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

Henrique S. Hippert (1)
James W. Taylor (2)

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. However, there are still no widely accepted strategies for designing the models and for implementing them, which makes the process of modelling by neural networks largely heuristic, dependent on the experience of the user. This paper evaluates techniques for automatic neural network modelling within a Bayesian framework, as applied to a dataset containing daily load and weather data for England and Wales. We analyse input selection as carried out by the Bayesian "automatic relevance determination", and also evaluate 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.

Keywords:
Bayesian neural networks; load forecasting; Bayesian methods

1) Introduction

Short-term load forecasting (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 minutes 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, purchasing of fuel, etc. 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 dedicated to this area. Among the techniques that have been tried for STLF, neural networks (NN) have been particularly conspicuous since the late 90’s. The reasons for this are not difficult to find – NNs are very flexible data-driven models, capable of approximating complex nonlinear functions to an arbitrary accuracy (Bishop, 1995), provided that the complexity of the model is consistent with the complexity of the problem at hand.
However, this same flexibility of NNs causes the task of finding the right model for a given problem to be a very difficult one. Though many advances have been made in the algorithms for the estimation of parameters, it is still far from straightforward to design a NN and to define the right degree of complexity needed in a model. If the model is too simple, it will perform poorly, both insample and out-of-sample; if it is too complex, it may overfit the data – that is, it may end by modelling not only the underlying function but also the noise in the data, and hence be unable to generalise (i.e., to produce adequate forecasts out-of-sample).

The complexity of a NN model is in the first instance dictated by its architecture. If one restricts oneself to NNs with one hidden layer with sigmoid neurons, and one output layer with linear neurons (the sort of NN more frequently used for STLF), the complexity would be indicated by the number of hidden neurons and inputs. 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 very small, the activation functions of the neurons will be operating in the central part of their ranges, which is practically linear. Thus, a large NN can in fact be no more complex than a linear regression model.

There are therefore two main ways of controlling the complexity of a NN: choosing the architecture (appropriate number of neurons), or using ‘regularisation’ techniques to ensure that the training algorithm leads, if appropriate, to NNs with small weights.

The architecture is usually chosen by means of procedures that change the number of neurons step by step, until an optimum value has been reached. 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). Both groups of techniques are based on the principle of parsimony (a model should be as complex as necessary for a given task, but not more), and both require some measure of NN complexity to be evaluated at each step.

For input selection, it is necessary to evaluate the influence of each input on the output. A comparison of methods for this is in Papadokonstantakis et al (2006), which considers 3 groups of methods: a) those that take only the data in 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, some measure of input relevance is needed – one that either measures the input’s ‘predictive importance’ (i.e., the increase in the generalisation error when that input is omitted) or the ‘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 my 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 other parameters, depending on the algorithm used, such as regularisation coefficients, momentum rate, learning rate, etc. A common way of setting these is by trial-and-error, on a ‘validation’ sample, distinct from the training and the testing ones; this procedure is called ‘cross-validation’.

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 has some disadvantages. 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, though, CV has other limitations. First, the estimations of generalisation ability obtained on a CV sample are noisy; a different CV sample might lead to very different conclusions. To overcome this, it is advisable to use several CV samples and average their results (e.g., k-fold cross-validation or leave-one-out method). For time series 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 several variables at the same time, the computational cost could easily become prohibitive.

Among the methods that have been proposed to deal with these difficulties in NN modelling, this paper intends to focus 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 methods 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).

The application of Bayesian techniques to NNs started with Buntine & Weigend (1991). Mackay (1992a, 1992b) introduced the ‘evidence approximation’ approach, which is based on a Gaussian approximations to the posterior distribution of the network weights. Reviews can be found in Thodberg (1996), Penny & Roberts (1999), Lampinen & Vehtari (2001), Titterington (2004).

In this paper we evaluate the application of Bayesian NNs to STLF, using the ‘evidence approximation’ approach. The dataset used contains a series of load and five series of weather variables for England and Wales. We compare the results to those obtained by CV methods, and to naïve models that serve as benchmarks.

The paper is organised as follows: Section 2, presents a short overview of the Bayesian approach to NN modelling; Section 3 describes the dataset and the routines used in the studies; Section 4 presents 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 $w$ of a NN are random variables, and assuming a prior distribution $p(w)$ for them. After the sample has been observed, a posterior distribution $p(w|D)$ is calculated by means of Bayes theorem,

$$ p(w | D) = \frac{p(D | w)p(w)}{p(D)} \tag{1} $$

where $w$ is the weight vector and $D=\{t\}^n$ is the set of target vectors. The probability $p(D|w)$ is the likelihood, and is 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:

$$ p(w | D) \propto \frac{1}{Z_s} \exp(-S(w)) \tag{2} $$

To find the most probable weight vector $w_{MP}$, one should minimise the exponent $S(w)$, given by:

$$ S(w) = \beta E_D + \alpha E_N = \frac{1}{n} \sum_{i=1}^{N} (y(x_i, w) - t_i)^2 + \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2 \tag{3} $$

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 $\alpha$ and $\beta$ 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 $S(w)$ by a quadratic expansion around its minimum value $S(w_{MP})$. This leads to a Gaussian form:

$$ p(w | D) = \frac{1}{Z_s} \exp \left[ -S(w_{MP}) - \frac{1}{2} \Delta w^T A \Delta w \right] \tag{4} $$

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:

\[
p(w \mid D) = \int \int p(w \mid \alpha, \beta, D) p(\alpha, \beta \mid D) d\alpha d\beta
\]  

(5)

There are two ways to work with this expression. One is to integrate out \( \alpha \) and \( \beta \). The other is to approximate the integral by a function of the most probable values of the hyperparameters, \( \alpha_{MP} \) and \( \beta_{MP} \). This is called evidence approximation (see Bishop, 1995, p. 417, for a short comparison of these two approaches).

In the ‘evidence approximation’ approach, it is necessary to find \( \alpha_{MP} \) and \( \beta_{MP} \), which requires the maximisation of the posterior distribution of the hyperparameters:

\[
p(\alpha, \beta \mid D) = \frac{p(D \mid \alpha, \beta) p(\alpha, \beta)}{p(D)}
\]  

(6)

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

\[
p(D \mid \alpha, \beta) = \int p(D \mid w, \alpha, \beta) p(w \mid \alpha, \beta) dw
\]  

(7)

Maximisation of (7) leads to expressions that allow the iterative estimation of the hyperparameters \( \alpha \) and \( \beta \).

2.3. Automatic relevance determination

The prior that leads to expression (3) is not the best choice, since it applies the same restrictions (\( \alpha \)’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 \( \alpha \)’s. For example, a NN could have the weights \( w \) that connect the input nodes to the hidden layer controlled by a different \( \alpha \) from the weights \( v \) that connect the hidden layer to the output node. This would result in an error function defined as:

\[
S(w) = \frac{\beta}{2} \sum_{n=1}^{N} \left( y(x_n, w) - t_n \right)^2 + \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2 + \frac{\alpha}{2} \sum_{j=1}^{V} v_j^2
\]  

(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 \( \alpha \) values. The size of an \( \alpha \) would then serve as an indicator of the relevance of the input controlled by it; if the \( \alpha \) 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. In principle, 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 however is a measure of relevance that is based on the absolute values of the weights, which may be criticised. For linear models with normalised inputs, the size of the coefficient is indeed an indicator of the variable relevance to the output; for nonlinear models, however, the impact of the coefficient is not so clear.

2.4. Bayesian comparison of models

In theory, it should not be necessary to tune the number of hidden neurons, since Bayesian regularisation would be able to achieve automatically the best level of complexity in the NN. The values of the weights that connect the superfluous neurons would be driven down to zero, and this would be equivalent to disconnecting those neurons. In practice, however, one may want to use as few neurons as possible, if only to relieve the computational burden.

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

$$p(H_i | D) = \frac{p(D | H_i) p(H_i)}{p(D)} \quad (9)$$

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:

$$p(D | H_i) = \int \int p(D | \alpha, \beta, H_i) p(\alpha, \beta | H_i) d\alpha d\beta \quad (10)$$

Supposing that the hyperparameters $\alpha$ and $\beta$ 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):

$$\ln p(D | H_i) \propto -\alphaMP E_w^{MP} - \betaMP E_D^{MP} - \frac{1}{2} \ln A | + \frac{1}{2} \sum_{i=1}^k W_i \ln \alpha_i^{MP} + \frac{N}{2} \ln \beta^{MP} +$$

$$+ \ln M! + M \ln 2 + \frac{1}{2} \sum_{i=1}^k \ln \left( \frac{2}{\gamma_i} \right) + \frac{1}{2} \ln \left( \frac{2}{N - \gamma} \right) \quad (11)$$

Where $M$: number of hidden neurons, $W$: number of weights in each group, $N$: number of input vectors, $\gamma$: number of ‘effective’ weights. The models (defined by their number of hidden neurons) can then be compared in terms of their evidences; the ones with larger evidences are the most probable ones, for a given dataset.
2.5. Summary

It may 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, (b) a method for estimating the regularisation hyperparameters, (c) a method for selecting the NN inputs, and (d) a method for selecting the optimum number of neurons. The method in (b) is already widely accepted, and has been implemented in Matlab; methods (c) and (d) however are still open to discussion, are this study evaluates their application to a load forecasting problem.
3. Materials and methods

3.1. The forecasting problem, and the data

In this paper, we study the application of NNs to forecasting demand (load) in England and Wales at midday, three days ahead. Midday is a particularly important period in the summer months because it is often when peak demand occurs; we chose to forecast three days ahead to make the problem more difficult, which should allow the performances of the compared methods to be more clearly differentiated. The data were supplied by the National Grid (NG), a company responsible for the transmission of electricity in England and Wales. A previous study of NN load forecasting using this dataset was done by Taylor & Buizza (2002).

Figure 1 shows a plot of electricity demand in England and Wales at midday for each day, from 01/01/1997 to 31/12/1998. 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. Another noticeable seasonal feature is the strong seasonality within each week: there is a consistent difference of about 6000 MW between weekday and weekend demand.

![Figure 1 – Midday loads, 1997-1998, England and Wales](image)

Also, there is unusual demand on a number of ‘special days’, including public holidays, such as 1 January. 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 forecast of later values.

As explanatory variables, we use the demand for previous middays, weather variables, and dummies. Short to medium-term forecasting models must accommodate the variation in demand due to the seasonal patterns shown in Figure 1 and due to weather. At the 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 previous hours.

$$TE_t = \frac{1}{2}TO_t + \frac{1}{2}TE_{t-1}$$

(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.

$$CP_t = \begin{cases} W_t^{3/2}(18.3-TO_t) & \text{if} \quad TO_t < 18.3 \degree C \\ 0 & \text{if} \quad TO_t \geq 18.3 \degree C \end{cases}$$

(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.

The remaining input variables in our models were lagged load values (lags 3 to 7); and dummies to represent the days of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat), the three summer weeks when a large amount of industry closes (Week1, Week2, Week3), and 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 thus: the first 658 days (94 weeks) were used for training, the next 182 days (26 weeks) for cross-validation, and again 182 days (26 weeks) for testing.
3.2. Methods and implementation

All NN models we used were multilayer perceptrons, having one hidden layer with sigmoid activation functions, and one linear output node. We experimented with six methods for selecting the inputs and the number of the hidden neurons in the NNs.

The methods were divided into two groups: Methods 1 to 3 select the number of neurons by CV; Methods 4 and 5, by Bayesian model evidence. 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 an ARD priors was used in all models.

Method 1: selection of number of 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 to them large values of $\alpha$. The best number of neurons is then selected by CV.

Method 2: selection of both number of neurons and inputs by CV

The measure used to determine the relevance of a input was the mean absolute percent 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: $v = [2 3 4 5 6 7 10 12]$.
2) Set the number of hidden neurons as $m = v(1)$.
3) Start with the NN with all the available inputs. Run the model 10 times, and compute the average CV error (MAPE).
4) Compute the relevance of each input at a time by evaluating the change in the output caused by the omission of that input:
   a. Remove the first input, and compute the average CV error (MAPE) over 10 runs;
   b. Return the first input to the NN, and remove the second; compute again the average CV error over 10 runs;
   c. 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;
5) 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.
6) Repeat steps 4 and 5, until the NN is reduced to just one input.
7) Find the number of inputs that resulted in the best CV performance for the NN with \( m \) neurons.
8) Set the number of hidden neurons as \( m = v(2) \).
9) Repeat steps 3 to 7.
10) Increase the number \( m \) of hidden neurons, until the maximum \( m = 12 \) is reached.
11) Compare the performances of the NN with different numbers of hidden neurons in terms of CV error. Find the best model (in terms of inputs and number of neurons).

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

**Method 3: selection of number of neurons by CV, of inputs by ARD + CV**

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

1) Set the vector of possible values of the number of hidden neurons: \( v = [2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 10 \ 12] \).
2) Set the number of hidden neurons as \( m = v(1) \).
3) Start the NN with all available inputs; compute the average CV errors and the average \( \alpha \)’s for each input over 10 runs.
4) Remove the least relevant input (the one with largest \( \alpha \)).
5) Compute the average CV error and the average \( \alpha \)’s for each input over 10 runs.
6) Repeat (4) and (5), until the last input is reached.
7) Set the number of hidden neurons as \( m = v(2) \).
8) Repeat steps 3 to 6.
9) Increase the number \( m \) of hidden neurons, until the maximum \( m = 12 \) is reached.
10) Compare the performances of the NN with different numbers of hidden neurons in terms of the CV error. Find the best model (in terms of inputs and number of neurons).

**Method 4: selection of number of 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, \( v = [2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 10 \ 12] \). The best model among the ones compared is the one that has the largest Bayesian evidence.
Method 5: selection of number of neurons by evidence, 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 for 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: \( v = [2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 10 \ 12] \).
2) Start with a NN with all available inputs and number of neurons \( m = v(1) \).
3) Compute the average evidences, the \( \alpha \)’s for the inputs, and the CV errors, for each model (i.e., for different numbers of neurons), across 10 runs.
4) Find the model with largest evidence; remove from the dataset the variable that this model indicates as the least relevant input.
5) Repeat steps (3) and (4) until all the inputs have been removed.
6) Find the model with smallest CV error overall. Test this model over the testing sample.

To decide which model is the best one overall, in step (6), one would be tempted to compare the evidences: the model with the largest evidence would be the best one. However, this comparison is not possible, since comparing the models according to their evidences requires the assumption that all models have the same \( p(D) \) in Eq. (9); therefore, only models that share the same inputs may be compared.

Method 6: naïve forecaster

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

\[
\hat{L}_{12,d} = L_{12,d-7}
\]

That is, the forecast for the load at midday on day \( d \) is given by the load observed at midday one week before day \( d \).
4. Results and discussion

Table 1 shows the mean MAPEs obtained by each of the five methods in 20 runs over the test samples. Figures 2 displays the distribution of the MAPEs of each method, over the 20 runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selection of # neurons</th>
<th>Selection of inputs</th>
<th>Best models</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CV</td>
<td>Soft-pruning</td>
<td>4 20 1.35</td>
</tr>
<tr>
<td>2</td>
<td>CV</td>
<td></td>
<td>4 13 1.44</td>
</tr>
<tr>
<td>3</td>
<td>Rank inputs by ARD, stop pruning by CV.</td>
<td>4 17 1.34</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Evidence</td>
<td>Soft-pruning</td>
<td>1 20 1.67</td>
</tr>
<tr>
<td>5</td>
<td>Delete variable indicated on model w/ largest evidence; CV stop</td>
<td>1 12 1.60</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Naïve forecaster</td>
<td></td>
<td>1.80</td>
</tr>
</tbody>
</table>

Figure 2 – Out-of-sample errors, over 20 runs

In Method 1, there is no input selection, but only ‘soft-pruning’. The irrelevant inputs are not actually discarded – the Bayesian regularisation is expected to drive down to zero the weights that connect them to the NN so that their effect on the output practically disappears. Interestingly, the results are comparable to the ones obtained in the models with reduced sets of inputs (Methods 2 and 3), 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 found that soft-pruning was not enough, and used ARD only as a guide for pruning. Penny & Roberts (1999) and Lauret et al (2008) selected arbitrary cut-off points, and reported improved performance. Silva
& Ferreira (2007) used a ‘probe’ to set this cut-off point: they introduced a normal random variable among the inputs, and considered irrelevant those inputs that had α’s larger that the random variable. We tried this method 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 transmission company), the probes were always the variables that received the largest alphas. This may also be the reason why soft pruning worked so well with our data – since there are no variables that are really irrelevant, hard pruning was not required.

Method 2, based on CV both for model and input selection, performed slightly worse than Method 3, based on ARD. The difference in performance may perhaps be not very significant, but Method 2 was clearly inferior to the others in terms of the computational effort involved: in order to produce the results displayed in one of the graphs, it required the running of no less than 16,720 NNs, against 1,600 for Methods 3, and only 80 for Method 1. Method 3 (hard pruning) had results practically identical to those of Method 1 (soft pruning).

The results produced by Method 3 suggest that both the Bayesian regularisation and the ARD techniques were quite effective in controlling model complexity. Figure 3 shows the MAPEs produced by models ranging from 2 to 10 neurons, with all inputs and ARD priors. It can be seen that regularisation ensured that, although the best NN was the one with 4 neurons, the larger ones produced very similar results, and had no problems with overfitting.

![Figure 3. Errors by models with 2 to 10 neurons and all inputs (Plan I)](image)

The ranking of input relevance provided by ARD was found to be quite comparable across models with different numbers of neurons. Also, we found that the out-of-sample error changed in a
consistent fashion as the inputs were discarded according to ARD indication. Figure 4 shows the evolution of CV error in the NN with four neurons. The smallest error was reached with 17 inputs, but the error actually remained more or less stable until the number of neurons 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 Table 1 that this NN had practically equivalent out-of-sample errors 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.

Methods 4 and 5 used the evidence for selecting models. Method 4 used soft-pruning; in Method 5, all the models were run on the same set of inputs, and the input dropped was the one indicated by the NN with largest evidence. CV was used to stop pruning. Both models performed poorly. Over this dataset, the Bayesian evidence was clearly not useful as a tool for model selection. 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 always pointed to the simplest model under test. As can be seen in Table 1, the two methods that used evidence (Methods 4 and 5) ended with NNs that had only one neuron – while CV suggested four and seven for Plans I and II, respectively.
5. Conclusions

In this study, we found that the Bayesian framework was only partially useful for NN modelling of the available dataset. Regularisation and ARD proved to be very useful; evidence-based selection proved to be ineffective.

Bayesian regularisation proved to be very efficient in controlling the NN complexity and avoiding overfitting, and the use of ARD priors for soft-pruning led good results in this study. Also, the estimates of input relevance given by ARD proved to be very reliable in guiding hard-pruning procedures. The difficulty is, 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. The evidence tended always to point to models that were too simple and that produced unsatisfactory results out-of-sample. Penny & Roberts (1999) found that this 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. Bishop (1995) suggested that the evidence might be a less robust technique than the ARD for guiding modelling, considering that the estimation of the ARD $\alpha$’s depended on the sum of the eigenvalues of the Hessian, while the evidence depended on the product of them; therefore, the evidence tended to be much more sensitive to the uncertainty in the estimation of these 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 required 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, but we would certainly support more research on the application to load forecasting in order to clarify the situations where Bayesian modelling could replace CV techniques with a reliable performance.
Acknowledgement

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 Said Business School – Oxford.

6. References