This technical report is a close-to-final draft of a paper which was published in
Neural Networks 14(4-5) May 2001, 427-437

Comparing Bayesian Neural Network Algorithms for Classifying Segmented Outdoor Images

Francesco Vivarelli
The Knowledge Lab - NCR Financial Solutions,
206 Marylebone Road,
London NW1 6LY, UK
email: francesco.vivarelli@ncr.com

Christopher K. I. Williams*
Division of Informatics,
The University of Edinburgh,
5 Forrest Hill,
Edinburgh EH1 2QL, UK
email: c.k.i.williams@ed.ac.uk

Preprint date: July 2001

*To whom correspondence should be addressed.

Comparing Bayesian Neural Network Algorithms for Classifying Segmented Outdoor Images

Abstract

In this paper we investigate the Bayesian training of neural networks for region labelling of segmented outdoor scenes; the data is drawn from the Sowerby Image Database of British Aerospace. Neural networks are trained with two Bayesian methods, (i) the evidence framework of MacKay (1992) and (ii) a Markov Chain Monte Carlo method due to Neal (1996). The performance of the two methods is compared by evaluating the empirical learning curves of neural networks trained with the two methods. We also investigate the use of the Automatic Relevance Determination method for input feature selection.

Keywords: Bayesian training of neural networks, Evidence framework, Markov Chain Monte Carlo, Automatic Relevance Determination, Empirical Learning curves.

1 Introduction

This paper is concerned with a comparison of two Bayesian methods for training neural networks; the evidence framework of MacKay (1992) and a Markov Chain Monte Carlo method due to Neal (1996). The networks are trained on the task of labelling regions of segmented outdoor scenes. The performance of the two methods is compared by evaluating the empirical learning curves of neural networks trained with the two Bayesian algorithms. It is important to carry out such a comparison if a neural network has to be applied to real-world tasks, in order to understand which algorithm can provide better generalisation performance. We also discuss the use of the Automatic Relevance Determination method for input feature selection.

The structure of the paper is as follows. In Section 2 we discuss various approaches that can be taken to the segmentation of images and explain the method used in this paper. Section 3 introduces the image database used for the training of the neural networks. The two implementations of Bayesian learning of neural networks are introduced in Section 4 and applied to the training of a Multi Layer Perceptron (MLP). Section 5 describes a preliminary experiment using both multiple logistic regression and multi-layer perceptron classifiers on the full data set. In Section 7 we investigate the empirical learning curves of the two methods, comparing classification performances on training sets of various sizes. In Section 8 results are reported for the use of the Automatic Relevance Determination technique (ARD) of MacKay and Neal (Neal, 1996) for feature selection. This paper (which is a revised and expanded version of Vivarelli and Williams (1997)) concludes with a discussion of the study undertaken.

2 Segmentation and Scene Analysis

This paper deals with the specific problem of investigating the use of neural networks for the classification of regions of outdoor scenes.

Scene interpretation is usually split in a two stage procedure, the segmentation and the interpretation of images. The segmentation process divides an image in a set of regions, where ideally each region corresponds to one object; the interpretation consists in the labelling of the regions. To carry out the task successfully it is important to classify each region not only using its own attributes (e.g. colour, shape, texture) but also taking into account the *context*. The context can be handled in two ways, either by searching for a consistent interpretation of the whole scene, or by classifying each single region according to its local context.

An example of the interpretation of the whole scene is based on Artificial Intelligence techniques (Ohta, 1985; Draper et al., 1989; McKeown et al., 1985). In this framework regions are described on the basis of their internal attributes and the context they are inserted in. The labelling process is carried out on the basis of a set of logic rules, so that each region is classified in one class or another according to which subset of rules are fulfilled.

An example of classification of each individual region on the basis of local context makes use of statistical methods (Wright, 1989; Wright et al., 1995; Mackeown, 1994; Clark, 1995). Segmented regions are classified individually by using statistical methods rather than a set of rules. Neural networks have been applied to this task and contextual information is used in the classification of each region.

The methods discussed above deal with generic scenes. However if specific information is available (e.g. airplane types), scene interpretation can use this knowledge by matching geometric models to image features. Similarly to the rule-based system, the matching of models can be performed by using logic rules. An example of vision system based on this approach was ACRONYM (Brooks, 1983), which was designed for the recognition of airplanes from aerial images.

An approach to image interpretation based on the matching of geometric models does not seem applicable to the Sowerby Image Database due to the great variety of objects represented. Even though a model-based approach may be suitable for recognition of man-made objects, it would be difficult to apply it to the classification of regions with complex shape (such as trees, vegetation). Rule-based systems may have problems when dealing with real valued attributes as their decisions are based upon binary logic.

The present study follows the same approach as Wright (1989).

3 The Database

The database consists of 96 coloured images extracted from the Sowerby Image Database of British Aerospace. The main subjects of the images are rural and urban scenes; the database was constructed so that objects with diverse characteristics appear in it (e.g. cars with different colours, different shapes of buildings, etc.). All the scenes have been photographed using small-grain 35mm transparency film. In order to ensure clear pictures, all the images have been shot with good atmospheric visibility, in dry and fully overcast conditions. Each image of the database has been digitised with a calibrated scanner, generating a high quality 24-bit colour representation of size 768 by 512 pixels.

The images have been segmented, splitting them up in a set of regions; the segmentation algorithm used in the Sowerby database is based on the region growing method of Gay (1989) and is described in Mackeown (1994).

The numerical description of the database is realised by computing a feature vector which encodes the characteristics of each region. The features chosen are divided in two groups, the internal and the contextual features. The internal features describe the interior characteristics of a given region such as colour, topology, size, position, shape and texture. The contextual features describe the relationships between a region and the regions around it. Following Wright et al. (1995), the contextual features are defined as the relative size, intensity and position of a region with respect to the four largest regions surrounding it. Because of the fixed length of the contextual features, if the number of surrounding regions is less than four there are some features which are not defined: in this case the undefined contextual features are set to the average of the defined components. Further details of the features used can be found in Vivarelli (1998).

All of the features have been rescaled using a linear normalising transformation, obtaining rescaled features which have mean 0 and variance 1.

The 96 images have been divided randomly in two independent sets: a training set and a testing set. Each region of the two sets has been hand labelled in one of the eleven categories. Table (1) shows a list of the 11 categories and the composition of the two sets.

4 Bayesian training of neural networks

A feed-forward neural network (Hertz et al., 1991) consists of processing units allocated in layers (Figure 1); each node of one layer is connected with all those of the previous layer and is characterised by a numerical value called *activation*.

In this paper neural networks with two layers of adaptable weights have mainly been used. The input layer is made up of 35 units whose activation values are the components of the feature vector.

In order to discriminate patterns belonging to one out of c classes, the output units should be interpreted as the posterior probabilities $P(C_k|\mathbf{x})$ that the pattern $\mathbf{x}$ belongs to the class C_k. This is achieved by imposing two conditions on the output functions: $y_k(\mathbf{x}) \in [0,1]$ and $\sum_{k=1}^{c} y_k(\mathbf{x}) = 1$. The activation of the output units is given by the softmax function

$$y_k(\mathbf{x}) = \frac{\exp[a_k(\mathbf{x})]}{\sum_{l=1}^{c} \exp[a_l(\mathbf{x})]}, \tag{1}$$

where $a_k(\mathbf{x})$ is given by the sum of the output values of all the units lying in the previous layer weighted by a set of adjustable parameters $\mathbf{w}$. In our experiments $c = 11$. The probability of misclassification is minimised by assigning the pattern $\mathbf{x}$ to the class k for which $y_k(\mathbf{x}) > y_l(\mathbf{x}) \, \forall l = 1, \ldots, c, l \neq k$. The activation function of the j-th hidden unit of the MLP is the sigmoid function $z_j(\mathbf{x}) = \tanh a_j(\mathbf{x})$.

It is natural to describe the Bayesian training of neural networks in a three-level hierarchy. The first level involves the probability distribution $p(\mathbf{w}|\boldsymbol{\alpha})$ over the synaptic weights of a neural network given the vector of *hyperparameters* $\boldsymbol{\alpha}$. The second level of the hierarchical description is concerned with the probability distribution of the hyperparameters $p(\boldsymbol{\alpha})$. A third hierarchical level allows the comparison between different statistical models. Given a set of models $\{H_i\}$, $p(H_i|\mathcal{D}_n)$ can be calculated by using Bayes' theorem and thus it is possible to choose the model which is more suitable to describe the dataset or to average predictions over the models. Due to limited time, issues related to model selection have not been considered in this paper.

In order to detect the relevant components of the input vector one hyperparameter can be associated with each group of weights which connects one input unit to all of the units in the next layer. This is the Automatic Relevance Determination (ARD) method of MacKay and Neal (Neal, 1996). Three further hyperparameters control the distribution of the hidden unit biases, the hidden-to-output weights and the biases of the output units of the MLP, as shown in Figure 1. The hyperparameter α_g controls the size of the group of weights g through a Gaussian prior distribution with zero mean and variance $\sigma_g^2 = 1/\alpha_g$. The larger the hyperparameter α_g the smaller the variance σ_g^2 of the weights belonging to the group g; the smaller α_g the larger σ_g^2. As we shall see below, these hyperparameters can be adapted during training of the neural network; a large value of α_g relative to the other groups indicates that the associated group of weights are less relevant.

After the training of the model on data $\mathcal{D}_n$, the distribution of the weights $\mathbf{w}$ conditional on $\boldsymbol{\alpha}$ is given by Bayes' theorem

$$p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n) = \frac{p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathcal{D}_n|\boldsymbol{\alpha})}, \tag{2}$$

where $p(\mathcal{D}_n|\mathbf{w})$ is the likelihood and

$$p(\mathcal{D}_n|\boldsymbol{\alpha}) = \int p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})\, d\mathbf{w}$$

is a normalising factor called the *evidence* (or marginal likelihood).

From the Equation 1, the output of a network $y_l(\mathbf{x}^i)$ can be interpreted as the posterior probability of assigning the input vector $\mathbf{x}^i$ to the class C_l. Considering the training point

$\left(\mathbf{x}^i, \mathbf{t}^i\right)$, where the l-th component of the target vector $t_l^i = \delta_{kl}$ if $\mathbf{x}^i$ belongs to the class C_k, the probability of observing $\mathbf{t}^i$ is given by $p\left(\mathbf{t}^i|\mathbf{w}, \mathbf{x}^i\right) = \prod_{l=1}^{c} \exp\left[t_l^i \ln y_l\left(\mathbf{x}^i\right)\right]$; the likelihood of a training set $\mathcal{D}_n$ composed by data independently drawn from the same distribution can be written as (see e.g. Bishop (1995))

$$p\left(\mathcal{D}_n|\mathbf{w}\right) = \prod_{i=1}^{n} p\left(\mathbf{t}^i|\mathbf{w}, \mathbf{x}^i\right) = \exp\left[\left(\sum_{i=1}^{n}\sum_{l=1}^{c} t_l^i \ln y_l\left(\mathbf{x}^i\right)\right)\right]. \tag{3}$$

Bayes' theorem also expresses the probability distribution of $\boldsymbol{\alpha}$ given the data $\mathcal{D}_n$:

$$p\left(\boldsymbol{\alpha}|\mathcal{D}_n\right) = \frac{p\left(\mathcal{D}_n|\boldsymbol{\alpha}\right) p\left(\boldsymbol{\alpha}\right)}{p\left(\mathcal{D}_n\right)}, \tag{4}$$

where $p\left(\boldsymbol{\alpha}\right)$ is the prior distribution. Equations 2 and Equation 4 show the hierarchical nature of the Bayesian approach: the term depending upon the data in the denominator of Equation 2 appears also in the numerator of Equation 4.

Bayesian prediction for a new input $\mathbf{x}$ is given by the *marginalisation* of the classification performed by the model $p\left(C_k|\mathbf{x}, \mathbf{w}\right)$ with respect to the posterior distribution of the parameters:

$$p\left(C_k|\mathbf{x}, \mathcal{D}_n\right) = \int p\left(C_k|\mathbf{x}, \mathbf{w}\right) p\left(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n\right) d\mathbf{w} d\boldsymbol{\alpha}. \tag{5}$$

The second term of the integrand represents the degree of belief in the values of the parameters $\mathbf{w}$ and $\boldsymbol{\alpha}$ given the data $\mathcal{D}_n$ and can be expressed as

$$p\left(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n\right) = p\left(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n\right) p\left(\boldsymbol{\alpha}|\mathcal{D}_n\right). \tag{6}$$

As the integrals in Equation 5 cannot be computed analytically for neural network models, we need to use numerical approximations. We compare two methods, the evidence framework (EF) and the Markov Chain Monte Carlo method (MCMC). The EF estimates the posterior distribution of the network's parameters approximating Equation 6 with a Gaussian function centred around the most probable value $\mathbf{w}^{mp}$ and optimising the hyperparameters $\boldsymbol{\alpha}$. The MCMC algorithm constructs a Markov chain whose equilibrium distribution is the desired posterior distribution. Samples from this chain are used to compute a Monte Carlo approximation of Equation 5. Both of the methods are briefly outlined in the following sections.

Barber and Bishop (1998) have suggested a different approach to the Bayesian training of neural networks based on the framework of the Ensemble learning (Hinton and van Camp, 1993). They approximated the posterior distribution of the network's parameters by optimising the Kullback-Leibler divergence between an ensemble $Q\left(\mathbf{w}, \boldsymbol{\alpha}\right)$ and the true posterior. Although it seems a promising approach, due to limited time we did not consider it in our experiments.

4.1 The evidence framework

The EF (based on work by Gull) has been discussed by MacKay (MacKay, 1992a; MacKay, 1992b) and is similar to the *type II maximum likelihood* method.

The EF computes an approximation to equation (6) by assuming that the posterior probability of the hyperparameters $p\left(\boldsymbol{\alpha}|\mathcal{D}_n\right)$ is sharply peaked around its maximum $\boldsymbol{\alpha}^{mp}$. To reflect the lack of *a priori* knowledge about the best value of $\boldsymbol{\alpha}$, the hyper-prior $p\left(\boldsymbol{\alpha}\right)$

is chosen to have the product form $p(\boldsymbol{\alpha}) = \prod_g p(\alpha_g)$ where g indexes a group, with each $p(\alpha_g)$ being constant on a logarithmic scale. Thus the value of $\boldsymbol{\alpha}$ maximising the log posterior $\log p(\boldsymbol{\alpha}|\mathcal{D}_n)$ can be found by maximising $\log p(\mathcal{D}_n|\boldsymbol{\alpha})$. $p(\mathcal{D}_n|\boldsymbol{\alpha})$ is obtained from the equation

$$p(\mathcal{D}_n|\boldsymbol{\alpha}) = \int p(\mathcal{D}_n|\mathbf{w})\, p(\mathbf{w}|\boldsymbol{\alpha})\, d\mathbf{w}. \tag{7}$$

In the EF the integral is approximated using Laplace's approximation around $\mathbf{w}^{mp}$, the value of $\mathbf{w}$ that maximises $p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n)$. It turns out that the components of the optimal values of $\boldsymbol{\alpha}^{mp}$ are given by

$$\alpha_g^{mp} = \frac{\gamma_g}{\sum_{i\in g}(w_i^{mp})^2}, \tag{8}$$

where $\gamma_g = W_g - \alpha_g \sum_{i\in g}\left(\left(\nabla\nabla G + \boldsymbol{\alpha}\mathbb{I}\right)^{-1}\right)_{ii}$, W_g is the total number of weights controlled by the hyperparameter α_g and G is the negative logarithm of the penalised likelihood ($G = -\ln p(\mathcal{D}_n|\mathbf{w}) - \ln p(\mathbf{w}|\boldsymbol{\alpha})$); the sum takes into account the elements of the matrix $\left(\nabla\nabla G + \boldsymbol{\alpha}\mathbb{I}\right)^{-1}$ corresponding to the weights belonging to the group g.

From the estimation of the optimal value of the hyperparameter α_g (Equation 8), we can construct a Gaussian approximation of $p(\mathcal{D}_n|\boldsymbol{\alpha})$. Since the hyperparameters are scale factors for the weights, their uncertainty is usually represented on a logarithmic scale. From the Gaussian approximation, the inverse of the variance for $\log \alpha_g$ turns out to be

$$\frac{1}{\sigma^2_{\log \alpha_g}} = -\alpha_g \frac{d}{d\alpha_g}\left(\alpha_g \frac{d}{d\alpha_g} p(\mathcal{D}_n|\alpha_g)\right) \simeq \frac{\gamma_g}{2}. \tag{9}$$

(see e.g. Bishop (1995), equations 10.82 to 10.84).

The EF proceeds by a two-step iterative procedure by computing the value $\mathbf{w}^{mp}$ maximising the penalised likelihood while periodically re-estimating the hyperparameter $\boldsymbol{\alpha}^{mp}$. In our experiments the first step has been achieved by optimising the penalised likelihood with a scaled conjugate algorithm for 70 iterations. The re-estimation of the hyperparameters $\boldsymbol{\alpha}^{mp}$ was carried out 10 times, by when the optimisation of $\boldsymbol{\alpha}$ had converged.

Some problems may arise during the implementation of EF. Since it is based on an approximation of the posterior around a local minimum, EF assumes that the Hessian matrix is positive definite at $\mathbf{w}^{mp}$. Sometimes this is not the case because the optimisation of the weights has not fully reached a local minimum and thus some of the eigenvalues can be negative, introducing instability in the implementation. In order to avoid this, we set the negative eigenvalues in $\nabla\nabla G$ to 0. Another problem is due to the fact that EF is based upon the computation of a Hessian matrix; for large networks the amount of storage required for this matrix is considerable, because its size grows as the square of the number of weights.

4.2 The Markov Chain Monte Carlo method

The second Bayesian algorithm computes Equation 5 by Markov Chain Monte Carlo (Neal, 1996). An MCMC algorithm constructs a Markov chain whose equilibrium distribution is the desired probability density $p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n)$. The Monte Carlo algorithm (see, e.g. Press et al., 1992) approximates an integral as $I = \int f(\mathbf{z})\, p(\mathbf{z})$ with the arithmetic mean $I \sim \sum_i f(\mathbf{z}^i)/N$, where $\{\mathbf{z}^i, i = 1 \ldots N\}$ are samples from the distribution $p(\mathbf{z})$. In our case we have $\mathbf{z} = (\mathbf{w}, \boldsymbol{\alpha})$, $f(\mathbf{z}) = p(C_k|\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha})$ and $p(\mathbf{z}) = p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n)$. (Strictly speaking $\boldsymbol{\alpha}$ need not appear in $\mathbf{z}$; however, it is a useful *auxiliary variable* that is used in constructing the Markov chain.) Although samples from the chain are not independent, they can be used for computing the necessary integrals.

The algorithm[1] was written by R. Neal (Neal, 1996); a brief outline is given below.

Samples from the posterior distribution $p(\mathbf{w}, \boldsymbol{\alpha}|\mathcal{D}_n) = p(\boldsymbol{\alpha}|\mathcal{D}_n)\, p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n)$ follow from a two-step procedure; in the first step the hyperparameters are held constant and the posterior distribution of the network weights is sampled using the *Hybrid Monte Carlo* method (Duane et al., 1987). This algorithm merges the Metropolis algorithm with a dynamical simulation. As it avoids random walks, it performs better than a simple Metropolis algorithm. The hyperparameters are changed during the second step, when a sample is obtained from the posterior distribution $p(\boldsymbol{\alpha}|\mathbf{w}, \mathcal{D}_n)$. Thus the two-step procedure carries out a block Gibbs sampling, first sampling from $p(\mathbf{w}|\boldsymbol{\alpha}, \mathcal{D}_n)$ then $p(\boldsymbol{\alpha}|\mathbf{w}, \mathcal{D}_n)$.

In the Hybrid Monte Carlo method each network variable has an associated fictitious momentum, thereby creating a dynamical system which is described in the phase space by a set of coordinates $(\mathbf{q}, \mathbf{p})$; for neural networks, the vector position $\mathbf{q}$ is interpreted as the network weights while $\mathbf{p}$ is the associated momentum. This system is described by an Hamiltonian function H given by the sum of the kinetic and the potential energies (denoted by $K(\mathbf{p})$ and $V(\mathbf{q})$ respectively). The kinetic energy is a function of the momentum vector $K(\mathbf{p}) = \mathbf{p}^T\mathbf{p}/2$. The potential energy is a function of the position $\mathbf{q}$. The Hybrid Monte Carlo method samples from the canonical distribution for $\mathbf{q}$ and $\mathbf{p}$ defined as

$$p(\mathbf{q}, \mathbf{p}) \propto \exp(-H(\mathbf{q}, \mathbf{p})) = \exp(-(K(\mathbf{p}) + V(\mathbf{q}))).$$

Provided that $V(\mathbf{q}) = V(\mathbf{w}) = -\ln p(\mathbf{w}|\mathcal{D}_n, \boldsymbol{\alpha})$, a set of values of $\mathbf{q}$ (whose posterior probability distribution is $p(\mathbf{q}) = p(\mathbf{w}|\mathcal{D}_n, \boldsymbol{\alpha})$ is obtained by generating samples from $p(\mathbf{q}, \mathbf{p})$ and ignoring $\mathbf{p}$.

Sampling from the joint distribution $p(\mathbf{q}, \mathbf{p})$ is achieved by generating new points in the phase space with constant value of H and then sampling the momentum variables $\mathbf{p}$ from the prior to change the value of H. The *leap-frog* method is used to approximate the Hamilton's first order differential equations. At the end of a chain of L leapfrog steps, the state of the system can be accepted or rejected depending upon the average value of the energy H over a window of states.

The prior distribution for each group of weights is a zero-mean Gaussian whose precision (α_g) is specified by a vague Gamma prior. In fact a two-level prior is used[2] so that there is a high-level prior controlling the overall magnitude of the α_g's. We also used scaling on the hidden-to-output weight as recommended by Neal (1996) so that the prior variances of the activations of the softmax units do not grow with the number of hidden units.

Although the specification of the prior for the α's is not exactly the same for the EF and the MCMC methods, note that the MCMC priors are vague so we would not expect this to cause large differences in the results.

[1]The source code of the implementation of Bayesian learning of neural networks is available at the ftp address: `ftp://ftp.cs.utoronto.ca/pub/radford/`.

[2]The actual prior specification used during our experiments is `net-spec mlp30-log 35 30 11 / - 0.05:0.5:0.5 0.05:0.5 - x0.05:0.5 - 0.05:0.5`

4.2.1 Practical issues for the MCMC method

For the results reported in sections 5, 7 and 8 the number of leap-frog steps L was 100.[3] The sampling phase of the MCMC simulations was run for 200 iterations; the first third of these have been discarded as "burn-in" (Gelman et al., 1995), with the aim of letting the simulation reach the equilibrium distribution. During the simulations, the rejection rates were around 1%.

We note that in general it is very difficult to know when a MCMC simulation has reached equilibrium, see, e.g. (Cowles and Carlin, 1996). We analysed the output of the MCMC simulations by looking at autocorrelation plots of the hyperparameters, and conducting the Geweke, Raftery-Lewis and Heidelberger-Welch tests in the CODA package[4] for the analysis of MCMC output as described in (Cowles and Carlin, 1996). In particular we looked at runs with small ($n = 91$) and large ($n = 2916$) amounts of data. From this analysis it was not possible to conclude that the equilibrium distribution had been reached.

We also investigated the convergence issue by running some experiments for much longer than the standard conditions stated above and comparing the results, specifically using HMC trajectories of length 1000 (as compared to the standard length of 100 steps). These longer runs used the same training sets as two of the runs described in Section 7. We compared the classification accuracy on an independent test set obtained under these two conditions, as given in Table 2 for both small and large amounts of data ($n = 91$ and $n = 2916$). These results indicate that the classification performances obtained for the two run lengths are not significantly different.

The lack of convergence is undesirable. However, the results below show that the MCMC method with relatively short run lengths can be viewed as a practical method which can yield superior performance compared to the EF method for the the size of network investigated below.

5 Preliminary investigations

Preliminary experiments were carried out using Multiple Logistic Regression (MLR) and MLP networks. The MLR network has only one layer of adaptable weights, and no hidden layer. A MLP network with 30 hidden units was used; this is a relatively large number of hidden units, chosen with a view to observing a difference between the EF and MCMC methods. This network has 1421 weights, and the maximum number of training examples available is 5832. Both the MLR and MLP networks were trained using both Bayesian methods.

In experiments where the ARD approach has been used with the MLP, this gave rise to 38 hyperparameters, 35 controlling the distribution of the input-to-hidden weights, one controlling the biases of the hidden units, one controlling the distribution of the hidden-to-output weights and one for the output biases. ARD with the MLR network gave rise to 36 hyperparameters, 35 controlling the distribution of the input-to-output weights and one for the output biases.

An initial analysis of the data was carried out, using the whole training set (5832 examples) for training, and testing on all 1505 test cases. Both MLR and MLP models were trained on this data, using ARD. The results are given in Table 3, for both region-based and

[3]Further details: the window size was 10 states and the step size correction factor (for the approximation of the Hamilton's equations) was 0.3. See (Neal, 1996) and the software documentation for details.

[4]The R-CODA package available from `http://www.ci.tuwien.ac.at/R/src/contrib/PACKAGES.html` was used.

area-based measures of performance. The results are quite similar for the MLR and MLP models. For the MLP model a small improvement is obtained using the MCMC training method over EF, and McNemar's test (Ripley, 1996; p. 77) shows that this difference is significant at the 95% level. We also investigated the effects of not using ARD; this gave rise to similar performance to the full ARD case.

6 Overview of Experiments

We compare and contrast the training of the EF and the MCMC with respect to two issues.

- Empirical learning curves. In order to assess how the training algorithms rely on the amount of training data, we trained the MLP on several independent training sets by varying the quantity of data available and assessing the sensitivity to the choice of the seed initialising the random number generator of the two algorithms. The study has been carried out comparing empirical learning curves of the MLP trained with the EF and the MCMC. These experiments are time consuming and have been running on our 200 MHz Silicon Graphics Challenge machine during several months before their completion; the results are presented in Section 7.

- Feature selection. In Section 8 we investigate the relevance of the various input features using the ARD method described above.

7 Empirical learning curves

A thorough investigation of the differences between the EF and the MCMC methods training of neural networks can be carried out by a statistical analysis of empirical learning curves. We should note in advance that the task of comparing two learning techniques, determining which one actually out-performs the other and detecting all the relevant sources of variation, is a difficult one (Rasmussen et al., 1996), (Dietterich, 1998).

Generalisation capabilities of neural networks depend upon many stochastic factors (see e.g. Rasmussen, 1996); for our problem, the variations in generalisation error are mainly due to the randomness in the learning algorithms and in the samples from the distribution of the test data and the training sets. The learning curve can be defined as the expectation of the generalisation error averaged over the distribution of all the stochastic components

$$E^g\left(n\right) = \int E^g\left(\mathcal{D}_n, r, \mathbf{x}, t\right) p\left(\mathcal{D}_n, r, \mathbf{x}, t\right) d\mathcal{D}_n dr d\mathbf{x} dt, \tag{10}$$

where $(\mathbf{x}, t)$ is the test point, $\mathcal{D}_n$ is the training set and r represents stochastic effects due to the training method; in our case, the stochastic effect r depends on the seed initialising the random number generator for the two algorithms.

Generalisation errors of the MLP trained with the two Bayesian algorithms have been estimated on the basis of the classification performances obtained on the original test set. At least two different evaluations of generalisation error can be employed. The most common measure is the 0/1 loss function; in this case the neural network is considered as a discriminant function assigning the test vector $\mathbf{x}$ to the class C_k for which $y_k\left(\mathbf{x}\right) > y_l\left(\mathbf{x}\right), \forall l = 1, \ldots c$. A second performance measure is based on the likelihood of the true class (see Equation 3) averaged over the test points.

Although the latter is more suitable to describe the class conditional distribution estimated by the neural network, the former is a more commonly-used measure of the overall performance of the classifier and in all our experiments we evaluate the generalisation error of the MLP with the 0/1 loss function.

We have conducted experiments evaluating the effect on the generalisation error of randomising over the choice of the dataset and the random seed; the aim of the experiments is to find evidence for which learning algorithm gives the better generalisation performance for the task at hand using a MLP predictor.

The experiments have been analysed following a two-way ANOVA design (DeGroot, 1984; Rasmussen et al., 1996). In the two-way ANOVA analysis, the generalisation error obtained after training the MLP with a given learning algorithms is modelled by

$$E^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right) = \mu_n + \alpha_n^i + \beta_n^j + \varepsilon_n^{ij}, \tag{11}$$

where μ_n is the mean generalisation error obtained with a training set with n data, α_n^i and β_n^j model the effect on μ_n due to choice of the training set $\mathcal{D}_n^i$ and to the test point $\left(\mathbf{x}^j, t^j\right)$, respectively. The residual variations in generalisation error due to stochastic effects and to the interaction between the training set and the test point are described by the noise term ε_n^{ij}. The stochastic effects include the variability of $E^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right)$ due to the choice of the random seed in the learning algorithm. We assume that the values of α, β and ε are normally distributed around zero with variance σ_α^2, σ_β^2 and σ_ε^2, respectively; these assumptions are not strictly applicable when the 0/1 loss function for classification is used, but a more appropriate model is not straightforward to analyse. However, according to Rasmussen (1996), these assumptions are commonly used and they lead to reasonable conclusions[5].

The superscripts i and j in Equation 11 label the training set $\mathcal{D}_n^i$ and the test points $j = 1 \dots 1505$, respectively. In order to obtain a number of training sets $\mathcal{D}_n^i$, the original training set made up by 5832 examples has been successively subdivided in smaller datasets with different sizes; the number of training sets varies with the amount of data n, being $i = 1, 2$ for $n = 2916$ data, $i = 1 \dots 4$ for $n = 1458$, $i = 1 \dots 8$ for $n = 729$ and $i = 1 \dots 10$ for $n = 365$, 182, 91 and 46. All the datasets were disjoint. For each training set one run was made for each method, with initial conditions specified by a random number generator. We do not carry out such analysis when the full amount of data is used (i.e. $n = 5832$) because we can not estimate the variance on the expected generalisation error generated by only one training set.

The estimate of the mean value μ_n and its variance have been computed according to the two-way ANOVA design with one observation for each coupling $\left(\mathcal{D}_n^i, j\right)$ as

$$\hat{\mu}_n = \frac{1}{N_t D} \sum_{i=1}^{D} \sum_{j=1}^{N_t} \hat{E}^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right), \tag{12}$$

$$\hat{\sigma}_{\hat{\mu}_n} = \sqrt{\frac{\hat{\sigma}_a^2}{D} + \frac{\hat{\sigma}_b^2}{N_t} + \frac{\hat{\sigma}_\varepsilon^2}{N_t D}} \tag{13}$$

(Rasmussen et al., 1996), where N_t and D indicate the overall number of test examples and training sets, respectively; $\hat{E}^g\left(\mathcal{D}_n^i, \mathbf{x}^j, t^j\right)$ indicates the generalisation error of the MLP evaluated for the training set $\mathcal{D}_n^i$ and the test point $\left(\mathbf{x}^j, t^j\right)$. The $\hat{}$ in Equations 12 - 13 denotes the empirical estimates of the variables; in particular, the values of σ_α^2, σ_β^2 and σ_ε^2 follow from the formulæ

$$\hat{\sigma}_\alpha^2 = \frac{(mse_\alpha - mse_\varepsilon)}{N_t}, \hat{\sigma}_\beta^2 = \frac{(mse_\beta - mse_\varepsilon)}{D}, \hat{\sigma}_\varepsilon^2 = mse_\varepsilon,$$

[5] An alternative approach to empirical study on difference in classification performance has been presented by Dietterich (1998).

where the *mean squared errors* are evaluated as

$$
\begin{aligned}
mse_\alpha &= \frac{N_t}{D-1}\sum_{i=1}^{D}\left(\hat{\mu}_{i\cdot}-\hat{\mu}_n\right)^2,\\
mse_\beta &= \frac{D}{N_t-1}\sum_{j=1}^{N_t}\left(\hat{\mu}_{\cdot j}-\hat{\mu}_n\right)^2,\\
mse_\varepsilon &= \frac{1}{(D-1)(N_t-1)}\sum_{i=1}^{D}\sum_{j=1}^{N_t}\left[\left(\hat{E}^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)-\hat{\mu}_n\right)-\left(\hat{\mu}_{i\cdot}-\hat{\mu}_n\right)-\left(\hat{\mu}_{\cdot j}-\hat{\mu}_n\right)\right]^2,
\end{aligned}
$$

and

$$
\hat{\mu}_{i\cdot}=\frac{1}{N_t}\sum_{j=1}^{N_t}\hat{E}^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right),\ \hat{\mu}_{\cdot j}=\frac{1}{D}\sum_{i=1}^{D}\hat{E}^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)
$$

(cf. Equations $12-17$ in Rasmussen et al. (1996)).

In order to quantify the difference in generalisation errors obtained by the MLP trained by EF and MCMC, we carried out also the comparison of the two methods by modelling the difference of the expected generalisation errors according to the two-way ANOVA design, i.e.

$$
\begin{aligned}
\Delta E^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right) &= E^g_{\mathrm{EF}}\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)-E^g_{\mathrm{MCMC}}\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)\\
&= \Delta\mu_n+\Delta\alpha_n^i+\Delta\beta_n^j+\Delta\varepsilon_n^{ij}.
\end{aligned}
\tag{14}
$$

In Equation 14 the variable $\Delta\mu_n$ models the variation in the mean generalisation error obtained with a training set with n data, $\Delta\alpha_n^i$ and $\Delta\beta_n^j$ model the variations of the effects due to choice of the training set $\mathcal{D}_n^i$ and to the test point $\left(\mathbf{x}^j,t^j\right)$, respectively; $\Delta\varepsilon_n^{ij}$ describes residual variations in the difference of the generalisation errors due to stochastic effects and to the interaction between the training set and the test point. After the substitution of the generalisation error $E^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)$ with the difference $\Delta E^g\left(\mathcal{D}_n^i,\mathbf{x}^j,t^j\right)$, the formulæ for the numerical estimation of $\Delta\hat{\mu}$ and its variance are similar to Equations 12 and 13, respectively.

7.1 Learning curve results

Figure 2(a) shows the empirical learning curves of the MLP trained with the EF and the MCMC algorithms, as calculated from Equations 12 and 13; the numerical estimation of $\Delta\hat{E}^g(n)$ with its 95% interval of variability as a function of the amount of training data is shown in Figure 2(b). The numerical values of $\hat{E}^g_{\mathrm{EF}}(n)$, $\hat{E}^g_{\mathrm{MCMC}}(n)$ and $\Delta\hat{E}^g(n)$ are reported in Table 4.

Evidence that one algorithm is significantly better than the other can be obtained by investigating whether the estimated overall difference $\Delta\hat{E}^g(n)$ is significantly different from zero; of course the conclusions drawn from this analysis depend upon the assumptions concerning the model adopted in Equation 14.

Using a quasi-F test (Lindman, 1974) (as recommended by Rasmussen et al., 1996), at the 0.05 level of significance we find that for most of the sizes of training set, the use of the MCMC algorithm has the effect of significantly improving the generalisation error of the MLP with respect to the one obtained by the MLP trained with the EF, although the difference varies with the number of training examples. In particular $\Delta\hat{E}^g(n)$ is significantly different from zero at the 5% level of significance for all datasets except $n = 365$. For large datasets the difference in generalisation error and its interval of variability are always below 7%; the difference is enlarged when small datasets are used (being in the range [20%, 30%] for $n = 91$ and [25%, 35%] for $n = 46$), although we note that these results have a larger variability.

Due to questions concerning the applicability of the Gaussian assumptions when using the two-way ANOVA model of equation 11 with 0/1 loss, we also analysed the results when j indexes not a single test case but a pool of test cases. Pool sizes of 2, 4, 8, 16 and 32 were used. The results obtained were very similar to those above with $\Delta\hat{E}^g(n)$ being significantly different from zero for training sets of size 1458, 729, 182, 91 and 46. The only difference was that $\Delta\hat{E}^g(n)$ for $n = 2916$ was now found to be not significantly different from zero.

The MLP trained using MCMC obtains better generalisation error than that obtained by the MLP trained with EF when small datasets are used; we expect that this is due to the inaccurate Gaussian approximation of the posterior distribution of the network's parameters estimated by the EF. This is not unexpected as it is improbable that the posterior can be well-approximated with a Gaussian distribution when the amount of data is small. It is perhaps surprising that the EF method is almost as effective as the MCMC method for $n \geq 182$, especially when the number of examples n is less than the number of parameters $|W| = 1421$. Generally one would expect to need $n/|W| > 1$ (and perhaps $\gg 1$) before the posterior is well localized. As there are 11 output units, there are only 30 hidden-to-output weights (plus one bias weight) per output unit and thus a relatively small n may provide adequate localization of these weights. However, this can only be a partial explanation as there are 1080 input-to-hidden weights.

Another possible explanation of the efficacy of the EF method for $n \geq 182$ relates to the fact that the MLR network described in section 5 was able to obtain good performance on the task used. It is possible for the MLP network to approximate a MLR network when the hidden units are working in the linear part of their operating range. Although there is a whole subspace of MLP weights that approximates a given MLR, this subspace might be modelled adequately by a Gaussian approximation. It is also possible that the predictions made by the network trained by the EF could be good although the Gaussian approximation is poor.

During our experiments we noticed another disadvantage of the EF for small data sets, namely that a large part of the computational effort is taken up with the inversion of the Hessian so that for small datasets the MCMC method yields its results using less CPU time. It is very difficult to compare precisely the execution time of the two algorithms as they have been implemented in two different programming languages (the EF has been written for MATLAB and the MCMC has been written in C). However, we can describe their relative computational complexity. Letting W be the number of parameters of the MLP, the EF takes time $O(nW) + O(W^3)$ per iteration as the forward and backpropagation passes are $O(nW)$ and the evaluation of the eigenvalues of the Hessian matrix is $O(W^3)$. The MCMC method scales as $O(nLW)$ per iteration as it requires the evaluation of forward and backpropagation passes for each of the L steps in the dynamical simulation.

8 Automatic Relevance Determination Results

The results in this section were obtained by training the network on the full training set of 5832 examples. Figure 3 reports the values of the hyperparameters (with their 95% confidence intervals) determined with the ARD prior for the MLP trained with both the

EF and the MCMC algorithms (see Figures 3(a) and (b) respectively). There is broad agreement between the two plots, although we note that the confidence intervals are usually larger for the MCMC case.

The graphs show that the features describing the colour, size and the y position of the regions (inputs 1, 2, 3, 7 and 9) are the most relevant in training the classifier; the hyperparameters corresponding to those features have small values for both the MCMC and the EF methods. The lesser relevance of the topological features (inputs 4-6) and the x position (x_8) of the regions is also recognised by both of the methods.

Concerning the textural features, both EF and MCMC point out the relative importance of the last feature x_{19}, whereas x_{17} and x_{18} seem less relevant for the labelling of the regions. Among the contextual features (inputs $20-35$), the relative intensity of one region ($x_{24} \dots x_{27}$) with respect to the four surrounding it appears the most useful.

All of the experiments have shown that the coordinates (x, y) of the centroids have a different weight in training the networks. Because images are taken with the y axis closely aligned to the vertical, the classification of the regions is unlikely to depend upon the x coordinate of the centroid; for example, regions representing cars appear in the data base in many positions along the x axis, whereas their y coordinates are ranged in a well defined interval. A similar consideration applies for the contextual features $28-35$, where a different relevance is accorded to the x and the y offsets of the relative position.

We note that the determination of the irrelevance of some of the features does not mean that those attributes are *absolutely* irrelevant. It may simply mean that the features chosen have not properly encoded the information about a region.

9 Discussion

In this paper we have compared and contrasted the evidence framework and a Markov Chain Monte Carlo method for training neural networks on the task of labelling regions of segmented outdoor scenes, using a network with 30 hidden units.

The generalisation performances obtained by the MLP trained using the EF and MCMC methods have been thoroughly investigated for different amounts of training data by analysing empirical learning curves using a two-way ANOVA design. Our results show that on the task of labelling images and for the given network size, the MCMC method obtains better generalisation performances than the EF, revealing statistically significant differences in the generalisation errors for most of the sizes of the datasets. For large amounts of training data, the difference in generalisation errors (with its 95% interval of variability) lies within the interval $\Delta E^g(n) \in [0\%, 7\%]$; the difference between the two methods becomes larger for datasets with 91 and 46 data (i.e. $\Delta E^g(n) \in [20\%, 30\%]$ and $\Delta E^g(n) \in [25\%, 35\%]$, respectively), although it is affected by large variance due to the choice of the datasets and the random seed.

Our results illustrate the breakdown of the Gaussian approximation used in the EF for small n (i.e. $n = 46$ and $n = 91$); for larger n the performance of the two methods is quite similar (although the difference is statistically significant for $n \geq 182$ except for $n = 365$). It would be very interesting to study at what value of n this transition occurs for other learning problems.

We are aware of only one other paper that compares the EF and MCMC methods, namely (Husmeier et al., 1999). That paper has a rather different focus to ours, in that the authors address issues of model selection (and multiple optima within the weight space of a model) and use multiple training tasks. However, they do not make use of the 2-way ANOVA design, which permits a careful assessment of the statistical significance between different error rates. We would recommend that any further investigation of the two methods should aim to benefit from the strengths of these two studies so that better comparisons can be made.

Acknowledgements

This research forms part of the "Validation and Verification of Neural Network Systems" project funded jointly by EPSRC (GR/K 51792) and British Aerospace. FV was supported by a studentship from British Aerospace. We thank Dr. Andy Wright of BAe for helpful discussions, Prof. Radford Neal for discussions concerning the two-way ANOVA analysis and Drs. Neill Campbell and William P. J. Mackeown who provided information about the database. We also thank the anonymous referees and action editor for their comments which have helped improve this paper. Much of this research was carried out when both authors were members of the Neural Computing Research Group at Aston University.

References

Barber, D. and Bishop, C. M. (1998). Ensemble learning for Multi-Layer Networks. In Mozer, M. C.and Jordan, M. I. and Petsche, T., editors, *Advances in Neural Information Processing Systems 10*. MIT Press.

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

Brooks, R. A. (1983). Model-based 3D interpretation of 2D images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):140–150.

Clark, A. (1995). Computer Vision for Outdoor Scene Analysis. Master's thesis, University of Bristol, Bristol, United Kingdom.

Cowles, M. K. and Carlin, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *J. American Statistical Assn.*, 91(434):883–904.

DeGroot, M. H. (1984). *Probability and Statistics*. Addison-Wesley, Reading, Mass., second edition.

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7).

Draper, B. A., Collins, R. T., Brolio, J., Hanson, A. R., and Riseman, E. M. (1989). The Schema System. *The International Journal of Computer Vision*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Gay, M. (1989). Segmentation using region merging with edges. In *Proceedings* 5^{th} *Alvey Vision Conference*, pages 115–119, Reading, UK. Sheffield University Press.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis*. Chapman and Hall.

Hertz, J., Krogh, A., and Palmer, G. (1991). *Introduction to the Theory of Neural Networks*. Addison-Wesley, Redwood City, CA.

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple by minimising the description length of the weights. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 5–13, New York, NY. ACM Press.

Husmeier, D., Penny, W. D., Roberts, S. J. (1999). An empirical evaluation of Bayesian sampling with hybrid Monte Carlo for training neural network classifiers. *Neural Networks*, 12:677-705.

Lindman, H. R. (1974). *Analysis of variance in Complex Experimental Design*. W. H. Freeman and Company, San Francisco.

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

MacKay, D. J. C. (1992b). The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736.

Mackeown, W. P. J. (1994). *A Labelled Image Database and its Application to Outdoor Scene Analysis.* PhD thesis, University of Bristol, Bristol, United Kingdom.

McKeown, D. M., Harvey, W. A., and McDermott, J. (1985). Rule-based interpretation of aerial imagery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(5).

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

Ohta, Y. (1985). *Knowledge based interpretation of Outdoor Natural Scenes.* Pittmann, London.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, Cambridge. Second edition.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression.* PhD thesis, Department of Computer Science, University of Toronto, Canada.

Rasmussen, C. E., Neal, R. M., Hinton, G. E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., and Tibshirani, R. (1996). *The DELVE Manual.* Department of Computer Science, University of Toronto, Canada, 1.1 edition. Delve is available at the `URL` address `http://www.cs.utoronto.ca/~delve/`.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Vivarelli, F. (1998). *Studies on generalisation in Gaussian processes and Bayesian neural networks.* PhD thesis, Neural Computing Research Group, Aston University, Birmingham, United Kingdom.

Vivarelli, F. and Williams, C. K. I. (1997). Using Bayesian neural networks to classify segmented images. In *Proceedings of the Fifth IEE International Conference on Artificial Neural Networks*, pages 268–273, London. Institution of Electrical Engineers.

Wright, W. A. (1989). Image labelling with a neural network. In *Proceedings* 5^{th} *Alvey Vision Conference*, pages 227–232, Reading, UK. Sheffield University Press.

Wright, W. A., Mackeown, W. P. J., and Greenway, P. (1995). The use of neural networks for region labelling and scene understanding. In Taylor, J. G., editor, *Neural Networks*, pages 165–192. Alfred Waller.

		Training set		Test set	
	Number of regions	5832		1505	
	Class	Region (%)	Area (%)	Region (%)	Area (%)
1	Clouds	7.90	8.18	6.38	11.03
2	Vegetation	27.35	35.37	26.25	33.19
3	Road Marking	1.68	0.20	1.79	0.17
4	Road Surface	15.29	39.30	13.02	40.40
5	Road Border	9.88	9.32	8.04	5.57
6	Building	20.82	4.20	24.98	5.19
7	Bounding object	7.13	2.08	6.45	2.17
8	Road sign	0.77	0.05	0.93	0.15
9	Telegraph Pole	2.45	0.26	2.79	0.23
10	Illumination Shadow	2.52	0.40	2.13	0.34
11	Mobile Object	4.20	0.65	7.24	1.56

Table 1: The table shows the labels used and the composition of the training and testing sets. For both data sets the first column gives the percentage of each class computed from the number of regions of the class in the data set against the total number of regions; the second column reports the percentage of each class computed from the total area of that class against the total area of all the regions in the data set.

n	$\hat{E}^g_{\mathrm{MCMC}}$ $(L = 100)$	$\hat{E}^g_{\mathrm{MCMC}}$ $(L = 1000)$
91	0.43 ± 0.02	0.43 ± 0.02
2916	0.32 ± 0.02	0.30 ± 0.03

Table 2: A comparison of the classification error rates using $L = 100$ leapfrog steps and $L = 1000$ leapfrog steps in each iteration of the Hybrid Monte Carlo algorithm, for data set sizes of $n = 91$ and $n = 2916$. The test set size was 1505. The figures after the $\pm$ symbol indicate 2 standard deviation confidence intervals.

	MLR		MLP	
	EF	MCMC	EF	MCMC
E^r (%)	34.8 ± 2.4	32.4 ± 2.4	32.8 ± 2.4	31.0 ± 2.3
E^a (%)	11.8	8.6	9.8	9.7

Table 3: The table shows the region-based and area-based misclassification rates achieved by the MLR and MLP methods on the test set. The overall region-based misclassification rate is defined as the fraction of regions which have not been correctly classified, i.e. $E^r = N/N_t$, where N is the number of regions incorrectly classified and N_t is the total number of regions of the test set. The two standard deviation error bars associated with the region-based misclassification rate can be estimated modelling the probabilities of incorrect classification with binomial probabilities distribution with parameters (N_t, E^r) (Ripley, 1996), thus the variance of E^r is $E^r\,(1 - E^r)\,/N_t$. The area-based misclassification rate is defined as $E^a = A/A_t$, where A is the area of the regions incorrectly labelled and A_t is the overall area of the regions of the test set.

n	$\hat{E}^g_{\mathrm{EF}}$	$\hat{E}^g_{\mathrm{MCMC}}$	$\Delta\hat{E}^g$
46	0.82 ± 0.05	0.51 ± 0.03	0.30 ± 0.05
91	0.70 ± 0.04	0.43 ± 0.02	0.26 ± 0.05
182	0.42 ± 0.02	0.39 ± 0.02	0.03 ± 0.02
365	0.37 ± 0.02	0.36 ± 0.02	0.01 ± 0.01
729	0.37 ± 0.02	0.34 ± 0.02	0.03 ± 0.01
1458	0.38 ± 0.03	0.32 ± 0.02	0.06 ± 0.01
2916	0.35 ± 0.03	0.32 ± 0.02	0.03 ± 0.02

Table 4: The table shows the average of the generalisation error reported by the MLP trained with EF and MCMC by varying the composition of the datasets; the third column shows the expected pairwise difference. The estimate of the means and their intervals of variability are obtained according to the two-way ANOVA analysis. The uncertainty associated with each value is the 95% confidence interval due to the stochastic effects of the training data, the test points and the initialisation of the random seeds.

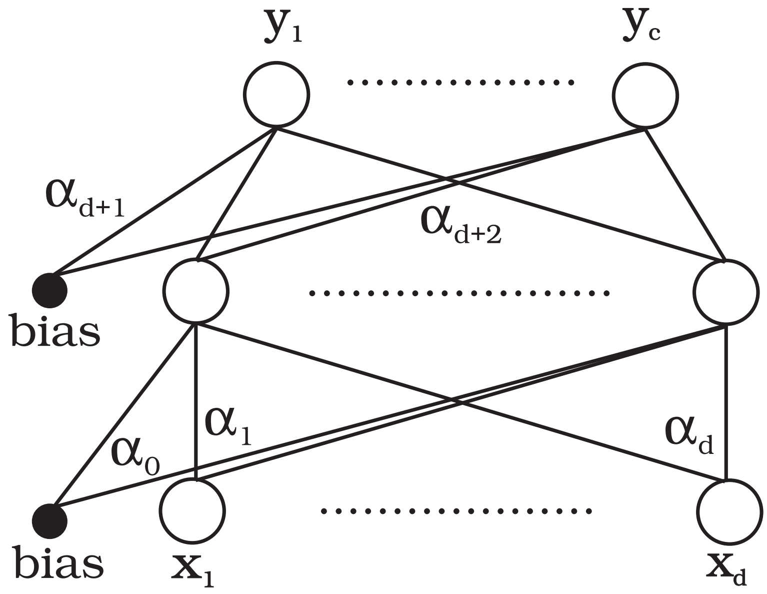
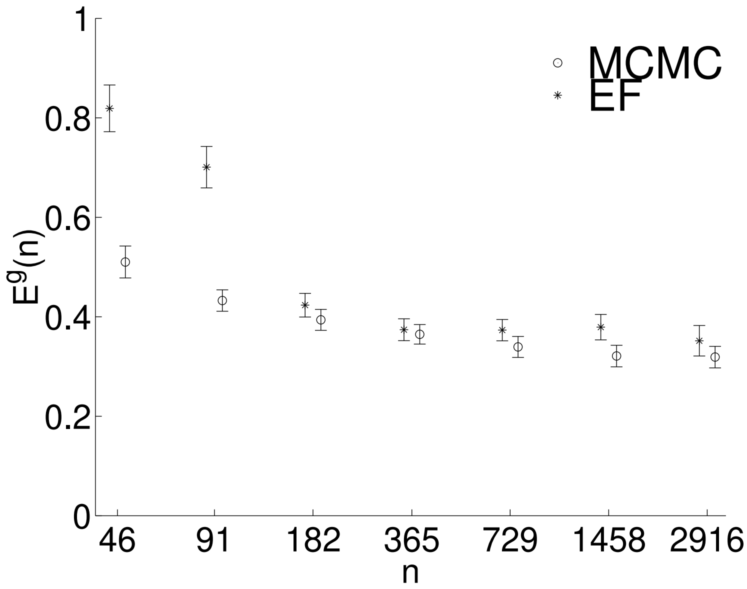
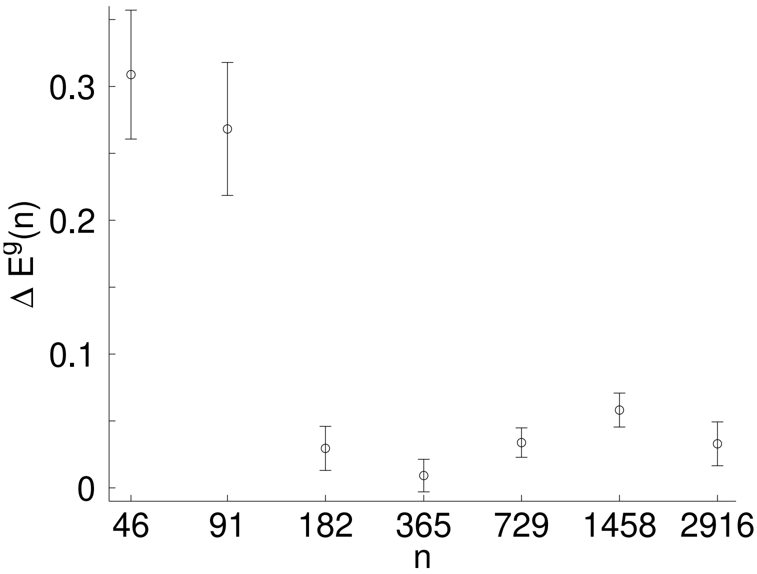


Figure 1: Graphical representation of a 2-layer neural network (MLP) with Automatic Relevance Determination. α_0 controls the biases of the hidden units; the hyperparameters $(\alpha_1, \dots \alpha_d)$ control the groups of synaptic weights connecting each input to the hidden layer; α_{d+1} controls the biases of the output layer and α_{d+2} controls the weights connecting the hidden to the output units.

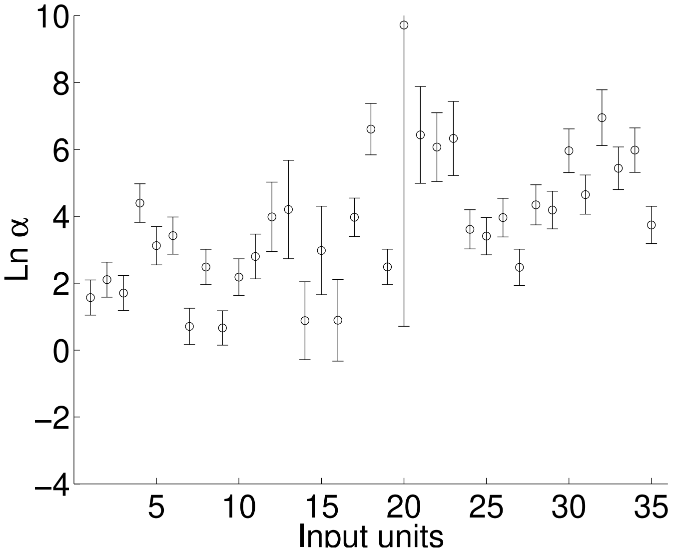


(a)

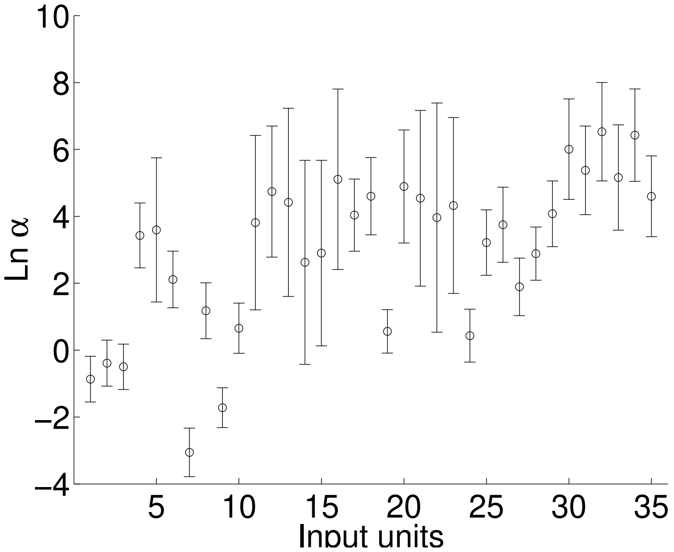


(b)

Figure 2: Figure 2(a) shows the empirical learning curves of the MLP trained with the EF and MCMC algorithms; the symbols $*$ and $\circ$ label the performances of the EF and MCMC respectively. For clarity, the symbols for the EF and MCMC methods have been displaced slightly to the left and right of the marked data set size. Figure 2(b) shows the graph of the empirical $\Delta E^g(n)$ as a function the amount of training data. The estimate of the means and their intervals of variability are obtained according to the two-way ANOVA analysis, as explained in the text. The error bars are generated by the 95% confidence interval due to the stochastic effects of the training data, the test points and the initialisation of the random seeds.



(a)



(b)

Figure 3: Graphs of the hyperparameters determined by EF (Figure 3(a)) and MCMC (Figure 3(b)) for the MLP with ARD. The labels of the input units are displayed on the x axis. The features are the mean intensity (input 1), hue angle (2, 3), topological descriptors $(4-6)$, size of the region (7), coordinates of the centroid (8,9), shape $(10-16)$ and texture $(17-19)$ descriptors, and contextual features describing the intensity ratios $(20-23)$, size ratios $(24-27)$, and the relative positions $(28-35)$ of the four largest surrounding regions. $\ln(\alpha_i)$, $i = 1, \ldots 35$ is reported on the y-axis along with two standard-deviation error bars. The error bars shown in Figure 3(a) have been estimated from Equation 9; those in Figure 3(b) have been computed from the sample variance. The upper portion of the error bar for α_{20} in panel(a) has been truncated.