An evaluation of Bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting

Henrique S. Hippert
Depto. de Estatistica, Universidade Federal de Juiz de Fora, Brazil

James W. Taylor
Saïd Business School, University of Oxford, UK

Neural Networks, 2010, Vol. 23, pp. 386-395.

Address for Correspondence:

Henrique Hippert
Depto. de Estatistica / ICE - UFJF
Universidade Federal de Juiz de Fora
Campus Universitario
36036 900 - Juiz de Fora, MG – Brazil
Phone: +55 32 3229 3306
Email: henrique.hippert@ufjf.edu.br

An evaluation of Bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting

Abstract:

Artificial neural networks have frequently been proposed for electricity load forecasting because of their capabilities for the nonlinear modelling of large multivariate data sets. Modelling with MLPs is not an easy task though; two of the main challenges are defining the appropriate level of model complexity, and choosing the input variables. This paper evaluates techniques for automatic neural network modelling within a Bayesian framework, as applied to six samples containing daily load and weather data for four different countries. We analyse input selection as carried out by the Bayesian 'automatic relevance determination', and the usefulness of the Bayesian 'evidence' for the selection of the best structure (in terms of number of neurons), as compared to methods based on cross-validation.

Key words: Bayesian neural networks; load forecasting; input selection; Bayesian model selection

1. Introduction

Neural networks (NNs) have frequently been proposed for short-term load forecasting (STLF), because of their capabilities for nonlinear modelling of large multivariate datasets. The family of NN models known as *multilayer perceptrons* are probably the most frequently used, since they have been shown to be *universal approximators* of functions (Haykin, 1999), and can be used to model the function that relates the electric load to its exogenous variables.

Modelling with MLPs is not an easy task though; two of the main challenges are defining the appropriate level of model complexity, and choosing the input variables. The complexity of a NN is dictated, in the first instance, by its architecture; for NNs with one hidden layer with sigmoid neurons, and an output layer with linear neurons (the sort of NN most frequently used for STLF), the complexity depends mostly on the number of hidden neurons. However, the architecture is but one aspect of the problem, since the size of the weights (their absolute values) must also be taken into account. If the weights are small, the activation functions of the neurons will be operating in the central part of their ranges, which is practically linear. A large NN, in this case, would be no more complex than a linear regression model.

In order to control the complexity of a NN, one has to work on these two aspects. First, one should choose the right architecture (the appropriate number of neurons). This may done by changing the number of neurons step by step, until an optimum value has been found. The initial model may be either a very large one, from which neurons are progressively removed ('pruning' algorithms), or a small one, to which neurons are progressively added ('growing' algorithms). The choice is based on the principle of parsimony (a model should be as complex as necessary for a given task, but not more), and it requires some measure of NN complexity to be evaluated at each step. Second, one should control the size of the NN weights by using 'regularisation' techniques. These techniques add a term to the NN cost function that penalises for large weights, which ensures that the training algorithm will lead to NN weights that are as small as possible (Haykin, 1999).

Another difficult task in NN modelling (as in all nonlinear modelling) is the selection of the subset of variables to be used as inputs. Some metrics for the influence of each variable on the output are needed; a comparison of methods to evaluate this influence is done by Papadokonstantakis *et al* (2006), who consider three groups of methods: a) those that take only the data into account, before the NN modelling (statistical methods such as PCA, etc.); b) those that affect the training (such as ARD, discussed below); and c) those applied on the trained NN. For this last group, two kinds of metrics of input relevance are considered –the ones that measure the input's 'predictive importance' (i.e., the increase in the generalisation error when an input is omitted), and the ones that measure its 'causal importance' (the change in the output caused by changes in the

input) (Lampinen & Vehtari, 2001). The predictive importance may be empirically evaluated on an independent sample; the causal importance is usually measured by analytical means, such as the second derivatives of the error, information theory measures, etc.

Besides choosing architecture and inputs, the NN designer also has to tune several parameters, depending on the algorithm used, such as regularisation coefficients, momentum rate, learning rate, etc. A common way of supporting these choices and tunings is to do trial-and-error simulations on a 'validation' sample, distinct from the training and the testing ones; this procedure is called 'cross-validation' (CV).

Cross-validation is an empirical technique that can be put to many uses - comparing different architectures or input sets, avoiding overfitting, tuning training parameters, etc. However, it also has its problems. On the theoretical side, Cataltepe *et al* (1999) showed that there is no guarantee that the model selected by CV is indeed the best one. Intuitively, this is easy to understand; given an infinite number of models, it is always possible to find one that overfits both the training and the validation set, and proves to be useless out-of-sample. In real world applications, however, the number of models to be compared is usually small, and this problem does not happen. On the practical side, CV has some limitations too. First, the estimates of generalisation ability obtained on a CV sample are noisy. A different CV sample may lead to different conclusions; to overcome this, it is advisable to use several CV samples and average their results, using methods such as *k-fold*, *leave-one-out*, or *bootstrap* (Lendasse *et al*, 2003; Efron and Tibshirani, 1993).

For forecasting applications, however, this is not possible, since the CV sample must always consist of data that are more recent than the training data, but older than the data used for out-of-sample testing; this reduces the choices of possible CV samples. Also, CV may only be used to tune one discrete variable at a time. If it is necessary to optimise n parameters, and each one can assume m distinct values, it will be necessary to run the simulation m^n times, and the computational cost might easily become prohibitive.

Among the methods that have been proposed to deal with these difficulties in NN modelling, this paper focuses on the Bayesian approach. In this method, any variables of interest (weights, regularisation coefficients, number of neurons, relevance of inputs, NN outputs, etc.) are modelled by random variables for which prior distributions are assumed; after the data have been collected, posterior distributions are derived, by means of the theorem of Bayes. In principle, there are several advantages to this approach, in comparison to more traditional ones based on CV: it is possible to obtain probability distributions for the variables of interest, and not only point estimates (which allows the researcher to quantify the uncertainty by means of confidence intervals); all available data can be used for training (since there is no need to reserve data for a CV sample); the

relevance of the inputs can be assessed after the training (by a technique called *Automatic Relevance Determination*, discussed below); and the optimum number of neurons can be found (by comparing the Bayesian 'evidence' of the models). Reviews of the Bayesian approach to NNs can be found in Thodberg (1996), Penny & Roberts (1999), Lampinen and Vehtari (2001), and Titterington (2004).

The application of Bayesian theory to neural networks was started by Buntine & Weigend (1991). Mackay (1992a-b) introduced the 'evidence approximation' framework, which is based on a Gaussian approximation for the posterior distribution of the network weights. This framework simplified the mathematical treatment, and allowed the derivation of expressions to estimate the most probable values of the hyperparameters, and the most probable model. Other authors avoided the approximation by integrating the posterior with a numerical technique known as Markov Chain Monte Carlo (MCMC) (Neal, 1996; Barber and Bishop, 1997; Muller and Rios Insua, 1998; Lampinen and Vehtari, 2001), with hybrids between MCMC and genetic algorithms (Chua and Goh, 2003; Liang, 2005), or with techniques that use neither Gaussian approximations or MCMC: the variational method (Titterington, 2004), and the Bayesian conjugate prior method (Vila *et al*, 2000; Rossi and Vila, 2006). Research on Mackay's evidence approximation approach was carried further by several authors, but the conclusions about the usefulness of this technique were conflicting. Thodberg (1996) was overall favourable, but Penny and Roberts (1999), and Lampinen and Vehtari (2001) tended to be sceptical, particularly for problems where the number of training patterns was small with respect to the number of weights. Recently, however, the evidence approximation framework was applied to STLF by Silva and Ferreira (2007) and Lauret *et al* (2008), with reportedly good results.

STLF is an essential task in the daily operation of electric power systems, both for technical and financial reasons. Forecasts with lead times ranging from a few hours to one week ahead are needed to support the decisions of the system operators and market agents, in performing tasks such as load dispatching, scheduling of generation, energy trading, and purchasing of fuel. Accurate forecasts have been shown to lead not only to more security in the operation but also to considerable cost savings (Bunn, 2000). All this has increased the demand for improved forecasting methods, and a great deal of research has been devoted to this area.

In this paper we evaluate the application of Mackay's evidence approximation framework, by comparison to the CV-based techniques, using six samples. Samples I and II are series of hourly values for loads and for five weather variables, from England and Wales; Samples III and IV are series of hourly loads and temperatures, from a utility in Rio de Janeiro. In order to confirm the findings from those data, we then repeat the study on two more samples, available online, which have already been used by several other researchers: one series of hourly loads and temperatures for

a North American utility (Sample V); and one series of hourly loads and temperatures for a utility in Slovakia (Sample VI). We compare the results of the Bayesian methods to those of methods based on CV, and to a naïve method that serves as a benchmark.

The paper is organised as follows: Section 2, presents a short overview of the Bayesian approach to NN modelling; Section 3 describes the routines used in the studies; Section 4 presents the datasets and discusses the results; and Section 5 is the conclusion.

2. The Bayesian approach to neural network modelling – an overview

This section provides a short introduction to the Bayesian approach to NN modelling, summarised from Bishop (1995). The idea is to introduce the concepts of *evidence* and of *automatic relevance determination*, on which the analyses in this study are based.

Bayesian NN modelling starts by postulating that the weights $\boldsymbol{w}$ of a NN are random variables, and assuming a prior distribution $p(\boldsymbol{w})$ for them. After a sample has been observed, a posterior distribution $p(\boldsymbol{w}|D)$ is calculated by means of Bayes theorem,

$$p(\mathbf{w} \mid D) = \frac{p(D \mid \mathbf{w})\, p(\mathbf{w})}{p(D)} \tag{1}$$

where $\boldsymbol{w}$ is the weight vector and $D = \{t\}^n$ is the set of target vectors. The probability $p(D|\boldsymbol{w})$ is the *likelihood*, usually called the *evidence* in this context. Expression (1) should have all the probabilities conditioned on the input data X ; however, since the NNs do not model X, and the conditionings affect both sides of the equation, this can be omitted from the notation.

2.1. Evaluating the posterior distribution of weights

To ensure smoothing mappings, exponential models are usually assumed both for the prior distribution and for the noise added to the targets; this leads to a posterior distribution of weights given by:

$$p(\mathbf{w} \mid D) \propto \frac{1}{Z_S} \exp(-S(\mathbf{w})) \tag{2}$$

where

$$S(\mathbf{w}) = \beta E_D + \alpha E_W = \frac{\beta}{2} \sum_{n=1}^{N} \{ y(\mathbf{x}_n, \mathbf{w}) - t_n \}^2 + \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2 \tag{3}$$

and Z_S is a scaling factor. To find the most probable weight vector $\mathbf{w}_{MP}$, one should minimise the exponent $S(\mathbf{w})$. This is called *Bayesian regularisation*, and is equivalent to minimising a cost function with *weight decay* regularisation, but with an important difference – the Bayesian framework provides a method for estimating the hyperparameters α and β without resorting to cross-validation (see section 2.2).

The evaluation of the NN outputs, and of the evidence of hyperparameters and models, will require integrating the posterior distribution in (2). In order to make this analytically tractable, Mackay (1992) suggested approximating the exponent $S(\boldsymbol{w})$ by a quadratic expansion around its minimum value $S(\boldsymbol{w}_{MP})$, which leads to the Gaussian form:

$$p(\mathbf{w} \mid D) = \frac{1}{Z_S^*} \exp\left[- S(\mathbf{w}_{MP}) - \frac{1}{2} \Delta \mathbf{w}^T A \Delta \mathbf{w} \right] \tag{4}$$

where *A* is the Hessian of the regularised error in (3) with respect to the weights.

2.2. 'Evidence approximation' for the hyperparameters

The posterior distribution of the weights may be written, with explicit hyperparameters:

$$p(\mathbf{w} \mid D) = \int\int p(\mathbf{w} \mid \alpha, \beta, D)\, p(\alpha, \beta \mid D) d\alpha d\beta \tag{5}$$

There are two ways of working with this expression. One is to integrate out α and β. The other is to approximate the integral by a function of the most probable values of the hyperparameters, α_{MP} and β_{MP}. This is called *evidence approximation* (see Bishop, 1995, p. 417, for a short comparison of these two approaches).

In this approach, α_{MP} and β_{MP} are found by the maximisation of the posterior distribution of the hyperparameters:

$$p(\alpha, \beta \mid D) = \frac{p(D \mid \alpha, \beta)\, p(\alpha, \beta)}{p(D)} \tag{6}$$

If a non-informative prior is chosen, maximising the posterior is equivalent to maximising the evidence $p(D|\alpha,\beta)$, since the denominator $p(D)$ is constant for a given NN. The evidence may be written as:

$$p(D \mid \alpha, \beta) = \int p(D \mid \mathbf{w}, \alpha, \beta)\, p(\mathbf{w} \mid \alpha, \beta) d\mathbf{w} \tag{7}$$

Maximisation of (7) leads to expressions, based on the eigenvalues of the Hessian A in eq. (4), which allow the iterative estimation of the hyperparameters α and β.

2.3. Automatic relevance determination

The prior that leads to expression (3) is not the best choice, since it applies the same restrictions (α's) to the weights of different layers, which is not consistent with the scaling properties of the NNs (Bishop, 1995). Better results can be obtained if different groups of weights are controlled by different α's. For example, a NN could have the weights ***w*** that connect the input nodes to the hidden layer controlled by a different α from the weights ***v*** that connect the hidden layer to the output node. This would result in an error function defined as :

$$S(\mathbf{w}) = \frac{\beta}{2} \sum_{n=1}^{N} \{ y(\mathbf{x}_n, \mathbf{w}) - t_n \}^2 + \frac{\alpha_1}{2} \sum_{i=1}^{W} w_i^2 + \frac{\alpha_2}{2} \sum_{i=1}^{V} v_i^2 \tag{8}$$

Pursuing this idea still further, a NN might have the weights that are connected to each input node gathered together in groups, controlled by their own separate α values. The size of an α would then serve as an indicator of the relevance of the input controlled by it; if the α is very large, the minimisation process would force the weights to be very small, and the input would therefore have low relevance to the output. This is called *Automatic Relevance Determination*. ARD is a 'soft-

pruning' method, which means the inputs of low relevance are not actually removed from the NN, as in the usual (hard) pruning methods; the algorithm just reduces the influence of the irrelevant inputs in the output, by reducing their weights.

The ARD coefficients, however, are measures of relevance that are based on the absolute values of the weights. This may be criticised, for although in linear models with normalised inputs the size of the coefficient is indeed an indicator of the variable relevance, in nonlinear models the relationship between coefficient and input relevance is not so clear.

2.4. Bayesian comparison of models

The Bayesian procedure for selecting the adequate number of neurons starts with a set H of candidate models (each one characterised by a number of hidden neurons, and all having the same inputs). The posterior probabilities of each model H_i is given by

$$p(H_i \mid D) = \frac{p(D \mid H_i)P(H_i)}{p(D)} \tag{9}$$

Where $p(D|H_i)$ is the evidence for the model H_i. If the priors are all the same, and the probability $p(D)$ is constant for all models, one may compare the evidences of the models, instead of comparing the posterior probabilities. The evidences may be written as:

$$p(D \mid H_i) = \iint p(D \mid \alpha, \beta, H_i)\, p(\alpha, \beta \mid H_i) d\alpha d\beta \tag{10}$$

Supposing that the hyperparameters α and β are independent, using the Gaussian approximation to their evidence, and assuming non-informative priors, one arrives at the log evidence of a model expressed as (deleting unnecessary constant terms):

$$\begin{aligned} \ln p(D \mid H_i) \propto -\alpha^{MP} E_W^{MP} - \beta^{MP} E_D^{MP} - \frac{1}{2}\ln \mid A \mid + \frac{1}{2}\sum_{i=1}^{k} W_i \ln \alpha_i^{MP} + \frac{N}{2}\ln \beta^{MP} + \\ + \ln M! + M \ln 2 + \frac{1}{2}\sum_{i=1}^{k} \ln\left(\frac{2}{\gamma_i}\right) + \frac{1}{2}\ln\left(\frac{2}{N-\gamma}\right) \end{aligned} \tag{11}$$

Where M: number of hidden neurons, W; number of weights in each group, N: number of input vectors, γ: number of 'effective' weights. For a given sample, the candidate models (defined by their number of hidden neurons) can then be compared in terms of their log-evidences; the ones with larger log-evidences are considered to be the most probable ones for modelling that sample.

2.5. The application of the Bayesian tools

It can be seen that the Bayesian approach features in different parts of the model specification process, since it offers: (a) a theoretical justification for the use of weight decay regularisation, and a method for estimating the regularisation hyperparameters (*Bayesian regularisation*), (b) a method for selecting the NN inputs (*ARD*), and (c) a method for selecting the optimum number of neurons (based on *Bayesian evidence*). The method in (a) is already widely accepted; methods (b) and (c) however are still open to discussion.

2.5.1. Bayesian regularisation

Several ways have been developed by researchers to apply these Bayesian methods, either in separate or in combination with other methods. Bayesian regularisation is easy to implement by itself, independently of the rest of the Bayesian tool kit (as for example, in the Matlab function *trainbr.m*). Chan et al (2006) and Saini (2008) compared it to other techniques to avoid overfitting in STLF. Bayesian regularisation was used in all the models we tried, but will not be discussed further in this paper.

2.5.2. Automatic relevance determination

ARD is somewhat more difficult to apply. In principle, it is a 'soft-pruning' technique, which means that the irrelevant inputs are not actually discarded. In practice, however, some authors have obtained better results by doing hard-pruning, i.e., by actually removing some of the less relevant inputs, guided by the relevance measures provided by ARD. The difficulty is, ARD can list the inputs in order of their relevance to the output, but cannot offer a cut-off point to separate the inputs that should be kept in the model from those that should be discarded. Penny & Roberts (1999) and Lauret *et al* (2008) selected arbitrary cut-off points. Silva & Ferreira (2007) used a 'probe' to set the cut-off point: they introduced a normal random variable among the inputs, and removed those inputs that were considered less relevant than the random variable.

2.5.3. Evidence-based model selection

In theory, it should not be necessary to limit the complexity of a NN (as indicated by the number of neurons) if the appropriate priors are used, since Bayesian regularisation should be enough to ensure that there would be no overfitting. (Bishop, 1995). The values of the weights that

connect the superfluous neurons would be driven down to zero, which would be equivalent to disconnecting those neurons. Very large models could therefore be used with good results. In practice, however, many researchers prefer to use as few neurons as possible, if only to relieve the computational burden. Bayesian evidence can be useful as a criterion for choosing the best NN (in terms of the number of neurons) to model a given sample. Alternatively, if several NNs are used in parallel (a committee of NNs), their evidences may be used as weights for the linear combination of their outputs. Thodberg (1996) first used the evidence to find the best 10 models, out of 20 candidates, then combined them into weighted committees. Penny and Roberts (1999) chose to use un-weighted committees. Since their datasets were small, they considered that the Gaussian approximation would not hold, and therefore the estimates of evidence would not be reliable enough to be used as weights. In applications to STLF, both Lauret *et al* (2008) and Silva and Ferreira (2007) used single NNs, instead of committees. Lauret *et al.* (2008) started with all 14 available inputs and a NN with 32 neurons. Using ARD and an arbitrary cut-off point, they reduced the number of inputs to 10. Then, using this subset of inputs, they run all the NNs and chose the one that had the largest evidence. Silva and Ferreira (2007) compared NNs with a number of neurons ranging from one to 10. For each number of neurons, they first run the NN with all the inputs, estimated their relevance by using ARD and a probe, and removed the less relevant ones. Then, they compared the models, and chose the number of neurons which led to the largest evidence.

3. Selection methods and implementation

3.1. Description of the methods

In our empirical study, we evaluated the use of Bayesian tools for input selection (ARD) and for model selection, separately and in combination; the benchmarks were provided by a CV-based method and by a naïve forecaster. In this section, we provide a short description of the methods we used; the algorithms for them are given in the next section.

All NN models we used were multilayer perceptrons, having one hidden layer with sigmoid activation functions, and one linear output node. All the analysis was run in Matlab; the codes for computing the evidences and for ARD were based on the routines written by Nabney (2004). Bayesian regularisation with ARD priors was used in all models. Since the estimation of the weights is sensitive to the choice of initial values, each model (i.e., each NN, as defined by its number of hidden neurons and its subset of inputs) was estimated 10 times, from random initial values. The metrics we used for the CV error was the *Mean Absolute Percent Error* (MAPE).

Methods 1 and 2 use CV for selecting the number of hidden neurons. Method 1 uses ARD as a soft-pruning technique. All the competing models (each defined by a different number of hidden neurons) are trained using all the available inputs, and then tested on the same CV sample; the one with the smallest error is the winner. Method 2 uses ARD as a guide for hard-pruning. Instead of using an arbitrary cut-off point (as in Penny and Roberts, 1999; Lauret *et al*, 2008), we use CV to determine when to stop pruning. All NNs (each with a different number of hidden neurons) are first run using all the inputs, which are ranked according to their relevance. The inputs are then deleted one by one, the least relevant ones first, and the CV error is computed after each deletion. When the error starts to increase, pruning is stopped. Then, the different NNs, each with its optimal set of inputs, are compared in terms of CV error. We experimented using a probe (as in Silva & Ferreira, 2007) but it did not work with our data. Perhaps because the initial input variables we considered were those that were already expected to be somewhat correlated to the load (as they were the variables considered by the electricity transmission company), the probes were always the variables considered to be the least relevant, and therefore the first ones to be discarded.

In Method 3, a benchmark, the choice of both the inputs and the number of hidden neurons is done by CV, using a method that is similar to the 'sequential zeroing of weights' in Papadokonstantakis *et al* (2006). For a given number m of hidden neurons, the NN is first trained with all the available k input variables and tested on the CV sample. Then, each of the k inputs is in turn removed, and the NN is trained and tested with the remaining k-1 inputs; the input whose removal leads to the least reduction in the CV error is considered to be the least relevant, and

discarded. The procedure is repeated, and at each step the least relevant variable is discarded, until the NN is left with only one variable. The subset of inputs that results in the smallest CV error is chosen, for a NN with *m* neurons. The whole procedure is repeated on NNs with different number of hidden neurons, until all the possible values of the number of neurons (previously defined) are tried. In the end, the CV errors obtained by each model (each number of hidden neurons and subset of inputs) are compared, and the model with smallest error is chosen. This method, totally empirical, is used as a benchmark, against which the methods that use Bayesian tools are compared.

Methods 4 and 5 use the Bayesian evidence for selecting the number of hidden neurons. Method 4 uses ARD as a soft-pruning technique. All the competing models (each defined by a different number of hidden neurons) are trained using all the available inputs, and have their evidences computed; the model with the largest evidence is the winner. Method 5 uses a combination of ARD and CV for hard-pruning, as in Method 2, but in a different way. In Method 2 the best subset of inputs is found for each model (defined by a number of neurons), and the models are then compared by means of CV. In Method 5, all models share the same set of inputs at each step. In the first step, all models are run using all inputs, and the evidences and input relevance measures are estimated for each of them. The winner is the model with the largest evidence; the variable considered by it to be the least relevant is removed from the set of inputs, and the CV error is computed. All NNs are then run again on the same reduced input set; at each step, the winner is the NN with the largest evidence, and the variable it indicates as the least relevant is removed from the common input set. The procedure is repeated until only one input is left. The CV errors produced by the winners of each step are compared; the overall winner is the model with the smallest CV error.

3.2. Algorithms for the methods

In this section, we provide detailed descriptions of each of the methods implemented in our empirical study.

Method 1 : selection of number of hidden neurons by CV, no input selection (soft pruning)

The inputs are not ‘selected’; all the available input variables are included in each model, and ARD is expected to be able to phase out the least relevant ones, by setting them to large values of α. The best number of hidden neurons is then selected by CV.

Method 2: selection of number of hidden neurons by CV, selection of inputs by ARD + CV

ARD was used to indicate the relevance of each input. The inputs were discarded one by one, according to their relevance; CV was used to find the optimum point where to stop pruning, and also to find the best number of hidden neurons. The algorithm was:

1) Set the vector of possible values of the number of hidden neurons: for example, v = [1 2 3 4 5 6 8 10 12].
2) Set i = 1;
3) While i < length(v)
    a. Set the number of hidden neurons as $m = v(i)$
    b. Run 10 times a NN with all the available inputs and m hidden neurons; compute the average CV error (MAPE) and the average α's for each input over the runs.
    c. Remove the least relevant input (the one with largest α)
    d. Compute the average CV error and the average α's for each input over 10 runs
    e. Repeat steps (c) and (d) until the NN is reduced to just one input.
    f. Find the number of inputs that resulted in the best CV performance for the NN with m hidden neurons
    g. Set $i \leftarrow i$+1
4) Compare the performances of the NNs with different number of hidden neurons in terms of the CV error.

Method 3: selection of both number of hidden neurons and inputs by CV

The measure used to determine the relevance of an input was the *mean absolute percentage error* (MAPE) over the CV sample. Since the NNs are sensitive to initialisation values, each model was run 10 times, and the MAPEs obtained across the 10 runs were averaged.
The algorithm was:

1) Set the vector of possible values of the number of hidden neurons; for example, v = [1 2 3 4 5 6 8 10 12].
2) Set i = 1;
3) While i < length(v)
    a. Set the number of hidden neurons as $m = v(i)$
    b. Run 10 times a NN with all the available inputs and m hidden neurons, and compute the average CV error (MAPE)
    c. Compute the relevance of each input at a time, by evaluating the change in the output caused by the omission of that input:

i. Remove the first input, and compute the average CV (MAPE) over 10 runs;
ii. Return the first input to the NN, and remove the second; compute again the average CV error over 10 runs;
iii. Continue in the same way, removing one input at a time and computing the average CV error over 10 runs, until the last input is reached;

d. Find the most irrelevant input, i.e., the one whose omission had the least effect on the NN performance on the CV sample. Remove that input from the NN.
e. Repeat steps (c) and (d) until the NN is reduced to just one input.
f. Find the number of inputs that resulted in the best CV performance for the NN with *m* hidden neurons.
g. Set $i \leftarrow i+1$

4) Compare the performances of the NN with different numbers of hidden neurons in terms of CV error. This reveals the best model (in terms of inputs and number of hidden neurons).

This method is similar to what Papadokonstantakis *et al* (2006) call 'sequential zeroing of weights'.

Method 4: selection of number of hidden neurons by evidence, no input selection (soft-pruning)

This method is an application of the *evidence framework*, as proposed by MacKay (1992b) for model selection. All the inputs are used, and the vector of possible values of the number of hidden neurons is the same as in the other methods, v = [1 2 3 4 5 6 7 10 12]. The best model among the ones compared is the one that has the largest Bayesian evidence.

Method 5: selection of number of hidden neurons by evidence, selection of inputs by ARD + CV

This method combines evidence-based model selection and ARD-based input selection. The input to be deleted is the one indicated by ARD+CV on the model that has the largest evidence at each iteration. The algorithm was:

1) Set the vector of possible values of the number of hidden neurons: v = [1 2 3 4 5 6 7 10 12].
2) Across 10 runs, compute the average evidences and the input α's for each model (i.e., for different numbers of neurons), using all the available inputs.
3) Find the winner, that is, the NN with the largest evidence. Compute the CV error of the winner.
4) Remove from the dataset the variable that the winner indicates as the least relevant.

5) Repeat steps 2 to 4 until all the inputs have been removed.
6) Find the model with smallest CV error overall.

Method 6: naïve forecaster

As a benchmark, we used the simple naïve forecaster defined by:

$$\hat{L}_t = L_{t-7}$$

where L_t is the load at midday on a day t. That is, the forecast for the load at midday on day t is given by the load observed at midday on the corresponding day of the previous week.

4. Empirical study

4.1. The forecasting problem, and the data

In this paper, we study the application of NNs to forecasting electricity demand (load), using six samples. Samples I and II come from England and Wales; Samples III and IV, from a utility in Brazil. To confirm the findings from those data, we repeated the study on two more samples, available online, which have already been used by several other researchers: Sample V, from a utility in the USA, and Sample VI, from a utility in Slovakia.

4.1.1. Samples I and II : data from England and Wales

These data were supplied by the National Grid (NG), the company responsible for the transmission of electricity in England and Wales. The two samples we used come from the same dataset, which we split in two different ways so that we could investigate the stability and consistency of the results. A previous study of NN load forecasting using these series was carried out by Taylor & Buizza (2002). This dataset is available online at *http://users.ox.ac.uk/~mast0315/*.

We were interested in forecasting the load at midday, three days ahead. Midday is a particularly important period in the summer months because it is often when peak demand occurs. As we discussed in the *Introduction*, load forecasts ranging from a few hours to a week ahead are needed by the industry. The most common applications of NN to load forecasting in the literature seem to be those that have a one-day ahead horizon. We chose to forecast three days ahead, instead, because we considered that at the longer range the impact of the weather variation on the load forecasts would be more easily observable. For NN training, we used the observed past values of

the weather variables; for the out-of-sample tests, however, we used forecasted values of these variables, in order to reproduce more accurately the kind of results that a forecasting system would obtain in a real application.

Figure 1a shows a plot of electricity demand in England and Wales at midday for each day, from 01/Jan/1997 to 31/Dec/1998. One clear feature of demand is the strong seasonality throughout the year, which results in a difference of about 5000 MW between typical winter and typical summer demand.

***** Figure 1a *****

Another noticeable seasonal feature typically found in load series is the strong seasonality within each week: there is a consistent difference of about 7 GW between weekday and weekend demand. Figure 1b shows the superimposed profiles (midday loads) of eight consecutives weeks (13/Jan/97 to 09/Mar/97).

***** Figure 1b *****

It should be noted that the weekly profiles may change considerably from dataset to dataset. In some series, the loads on Mondays or on Fridays may be significantly lower than the loads on the middle days of the week (Tuesday to Thursday); in other series (as the NG ones), all weekdays may have similar loads. Forecasters may have to develop heuristics to model the individual characteristics of a particular load profile. In this study, since we are interested only in automatic methods for determining variable relevance (by ARD or by CV), we simply modelled all the days of the week by binary dummies, without trying to distinguish among them.

There is unusual demand on a number of 'special days', including public holidays, such as New Year's Day. In practice, NG forecasts demand on these days using judgemental methods. In this study, we elected to smooth out these special days, as their inclusion is likely to be unhelpful in our analysis of the relationship between demand and weather. Alternatives to this would be treating the special days as missing observations, or modelling them explicitly, as in Cancelo *et al* (2007). In this study, the load values on the special days are not used as targets for forecasting, but they are included in the training samples for the NN; i.e., we do not evaluate forecast accuracy for these periods, but their smoothed observed values may be used as input for the forecasting of later values.

As explanatory variables, we use lagged demand, weather variables, and dummies. Short to medium-term forecasting models must accommodate the variation in demand due to the seasonal patterns shown in Figure 1a, and due to weather. At NG, demand is modelled using three weather variables: effective temperature, cooling power of the wind and effective illumination. These

variables are constructed by transforming standard weather variables in such a way as to enable efficient modelling of weather-induced demand variation (Baker, 1985). *Effective temperature* (TE_t) for day t is an exponentially smoothed form of TO_t, which is the mean of the spot temperature recorded for each of the four hours prior to midday:

$$TE_t = \tfrac{1}{2} TO_t + \tfrac{1}{2} TE_{t-1} \qquad (1)$$

The influence of lagged temperature aims to reflect the delay in response of heating appliances within buildings to changes in external temperature. *Cooling power of the wind* (CP_t) is a non-linear function of wind speed W_t, and of average temperature TO_t. It aims to describe the draught-induced load variation.

$$CP_t = \begin{cases} W_t^{\frac{1}{2}} (18.3 - TO_t) & if \quad TO_t < 18.3\ ^oC \\ 0 & if \quad TO_t \geq 18.3\ ^oC \end{cases} \qquad (2)$$

Effective illumination is a complex function of visibility, number and type of cloud and amount and type of precipitation. Since we had no data for the effective illumination we replaced this variable by cloud cover; also, we used spot temperature, instead of average temperature TO_t, to construct effective temperature and cooling power of the wind from NG's formulae in expressions (1) and (2).
The loads and all weather data were normalised before being input into the models.

There were thus five weather variables (*temperature*, *wind speed*, *cloud cover*, *effective temperature*, and *cooling power*). In addition, the models used five lagged load values (*load_lag_3* to *load_lag_7*); six dummy variables to represent the days of the week (*Mon, Tue, Thu, Fri, Sat, Sun*; we dropped one dummy, *Wed*, to avoid colinearity), three dummies to represent the summer weeks when a large amount of industry closes (*week1, week2, week3*), and another dummy to represent the days included in the British Summer Time (*BST*).

The load data set consisted of midday load observations for 1022 days, from 01/Jan/1997 to 30/Apr/2000, inclusive. These data were split according to two plans, generating two samples. In Sample I, the first 658 days (94 weeks) of the series were used for training, the next 182 days (26 weeks) for cross-validation, and again 182 days (26 weeks) for testing. In Sample II, the CV and test sample were moved forward 182 days; the training sample was expanded to include 840 days (120 weeks), while the CV and training sample had the same size as before. For those models that did not use CV, the CV samples were added to the training samples.

4.1.2. Samples III and IV: data from Rio de Janeiro, Brazil

These data were supplied by a utility that serves the city of Rio de Janeiro, Brazil. We extracted two samples from the dataset, by splitting the series (load and temperature) in two

different ways, so that we could investigate the stability and consistency of the results. A previous study of NNs forecasting using this dataset is in Hippert *et al* (2005). This dataset is available online at *http://users.ox.ac.uk/~mast0315/*.

We were interested in forecasting the load at midday, one day ahead. Figure 2 shows a plot of electricity demand in Rio at midday for each day of 1996 and 1997. This load series also exhibits marked seasonal patterns within the year and within the week. The 'special days' were smoothed out from this series, as had also been done in the Samples I and II. As explanatory variables, we use seven lagged load values (*load_lag_1* to *load_lag_7*), the observed hourly temperature, and six dummy variables to represent the days of the week (*Mon, Tue, Thu, Fri, Sat, Sun*). Since the observed hourly temperature was the only weather variable available to us, we chose to forecast the load one day ahead (instead of three days ahead, as in Samples I and II), as the difference between observed and forecast values for weather variables would be much smaller, at this closer range.

The dataset consisted of midday load and temperature observations for 637 days, from 01/Apr/1996 to 31/Dec/1997, inclusive. These data were split according to two plans, generating Samples III and IV. In Sample III, the first 301 days (43 weeks) of the available series were used for training, the next 112 days (16 weeks) for cross-validation, and again 112 days (16 weeks) for testing. In Sample IV, the CV and test sample were moved forward 112 days; the training sample was expanded to include 413 days (59 weeks), while the CV and training sample had the same size as before. For those models that did not use CV, the CV samples were added to the training samples.

**** Figure 2 ****

4.1.3. Samples V and VI: data from an American and a Slovakian utilities

To confirm the findings from Samples I to IV, we repeated the study on two more samples, extracted from datasets which have been used previously for load forecasting competitions, and which are available online: Sample V, from a utility in the USA, and Sample VI, from a utility in Slovakia. We used both for testing models that forecast the midday load, one day ahead.

The two series in Sample V (hourly load and temperature) come from a North American utility, the Puget Sound Power and Light Company. The data has been the subject of a load profile forecasting competition (Ramanathan *et al*, 1997). The series are 2,842 days long; we used one year (52 weeks = 364 days) for cross-validation, another year for testing, and the remaining data for network training (2,114 days).

The two series in Sample VI (half-hourly loads, and daily temperatures) come from the Eastern Slovakian Electricity Corporation, and were used for a daily peak load forecasting

competition (Chen *et al*, 2004). This dataset includes a number of "special days", the loads of which we smoothed out, as described in Section 4.1.1. The data comprises 761 days; we used 112 days (16 weeks) for cross-validation, another 112 for testing, and the remaining data (537 days) for training.

4.2. Results and discussion

The final NN model (as defined by the number of hidden neurons and subset of inputs) selected by each of the six methods described in Section 3 was re-estimated and tested 20 times over each of the six samples. The metrics for the out-of-sample error were the *mean absolute error* (MAE), the *mean absolute percent error* (MAPE), and the *root mean square error* (RMSE). Table 1 shows the number of hidden neurons and the number of inputs of each model; Table 2 shows the average of the 20 MAPEs; Table 3 shows the average of the 20 MAEs for each model; Table 4 shows the average of the 20 RMSEs. Figure 3 displays the distribution of the MAPEs. We deemed it uninteresting to include in the figure the MAPEs produced by the naïve method (Method 6), because these errors were much larger than those of the other methods, and would therefore require stretching out the error axis, making the graph too sparse.

**** Table 1 ****

**** Table 2 ****

**** Table 3 ****

**** Table 4 ****

**** Figure 3 ****

In Method 1, there is no input selection, but only 'soft-pruning'. Interestingly, its results were comparable to the ones obtained by the model with a reduced set of inputs (Method 2), over all the six samples, proving that ARD did a good job in controlling the complexity of the NN. This is somewhat inconsistent, however, with what was found by previous researchers, who deduced that soft-pruning was not enough, and used ARD only as a guide for pruning (Penny and Roberts, 1999; Silva and Ferreira, 2007; Lauret *et al*, 2008).

The results produced by Method 2 suggest that both the Bayesian regularisation and the ARD techniques were quite effective in controlling model complexity. As an example of this, Figure 4 shows the MAPEs produced by models ranging from 1 to 20 neurons, with all inputs and ARD priors, over Sample I. It can be seen that regularisation ensured that, although the best NN was the one with 4 neurons, NNs with more neurons still produced rather similar results. This does not mean, however, that the choice of the number of neurons is inconsequential; finding the right number is still necessary - first, because overfitting still may occur, for larger NNs; second, because it has been shown that in load forecasting even relatively small improvements in accuracy have noticeable economic significance (Ranaweera *et al*, 1997; Hobbs *et al*, 1998).

**** Figure 4 ****

The ranking of input relevance provided by ARD in Method 2 was found to be quite comparable across models with different numbers of neurons. Although this ranking was made automatically, according to statistical criteria, it is instructive to see which variables were chosen by the best models. Some of the choices were the ones a practitioner would expect: all models we tested, with number of hidden neurons ranging from 4 to 12, pointed to the *effective temperature* as the most relevant weather variable; also, to the dummies that marked the weekend (*Sat* and *Sun*) as the most relevant calendar dummies. However, the explanation for other choices is not so clear: almost all models included either *load_lag_3* or *load_lag_5* (frequently both) among the three most relevant variables; however, *load_lag_4* was always considered irrelevant.

We found that the out-of-sample error changed in a consistent fashion as the inputs were discarded according to ARD indication. Figure 5 shows the evolution of CV error in the NN with four neurons (Method 2), over Sample I. The smallest error was reached with 17 inputs, but the error actually remained more or less stable until the number of inputs was dropped to less than eight, when the NN became too simple for the problem, and the error started to climb. It can be seen in Tables 1 and 2 that the NN with four neurons in Sample I had practically the same out-of-sample mean MAPE before the pruning (with 20 inputs) and after it (17 inputs). This suggests that the measure of relevance given by ARD is indeed consistent with the model CV error, and that the inputs dropped were indeed irrelevant.

**** Figure 5 ****

The performance of Method 3, based on CV both for model and input selection, was reasonably similar to that of Methods 1 and 2, based on CV and ARD, over all six samples. However, Method 3 was clearly inferior to the others in terms of the computational effort involved; for Sample I, for example, it required the training of no less than 16,720 NNs, against 1,600 for Method 2, and only 80 for Method 1.

Methods 4 and 5 used the Bayesian evidence for selecting models; Method 4 used soft-pruning, Method 5 used hard-pruning. Both methods had very similar results, except on Sample V, where Method 4 resulted in a one-neuron NN, while Method 5 resulted in a three-neuron NN. As one can see by inspecting Tables 2 to 4, Methods 4 and 5 were out-performed by CV-based Methods 1 and 2 in five out of the six test samples. It is well known that the evidence favours the simpler models, since it penalises model complexity; however, in this study, the result was that the evidence almost always pointed to the simplest model under test, the one-neuron NN. This result

does not suggest that a NN with only one hidden neuron can solve the problem adequately; it just shows that the evidence-based procedure for selecting the number of neurons is biased towards smaller nets. Also, it suggests that Bayesian evidence was not useful as a tool for model selection for these data. The fact that these one-neuron NN achieved results rather close the best ones (particularly on Sample VI) may suggest that the problem is not strongly nonlinear, and that perhaps reasonable results could also be obtained by linear methods on some of the samples.

Most of the studies that reported good results with the evidence approximation framework were based on applications where the number of inputs was rather small (with the exception of Thodberg, 1996, who used up to 18 inputs). In order to check the effect that this number could have on the performance of the framework, we run a few experiments on Sample 1, with NNs that had just a small subset of inputs selected among the ones that are usually considered to be the most relevant for STLF. We ran NNs ranging from one to 10 neurons, using only four inputs: the most recent observed value of load, the temperature, and the two dummies for the weekend. The model chosen by evidence (4 neurons) had an out-of-sample MAPE of 1.59 %; the model selected by CV (6 neurons) had a practically identical MAPE of 1.58 %. These results suggest that evidence-based methods may be more likely to achieve good results (i.e., results close to those of CV-based methods) in problems with low dimensionality.

An increase in the number of inputs leads not only to an increase in the complexity of the surface of the NN response, but also to an increase in the number of NN weights to be estimated. Since the evidence framework is based on Gaussian approximations, the larger the samples, with respect to the number of parameters, the better one would expect the results to be. (Mackay, 1992b, suggested that the number of data points should be at least three times as large as the number of NN weights, as a rule of thumb). In our experiments with models on reduced input sets, the sample sizes were larger (relative to the number of weights), and the problem was less complex (reduced dimensionality); those are perhaps the reasons why the evidence framework achieved a better performance. From a practical point of view, however, these results are not very encouraging, since STLF problems frequently involve large dimension input vectors, and relatively short data series; exactly the situations where the evidence framework, in our study, resulted in a disappointing performance.

5. Conclusions

The Bayesian approach has been proposed as a way to control the NN complexity, without resorting to CV or to pruning/growing techniques – it offers ways to avoid overfitting (by regularisation), to decide on the number of neurons (by comparing the model evidences), and to deal with the inputs (by soft-pruning). In this study, we found that the Bayesian framework was only partially useful for the modelling of the available datasets; regularisation and ARD proved to be very useful; evidence-based selection proved to be ineffective.

Bayesian regularisation proved to be very efficient in controlling NN complexity and avoiding overfitting, and the use of ARD priors for soft-pruning led to good results. Also, the estimates of input relevance given by ARD proved to be very reliable in guiding hard-pruning procedures. The difficulty is that there is no obvious way to decide when to stop pruning, and CV had to be used for that. Penny & Roberts (1999) also found that ARD was effective in ranking input relevance, though this was only useful in large NNs with many irrelevant inputs.

Model selection by evidence, however, proved to be ineffective in this study. Penny & Roberts (1999) found that the issue of whether or not the evidence is useful as a measure to guide model selection depends on the problem under study and on the amount of available data; they found correlation between the evidence and the test error in only two out of eight datasets they considered. Thodberg (1996) discusses the reasons why the evidence may not necessarily be correlated to the out-of-sample error, especially if the samples have relatively small sizes. In our study, the sizes of the samples were adequate, but the evidence tended always to point to models that were too simple and that produced unsatisfactory results out-of-sample. Bishop (1995) suggests that the evidence might be a less robust technique than the ARD for guiding modelling, considering that the estimation of the ARD relevance measures depends on the sum of the eigenvalues of the Hessian, while the estimation of the evidence depends on the product of them; therefore, the evidence tends to be much more sensitive to the uncertainty of the eigenvalues.

Overall, although they achieved good results with Bayesian modelling in their study, Lampinen & Vehtari (2001) concluded that the Bayesian framework does not automatically guarantee better results than the traditional (CV-based) NN fitting procedures, and that it requires more expertise during the modelling. MacKay (1992b), after using the Bayesian framework on a small problem with synthetic data, remarked that it "would be interesting to see the results of evaluating the evidence for networks applied to larger real-world problems". This paper provides some such results. It is clear, however, that an empirical study cannot produce a definitive conclusion; the best it can do is to add to the body of evidence concerning the usefulness of a proposed hypothesis or method. We would certainly support more research on the application to

load forecasting in order to identify the situations where Bayesian modelling could replace CV techniques with reliable performance.

Acknowledgements

This study was supported by a grant from the Brazilian agency CAPES (no. 2797-07-0), which allowed H. S. Hippert to work as a visitor at the Saïd Business School – Oxford, UK. We thank the European Centre for Medium-range Weather Forecasts and the National Grid for making available the data relating to the modelling of load in England and Wales.

6. References

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Bunn, D. W. (2000). Forecasting loads and prices in competitive power markets. *Proc IEEE*, *88* (2), 163-169.

Buntine, W. L., & Weigend, A. S. (1991). Bayesian back-propagation. *Complex systems*, *05* (6), 603.

Cancelo, J. R., Espasa, A., & Grafe, R. (2007). Forecasting from one day to one week ahead for the Spanish system operator. *International Journal of Forecasting*, 24: 588-602.

Cataltepe, Z., Abu-Mostafa, Y. S., & Magdon-Ismail, M. (1999). No free lunch for early stopping. *Neural computation*, *11*, 995–1009**.**

Chan, Z. S. H., Ngan, H. W., Rad, A. B, David, A. K., & Kasabov, N. (2006). Short-term artificial neural network load forecasting from limited data using generalization learning strategies. *Neurocomputing*, *70*, 409-419.

Chen, B.-J., Chang, M.-W., & Lin, C.-J. (2004). Load forecasting using support vector machines: A study on EUNITE competition 2001. *IEEE transactions on power systems*, *19* (4), 1821–1830.

Chua, C. G., & Go, A. T. C. (2003). A hybrid Bayesian back-propagation neural network approach to multivariate modelling. *International journal for numerical and analytical methods in geomechanics, 27*, 651–667.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. London: Chapman & Hall.

Haykin, S. (1999). *Neural networks – a comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Hippert, H. S., Bunn, D. W., & Souza, R. C. (2005). Large neural networks for electricity load forecasting: are they overfitted? *International journal of forecasting*, *21* (3), 425-434.

Hobbs, B. F., Helman, U., Jitprapaikulsarn, S., Konda, S., & Maratukulam, D. (1998). Artificial neural networks for short-term energy forecasting: accuracy and economic value. *Neurocomputing*. 23: 71-84.

Lampinen, J., & Vehtari, A. (2001). Bayesian approach for neural networks – review and case studies. *Neural networks*, *14*, 257-274.

Lauret, P., Fock, E., Randrianarivony, R. N., & Manicom-Ramsamy, J.-F. (2008). Bayesian neural network approach to short term load forecasting. *Energy conversion and management*, *49*, 1156-1166.

Lendasse, A., Wertz, V., & Verleysen, M. (2003). Model selection with cross-validations and bootstraps - application to time series prediction with RBFN models. ICANN/ICONIP 2003, Istanbul (Turkey). *Lecture Notes in Computer Science*, vol. 2714, pp. 573-580. Springer-Verlag.

Liang, F. (2005). Bayesian neural networks for nonlinear time series forecasting. *Statistics and computing*, *15*, 13–29.

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.

Nabney, I. T. (2004). Netlab : Neural network software [WWW page]. URL http://www.ncrg.aston.ac.uk/netlab/

Papadokonstantakis, S., Lygeros, A., & Jacobsson, S. P. (2006). Comparison of recent methods for inference of variable influence in neural networks. *Neural networks, 19*, 500-513.

Penny, W. D., & Roberts, S. J. (1999). Bayesian neural networks for classification: how useful is the evidence framework? *Neural networks*, *12*, 877-892.

Ramanathan, R., Engle, R., Granger, C. W. J., Vahid-Araghi, F., & Brace, C. (1997). Short-run forecasts of electricity loads and peaks. *International journal of forecasting*, *13* (2), 161–174.

Ranaweera, D. K., Karady, G. G., & Farmer, R. G. (1997). Economic impact analysis of load forecasting. *IEEE Transactions on power systems*, vol. 12, no. 3, pp. 1388-1392.

Rossi, V., & Vila, J-P. (2005). Bayesian multioutput feedforward neural networks comparison: a conjugate prior approach. *IEEE transactions on neural networks*, *17* (1), 35-47.

Saini, L. M. (2008). Peak load forecasting using Bayesian regularization, resilient and adaptive backpropagation learning based artificial neural networks. *Electric power systems research*, *78*, 1302-1310.

Silva, A. P. A., & Ferreira, V. H. (2007). Towards estimating autonomous neural network-based electric load forecasters. *IEEE transactions on power systems, 22* (4), 1554-1562.
Taylor, J. W., & Buizza, R. (2002). Neural network load forecasting with weather ensemble predictions. *IEEE transactions on power systems*, *17*, 626-632.
Thodberg, H. H. (1996). A review of Bayesian neural networks with an application to near infrared spectroscopy. *IEEE transactions on neural networks*, *7* (1), 56-72.
Titterington, D. M. (2004). Bayesian methods for neural networks and related models. *Statistical science*, *19* (1), 128-139.
Vila, J-P., Wagner, V., & Neveu, P. (2000). Bayesian nonlinear model selection and neural networks: a conjugate prior approach. *IEEE transactions on neural networks*, *11* (2), 265-278.

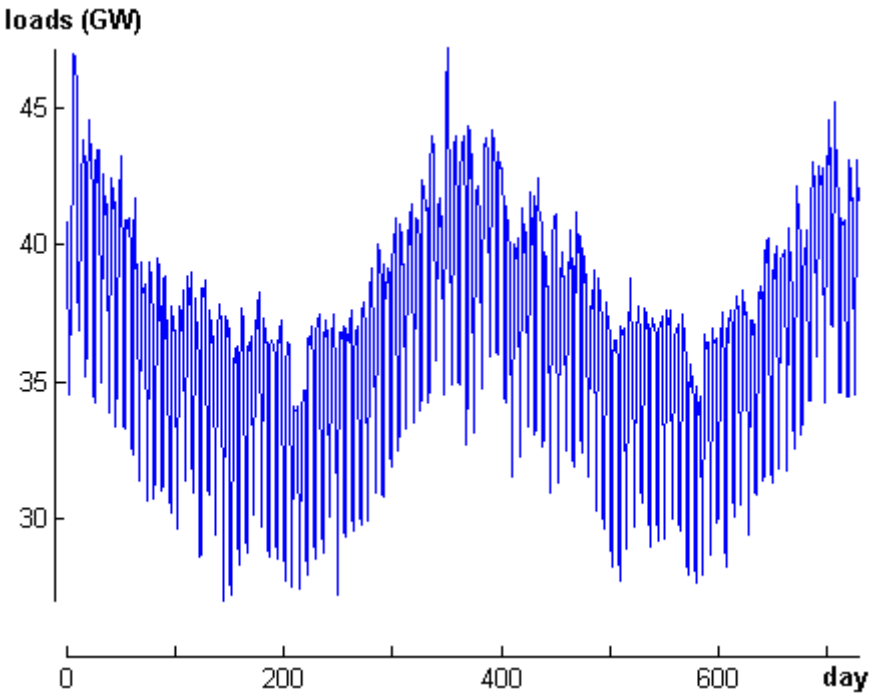

Figure 1a – Midday loads, 1997-1998, England and Wales

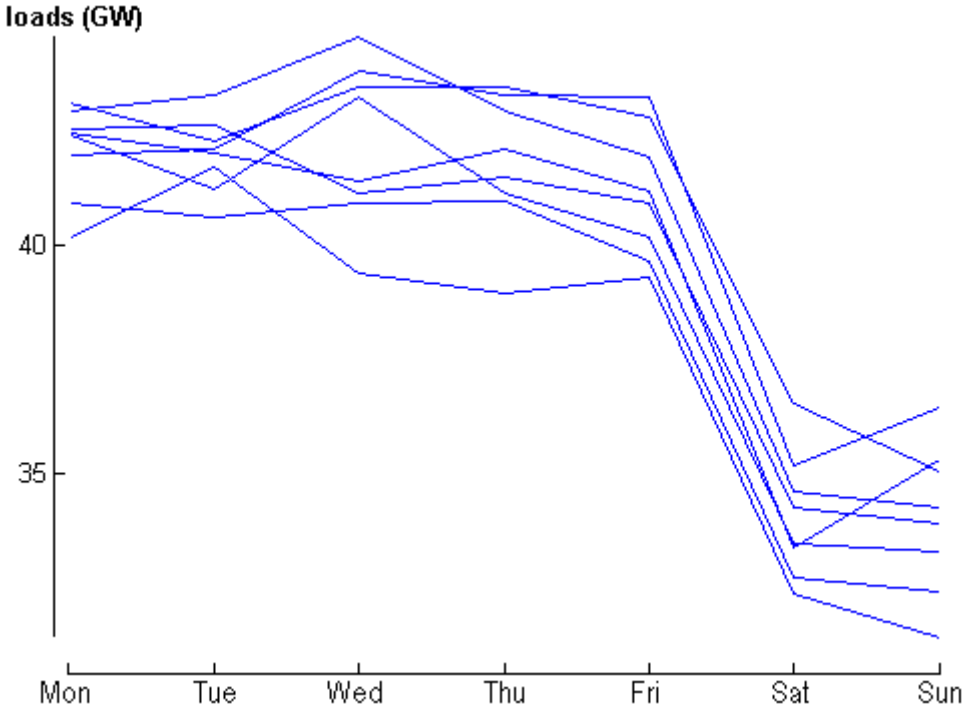

Figure 1b – Midday loads for eight weeks (13/Jan/97 to 09/Mar/97), showing weekly pattern

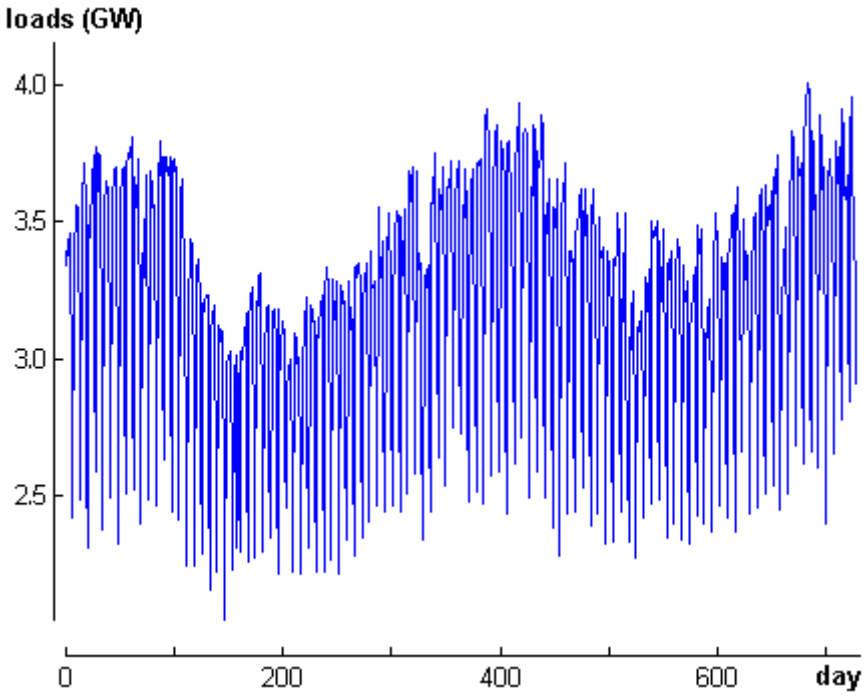

Figure 2 – Midday loads, 1996-1997, Rio de Janeiro

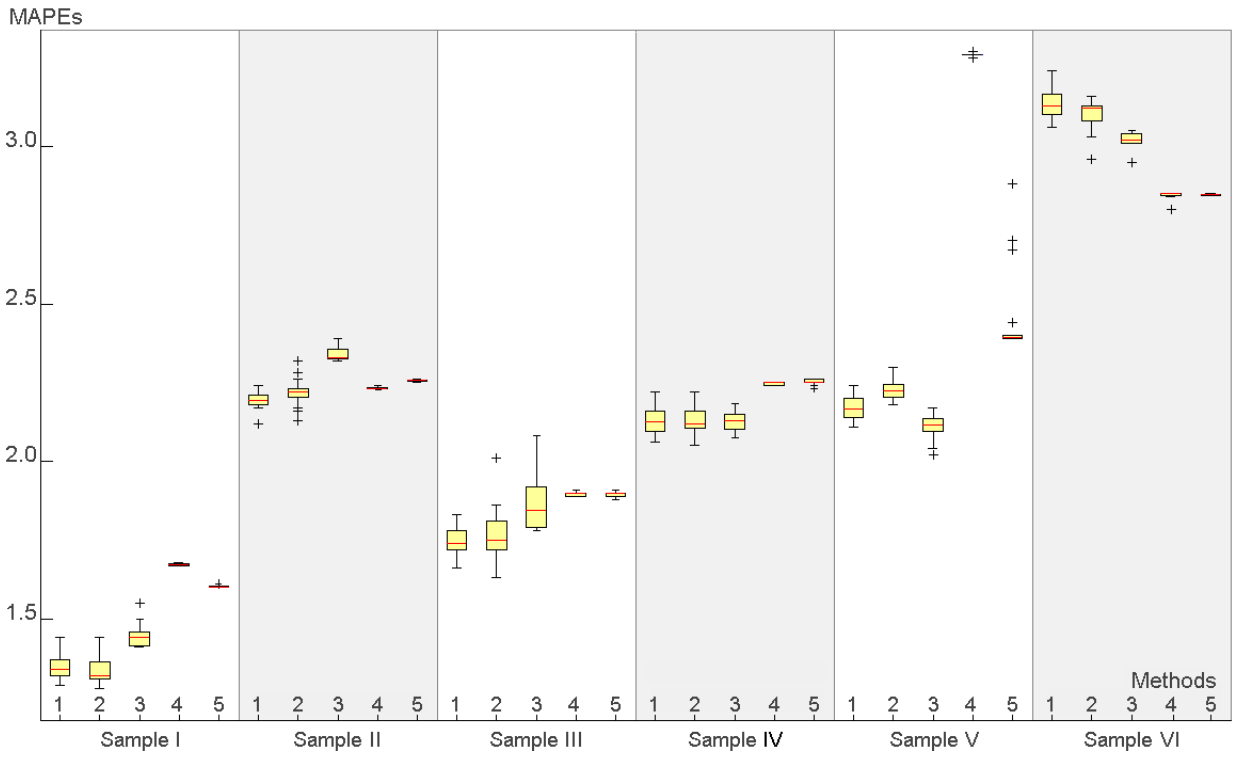

Figure 3. Out-of-sample MAPEs obtained by the models in 20 runs over each sample

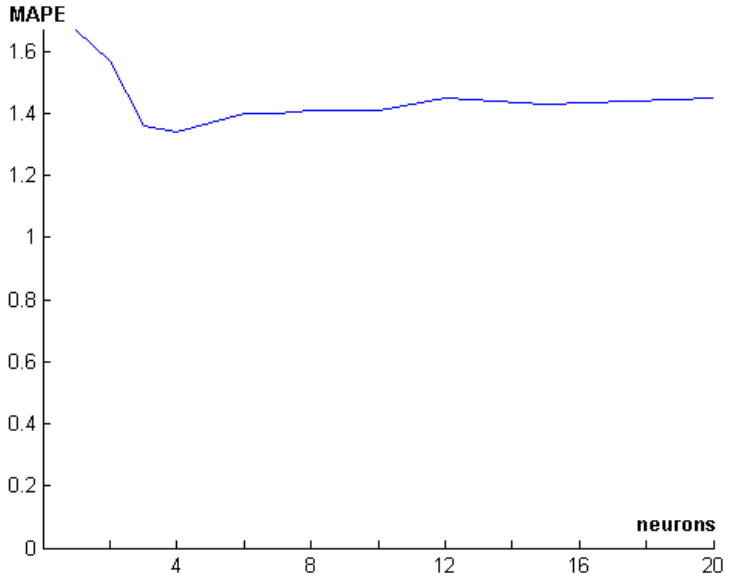

Figure 4. Out-of-sample MAPEs for models with 1 to 20 neurons and all inputs (Sample I)

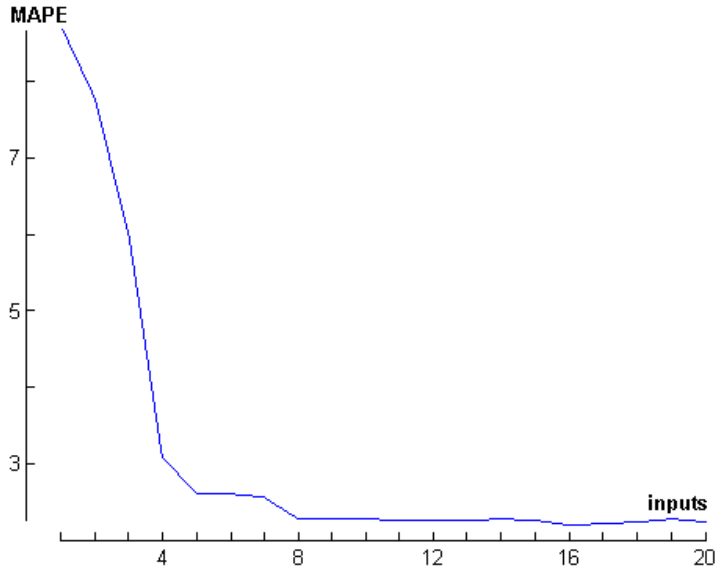

Figure 5 – Evolution of MAPE along pruning based on ARD (NN with 4 neurons; Sample I)

Table 1 – Number of hidden neurons / number of inputs of the tested methods

	Selection # neurons	Selection of # inputs	Samples I	II	III	IV	V	VI
1	CV	Soft-pruning	4 / 20	4 / 20	2 / 14	3 / 14	7 / 18	5 / 14
2		Rank inputs by ARD; stop pruning by CV.	4 / 17	7 / 13	2 / 10	5 / 9	5 / 14	3 / 14
3		CV	4 / 13	7 / 15	5 / 5	5 / 8	14 / 13	2 / 8
4	Evidence	Soft-pruning	1 / 20	1 / 20	1 / 14	1 / 14	1 / 14	1 / 14
5	Evid.+CV	ARD on model with largest evidence; CV stop	1 / 12	1 / 12	1 / 13	1 / 8	3 / 7	1 / 14
6	Naïve forecaster		---	---	---	---	---	---

Table 2 – Mean MAPEs

	Selection # neurons	Selection of # inputs	Samples I	II	III	IV	V	VI
1	CV	Soft-pruning	1.35	**2.19**	**1.75**	**2.13**	2.17	3.13
2		Rank inputs by ARD; stop pruning by CV.	**1.34**	2.22	1.77	**2.13**	2.23	3.10
3		CV	1.44	2.34	1.87	**2.13**	**2.11**	3.02
4	Evidence	Soft-pruning	1.67	2.23	1.90	2.25	3.29	**2.84**
5	Evid.+CV	ARD on model with largest evidence; CV stop	1.60	2.26	1.90	2.25	2.45	2.85
6	Naïve forecaster		1.80	3.12	4.07	4.41	5.10	4.11

Table 3 – Mean MAEs

	Selection # neurons	Selection of # inputs	Samples I	II	III	IV	V	VI
1	CV	Soft-pruning	473	**872**	**52.5**	71.3	54.6	22.0
2		Rank inputs by ARD; stop pruning by CV.	**462**	882	54.0	70.5	56.4	22.0
3		CV	507	919	57.7	**69.9**	**53.8**	21.4
4	Evidence	Soft-pruning	588	893	57.2	74.8	81.6	**20.1**
5	Evid.+CV	ARD on model with largest evidence; CV stop	563	905	56.7	75.1	60.6	**20.1**
6	Naïve forecaster		633	1235	128.7	149.1	131.4	29.0

Table 4 – Mean RMSEs

	Selection # neurons	Selection of # inputs	Samples I	II	III	IV	V	VI
1	CV	Soft-pruning	652	**1086**	**69.7**	93.5	**75.9**	27.5
2		Rank inputs by ARD; stop pruning by CV.	**638**	1113	71.5	**93.2**	77.2	27.3
3		CV	699	1124	78.3	94.0	76.4	26.3
4	Evidence	Soft-pruning	767	1119	74.9	97.6	106.8	**25.2**
5	Evid.+CV	ARD on model with largest evidence; CV stop	745	1137	75.0	96.9	83.4	**25.2**
6	Naïve forecaster		879	1515	159.0	188.8	180.0	35.4