1999 IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications
Kuusamo, Finland, June 16–18, 1999

# Bayesian Neural Networks for Industrial Applications

Aki Vehtari and Jouko Lampinen
Laboratory of Computational Engineering, Helsinki University of Technology
P.O.Box 9400, FIN-02015 HUT, FINLAND

*Abstract*— **We demonstrate the advantages of using Bayesian neural networks in regression, inverse and classification problems, which are common in industrial applications. The Bayesian approach provides consistent way to do inference by combining the evidence from data to prior knowledge from the problem. A practical problem with neural networks is to select the correct complexity for the model, i.e., the right number of hidden units or correct regularization parameters. The Bayesian approach offers efficient tools for avoiding overfitting even with very complex models, and facilitates estimation of the confidence intervals of the results. In this contribution we review the Bayesian methods for neural networks and present comparison results from case studies in prediction of the quality properties of concrete (regression), electrical impedance tomography (inverse problem) and forest scene analysis (classification). The Bayesian networks provided consistently better results than other methods.**

## I. Introduction

In classification and non-linear function approximation neural networks have become very popular in recent years. With neural networks the main difficulty is in controlling the complexity of the model. Another problem of standard neural network models is the lack of tools for analyzing the results (confidence intervals, like 10 % and 90 % quantiles, etc.).

The Bayesian approach provides consistent way to do inference by combining the evidence from data to prior knowledge from the problem. Bayesian methods use probability to quantify uncertainty in inferences and the result of Bayesian learning is a probability distribution expressing our beliefs regarding how likely the different predictions are. Predictions are made by integrating over the posterior distribution. In case of insufficient data the prior dominates the solution, and the effect of the prior diminishes with increased evidence from the data.

For neural networks, MacKay introduced Bayesian approach [1] based on Gaussian approximation. Neal has introduced hybrid Monte Carlo method [2] that facilitates Bayesian learning for neural networks with no compromising approximations. The main advantages of Bayesian neural networks are:

- Automatic complexity control: Bayesian inference techniques allow the values of regularization coefficients to be selected using only the training data, without the need to use separate training and validation data.
- Possibility to use prior information and hierarchical models for the hyperparameters.
- Predictive distributions for outputs.

In this contribution we demonstrate the advantages of Bayesian neural networks in three case problems. First we briefly review Multi Layer Perceptron network in section II and in section III we give a review of the Bayesian methods for neural networks. In section IV we present results using Bayesian neural networks in a regression problem for predicting the quality properties of concrete. In section V we present results using Bayesian neural networks to solve inverse problem in image reconstruction and void fraction estimation in electrical impedance tomography. Results comparing Bayesian neural networks and other classification methods for classification of objects in forest scenes are presented in section VI.

## II. Multi Layer Perceptron

In this section we briefly review Multi Layer Perceptron (MLP) neural network. See [3] for thorough introduction to MLPs. We concentrate here to one hidden layer MLP networks with hyperbolic tangent (tanh) activation function, but Bayesian methods described can be used for other types of neural networks too. Basic MLP network model with $k$ outputs is

$$f_k(\mathbf{x}, \mathbf{w}) = w_{k0} + \sum_{j=1}^{m} w_{kj} \tanh\left(w_{j0} + \sum_{i=1}^{d} w_{ji} x_i\right), \tag{1}$$

where $\mathbf{x}$ is a $d$-dimensional input vector, $\mathbf{w}$ denotes weights and indices $i$ and $j$ correspond to hidden and output units, respectively.

MLP is often considered as a generic semiparametric model which means that the effective number of parameters may be less than the number of available parameters. Effective number of parameters determines the complexity of the model. For small weights the network mapping is almost linear and has low effective complexity, since the central region of sigmoidal activation function can be approximated by linear transformation. Traditionally complexity of MLP has been controlled with early stopping or weight decay [3].

In early stopping weights are initialized to very small values. Part of the training data is used to train the MLP and the other part is used to monitor the validation error. Iterative optimization algorithms used for minimizing the training error gradually take parameters in use. Training is stopped when the validation error begins to increase. Since training is stopped before a minimum of the training error, the effective number of parameters remains less than the number of available parameters.

The basic early stopping is rather inefficient, as it is very sensitive to the initial conditions of the network

and only part of the available data is used to train the model. These limitations can easily be alleviated by using a committee of early stopping networks, with different partitioning of the data to training and stopping sets for each network. When used with caution MLP early stopping committee is good baseline method for neural networks.

In weight decay penalizing term is added to the error function. Using sum of squares of weights the weights are encouraged to be small. In practice each layer in an MLP should have different regularization parameter [3], giving the penalty term

$$\alpha_1 \sum_{j,i} w_{ji}^2 + \alpha_2 \sum_{j,k} w_{kj}^2. \tag{2}$$

Problem is how to select good values for $\alpha_i$. Traditionally this has been done with cross validation (CV). Since CV gives noisy estimate for error, it does not guarantee that good values for $\alpha_i$ can be found. Also it becomes easily computationally prohibitive as computational expenses grow exponentially with number of parameters to be selected.

## III. Bayesian Learning for MLP

Bayesian methods use probability to quantify uncertainty in inferences and the result of Bayesian learning is a probability distribution expressing our beliefs regarding how likely the different predictions are. Bayesian paradigm offers consistent way to do inference using models with even very large number of parameters. See, e.g., [4] for good introduction to Bayesian methods.

### A. Bayesian Learning

Consider a regression or classification problem involving the prediction of a noisy vector $\mathbf{y}$ of target variables given the value of a vector $\mathbf{x}$ of input variables.

The process of Bayesian learning is started by defining a model, $\mathcal{M}$, and prior distribution $p(\theta)$ for the model parameters. Prior distribution expresses our initial beliefs about parameter values, before any data has observed. After observing new data $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$, prior distribution is updated to the posterior distribution using Bayes' rule

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto L(\theta|D)p(\theta), \tag{3}$$

where the likelihood function $L(\theta|D)$ gives the probability of the observed data as function of the unknown model parameters.

To predict the new output $\mathbf{y}^{(n+1)}$ for new input $\mathbf{x}^{(n+1)}$, predictive distribution is obtained by integrating the predictions of the model with respect to the posterior distribution of the model parameters

$$p(\mathbf{y}^{(n+1)}|D) = \int p(\mathbf{y}^{(n+1)}|\mathbf{x}^{(n+1)}, \theta)p(\theta|D)d\theta. \tag{4}$$

This is same as taking the average prediction of all the models weighted by their goodness.

Note that predictive distribution for $\mathbf{y}^{(n+1)}$ is implicitly conditioned on hypotheses that hold throughout and to be more explicit notation as the following might be used

$$p(\mathbf{y}^{(n+1)}|D, H) = \int p(\mathbf{y}^{(n+1)}|\mathbf{x}^{(n+1)}, \theta, H)p(\theta|D, H)d\theta, \tag{5}$$

where $H$ refers to the set of hypotheses or assumptions used to define the model.

### B. Models

Statistical model is defined with the likelihood function, which in case of independent and exchangeable data points is given by

$$L(\theta|D) = \prod_{i=1}^{n} p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \theta), \tag{6}$$

where $n$ is the number of data points.

In the likelihood equation the term $p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \theta)$ depends on our problem. In regression problems, it is generally assumed that the distribution of target data can be described by a deterministic function of inputs, corrupted by additive Gaussian noise of a constant variance. Probability density for a target $y_j$ is then

$$p(y_j|\mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{1}{2\sigma_j^2}[y_j - f_j(\mathbf{x}, \mathbf{w})]^2\right), \tag{7}$$

where $\sigma_j^2$ is the noise variance for the target. See [2] for per-case normal noise variance model. For a two class classification (logistic regression) model, the probability that a binary-valued target, $y_j$, has the value 1 is

$$p(y_j = 1|\mathbf{x}, \mathbf{w}) = [1 + \exp(-f_j(\mathbf{x}, \mathbf{w}))]^{-1} \tag{8}$$

and for many class classification (softmax) model, the probability that a class target, $y$, has value $j$ is

$$p(y = j|\mathbf{x}, \mathbf{w}) = \frac{\exp(f_j(\mathbf{x}, \mathbf{w}))}{\sum_k \exp(f_k(\mathbf{x}, \mathbf{w}))}. \tag{9}$$

In (7), (8) and (9) function $f_j(\mathbf{x}, \mathbf{w})$ is in this case an MLP network. Traditionally in many methods one of the problems has been to find a good topology for the MLP. In Bayesian approach we could use infinite number of hidden units [2]. We do not need to restrict the size of the MLP based on the size of the training set, but in practice, we will have to use finite number of hidden units due to computational limits. MCMC methods (section III-E) produce the correct answer eventually, but it may sometimes take unreasonable amount of time[2].

### C. Priors

Next, we have to define the prior information about our model parameters, before any data has been seen. Usual prior is that the model has some unknown complexity but the model is not constant or extremely flexible. To express this prior belief we set hierarchical model specification.

Parameters $\mathbf{w}$ define the model $f(\mathbf{x}, \mathbf{w})$. As discussed in section II, complexity of the MLP network can be controlled by controlling the size of the weights $\mathbf{w}$. Corresponding prior to weight decay is to use Gaussian prior distribution for weights $w$ given hyperparameter $\alpha$

$$p(\mathbf{w}|\alpha) = (2\pi)^{-m/2}\alpha^{m/2}\exp(-\alpha\sum_{i=1}^{m} w_i^2/2). \quad (10)$$

This prior states that smaller weights are more probable, but how much more is determined by the value of hyperparameter $\alpha$. Since we do not know the correct value for hyperparameter $\alpha$, we set a vague hyperprior $p(\alpha)$ expressing our belief that complexity controlled by $\alpha$ is unknown but the model is not constant or extremely flexible. A convenient form for this hyperprior is vague Gamma distribution with mean $\mu$ and shape parameter $a$

$$p(\alpha) \sim \text{Gamma}(\mu, a) \propto \alpha^{a/2-1}\exp(-\alpha a/2\mu). \quad (11)$$

In order to have prior for weights which is invariant under the linear transformations of data, separate priors (each having its own hyperparameters $\alpha_i$) for different weight groups in each layer of a MLP are used.

In MLP networks, the weights from less important inputs are typically smaller than weights from more important inputs[1]. Prior belief that some inputs are likely to be more relevant than others can be implemented by using different priors for weight groups from each input, and hierarchical hyperpriors for these priors. The posteriors for hyperparameters should then adjust according to relevance of the inputs. This prior is called Automatic Relevance Determination (ARD) [5, 2].

For regression models we need prior for $\sigma$ in (7), which is often specified in terms of corresponding precision, $\tau = \sigma^{-2}$. As for $\alpha$, our prior information is usually quite vague, stating that $\sigma$ is not zero or extremely large. This prior can be expressed with vague Gamma-distribution with mean $\mu$ and shape parameter $a$

$$p(\tau) \sim \text{Gamma}(\mu, a) \propto \tau^{a/2-1}\exp(-\tau a/2\mu). \quad (12)$$

### D. Prediction

After defining the model and prior information, we combine the evidence from the data to get the posterior distribution for the parameters

$$p(\mathbf{w}, \alpha, \tau|D) \propto L(\mathbf{w}, \alpha, \tau|D)p(\mathbf{w}, \alpha, \tau). \quad (13)$$

Predictive distribution for new data is then obtained by integrating over this posterior distribution

$$p(\mathbf{y}^{(n+1)}|\mathbf{x}^{(n+1)}, D) = \int p(\mathbf{y}^{(n+1)}|\mathbf{x}^{(n+1)}, \mathbf{w}, \alpha, \tau)p(\mathbf{w}, \alpha, \tau|D)\, d\mathbf{w}\alpha\tau. \quad (14)$$

We can also evaluate expectations of various functions with respect to the posterior distribution for parameters. For example in regression we may evaluate the expectation for a component of $\mathbf{y}^{(n+1)}$

$$\hat{\mathbf{y}}_k^{(n+1)} = \int f_k(\mathbf{x}^{(n+1)}, \mathbf{w})p(\mathbf{w}, \alpha, \tau|D)\, d\mathbf{w}\alpha\tau, \quad (15)$$

which corresponds to the best guess with squared error loss.

The posterior distribution for the parameters $p(\mathbf{w}, \alpha, \tau|D)$ is typically very complex, with many modes. Evaluating the integral of (15) is therefore a difficult task.

The integral can be approximated with Gaussian approximations to modes. Then predictive distribution is approximated by the corresponding integral with respect to the Gaussian [1, 6]. Or we can use Monte Carlo methods, described next, to numerically approximate the integral.

### E. Markov Chain Monte Carlo method

Neal has introduced implementation of Bayesian learning for neural networks in which the difficult integration of (15) is performed using Markov Chain Monte Carlo (MCMC) methods [2]. In [7] there is a good introduction to basic MCMC methods and many applications in statistical data analysis.

MCMC methods make no assumptions about the form of the posterior distribution. They may in some circumstances require a very long time to converge to the desired distribution.

The integral of (15) is the expectation of function $f_k(\mathbf{x}^{(n+1)}, \mathbf{w})$ with respect to the posterior distribution of the parameters. This and other expectations can be approximated by Monte Carlo method, using a sample of values $\mathbf{w}^{(t)}$ drawn from the posterior distribution of parameters

$$\hat{\mathbf{y}}_k^{(n+1)} \approx \frac{1}{N}\sum_{t=1}^{N} f_k(\mathbf{x}^{(n+1)}, \mathbf{w}^{(t)}). \quad (16)$$

Note that samples from the posterior distribution are drawn during the learning phase and predictions for new data can be calculated quickly using the same samples and (16).

In the MCMC, samples are generated using a Markov chain that has the desired posterior distribution as its stationary distribution. Difficult part is to

[1]Note that in the non-linear network the effect of an input may be small even if the weights from it are large and vice verse, but in general the size of the weights roughly reflects the relevance of the input.

create Markov chain which converges rapidly and in which states visited after convergence are not highly dependent.

Neal used the hybrid Monte Carlo (HMC) algorithm [8] for parameters and Gibbs sampling for hyperparameters. HMC is an elaborate Monte Carlo method, which makes efficient use of gradient information to reduce random walk behavior. The gradient indicates in which direction one should go to find states with high probability. Use of Gibbs sampling for hyperparameters helps to minimize the amount of tuning that is needed to obtain good performance in HMC.

When the amount of data increases, the evidence from data causes the probability mass to concentrate to the smaller area and we need less samples from the posterior distribution. Also less samples are needed to evaluate the mean of the predictive distribution than the tail-quantiles like, 10% and 90% quantiles. So depending on the problem 10–20 samples may be enough (given that samples are not too highly dependent).

In our examples of Bayesian learning for neural networks with MCMC we have used Flexible Bayesian Modeling (FBM), software[2], which implements the methods described in [2].

## IV. Case I: Regression in Concrete Quality Estimation

The goal of the project was to develop a model for predicting the quality properties of concrete. The quality variables contained, e.g., compression strengths and densities for 1, 28 and 91 days after casting, bleeding (water extraction) and spread and slump that measure softness of the fresh concrete. These quality measurements depend on the properties of the stone material (natural or crushed, size and shape distributions of the grains, mineralogical composition), additives, and amount of cement and water. In the study we had 7 target variables and 19 explanatory variables.

Collecting the samples for statistical modeling is rather expensive in this application, as each sample requires preparation of the sand mixture, casting the test pieces and waiting for 91 days for the final tests. Thus available samples must be used as efficiently as possible, which makes Bayesian techniques tempting alternative, as they allow fine balance of prior assumptions and evidence from samples. In the study we had 149 samples designed to cover the practical range of the variables, collected by a concrete manufacturer company.

MLP networks containing 6 hidden units were used. Different MLP models tested were:

*MLP ESC* : Early stopping committee of 20 MLP networks, with different division of data to training and stopping sets for each member. The networks were initialized to near zero weights to guarantee that the mapping is smooth in the beginning.

[2]<URL:http://www.cs.toronto.edu/~radford/fbm.software.html>

TABLE I
Ten fold cross-validation error estimates for predicting the slump of concrete.

| Method | Root mean square error |
|---|---|
| MLP ESC | 37 |
| Bayes MLP | 34 |
| Bayes MLP +ARD | 27 |

*Bayes MLP* : Bayesian neural network with FBM-software, using vague priors and MCMC-run specifications similar as used in [2]. 20 networks from the posterior distribution of network parameters were used.

*Bayes MLP +ARD:* Similar Bayesian neural network to the previous, but using also ARD prior.

Error estimates for predicting the slump are collected in Table I. Note that use of ARD prior gives much better results.

## V. Case II: Inverse Problem in Electrical Impedance Tomography

In this section we report results on using Bayesian neural networks for solving the ill-posed inverse problem in electrical impedance tomography, EIT. The full report of the proposed approach is presented in [9].

The aim in EIT is to recover the internal structure of an object from surface measurements. Number of electrodes are attached to the surface of the object and current patterns are injected from through the electrodes and the resulting potentials are measured. The inverse problem in EIT, estimating the conductivity distribution from the surface potentials, is known to be severely ill-posed, thus some regularization methods must be used to obtain feasible results [10].

Fig. 1 shows a sample bubble and resulting equipotential curves. The potential signals from which the image is to be recovered are shown in Fig. 2.

In [9] we proposed a novel feedforward solution for the reconstruction problem.The approach is based on transformation of both input and output data by principal component projection and application of the neural network in this lower dimensional eigenspace.

The reconstruction was based on 20 principal components of the 128 dimensional potential signal and 60 eigenimages with resolution $41 \times 41$ pixels. The training data consisted of 500 simulated bubble formations with one to ten overlapping circular bubbles in each image. To compute the reconstructions MLP networks containing 30 hidden units were used. Models tested were *MLP ESC* and *Bayes MLP* (see section IV).

Fig. 3 shows examples of the image reconstruction results. Table II shows the quality of the image reconstructions with models, measured by error in the void fraction and percentage of erroneous pixels in the segmentation, over the test set.

An important goal in the studied process tomography application was to estimate the void fraction, which is the proportion of gas and liquid in the image. With the proposed approach such goal variables can be estimated directly without explicit reconstruction

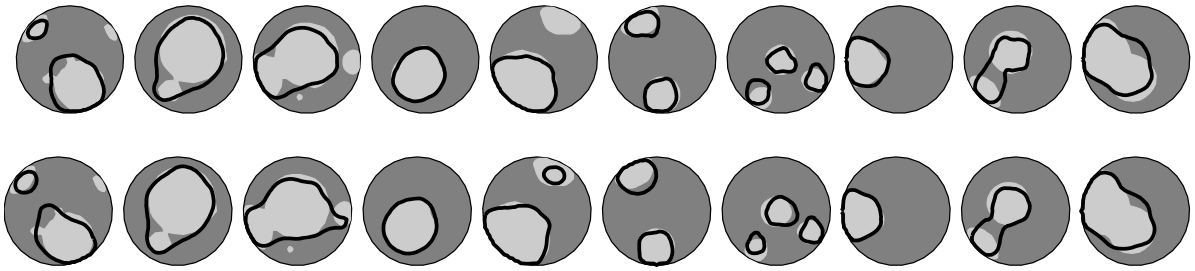

Fig. 3. Example of image reconstructions with MLP ESC (upper row) and the Bayesian MLP (lower row)

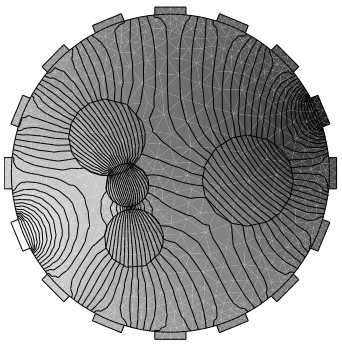

Fig. 1. Example of the EIT measurement. The simulated bubble formation is bounded by the circles. The current is injected from the electrode with the lightest color and the opposite electrode is grounded. The gray level and the contour curves show the resulting potential field.

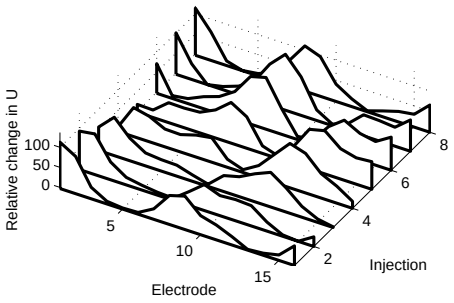


Fig. 2. Relative changes in potentials compared to homogenous background. The eight curves correspond to injections from eight different electrodes.

TABLE II

Errors in reconstructing the bubble shape and estimating the void fraction from the reconstructed images. See text for explanation of the models.

| Method | Classification error % | Relative error in VF % |
|---|---|---|
| MLP ESC | 6.7 | 8.7 |
| Bayes MLP | 5.9 | 8.1 |
| Bayes MLP, direct VF | | 3.4 |

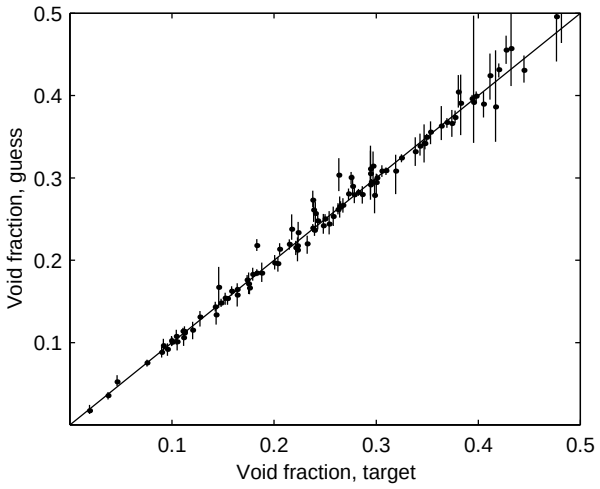


Fig. 4. Scatterplot of the void fraction estimate with 10% and 90% quantiles.

of the image. The bottom row in Table II shows the relative absolute error in estimating the void fraction directly from the projections of the potential signals.

With Bayesian methods we can easily calculate confidence intervals for outputs. Fig. 4 shows the scatter plot of the void fraction versus the estimate by the Bayesian neural network. The 10% and 90% quantiles are computed directly from the posterior distribution of the model output.

See [9] for results for effect of additive Gaussian noise to the performance of the method.

## VI. Case III: Forest Scene Analysis

In this section we report results of using Bayesian neural networks for classification of forest scenes, to accurately recognize and locate the trees from any background.

Forest scene classification task is demanding due to the texture richness of the trees, occlusions of the forest scene objects and diverse lighting conditions under operation. This makes it difficult to determine which are optimal image features for the classification. A natural way to proceed is to extract many different types of potentially suitable features.

In [11] we extracted total of 84 statistical and Gabor features over different sized windows at each spectral channel. Due to great number of features used, many classifier methods would suffer from the curse of dimensionality, but Bayesian neural networks manage well in high dimensional problems.

The image data for teaching and testing of the classifiers was collected by using an ordinary digital camera

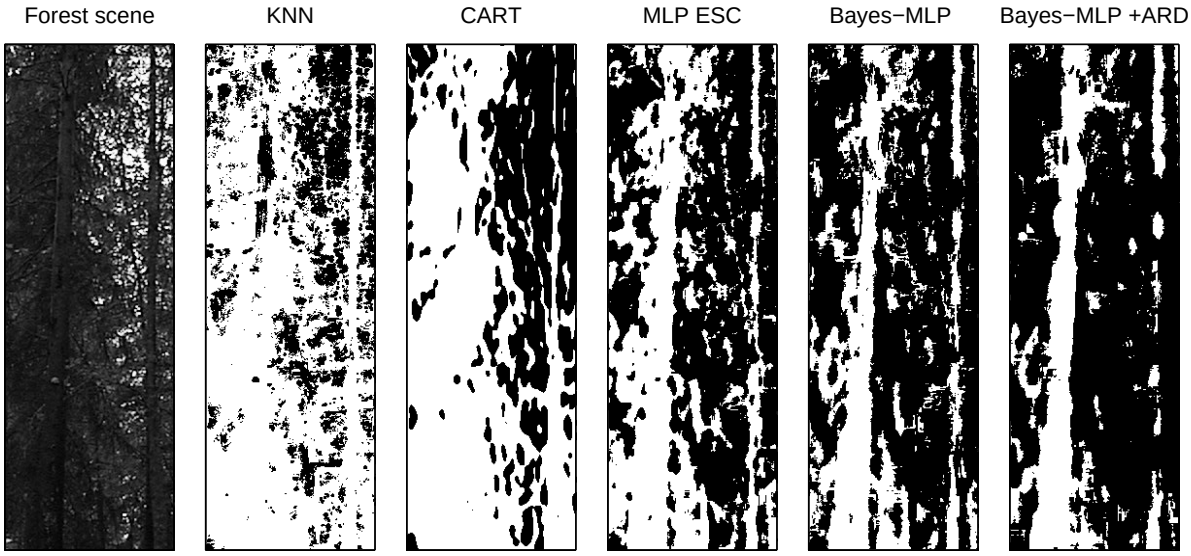


Fig. 5. Examples of classified forest scene. See text for explanation of the different models.

TABLE III
CV error estimates for forest scene classification. See text for explanation of the different models.

| | *Error%* |
|---|---|
| KNN LOOCV | 20 |
| CART | 30 |
| MLP ESC | 13 |
| Bayes MLP | 12 |
| Bayes MLP +ARD | 11 |

in varying weather conditions. Ideal weather conditions were not searched, as the aim was to test the viability and the robustness of the methods. Total of 48 images were taken. The labeling of the image data was done by hand via identifying many types of tree and background image blocks with different textures and lighting conditions. In this study only pines were considered.

Addition to 20 hidden unit MLP models *MLP ESC*, *Bayes MLP* and *Bayes MLP +ARD* (see section IV ) the models tested were:

*KNN LOOCV* : K-nearest-neighbor. K is chosen by leave-one-out cross-validation on the training set.

*CART* : Classification And Regression Tree [12].

Eight folded cross-validation (CV) error estimates are collected in Table III. Fig. 5 shows example image classified with different methods.

## VII. Summary discussion

Above case problems in industrial applications illustrate the advantages of using Bayesian neural networks. The approach contains automatic complexity control, without the need to use separate training and validation data. We can use large number of inputs. It is possible to use prior information, like ARD. The Bayesian approach gives the predictive distributions for outputs, which can be used to estimate reliability of the predictions.

## Acknowledgments

This study was partly funded by TEKES Grant 40888/97 (Project *PROMISE, Applications of Probabilistic Modeling and Search*).

## References

[1] David J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[2] Radford M. Neal, *Bayesian Learning for Neural Networks*, vol. 118 of *Lecture Notes in Statistics*, Springer-Verlag, July 1996.

[3] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[4] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald R. Rubin, *Bayesian Data Analysis*, Texts in Statistical Science. Chapman & Hall, 1995.

[5] David J. C. MacKay, "Bayesian non-linear modelling for the prediction competition," in *ASHRAE Transactions, V.100, Pt.2*, Atlanta Georgia, 1994, pp. 1053–1062, ASHRAE.

[6] David J. C. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.

[7] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, Eds., *Markov Chain Monte Carlo in Practice*, Chapman & Hall, 1996.

[8] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, pp. 216–222, 1987.

[9] Jouko Lampinen, Aki Vehtari, and Kimmo Leinonen, "Using Bayesian neural network to solve the inverse problem in electrical impedance tomography," in *Proceedings of 11th Scandinavian Conference on Image Analysis SCIA'99*, Kangerlussuaq, Greenland, June 1999.

[10] M. Vauhkonen, J. P. Kaipio, E. Somersalo, and P. A. Karjalainen, "Electrical impedance tomography with basis constraints," *Inverse Problems*, vol. 13, no. 2, pp. 523–530, 1997.

[11] Aki Vehtari, Jukka Heikkonen, Jouko Lampinen, and Jouni Juujärvi, "Using Bayesian neural networks to classify forest scenes," in *Intelligent Robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision*, David P. Casasent, Ed., Boston, MA, USA, November 1998, vol. 3522 of *Proceedings of SPIE*, pp. 66–73.

[12] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone, *Classification and regression trees*, Chapman and Hall, 1984.