International Journal of Neural Systems, Vol. 11, No. 1 (2001) 1–10
© World Scientific Publishing Company

Improved long-term temperature prediction by chaining of neural networks

M. DUHOUX, J. SUYKENS, B. DE MOOR, J. VANDEWALLE
Katholieke Universiteit Leuven, Department of Electrical Engineering, ESAT-SISTA.
Kardinaal Mercierlaan 94, B-3001 Heverlee (Leuven), Belgium.
Tel.: 32/16/321160, Fax: 32/16/321970.
E-mail: {*duhoux,suykens,demoor,vandewalle*}*@esat.kuleuven.ac.be*

When an artificial neural network (ANN) is trained to predict signals p steps ahead, the quality of the prediction typically decreases for large values of p. In this paper, we compare two methods for prediction with ANNs: the classical recursion of one-step ahead predictors and a new kind of chain structure. When applying both techniques to the prediction of the temperature at the end of a blast furnace, we conclude that the chaining approach leads to an improved prediction of the temperature and avoidance of instabilities, since the chained networks gradually take the prediction of their predecessors in the chain as an extra input. It is observed that instabilities might occur in the iterative case, which does not happen with the chaining approach. To select relevant inputs and decrease the number of weights in this approach, Automatic Relevance Determination (ARD) for multilayer perceptrons is applied.

1. Introduction

In industrial applications, the prediction of measurement outputs is sometimes very valuable to improve control[9]. In model predictive control (MPC) the determination of actions to steer future behaviour of the plant is automated by solving an optimization problem at each time step using the model[2]. When this control strategy is not automated, the operator can make decisions on what actions to undertake by knowing tendencies of a certain controlled variable in the future. The need for good predictors is thus based on the requirements of an operator to steer the plant in a desired way. A prerequisite is then that the experience of the operator is sufficient to interpret the results, and that the predicting structure is presenting these results in an understandable way.

When a p-step ahead predicting model is trained to predict signals p steps ahead in *one* stage, one sometimes notices that the quality of the prediction decreases fast as p increases. One of the reasons for this is the fact that the information in the inputs does not contain much information about the output, if this output lies far ahead in the future.

In this paper, we apply methods of Bayesian learning and input selection to a new predictive neural network architecture. The results are compared with the classical iterative approach. Both structures are thoroughly explained and examined by testing the predictive capabilities on industrial data, which were measured on a blast furnace installation. The goal is to achieve a good tendency prediction of the Hot Metal Temperature (HMT) 3 hours in advance.

In many cases the use of nonlinear models in an iterative way leads to the occurrence of instabilities. Model instabilities in neural networks can be avoided using NL_q theory[11]. The chaining approach we propose avoids prediction instabilities, but requires p different networks, whereas the iterative approach needs only one. Therefore, the weights and biases of the networks in the chain should be limited in number or be partially copied from well predicting networks in the chain. To the first end, we illustrate the concept of Automatic Relevance Determination (ARD), answering the question: which measured signals need to be included as network inputs, and what measured variables are irrelevant? The Bayesian approach followed here analyses which measurements and how many past samples of these are relevant inputs to the model of the future values of the hot metal temperature[3]. This technique proved to work better than other frequently used methods on the experimental data from the furnace.

This paper is organized as follows. In Section 2, we will explain how input-output predictive model chains are built. In Section 3, we review aspects of Bayesian learning, including input node weighting when applying ARD. In Section 4, we will give results on experimental data.

2. Chaining of model structures

Consider the nonlinear input-output model $f(\mathbf{y_k}, \mathbf{u_k})$, with $\mathbf{y_k}$ the vector $[y_k, y_{k-1}, \ldots, y_{k-q+1}]^T$ containing q past values of the measurement to be predicted, and $\mathbf{u_k}$ other relevant inputs with their necessary past included. To predict a signal *one*-step ahead, f can be trained using y_{k+1} as targets. For now, we do not discuss the methods for training the model f, but discuss generally different possible architectures. If the signal is to be predicted p samples ahead, the one-step ahead predictor can be extended to the same training principle, but now with y_{k+p} as targets for the training. The result is a one-*stage* p-*step* ahead predictor. However, this results in an information gap, since all (estimated) information $\hat{y}_{k+1}, \ldots, \hat{y}_{k+p-1}$ is not used. We will examine if results improve when one also uses the intermediate estimates, using two approaches, that could be summarised as *multi*-stage predictors.

There are two ways to train input-output models:

- We can train one single one-step ahead predictor and then use this model iteratively p times while shifting the inputs appropriately each time a previous prediction of the model is available (Figure 1). This is the standard iterative approach.
- One can also train p different predictors, each using the normal inputs and targets, but now with the prediction of the previous steps as *extra* inputs (Figure 2). This is some kind of chain architecture and only this implementation will result in a stable prediction for the studied industrial problem.

These two alternatives for p-step ahead prediction will now be explained in more detail.

2.1. *Iterative prediction*

If no separate predictive networks are trained, an iterative approach could be followed as indicated in Figure 1. Shifting new estimates through the input vector, and thus needing only one and the same one-step ahead predictor, new predictions are only based on a group of previous ones. When using this *sliding input* system, one trains the model f as:

$$\hat{y}_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-q+1}, \\ u_k, u_{k-1}, \ldots, u_{k-r}), \tag{1}$$

with q the number of temperature values we wish to include from the past, and r the same for other inputs. To calculate the p-step ahead prediction (by the last network), we use:

$$\begin{cases} \hat{y}_{k+p} &= f(\hat{y}_{k+p-1}, \hat{y}_{k+p-2}, \ldots, \hat{y}_{k+p-i}, \\ & \quad y_{k+p-i-1}, \ldots, y_{k+p-q}, u_k, u_{k-1}, \\ & \quad \ldots, u_{k-r}) \text{ if } i < q, \\ \hat{y}_{k+p} &= f(\hat{y}_{k+p-1}, \hat{y}_{k+p-2}, \ldots, \hat{y}_{k+p-q}, \\ & \quad u_k, u_{k-1}, \ldots, u_{k-r}) \text{ if } i \geq N, \end{cases} \tag{2}$$

where i depends on how many estimates have been calculated in a previous run: beginning with $i = 1$, one gradually has to include more previous estimates for the output $\hat{y}$, until one finally arrives at the p-th sample prediction, $\hat{y}_{k+p}$. Notice that the variable to be predicted uses a constant number of (estimated) previous values, namely q. That is the reason why the network f can be used in all the steps, i.e. *iteratively*. In fact it is trained in a feedforward way and used as a recurrent model to generate the prediction.

2.2. *Chaining of one-step ahead predictors*

Separately trained predictors f_j can also be used to calculate individual intermediate predictions. Therefore, it is necessary to train p networks f_j, $j = 1 \ldots p$, augmenting the vector $\mathbf{y_k}$ with estimates

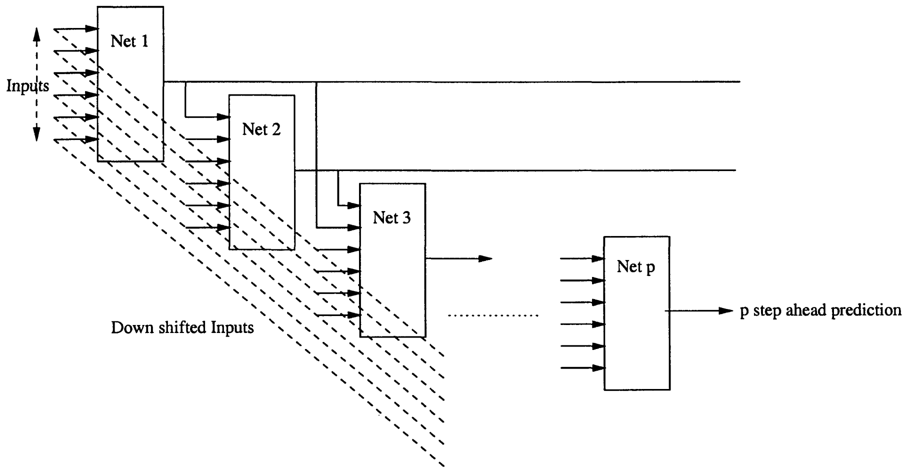

Fig. 1. In the standard classical approach, one iterates the model. New estimated outputs are shifted through the input vector and old inputs are discarded. All neural networks are identical.

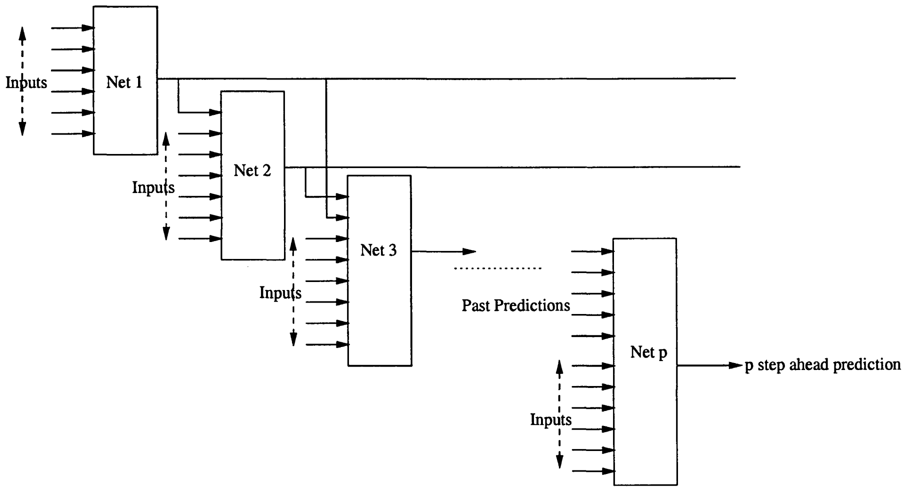

Fig. 2. A visual representation of the new approach using chains of neural networks: beginning with a classical one-step ahead predictor, the outputs are inserted in a next one-step ahead predictor, by adding the one-step ahead prediction to the input vector of the subsequent predictor. In each step the input vector of the neural networks grows with one additional component.

$\hat{y}_{k+j-1}$ from previous networks $f_{j-1\ldots1}$. The principle is illustrated by Figure 2. Be aware that it is necessary to train exactly p different networks, and that the model is therefore *not* used in an iterative way for predicting. In fact, one has a chain of one-step ahead predictors, where each k-th model is retrained, eventually with the final values (weights and biases) of the parameter vector $\theta_{\mathbf{k-1}}$ of its predecessor as a part of its own initial guess $\theta_{\mathbf{k0}}$. When using neural networks, the weights and biases of the previous network might form a good partial basis for the weights and biases of the next network in the chain.

3. Bayesian learning with neural network models

In this Section, we will review several Bayesian methods for the training of neural networks and the optimal selection of their inputs.

Multi-layer perceptrons (MLPs) neural networks correspond to a nonlinear mapping from input to output data[1,10]:

$$y = \mathbf{w}^{\mathbf{T}} \tanh(\mathbf{V}\mathbf{x} + \mathbf{b}), \tag{3}$$

with weights $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{V} \in \mathbb{R}^{n\times m}$, biases $\mathbf{b} \in \mathbb{R}^n$, inputs $\mathbf{x} \in \mathbb{R}^m$ and output $y \in \mathbb{R}$. From a Bayesian perspective, a choice of a neural network model can be seen as defining a prior probability distribution over the weights, and the neural network's learning process can be interpreted in terms of posterior probability distribution over the unknown mapping[6].

Bayesian regularization is related to a Levenberg-Marquardt optimization, using Bayesian theory for finding optimal training parameters[3]. Setting the weights $\mathbf{w}$ and $\mathbf{V}$ and the biases $\mathbf{b}$ as the unknown vector θ, the objective function to be minimized, classically a squared error $E_D = \sum_{k=1}^{N}(\hat{y}_k - y_k)^2$ where $\hat{y}_k$ stands for the predicted estimate of the true value y_k, is extended with a regularization term $E_\Theta = \theta^T\theta$ and becomes:

$$F(\theta) = \beta E_D + \alpha E_\Theta. \tag{4}$$

To determine the optimal value for α and β, one uses a Bayesian approach. Let $\mathcal{H}$ be a representation for the probabilistic model that specifies the functional form of the network[7]. Interpreting the error function E_D as defining a probability function of a noise model $P(D|\theta,\beta,\mathcal{H})$ with Gaussian characteristics (β defining a noise level $\sigma_\nu = \frac{1}{\beta}$), and interpreting the regularization term E_Θ as defining the same sort of Gaussian probability model $P(\theta|\alpha,\mathcal{H})$, then we can use Bayes' rule to write an equation for the inference of the parameters θ:

$$\begin{aligned} P(\theta|D,\alpha,\beta,\mathcal{H}) &= \frac{P(D|\theta,\beta,\mathcal{H})\mathcal{P}(\theta|\alpha,\mathcal{H})}{P(D|\alpha,\beta,\mathcal{H})} && (5)\\ &= \frac{1}{Z_F}\exp\left(-F(\theta)\right). && (6)\end{aligned}$$

When needing the best possible guess for θ, the cost function $F(\theta)$ is minimized. From (6), it can be seen that this corresponds to the most probable θ.

In a similar way one infers the hyperparameters α and β for the given data:

$$P(\alpha,\beta|D,\mathcal{H}) = \frac{\mathcal{P}(\mathcal{D}|\alpha,\beta,\mathcal{H})\mathcal{P}(\alpha,\beta|\mathcal{H})}{\mathcal{P}(\mathcal{D}|\mathcal{H})}, \tag{7}$$

where $P(D|\alpha,\beta,\mathcal{H})$ is called the *evidence* for α and β. Taking logarithms of the exponential Gaussian approximation for this posterior probability distribution *evidence* it can be shown[7] that this results in:

$$\begin{aligned} \alpha &= \frac{\gamma}{2E_\Theta(\theta)}, && (8)\\ \beta &= \frac{N-\gamma}{2E_D(\theta)}, && (9)\end{aligned}$$

with γ equal to the effective numbers of well-defined parameters

$$\gamma = l - 2\alpha \mathrm{Trace}(\mathbf{H})^{-1}, \tag{10}$$

representing a measure for how many parameters in the neural network are effectively used to reduce the error function[3]. l is the total number of parameters, N is the total number of training samples and $\mathbf{H}$ is the Hessian matrix of the objective function:

$$\mathbf{H} = \nabla^2 F(\theta) \approx 2\beta\mathbf{J}^T\mathbf{J} + 2\alpha\mathbf{I}_N, \tag{11}$$

where the Gauss-Newton approximation is calculated using $\mathbf{J}$, the Jacobian of the training set errors[5]. This Jacobian is available from the Levenberg-Marquardt (LM) training algorithm.

Extending the above to all input nodes, one gets Automatic Relevance Determination: all input nodes in the nonlinear input-output model receive a form of weighting by hyperparameters α_i. One shows that these correspond to an inverse variance value $\frac{1}{\sigma_i^2}$, that can be interpreted as the relative unimportance of that input/hidden/output-node. The more variance a node gets, the more influence it has on the output. This is in fact nothing but an extension on the derivations above[5,7], whereby now *every* input node in the network (weight and bias value) gets an a priori Gaussian distribution with the already mentioned variance $\sigma_i = \frac{1}{\alpha_i}$. In fact the cost function $F(\theta)$ has now regularization terms per parameter θ_i, and thus we now have:

$$F(\theta) = \beta E_D + \sum_{i=1}^{m} \alpha_i \theta_i^T \theta_i + \alpha_{m+1} E_H, \qquad (12)$$

where θ_i includes the appropriate components of the weight and bias vectors and matrix related to input node i, and E_H represents the same regularization terms for the remaining part of the hidden and output layer. Comparing with (4), we see that αE_Θ is changed to a product of two *vectors*, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$ and $\theta = [\theta_1, \ldots, \theta_m]^T$. The hyperparameters α_i can be calculated as follows:

1. Infer the parameters θ, for given (initially chosen) values of α_i and β.
2. Infer α_i and β.
3. Compare the models to see if there is an improvement in the cost (re-estimation).

All inference is done by using equations similar to (7).

For more information about ARD, we refer to[1,7]. More about Gaussian Processes can be found in[6].

4. Experimental results

The goal was to predict the future values of some temperature signal in the industrial furnace installation 3 hours in advance. We cannot give the precise origin nor exact values of all used measurements due to confidentiality agreements. All data values are scaled to the interval $[-1, 1]$. Since the data is sampled every 15 minutes, there are 12 samples to be predicted. Since the system under study has slow dynamics, it was necessary to also take the values $u_j(k-q)$ as inputs, the q-th previous sample of input $u_j(k)$, and this for $q = 1 \ldots 12$ to 24 for the temperature itself. One can easily see then that because of this, one still has a large input space of the network, even when the number of signals is not large.

The test set had 500 samples, taken from the series immediately following the training set. All training has been done in Matlab on 1300 to 3500 points of the four remaining signals. The `trainbr` routine in Matlab uses the results of the previous Section. An extended usage of Bayes' rule towards input selection is in the weighting of nodes in the network, implemented as `evidence` in the Netlab toolbox for Matlab implemented by Bishop and Nabney, available as a Matlab toolbox[1], also reviewed in the previous Section. In Netlab, two models $\mathcal{H}$ are possible: using a multilayer perceptron training. or by calculating a Gaussian Process.

(i) In the first case a Gaussian prior is set up for the prior over the weights. For an MLP, this boils down to the weighting of each input node i with a value α_i and the weighting of hidden and output nodes using β_i, γ_i respectively. Then the MLP is trained a couple of times, each time re-estimating the hyperparameters. The final collection is to be taken as the solution of the ARD routine.

(ii) Also in the second case, the Gaussian Process gets a separate hyperparameter for each input. After training the extended learning structure, one again has an idea of the relative importance of the inputs. We mean extended in the sense that more unknown variables, not only weights and biases but *also* hyperparameters, are to be calculated by the training algorithm.

When using '**trainbr**' on a one-layer neural network with five hidden neurons (tanh activation), each being delayed 24 to 36 delayed inputs, in total 124 inputs are used. All Figures have the (sometimes) scaled temperature on the y-axis, and a time index on the x-axis. When using the first iterative approach of Section 2.1, stability problems are becoming very destructive as can be seen on Figure 3. To solve this, we used the network chain as in Figure 2. From Figure 4, it is clear that the results are much better, because the model becomes stabilized.

When trying to validate the predictive capabilities in industrial context, we must use Zero Order Hold (ZOH) inputs, since interpolation is no longer possible in reality: the second value, lying in the

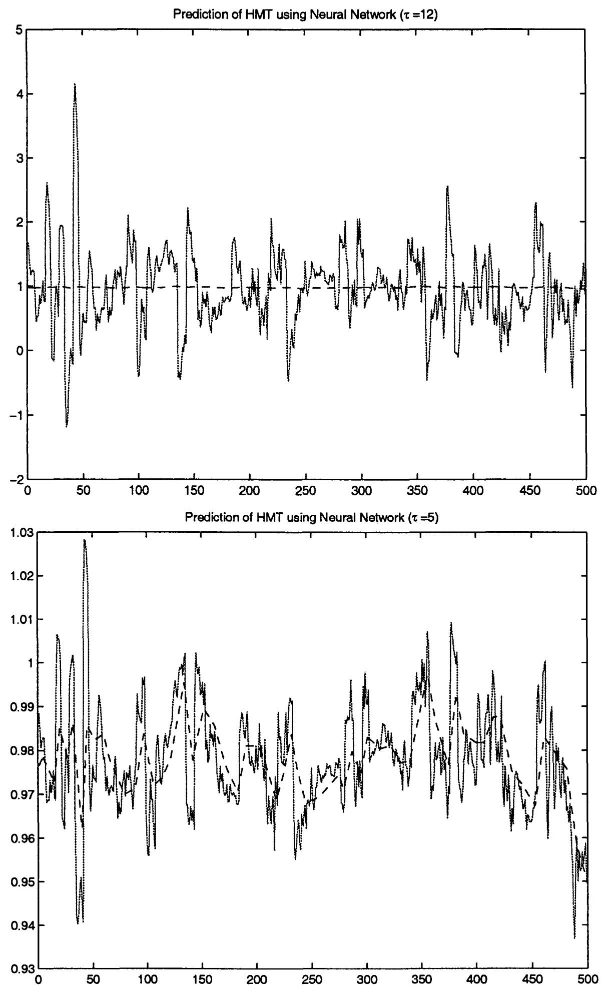

Fig. 3. Using the method of Figure 1, stability problems seem inherent for long term (full line) predictions. (**Top**) $p = 12$ or three hours. Only when $p \leq 4$, a reasonable prediction is possible. Further predictions are destroyed by growing instabilities. (**Bottom**) starting from $p = 5$ or 1h15 in advance. The dashed line is the true continuation.

future, cannot be used. From correlation analysis between predictor output values $\hat{y}$ and true temperature value y, we can see that the small cloud gradually changes into a broader cloud of points, but that the general quality of the prediction is still acceptable and much higher than the iterative use of one-step ahead predictors. We illustrate these results in Figure 5 in the same manner as before.

We investigated ARD as a robust input selection method. We verified this method on the same input input space as the one with four inputs we originally selected using linear correlation techniques, but with more available data. For a one-step ahead predictor, the hyperparameters are visualized in Figure 6. From this figure, we can conclude that some inputs are relatively more relevant than others and this in a more precise way than with linear correlation. However, the iterative use of a model with inputs weighted following an ARD analysis gives an unstable prediction, which is overcome by the chaining method.

5. Conclusion

In this paper we compared two methods for the long-term prediction of a signal in an industrial application and concluded that the standard *iterative* methods might not be ideal in some cases. A major im-

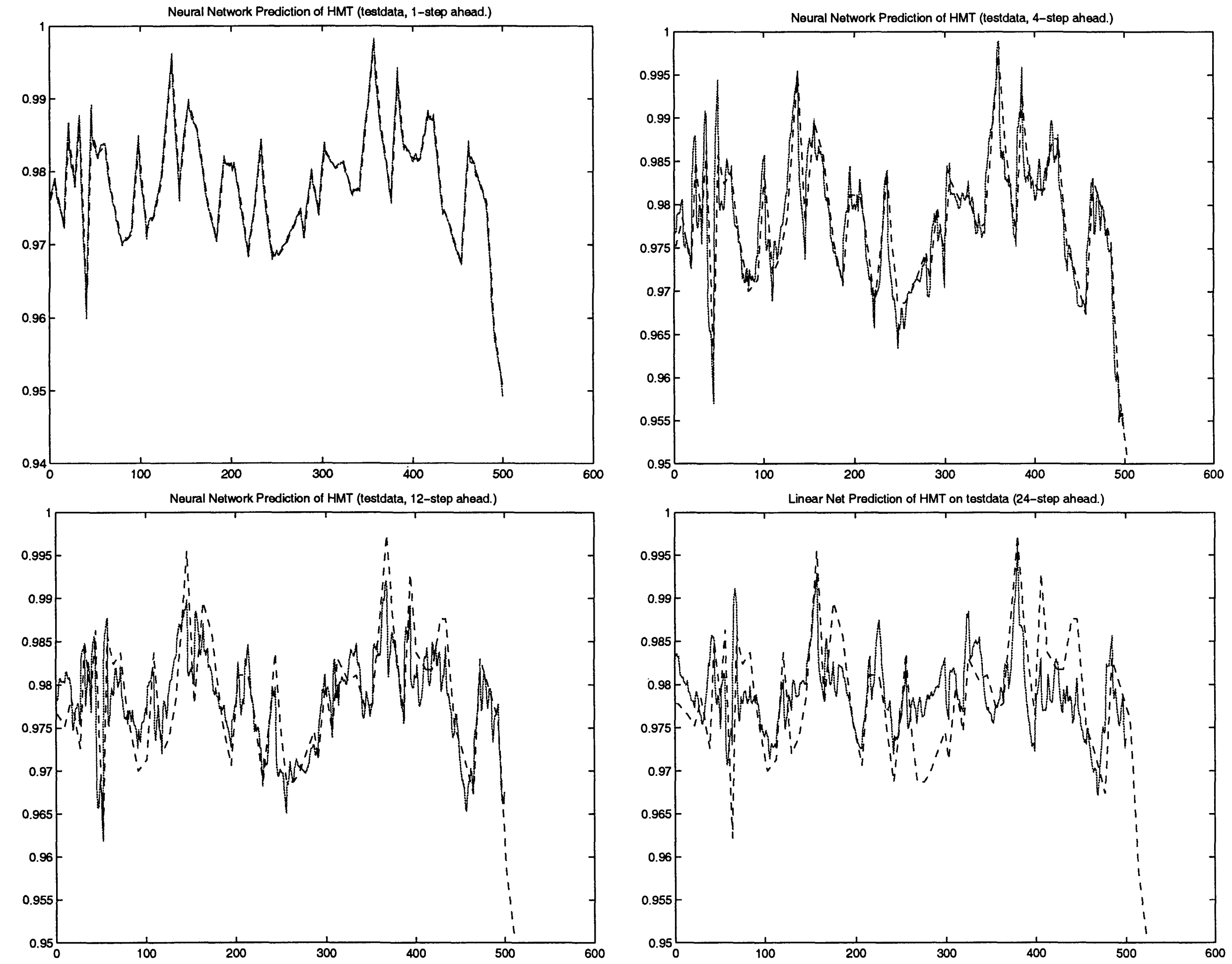

Fig. 4. Using the neural network chain of Figure 2, no instabilities occur, even for reasonably long term predictions. (**Top**) Prediction of one sample (left: 15 minutes) and four samples (right: one hour) ahead. (**Bottom**) The required prediction horizon: p=12, i.e. 3 hours (left). Even for very long term predictions, a vague tendency is still present: p=24, i.e. 6 hours (right).

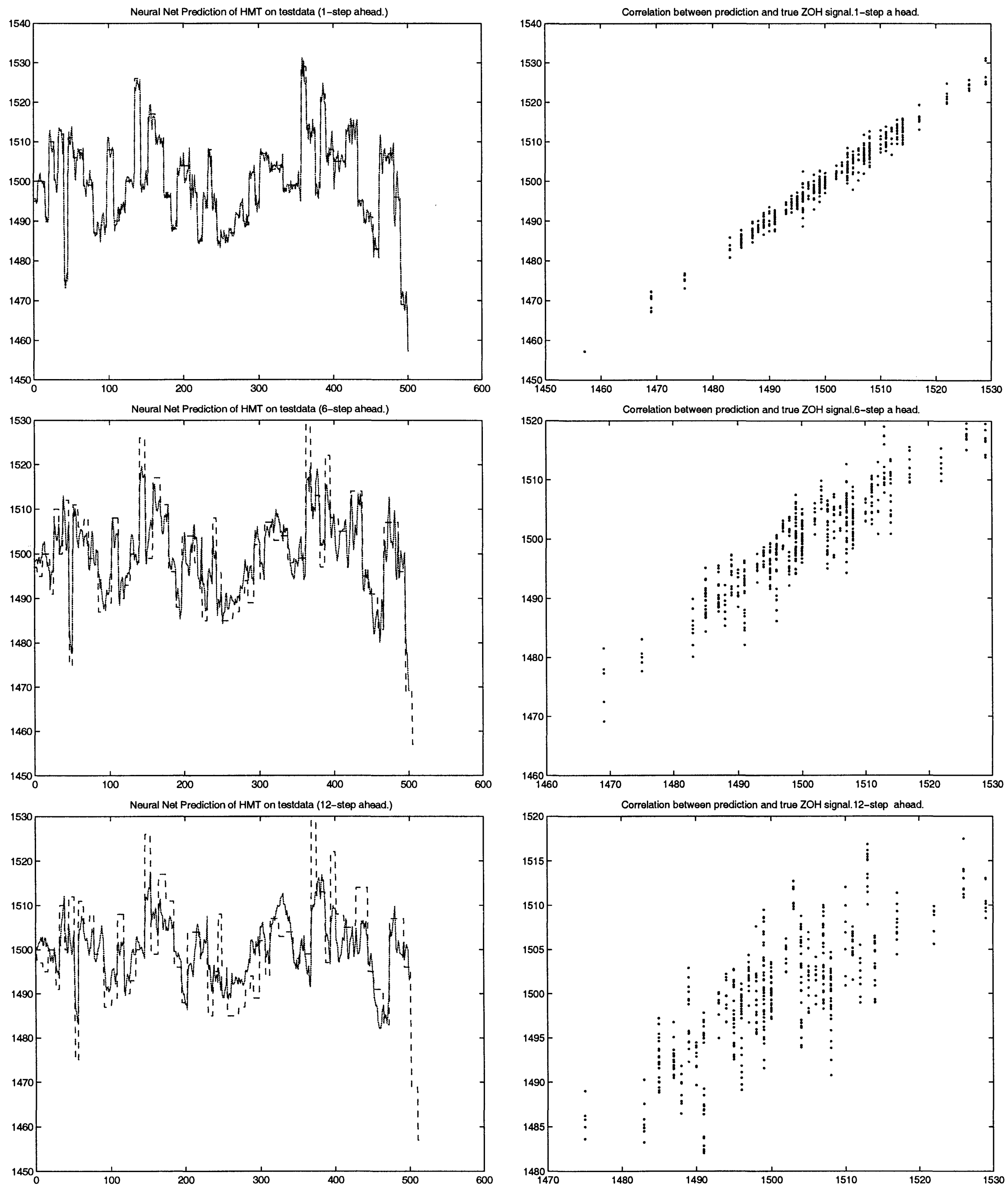

Fig. 5. The sample and hold input results : predictions and correlations are shown. Simulations shown here for p equal to 1, 6 and 12 samples, or 15 minutes, 1h30 and 3h in the future. In this case the temperature is not scaled.

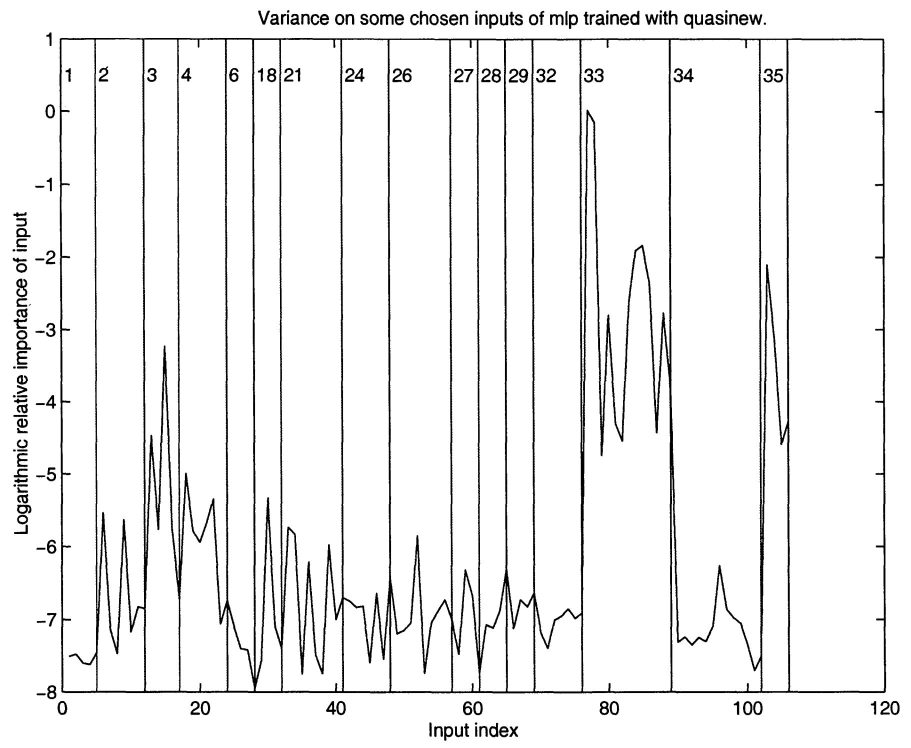

Fig. 6. Using automatic relevance determination (ARD) on the same input space as the first experiments, one observes that a lot of inputs are relatively irrelevant compared to a few others. We have plotted the logarithm of the relevance value with respect to the index of the input of the neural network. A broad selection of inputs was taken (the number at the top of each band), together with the past of these inputs, located on the index axis towards the left side of the band. The HMT has index 33. Other input spaces confirmed the relevance of the temperature and revealed some other relevant variables.

provement is achieved when several neural networks, trained separately to predict intermediate signals, are chained in such a manner that past predictions are inserted along the chain. In this way, the previous estimates are taken as extra inputs. This seems to avoid instabilities often occurring in the classical approach, and basically makes it much more useful.

In order to minimize the number of inputs and thus reduce the large training problem for the chaining approach in size, input selection is necessary. This input selection problem was illustrated using Automatic Relevance Determination. It seems that the weighting of the input nodes is no guarantee whatsoever that instabilities will not occur when using predictive models iteratively, but an acceptable result can nevertheless be achieved when using less inputs.

Problems that result from the proposed approach, are first that the amount of networks to be trained equals the number of samples to be predicted. The exact value of the prediction is mostly not achieved. One should also be able to detect the reasons for instabilities to occur in the standard iterative approach. These instabilities might be avoided by imposing NL_q stability theory during training of the network. In this paper we have shown that chaining of neural networks can be a good alternative for this.

Acknowledgments

Maarten Duhoux is a Research Assistant with the K. U. Leuven. Johan Suykens is a postdoctoral researcher with the F.W.O. (Fund for Scientific Research-Flanders). Bart De Moor and Joos Vandewalle are Full Professor at the K.U.Leuven. This work is supported by several institutions: the Flemish Government (Research Council K.U.Leuven : Concerted Research Action Mefisto-666; the FWO projects G.0240.99, G.0256.97, and Research Communities: ICCoS and ANMMM; IWT projects: EUREKA 1562-SINOPSYS, EUREKA 2063-IMPACT, STWW), the Belgian State, Prime Minister's Office (IUAP P4-02 (1997-2001) and IUAP P4-24 (1997-2001)), the European Commission (TMR networks: ALAPEDES and System Identification; Brite/Euram Thematic network: NICONET), and the Center for Research in Metallurgy in Liege. The scientific responsibility is assumed by its authors.

6. References

1. Bishop C.M., *Neural networks for pattern recognition*, Oxford University Press, 1995.
2. Camacho E.F., Bordons C., *Model Predictive Control*, Springer-Verlag, 1998.
3. Foresee D.F., Hagan M.T., "Gauss-Newton Approximation to Bayesian learning", *International Joint Conference on Neural Networks*, **3**, 1930-1935 (1997).
4. Simon Haykin, "*Neural Networks. A Comprehensive Foundation.*", MacMillan, 1994.
5. MacKay D.J.C., "Bayesian Interpolation", *Neural Computation*, **4**, 415-447 (1992).
6. MacKay D.J.C., "Probable Networks and Plausible Predictions — A Review of Practical Bayesian Methods for Supervised Neural Networks", *Network of Computation in Neural Systems*, **6**, 469-505 (1995).
7. MacKay D.J.C., "Bayesian Non-linear modelling with neural networks", Cambridge University Programme for Industry, `http://wol.ra.phy.cam.ac.uk/mackay/`, 1995.
8. MacKay D. J. C., "Bayesian non-linear modelling for the 1993 energy prediction competition", *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, Kluwer, Dordrecht, 221-234 (1996).
9. Suykens J.A.K., Vandewalle J. and De Moor B., *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Kluwer Academic Publishers, Boston, 1996.
10. Suykens J.A.K., Vandewalle J. (Eds.), *Nonlinear Modelling: advanced black box techniques*, Kluwer Academic Publishers, Boston, 1998.
11. Suykens J., Vandewalle J., De Moor B., "NLq theory: checking and imposing stability of recurrent neural networks for nonlinear modelling", *IEEE Transactions on Signal Processing*, **45**, 11, 2682-2691 (1997).
12. A. S. Weigend and Gershenfeld, N. A. (Eds.), "The Future of Time Series: Learning and Understanding.", *Time Series Prediction: Forecasting the Future and Understanding the Past*, 1–70, Addison-Wesley, Reading, MA, 1993.