in principle, be tested by repeating the simulations on new data sets generated from the known underlying process. For the more interesting real-world problems, however, this method is intractable because the underlying data-generating process is not known.

[10] Equation (73) indicates the critical region for a so-called *two-sided* test, where we test H_0 against the alternative hypothesis that *ARD* and *not using ARD* lead to *different* results. In fact, the results of the simulations allow the testing of a stricter alternative hypothesis, namely that ARD is either *better* or *worse* than not using ARD. This leads to a so-called *single-sided* test; see (Hoel, 1984) for details. In order to keep the discussion simple we do not consider this distinction here. The justification is that the critical value for a two-sided test is larger than for a single-sided test, that is, when using a two-sided test we are more conservative and less likely to accept our results as being significant.

To get an estimate of the variability of the results as a consequence of resampling the data, we resorted to bootstrapping and drew N_{test} exemplars with replacement from the tremor test set (where N_{test} denotes the number of test exemplars). The generalisation performance was then estimated on the modified test data, and this procedure was repeated several times for different bootstrap samples. In every case the t *statistic* based on the matched-pair test was calculated as before. The only difference was that for estimating the network performance according to (12), we chose distinct time intervals $201 \le t \le 300$ and $301 \le t \le 400$ rather than overlapping intervals $201 \le t \le 400$ and $301 \le t \le 400$. As discussed before, this improves the reliability of the t-test, although the difference in the results was not large. The results thus obtained are shown in Table 17. It is seen that, except for two bootstrap samples, the *ARD* performance is always significantly worse than the performance *without ARD*. More formally, we can apply the standard bootstrap procedure for bias correction and variance estimation. Let t denote the t *statistic* obtained from the unmanipulated data, and $\bar{t}_{Boot}$ the average t *statistic* obtained from the bootstrap samples. Then the bias estimate is given by the difference $\bar{t}_{Boot} - t$, which is to be subtracted from t to give the true value[11]

$$t_{true} \quad = \quad 2t - \bar{t}_{Boot} \tag{74}$$

The standard deviation is estimated by the standard deviation over the bootstrap samples. Following this procedure we obtain:

- Percent misclassification: 4.12 ± 1.01
- Negative log likelihood: $t = 6.72 \pm 1.45$

[11] See, for instance, (Efron and Gong, 1983).

Both values are significantly larger than the critical value of $t_c = 2.37$, which suggests that the deterioration observed when applying ARD is not a consequence of any chance correlations or particular characteristics of the test data, but a significant fact inherent to the ARD scheme.

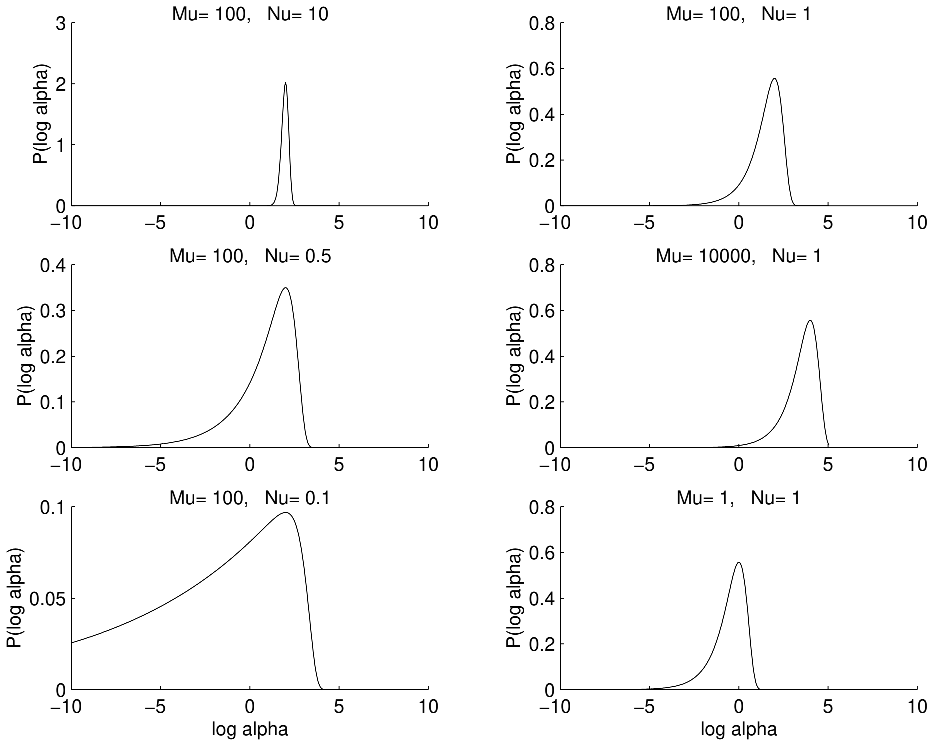

Figure 1: Gamma prior for weight decay parameters. The graphs show different prior distributions for the weight decay parameter α ($\log_{10}$ scale). Note that μ determines the location of the distribution, whereas ν determines its width. An increase of ν reduces the vagueness of the prior.

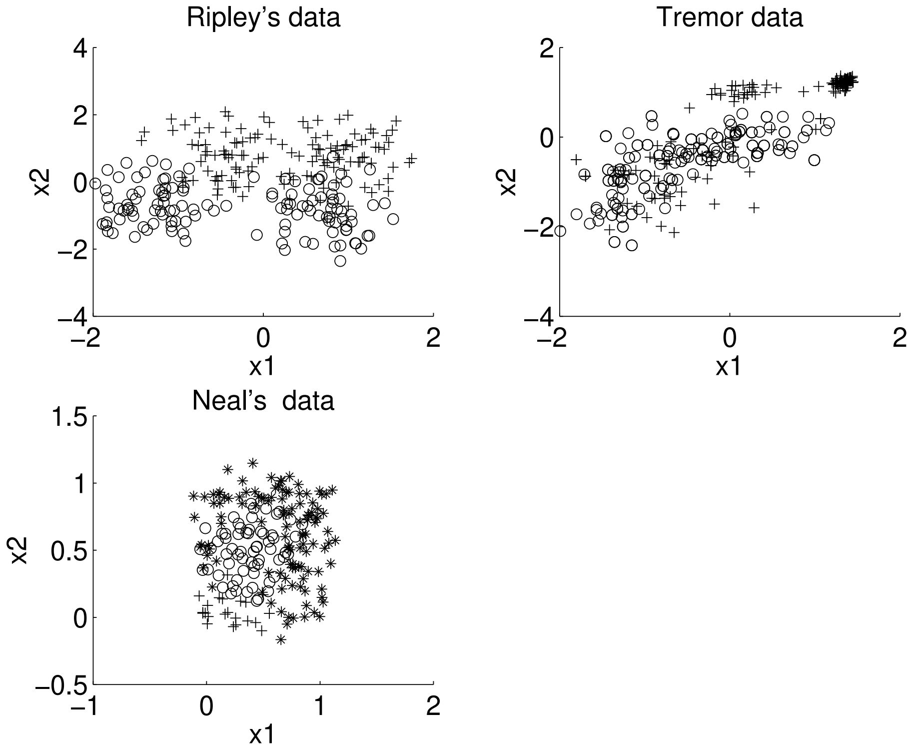

Figure 2: Classification problems. The plots show some of the classification problems applied in the present study (those depending only on two input variables). Top left: Ripley's synthetic data set. Top right: Tremor data. Bottom: Neal's three-way classification problem.

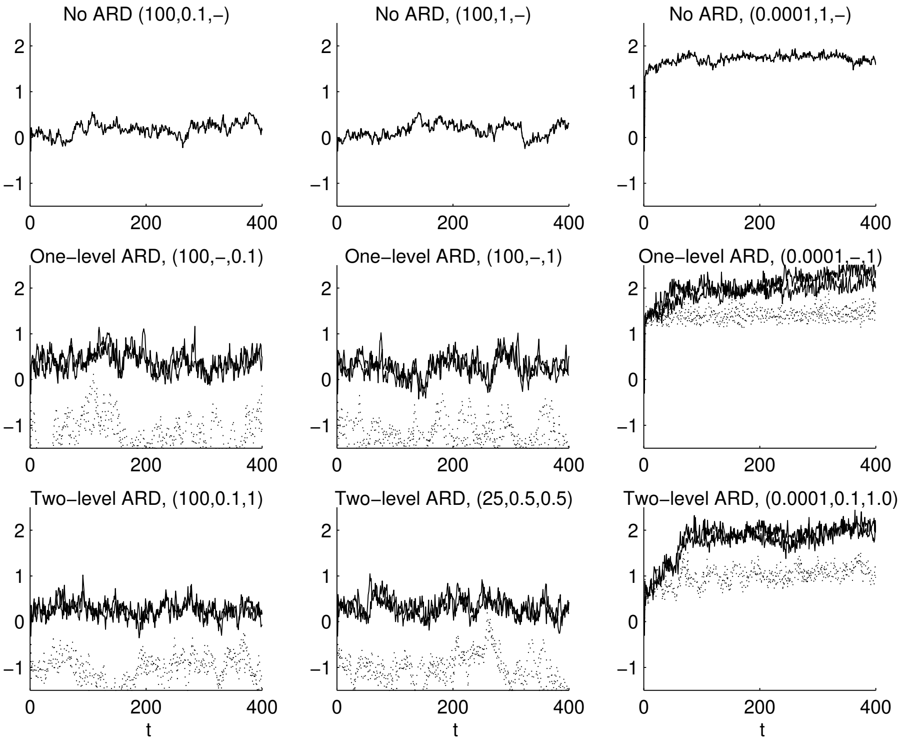

Figure 3: Hyperparameters for the synthetic three-way classification problem. The graphs show the evolution of the square root of the inverse weight decay parameters, $\sqrt{(\frac{1}{\alpha_g})}$ for each of the input-to-hidden layer weight groups ($\log_{10}$ scale). For the plots in the top row no ARD was used, so all hyperparameters are the same. Rows in the middle row were obtained with a one-level ARD scheme, whereas those in the bottom row result from application of a two-level scheme. In both cases, solid lines refer to relevant and dotted lines to irrelevant inputs. The graphs in the right column correspond to a rather unrealistic hyperprior which supports very small values of $\alpha_g \approx 10^{-4}$; see text for details. Note the similarities of all these graphs shown here with those of Figure 4. The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\text{high}}, \nu_{\text{low}})$.

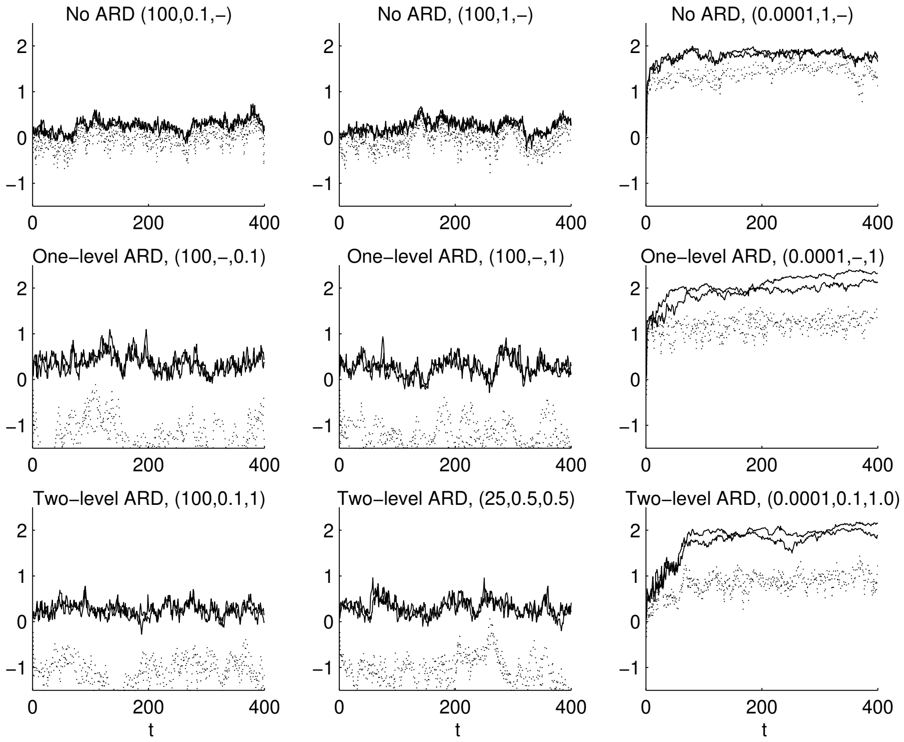

Figure 4: Weight magnitude for the synthetic three-way classification problem. The graphs show $\sqrt{\langle w^2 \rangle_g}$, the square root of the average square magnitude of weights on connections out of each of the four input units ($\log_{10}$ scale). Solid lines refer to relevant inputs, dotted lines to irrelevant ones. The graphs in the upper row show the magnitudes when ARD is not used, graphs in the middle row were obtained with a one-level ARD scheme, whereas for the graphs in the bottom row a two-level ARD scheme was applied. It is clearly noticed that the application of ARD results in a stronger suppression of the irrelevant inputs. The graphs in the right column were obtained with an unrealistic hyperprior, as discussed in the text. The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\text{high}}, \nu_{\text{low}})$.

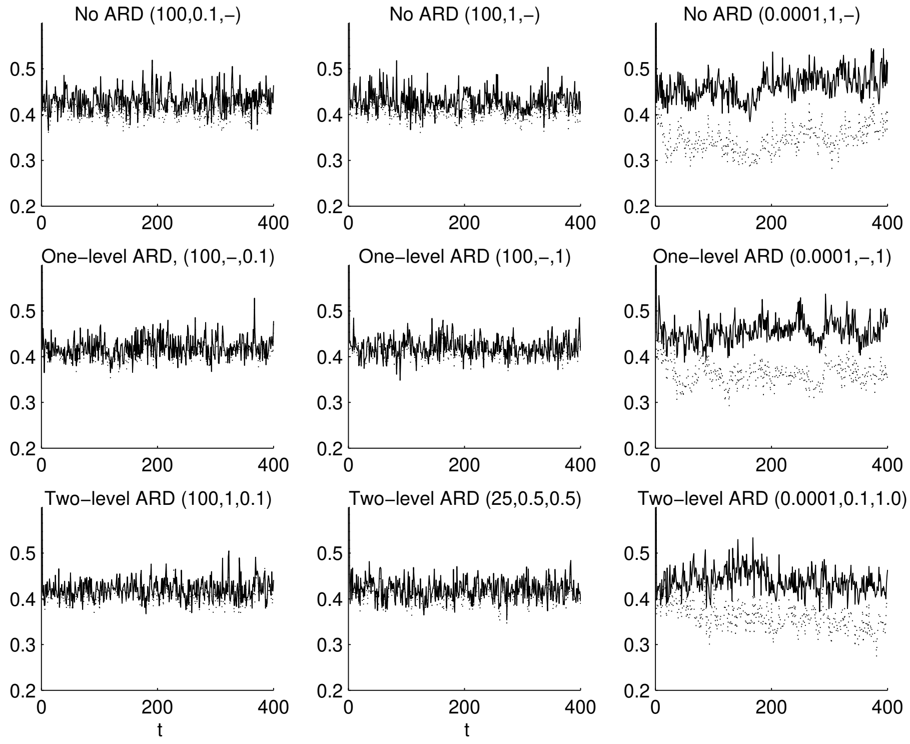

Figure 5: Prediction error for the synthetic three-way classification problem. The graphs show, on a $\log_{10}$ scale, the average absolute deviation between the target vector (1-of-K coding) and the vector of network outputs. Solid lines show the performance on the test set, dashed lines that on the training set. Note that the deviation between these two errors only becomes noticeable (overfitting!) when choosing an unrealistic hyperprior that assumes unrealistically small values for the weight decay constants (right column). The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\text{high}}, \nu_{\text{low}})$.

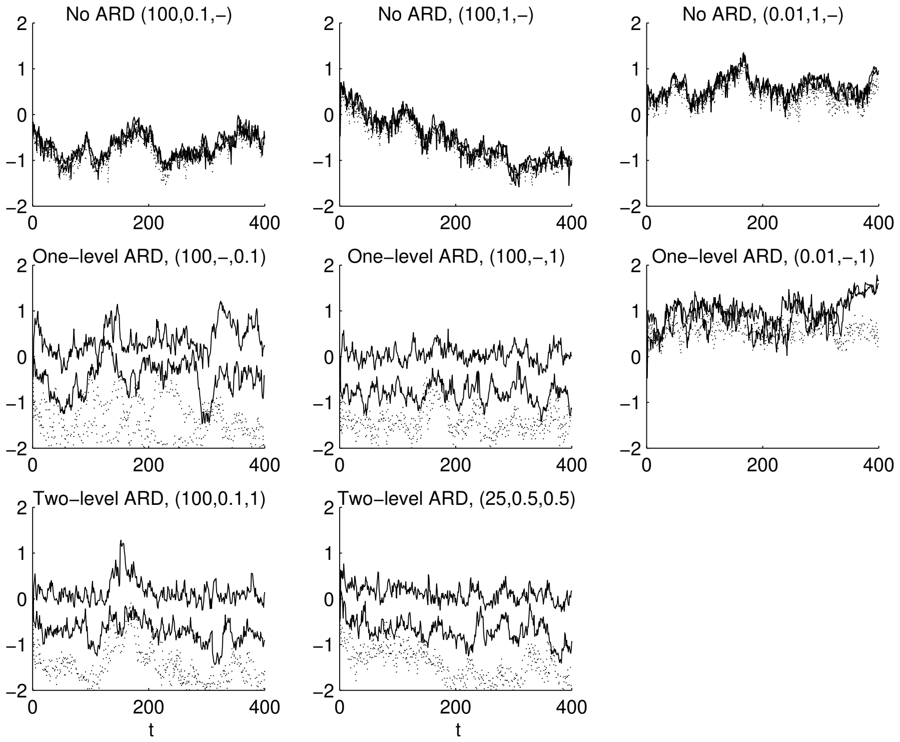

Figure 6: Weight magnitude for Ripley's synthetic classification problem. The graphs show $\sqrt{\langle w^2 \rangle_g}$, the square root of the average square magnitude of weights on connections out of each of the four input units ($\log_{10}$ scale). Solid lines refer to relevant inputs, dotted lines to irrelevant ones. The graphs in the upper row show the magnitudes when ARD is not used, graphs in the middle row were obtained with a one-level ARD scheme, whereas for the graphs in the bottom row a two-level ARD scheme was applied. It is clearly noticed that without ARD all weights tend to have a similar size, whereas the application of ARD significantly reduces the magnitude of the weights belonging to the irrelevant inputs. The different columns in the first two rows represent different vaguenesses of the hyperprior $P(\alpha)$. Left column: vague hyperprior. Middle column: informative hyperprior, supporting large weight-decay hyperparameters about $\alpha = 100$. Right column: informative hyperprior, supporting small weight-decay hyperparameters about $\alpha = 0.01$. Note that this difference between the weight decay constants is reflected in the average weight size. The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\text{high}}, \nu_{\text{low}})$.

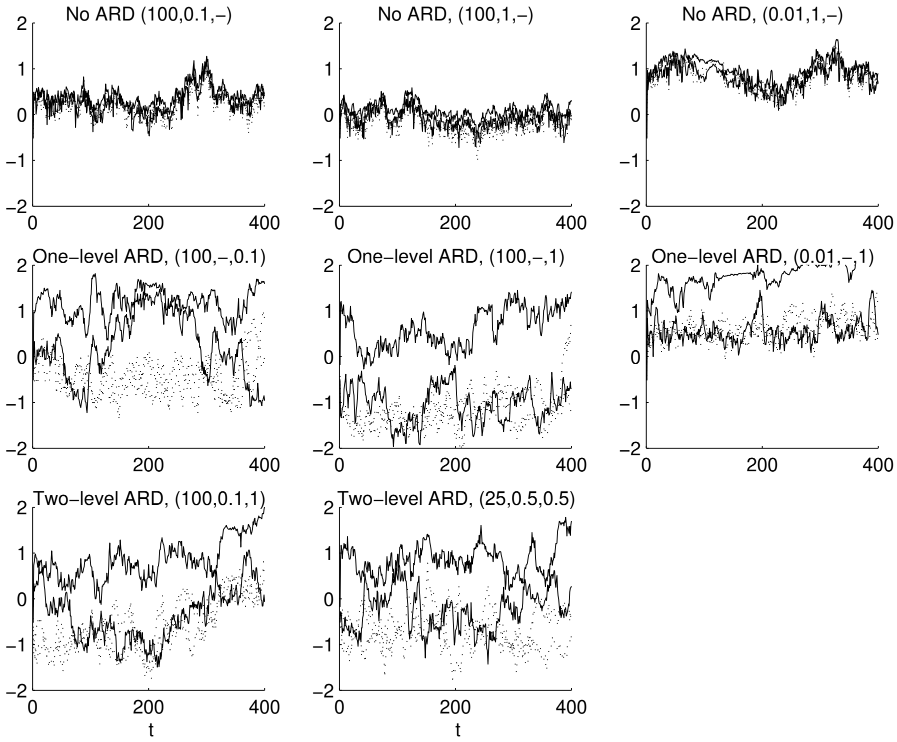

Figure 7: Weight magnitude for the tremor data. The graphs show $\sqrt{\langle w^2 \rangle_g}$, the square root of the average square magnitude of weights on connections out of each of the four input units ($\log_{10}$ scale). Solid lines refer to relevant inputs, dotted lines to irrelevant ones. The graphs in the upper row show the magnitudes when ARD is not used, graphs in the middle row were obtained with a one-level ARD scheme, whereas for the graphs in the bottom row a two-level ARD scheme was applied. For further details, see the caption of Figure 6. Note that for this data the ARD scheme leads to a significant differentiation between the two relevant inputs. The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\mathrm{high}}, \nu_{\mathrm{low}})$.

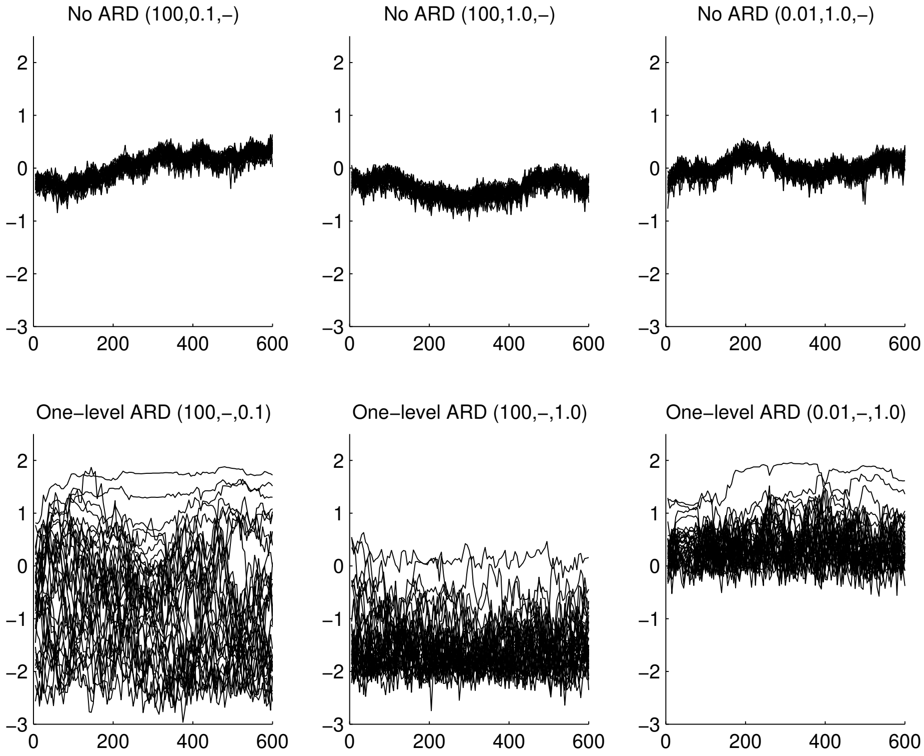

Figure 8: Weight magnitude for the ionosphere data. The graphs show $\sqrt{\langle w^2 \rangle_g}$, the square root of the average square magnitude of weights on connections out of each of the 34 input units ($\log_{10}$ scale). Top row: No ARD, the differences between the trajectories of different weight groups are only small. Bottom row: One-level ARD scheme. The weight-magnitude spectrum is significantly broadened. Note that the spectrum is wider for the vague hyperprior $P(\alpha)$ (bottom row, left) than for the informative hyperprior (bottom row, middle and right). The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\rm high}, \nu_{\rm low})$.

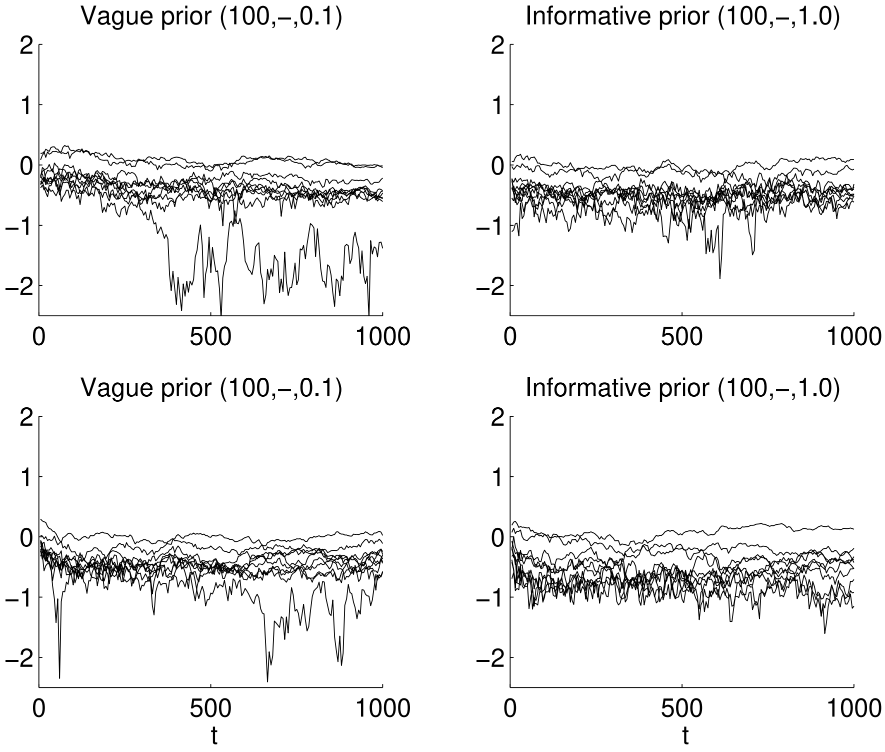

Figure 9: Weight magnitude for the vowel data. The graphs show $\sqrt{\langle w^2 \rangle_g}$, the square root of the average square magnitude of weights on connections out of each of the 10 input units ($\log_{10}$ scale). The graphs in the left column were obtained with a vague hyperprior, those in the right with a more informative one. The triplets over the graphs indicate the hyper-hyperparameters $(\mu/n, \nu_{\mathrm{high}}, \nu_{\mathrm{low}})$. Two different sampling schemes were applied. Top: the standard method described in the text. Bottom: a slightly modified version using momentum persistence (described in (Neal, 1996)).

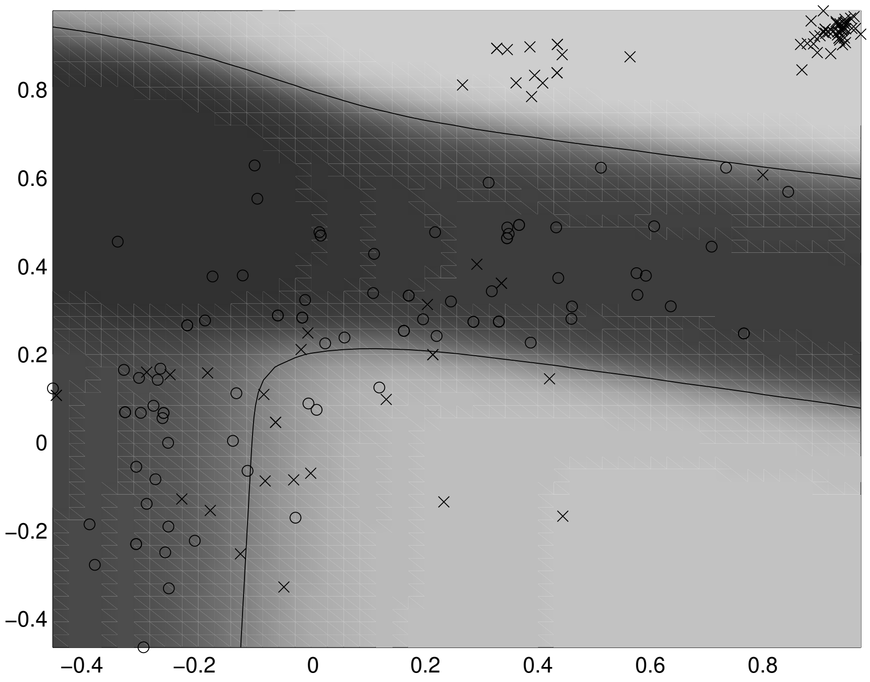

Figure 10: Classification boundaries for the tremor data. Three regimes in the data space can be distinguished: (i) a cluster of '+'-exemplars *above* the upper decision boundary, (ii) a mixed cluster dominated by 'o'-exemplars *between* the two decision boundaries (shown in black), (iii) a mixed cluster dominated by '+'-exemplars *below* the lower decision boundary.

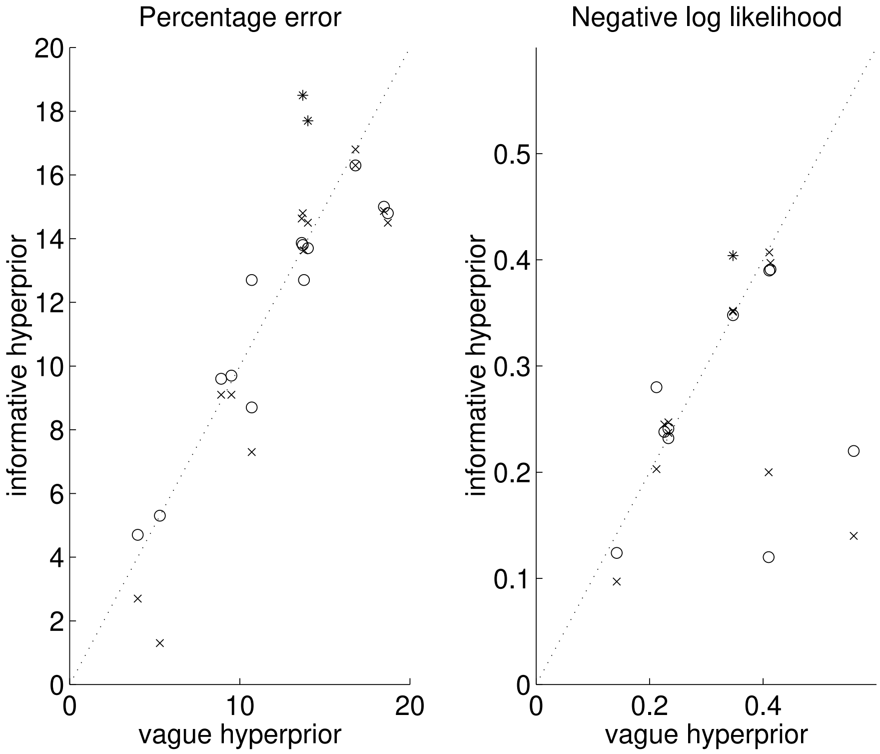

Figure 11: Dependence of the network prediction on the vagueness of the hyperprior. The graphs show two scatter plots of the estimated generalisation performance obtained with a vague hyperprior ($\nu = 0.1$) against that obtained with a more informative hyperprior ($\nu = 1.0$). The results are taken from simulations with a one-level ARD scheme, applied to all problems of this study. *Left:* Generalisation performance measured in percent misclassification. *Right:* Generalisation performance in terms of the negative log likelihood of the true class. Different informative hyperpriors were employed. *Circles:* hyperprior supporting large weight decay parameters ($\mu/n = 100$); *crosses:* hyperprior supporting medium ($\mu/n = 1$) or small ($\mu/n = 0.01$) weight decay parameters; *stars:* hyperprior supporting extremely small weight decay parameters ($\mu/n = 0.0001$, only used in one simulation). Values above the dotted line indicate that the performance is improved by applying a vague hyperprior, values below the dotted line point to a deterioration as a consequence of using a vague hyperprior. The values for the vowel data were scaled (by a factor of 0.3 in the left plot, and by a factor of 0.1 in the right plot) to fit into the scale.

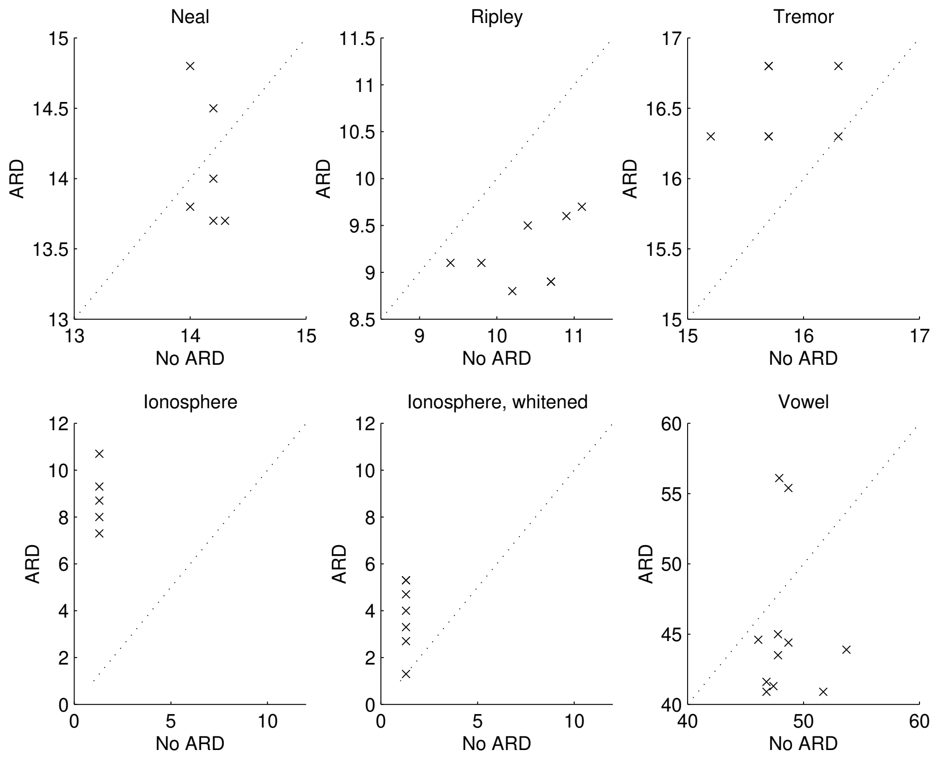

Figure 12: Comparison of the classification results obtained with and without ARD. The figures show scatter plots of the percent misclassification generalisation error without ARD against that obtained with a one-level ARD scheme. The hyper-hyperparameters μ and ν for two corresponding simulations were identical, with the difference that without ARD, these hyper-hyperparameters determine the prior distribution of one weight-decay hyperparameter α common to all input-to-hidden layer weights, whereas for the one-level ARD scheme, different weight-decay hyperparameters α_g are drawn from separate distributions (which have, however, the same hyper-hyperparameters μ and ν). Values below the dotted line indicate a performance enhancement by applying ARD, values above the dotted line indicate a deterioration as a result of applying ARD. *Top left:* Neal's synthetic three-way classification problem; *top middle:* Ripley's synthetic problem; *top right:* tremor data; *bottom left:* ionosphere data; *bottom middle:* ionosphere data after whitening; *bottom right:* vowel data.

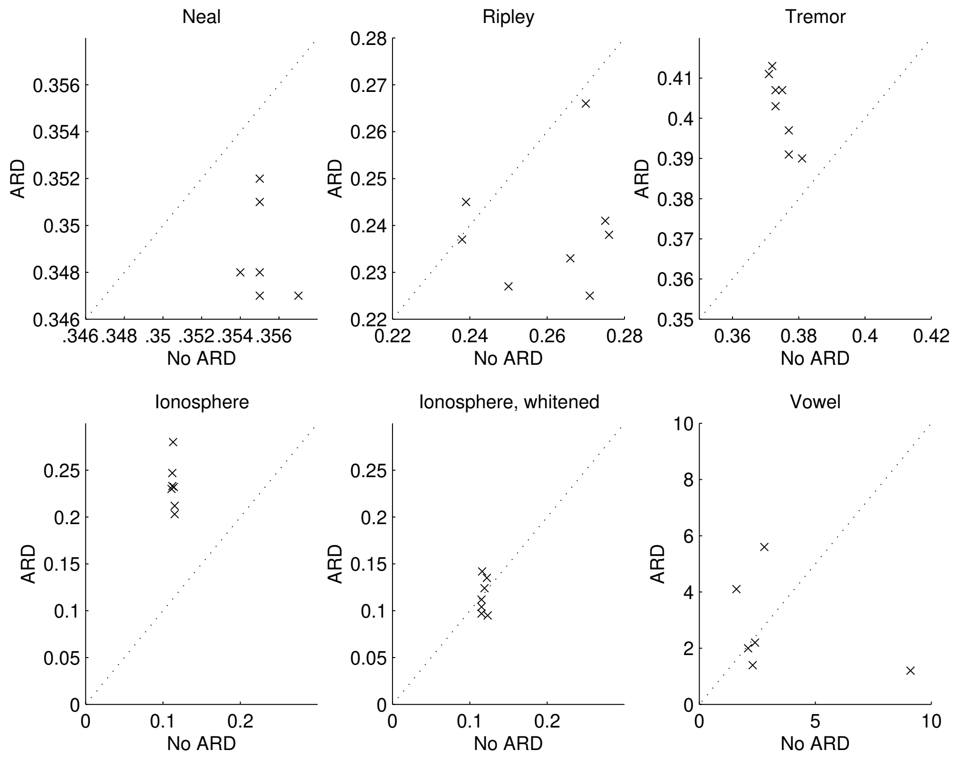

Figure 13: Comparison of the classification results obtained with and without ARD. Similar to Figure 12, the graphs show scatter plots of the estimated generalisation error without ARD against that obtained with a one-level ARD scheme, where the generalisation error is now measured in terms of the negative log likelihood of the true class. Values below the dotted line indicate a performance enhancement by applying ARD, values above the dotted line indicate a deterioration as a result of applying ARD. As in Figure 12, the plots refer to the following data sets: *Top left:* Neal's synthetic three-way classification problem; *top middle:* Ripley's synthetic problem; *top right:* tremor data; *bottom left:* ionosphere data; *bottom middle:* ionosphere data after whitening; *bottom right:* vowel data.

Problem	Characterisation		Number of inputs		Classes	Number of exemplars	
	synthetic	real	total	relevant		training	testing
Neal	•		4	2	3	400	600
Ripley	•		4	2	2	250	1000
Tremor		•	4	2	2	179	178
Ionosphere		•	34	unknown	2	200	150
Vowel		•	10	unknown	11	528	462

Table 1: Overview of the classification problems used in this study.

Data	Bayesian sampling, MCMC	Conventional training, comittees
Neal	2 hours	30 minutes
Ripley	1 hour 20 minutes	14 minutes
Tremor	40 minutes	10 minutes
Ionosphere	2 hours 30 minutes	1 hour
Vowel	27 hours	41 hours

Table 2: Typical simulation times for Bayesian sampling with MCMC (second column) and the conventional approach of training a neural network committee (third column). In the latter case, different architectures were employed (Ripley, tremor, and ionosphere data: 2,3,4, and 5 hidden units; Neal's data: 3,4,5,6, and 8 hidden nodes, vowel data: 3, 5, 7, 9, 11, 13, 15, and 22 hidden units), and for each architecure ten different initialisations were generated. The optimisation was done with conjugate gradients, where the hyperparameters were updated in regular intervals according to the evidence scheme. All simulations were carried out on a SUN ULTRA 1 workstation with programs written in C.

Problem	2 hidden nodes		4 hidden nodes		6 hidden nodes		8 hidden nodes	
Ripley	10.1	9.8	10.2	10.1	9.8	10.0	9.3	9.3
Tremor	16.9	16.3	16.9	16.3	15.7	16.9	15.7	16.3
Ionosphere	–	–	8.7	10.0	10.0	10.7	8.7	8.0

Table 3: Dependence of the classification performance on the network complexity. The table shows the classification error (in percentage) for different numbers of hidden units. The values in the left columns were obtained by averaging over the trajectories of the last 100 configurations, the values in the right columns by averaging over the final 200 configurations. For all simulations, two-level ARD hyperpriors with $\mu/n = 100$, $\nu_{\text{high}} = 0.5$ and $\nu_{\text{low}} = 1.0$ were applied.

Characterisation of simulation	Hyperprior μ/n	ν_{high}	ν_{low}	Classification error last 200	last 100
No ARD, vague hyperprior	100	0.1	-	14.3	14.2
No ARD, informative hyperprior	100	1.0	-	14.0	14.2
No ARD, informative hyperprior	1.0	1.0	-	14.0	14.2
No ARD, informative hyperprior	0.0001	1.0	-	18.0	17.7
ARD1, vague hyperprior	100	-	0.1	13.7	14.0
ARD1, informative hyperprior	100	-	1.0	13.8	13.7
ARD1, informative hyperprior	1.0	-	1.0	14.8	14.5
ARD1, informative hyperprior	0.0001	-	1.0	18.5	17.8
ARD2	100	0.1	1.0	14.0	13.5
ARD2	25	0.5	0.5	14.3	13.8
ARD2	0.0001	0.1	1.0	17.0	16.3

Table 4: Percent misclassification on the test set for the synthetic three-way classification problem. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	last 200	last 100
No ARD, vague hyperprior	100	0.1	-	10.7	10.4
No ARD, informative hyperprior	100	1.0	-	10.9	11.1
No ARD, informative hyperprior	1.0	1.0	-	10.2	10.7
No ARD, informative hyperprior	0.01	1.0	-	9.4	9.8
ARD1, vague hyperprior	100	-	0.1	8.9	9.5
ARD1, informative hyperprior	100	-	1.0	9.6	9.7
ARD1, informative hyperprior	1.0	-	1.0	8.8	8.9
ARD1, informative hyperprior	0.01	-	1.0	9.1	9.1
ARD2	100	0.1	1.0	9.3	9.3
ARD2	25	0.5	0.5	9.6	9.6

Table 5: Percent misclassification on the test set for Ripley's synthetic binary classification problem. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	last 200	last 100
No ARD, vague hyperprior	100	0.1	-	15.7	16.3
No ARD, informative hyperprior	100	1.0	-	15.7	15.7
No ARD, informative hyperprior	1.0	1.0	-	16.3	15.7
No ARD, informative hyperprior	0.01	1.0	-	15.7	15.2
ARD1, vague hyperprior	100	-	0.1	16.8	16.8
ARD1, informative hyperprior	100	-	1.0	16.3	16.3
ARD1, informative hyperprior	1.0	-	1.0	16.3	16.3
ARD1, informative hyperprior	0.01	-	1.0	16.8	16.3
ARD2	100	0.1	1.0	16.3	16.3
ARD2	25	0.5	0.5	16.3	16.3

Table 6: Percent misclassification on the test set for the tremor data. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of simulation	Hyperprior μ/n	ν_{high}	ν_{low}	Standard 200	100	Whitened 200	100
No ARD, vague hyperprior	100	0.1	-	1.3	1.3	1.3	1.3
No ARD, informative hyperprior	100	1.0	-	1.3	1.3	1.3	1.3
No ARD, informative hyperprior	1.0	1.0	-	1.3	1.3	1.3	1.3
No ARD, informative hyperprior	0.01	1.0	-	1.3	1.3	1.3	1.3
ARD1, vague hyperprior	100	-	0.1	10.7	10.7	4.0	5.3
ARD1, informative hyperprior	100	-	1.0	12.7	8.7	4.7	5.3
ARD1, informative hyperprior	1.0	-	1.0	8.0	9.3	3.3	3.3
ARD1, informative hyperprior	0.01	-	1.0	7.3	7.3	2.7	1.3
ARD2	100	0.1	1.0	9.3	12.0	3.3	5.3
ARD2	25	0.5	0.5	7.3	10.7	6.0	8.0

Table 7: Percent misclassification on the test set for the ionosphere data. Results are shown for the standard data (input variables normalised seperately) and for whitened data (that is, the data was subjected to a Karhunen-Loeve transform). Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of the simulation	Hyperprior			Classification error		Classification error	
	μ/n	ν_{high}	ν_{low}	last 500	last 100	last 500	last 100
No ARD, vague hyperprior	100	0.1	-	47.9	48.7	47.4	46.8
No ARD, informative hyperprior	100	1.0	-	48.7	47.8	46.8	46.8
No ARD, informative hyperprior	1.0	1.0	-	47.8	46.1	51.7	53.7
ARD1, vague hyperprior	100	-	0.1	56.1	55.4	41.3	40.9
ARD1, informative hyperprior	100	-	1.0	44.4	45.0	38.1	41.6
ARD1, informative hyperprior	1.0	-	1.0	43.5	44.6	40.9	43.9
ARD2	100	0.1	1.0	44.2	45.0	33.8	38.5
ARD2	25	0.5	0.5	45.2	46.5	35.7	35.9

Table 8: Percent misclassification on the test set for the vowel classification problem. Results are stated for (i) the final 500 configurations and (ii) the last 100 configurations. The simulations were repeated twice, with the results on the right obtained from a slightly modified sampling scheme using momentum persistence (see (Neal, 1996)).

Characterisation of simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	last 200	last 100
No ARD, vague hyperprior	100	0.1	-	0.355	0.357
No ARD, informative hyperprior	100	1.0	-	0.354	0.355
No ARD, informative hyperprior	1.0	1.0	-	0.355	0.355
No ARD, informative hyperprior	0.0001	1.0	-	0.417	0.417
ARD1, vague hyperprior	100	-	0.1	0.347	0.347
ARD1, informative hyperprior	100	-	1.0	0.348	0.348
ARD1, informative hyperprior	1.0	-	1.0	0.351	0.352
ARD1, informative hyperprior	0.0001	-	1.0	0.404	0.404
ARD2	100	0.1	1.0	0.349	0.350
ARD2	25	0.5	0.5	0.349	0.347
ARD2	0.0001	0.1	1.0	0.395	0.407

Table 9: Generalisation performance, measured as the negative log likelihood of the correct class, for the synthetic three-way classification problem. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	last 200	last 100
No ARD, vague hyperprior	100	0.1	-	0.270	0.266
No ARD, informative hyperprior	100	1.0	-	0.276	0.275
No ARD, informative hyperprior	1.0	1.0	-	0.250	0.271
No ARD, informative hyperprior	0.01	1.0	-	0.239	0.238
ARD1, vague hyperprior	100	-	0.1	0.226	0.233
ARD1, informative hyperprior	100	-	1.0	0.238	0.241
ARD1, informative hyperprior	1.0	-	1.0	0.227	0.225
ARD1, informative hyperprior	0.01	-	1.0	0.245	0.237
ARD2	100	0.1	1.0	0.227	0.228
ARD2	25	0.5	0.5	0.235	0.237

Table 10: Generalisation performance, measured as the negative log likelihood of the correct class, for Ripley's synthetic binary classification problem. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	last 200	last 100
No ARD, vague hyperprior	100	0.1	-	0.371	0.372
No ARD, informative hyperprior	100	1.0	-	0.381	0.377
No ARD, informative hyperprior	1.0	1.0	-	0.373	0.373
No ARD, informative hyperprior	0.01	1.0	-	0.375	0.377
ARD1, vague hyperprior	100	-	0.1	0.411	0.413
ARD1, informative hyperprior	100	-	1.0	0.390	0.391
ARD1, informative hyperprior	1.0	-	1.0	0.407	0.403
ARD1, informative hyperprior	0.01	-	1.0	0.407	0.397
ARD2	100	0.1	1.0	0.395	0.397
ARD2	25	0.5	0.5	0.386	0.378

Table 11: Generalisation performance, measured as the negative log likelihood of the correct class, for the tremor data. Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

ARD scheme	Hyperprior μ/n	ν_{high}	ν_{low}	Standard 200	100	Whitened 200	100
No ARD, vague hyperprior	100	0.1	-	0.115	0.112	0.116	0.119
No ARD, informative hyperprior	100	1.0	-	0.113	0.114	0.119	0.122
No ARD, informative hyperprior	1.0	1.0	-	0.108	0.111	0.115	0.115
No ARD, informative hyperprior	0.01	1.0	-	0.115	0.112	0.115	0.123
ARD1, vague hyperprior	100	-	0.1	0.212	0.233	0.142	–
ARD1, informative hyperprior	100	-	1.0	0.280	0.232	0.124	0.135
ARD1, informative hyperprior	1.0	-	1.0	–	0.230	0.104	0.112
ARD1, informative hyperprior	0.01	-	1.0	0.203	0.247	0.097	0.095
ARD2	100	0.1	1.0	0.241	0.474	0.124	0.135
ARD2	25	0.5	0.5	0.237	0.530	0.145	0.181

Table 12: Generalisation error, measured as the negative log likelihood of the correct class, for the ionosphere data. Results are shown for the standard data (input variables normalised seperately) and for whitened data (input vectors subjected to a Karhunen-Loeve transform). Results are stated for (i) the final 200 configurations and (ii) the last 100 configurations.

Characterisation of the simulation	Hyperprior			Classification error	
	μ/n	ν_{high}	ν_{low}	A	B
No ARD, vague hyperprior	100	0.1	-	2.8	1.6
No ARD, informative hyperprior	100	1.0	-	2.4	9.1
No ARD, informative hyperprior	1.0	1.0	-	2.3	2.1
ARD1, vague hyperprior	100	-	0.1	5.6	4.1
ARD1, informative hyperprior	100	-	1.0	2.2	1.2
ARD1, informative hyperprior	1.0	-	1.0	1.4	2.0
ARD2	100	0.1	1.0	2.6	1.0
ARD2	25	0.5	0.5	3.1	1.1

Table 13: Test-set performance for the vowel classification problem, measured as the negative log likelihood for the correct class. The results were obtained from the final 500 configurations of the trajectories. Two different sampling schemes were applied: the standard method described in the text (A), and a slightly modified version using momentum persistence (B, see (Neal, 1996)).

Data	Hyperparameters (μ/n, ν_{high}, ν_{low})	ARD (normal)	ARD (long)	no ARD (normal)
Tremor	(100,-,0.1)	0.411	0.378	0.371
Tremor	(100,-,1.0)	0.390	0.386	0.381
Tremor	(1.0,-,1.0)	0.407	0.391	0.373
Ionosphere	(1.0,-,1.0)	0.230	0.206	0.108
Ionosphere	(0.01,-,1.0)	0.203	0.181	0.115

Table 14: Results for the tremor and ionosphere data for longer simulation times. The fourth column (*'ARD long'*) shows the estimated generalisation error (in terms of the negative log likelihood) obtained with ARD after a substantial increase of the equilibration and sampling periods. For the tremor data, the simulation time was increased by a factor of 25 (that is, 25 times 400 hybrid Monte-Carlo trajectories over 1000 leapfrog steps each were computed), for the ionosphere data the simulation time was increased by a factor of 50 (that is, 50 times 600 hybrid Monte-Carlo trajectories over 1000 leapfrog steps each were computed). The first half of the configurations were discarded (equilibration period). The columns entitled *'ARD normal'* and *'no ARD normal'* repeat the results of the earlier simulations with 'normal' simulation times.

Data set	Bayesian sampling (MCMC)			Evidence scheme		Other methods
	no ARD	ARD 1	ARD 2	single	committee	
Neal	14.2	14.25	14.15	14.6	15.3	–
Tremor	15.7	16.3	16.3	16.3	15.7	15.5
Ripley	10.5	9.1	9.4	9.3	9.3	9.4
Ionosphere	1.3	3.7	5.7	7.3	7.3	7.3
Vowel	47.6	43.7	41.4	70.1	47.8	49.0

Table 15: Comparison of percent misclassification obtained with Bayesian sampling, the evidence scheme and other, non-Bayesian methods. Three different values are stated for Bayesian sampling. *No ARD:* median value of simulations without ARD; *ARD 1:* median of simulations with a one-level ARD prior; *ARD 2:* median of simulations with a two-level ARD prior. For the evidence scheme, two values are shown. *Single:* classification performance of the network with the largest value for the evidence; *committee:* classification error of a committee of networks, selected on the basis of the evidence. The term *'other methods'* refers to various alternative non-Bayesian methods briefly described in the text.

Data Set	Criterion	$n-1$	t	t_c	Reject H_0	Result
Neal	class error	7	0.15	±2.37	no	undecided
	likelihood	7	-5.99	±2.37	yes	ARD better
Ripley	class error	7	-6.41	±2.37	yes	ARD better
	likelihood	7	-3.88	±2.37	yes	ARD better
Tremor	class error	7	5.14	±2.37	yes	ARD worse
	likelihood	7	6.55	±2.37	yes	ARD worse
Ionosphere	class error	7	5.01	±2.37	yes	ARD worse
(whitened)	likelihood	6	-0.33	±2.45	no	undecided
Vowel	class error	11	-2.22	±2.20	yes	ARD better
	likelihood	5	-0.40	±2.57	no	undecided

Table 16: t statistic. The 4th column shows, for the data sets listed in column 1, the *t statistics* of both performance measures (percent misclassification and negative log likelihood, as indicated in column 2). The 3rd column shows the number of degrees of freedom, which determines the critical value t_c (given in column 5). Negative values for the *t statistic* indicate that ARD achieves an improvement, which is significant (at a 95% confidence level) if $t < -t_c$. Positive values for the *t statistic* suggest that the application of ARD results in a deterioration of the performance, which is significant (at a 95% confidence level) if $t > t_c$. If $-t_c < t < t_c$, the null hypothesis H_0 is to be accepted, that is, *ARD* achieves the same performance as *not using ARD*.

Data Set	Criterion	t	t_c	Reject H_0	Result
Original	class error	3.74	2.37	yes	ARD worse
	neg. log likelihood	5.48	2.37	yes	ARD worse
Bootstrap 1	class error	3.05	2.37	yes	ARD worse
	neg. log likelihood	5.40	2.37	yes	ARD worse
Bootstrap 2	class error	3.28	2.37	yes	ARD worse
	neg. log likelihood	4.95	2.37	yes	ARD worse
Bootstrap 3	class error	3.33	2.37	yes	ARD worse
	neg. log likelihood	4.52	2.37	yes	ARD worse
Bootstrap 4	class error	4.61	2.37	yes	ARD worse
	neg. log likelihood	4.96	2.37	yes	ARD worse
Bootstrap 5	class error	2.16	2.37	no	undecided
	neg. log likelihood	3.44	2.37	yes	ARD worse
Bootstrap 6	class error	3.41	2.37	yes	ARD worse
	neg. log likelihood	4.77	2.37	yes	ARD worse
Bootstrap 7	class error	2.38	2.37	yes	ARD worse
	neg. log likelihood	0.64	2.37	no	undecided
Bootstrap 8	class error	2.76	2.37	yes	ARD worse
	neg. log likelihood	4.78	2.37	yes	ARD worse
Bootstrap 9	class error	5.29	2.37	yes	ARD worse
	neg. log likelihood	4.72	2.37	yes	ARD worse

Table 17: Bootstrapping the tremor test set. The third column shows the t *statistics* of the classification error and the negative log likelihood for different bootstrap samples. The critical value t_c is shown in the fourth column. The null hypothesis H_0 – which assumes equal performance with and without ARD – is to be rejected (at a 95% confidence level) when $t > t_c$. It is seen that except for two bootstrap samples this is always the case.