Methodol Comput Appl Probab (2010) 12:623–645
DOI 10.1007/s11009-009-9153-3

Level Crossing Prediction with Neural Networks

Halfdan Grage · Jan Holst · Georg Lindgren · Mietek Saklak

Received: 12 August 2008 / Accepted: 28 July 2009 /
Published online: 13 August 2009

Abstract A level crossing predictor or alarm system with prediction horizon k is said to be optimal if it, at time t detects that an upcrossing will occur at time $t + k$, with a certain high probability and simultaneously gives a minimum number of false alarms. For a Gaussian stationary process, the optimal level crossing predictor can be explicitly specified in terms of the predicted value of the process itself and of its derivative. To the authors knowledge this simple optimal solution has not been used to any substantial degree. In this paper it is shown how a neural network can be trained to approximate an optimal alarm system arbitrarily well. As in other methods of parametrization, the choice of model structure, as well as an appropriate representation of data, are crucial for a good result. Comparative studies are presented for two Gaussian ARMA-processes, for which the optimal predictor can be derived theoretically. These studies confirm that a properly trained neural network can indeed approximate an optimal alarm system quite well – with due attention paid to the problems of model structure and representation of data. The technique is also tested on a strongly non-Gaussian Duffing process with satisfactory results.

Keywords ARMA-process · Detection probability · Duffing oscillator · False alarm · Gaussian process · Operating characteristic · Optimal alarm · Weight decay

AMS 2000 Subject Classifications Primary 62M45; Secondary 60G25 · 60G70 · 62M20

Jan Holst, deceased March 3, 2006.

H. Grage
Novo Nordisk A/S, Bagsværd, Denmark

J. Holst · G. Lindgren (✉)
Mathematical Statistics, Lund University, Lund, Sweden
e-mail: georg@maths.lth.se

M. Saklak
Visma Software AB, Limhamn, Sweden

1 Introduction

In a surveillance system one important objective is to predict whether or not a critical variable will exceed some predetermined level sometime in the future. The prediction horizon may depend on the purpose of the surveillance system, from being an early warning system to an acute emergency alarm. In any case, it is the level crossing events, not the exact value of the variable, that are of interest in this article. To allude to the sometimes drastic consequences of a level crossing, we will use the term *catastrophe* for the exceedance event. Practical examples are warning systems for bad air quality, extreme floods, and health surveillance systems.

Neural network algorithms are often successfully used to produce predictions of non-Gaussian time series, usually minimizing a quadratic loss function. When the aim is to predict whether or not the time series will exceed a certain fixed level, the mean square error is not very useful as a loss function, and several ad hoc modifications of the standard neural network algorithms have been proposed in the literature to improve performance.

An alarm system is a device which, based on current information, predicts whether a catastrophe is going to occur at a specified time in the future. At each moment the alarm system signals whether or not a catastrophe is going to happen. An alarm is *false* if no catastrophe (level crossing) occurs at the specified time, a catastrophe is *undetected* if the alarm is not sound at the required prewarning time. The success of an alarm system is measured by its detection probability and its false alarm rate. The following definition specifies what we mean by optimal alarm prediction.

Optimal alarm: *An optimal alarm system for a specified set of available data is defined as a system which, for a given probability of detecting a catastrophe, has the highest probability of correct alarm.*

By exploiting the analogy between alarm systems and hypothesis testing, de Maré derived, in a general context, an optimal alarm system based on a likelihood-ratio argument, and for prediction of level crossings in Gaussian processes, Lindgren (1985) restated this result and gave an explicit formulation of the optimal alarm in terms of the pair of the predicted value and the predicted growth rate of the process; (de Maré 1980; Lindgren 1985). The optimal alarm system should give alarm when the prediction exceeds a variable alarm level that adjusts according to the expected growth rate of the process. Formulated in terms of the predicted value and growth rate, the optimal alarm is simply defined by a certain region in $\mathbb{R}^2$, the *alarm region*.

Central to the problem are the *operating characteristics* (OC) "probability of correct alarm" and "probability of detected catastrophe", introduced in Lindgren (1975), also treated in Beckman et al. (1990). The idea of including the expected growth rate of the process into the alarm system was exploited in Svensson et al. (1996), which also contained a comparison with the naive alarm system, based solely on predictions of the process-value. The works by de Maré and Svensson et al. are those of most immediate concern to the investigations in this article.

The aim of the present article is to investigate how well an artificial neural network can identify the optimal alarm region, both for the case when the full information is used as input to the network, and for the restricted case when only the pre-calculated value and growth rate predictions are used as inputs. For a Gaussian time series, the correct form of the alarm region is exactly known, and hence one can directly check

the accuracy. We also use the network on a highly non-Gaussian process, for which no exact solution is known.

The article is organized as follows: In Section 2 the alarm problem is described and the definition of an optimal alarm system is presented. In Section 3 the neural network is introduced as an alarm system. The procedure of finding a good network is quite involved, including considerations of model order and local minima of the error function, and this is the theme of Section 4. Four examples are presented in Section 5. Three of them are based on data generated by Gaussian ARMA-processes, which makes it possible to compare our results with the results in Svensson et al. (1996) on the optimal predictor. The fourth example is a non-linear process of Duffing-type, and the aim is to see if the methodology based on the Gaussian theory can be of any use in the study of a non-Gaussian process as well.

Most of the current neural network and forecasting literature deals with prediction of future values of a time series, and most neural network algorithms aim at minimizing the prediction error. The specific problem of warning for exceedance of extreme levels has recently received attention in the environmetric literature by suitable modifications of the network algorithms; see Nunnari (2006), Cawley et al. (2007), Dutot et al. (2007).

2 The Optimal Catastrophe Predictor

2.1 Alarm Characteristics

We consider a stationary stochastic time series $X(t),\ t \in \mathbb{Z}$. In this work a catastrophe at time t is defined as the event that $X(t)$ has an upcrossing of a critical level u at time t, denoted

$$C_t = \{X : X(t-1) \le u < X(t)\}\ .$$

An alarm system is then an algorithm which predicts such an event.

Let $A_{t,k}$ denote the event that an alarm is given at time $t-k$, predicting a catastrophe C_t at time t. Let $\widehat{X}_k(t)$ be a k-step prediction of $X(t)$—i.e. the process $\widehat{X}_k(t)$ predicts the value of $X(t)$ using data from time $t-k$ and older. A "naive" approach to the alarm problem would be to use such a k-step predictor and give alarm for a catastrophe C_t if and only if $\widehat{X}_k(t)$ has an upcrossing of some specified alarm level $\widehat{u}$ at time t. The event of alarm, defined in this way, can be written

$$A_{t,k}^{naive} = \left\{\widehat{X}_k : \widehat{X}_k(t-1) \le \widehat{u} < \widehat{X}_k(t)\right\}\ . \tag{1}$$

When judging the performance of such an alarm system, the quantity of interest is not so much how good $\widehat{X}_k(t)$ is as a prediction of $X(t)$, but rather the ability of the system to detect catastrophes without giving too many false alarms. Figure 1 shows a simulated ARMA(6,4)-process together with mean-square optimal predictor. This process is to be studied in more detail in Section 4.

For a good alarm system, the probabilities

$$P(C_t \mid A_{t,k}) = P(\textit{correct alarm})$$

$$P(A_{t,k} \mid C_t) = P(\textit{detected catastrophe})$$

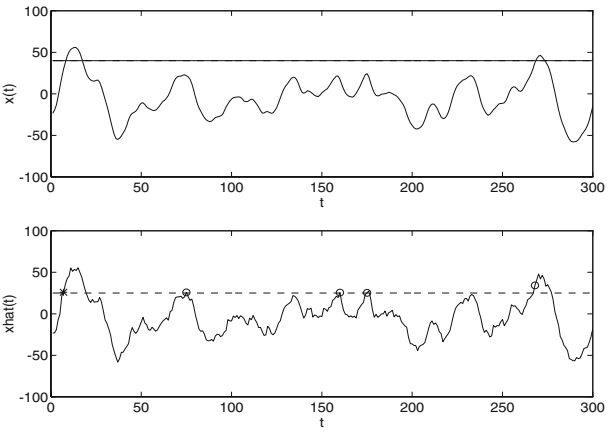

Fig. 1 Illustration of the "naive" alarm approach. *Top*: an ARMA(6,4)-process $x(t)$ and catastrophe level *solid line*, $u = 40$. *Bottom*: two-step prediction of the process and alarm level *broken line*, $\widehat{u} = 25$. *asterisks* = correct alarm (two-step prediction, i.e. $k = 2$), *circles* = false alarm (the last alarm maybe should be counted as late rather than false, since it comes only one time unit before the catastrophe)

both should be as high as possible. In Lindgren (1975) these were termed *the operating characteristics*, OC, of the alarm system. If we use these alarm characteristics as a criterion, as in de Maré (1980) and Lindgren (1985), we can say that an alarm system $\mathcal{A}_k = \{A_{t,k}\}$ is optimal, if for each t,

$$P(C_t \mid A_{t,k}) = \sup_{B_{t,k}} \left\{ P(C_t \mid B_{t,k}) : P(B_{t,k} \mid C_t) = P(A_{t,k} \mid C_t) \right\} . \tag{2}$$

With this definition, an alarm system is optimal if it, among all systems $\mathcal{B}_{t,k}$ with the same ability to detect a catastrophe, has a the smallest false alarm probability.

2.2 Available Information and the Optimal Alarm

Now, let the information available at time $t - k$ be condensed in the value at time t of a stochastic process $\mathbf{Y}$ with values in $\mathbb{R}^n$. This information process could, for instance, consist of n consecutive values of the X-process: $\mathbf{Y}(t) = (X(t - k - n + 1), \ldots, X(t - k))^\tau$, but it may also contain other explanatory variables. Then, let alarm be given at time $t - k$ for a catastrophe at time t if $\mathbf{Y}(t) \in A_{t,k}$ (cf. Eq. 1). The formal similarity to power considerations in test theory suggests that some likelihood ratio could be useful. According to de Maré (1980), an alarm system is optimal in the above sense Eq. 2, if it gives alarm for all outcomes $\mathbf{y} = \mathbf{y}(t)$ such that

$$\frac{dP_{\mathbf{Y}(t)}(\mathbf{y} \mid C_t^*)}{dP_{\mathbf{Y}(t)}(\mathbf{y} \mid C_t)} \leq c' = constant , \tag{3}$$

where C_t^* is the complement of C_t, i.e. no catastrophe at time t, and $P_{\mathbf{Y}(t)}(\mathbf{y} \mid C_t)$ is the distribution of $\mathbf{Y}(t)$ conditioned on C_t. If the process $X(t)$ is strictly stationary, so that the probabilities $P(C_t)$ and $P(C_t^*)$ are constant in time, it can be shown, as in Svensson et al. (1996), that the inequality (3) is equivalent to

$$P(C_t \mid \mathbf{Y}(t)) \geq c . \tag{4}$$

Thus, an optimal alarm system gives alarm when the conditional probability of catastrophe, given the available information, exceeds a specified level. The constant c determines the boundary of the alarm region in the "$\mathbf{Y}$-space", when we use the

inequality (4) as an alarm criterion, and it will henceforth be called the *boundary probability*.

2.3 The Alarm Region for the Gaussian Case

When $X(t)$ and $\mathbf{Y}(t)$ are jointly Gaussian, with continuous time parameter t, optimal alarm can be formulated simply. We get the most simple formulas for a differentiable process in continuous time, and we state the explicit result for such a process.

Let $\widehat{x}(t \mid \mathbf{y}(t))$ be the best linear predictor of $x(t)$ given $\mathbf{y}(t)$, and write

$$\widehat{x}_{u,\mathbf{y}}(t)' = \widehat{x}(t \mid \mathbf{y}(t), x(t) = u)'$$

for the best linear predictor of the derivative $x(t)'$ given $\mathbf{y}(t)$ and given $x(t) = u$. This means that

$$\widehat{x}(t \mid \mathbf{y}(t), x(t) = u)'$$

is the expected growth rate given the available information, under the further condition that there is actually an upcrossing by $x(t)$. Further, let $\sigma^2_{x\cdot\mathbf{y}}$ be the conditional variance of $X(t)$ given $\mathbf{Y}(t)$, $\sigma^2_{x'\cdot x\mathbf{y}}$ be the conditional variance of $X'(t)$ given $X(t), \mathbf{Y}(t)$, and write $\Psi(x) = \phi(x) + x\Phi(x)$, where $\phi(x)$ and $\Phi(x)$ are the standardized normal density and distribution functions.

Theorem 1 (Corollary 4,6, (Lindgren 1985)) *The optimal alarm for a u-upcrossing starts as soon as the predicted value* $\widehat{x}(t \mid \mathbf{y}(t))$ *exceeds the variable lower alarm level*

$$u^- = u - \sigma_{x\cdot\mathbf{y}} \sqrt{2 \log \Psi \left[\frac{\widehat{x}_{u,\mathbf{y}}(t)'}{\sigma_{x'\cdot x\mathbf{y}}} \right] + 2 \log \frac{\sigma_{x'\cdot x\mathbf{y}}}{\sigma_{x\cdot\mathbf{y}}} + K} \tag{5}$$

for some constant K.

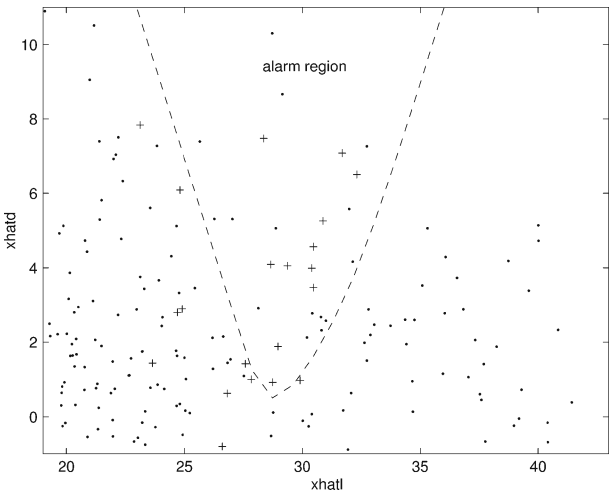

Fig. 2 Example of an optimal alarm region in two dimensions. *plus signs* = catastrophe points, *dots* = non-catastrophe points, *broken line* border of alarm region (determined by the boundary probability c, here $c = 0.3$). The data are from the same ARMA(6,4)-process as was used in Fig. 1

Table 1 Detection and reliability parameters for the ARMA(6,4)-process in Figs. 1 and 2 with boundary probability $c = 0.3$

Alarms		Catastrophes	
Correct	11 (0.44)	Detected	11 (0.52)
False	14 (0.56)	Undetected	10 (0.48)
Total	25	Total	21

Numbers in parentheses give relative frequencies within each class. The OC-estimates (according to Eq. 7) are $\widehat{P}(A_2 \mid C) = 0.52$ and $\widehat{P}(C \mid A_2) = 0.44$. The data set consists of 2000 points

(The alarm should stop when the the predicted value $\widehat{x}(t \mid \mathbf{y}(t)$ exceeds an upper alarm level u^+ with the minus sign in Eq. 5 replaced by a plus sign.) This means that it is expected that the process is already well above the catastrophe level, so no upcrossing is foreseen!) The essence of this result is that the condition $u^- < \widehat{x}(t \mid \mathbf{y}(t)) < u^+$ defines a condition on the $\mathbf{y}(t)$-values that guarantees that Eq. 4 is satisfied. The constant K depends on the boundary probability c.

Returning to the discrete time case we just replace the process value and derivative by an average and a difference:

$$\begin{aligned} x_L(t) &= \frac{x(t) + x(t-1)}{2}\,, \\ x_D(t) &= x(t) - x(t-1)\,. \end{aligned} \tag{6}$$

The occurrence of a u-level upcrossing is then equivalent to the event $|x_L(t) - u| < x_D(t)/2$, and the continuous time alarm condition (5) is replaced by a similar, but less explicit expression for the alarm region. The pairs $(\widehat{x}_L(t), \widehat{x}_D(t)) \in \mathbb{R}^2$ can be represented as points in the plane, as in Fig. 2. Some of these prediction points will be associated with real catastrophes and these are marked by $+$ in the figure.

In this way the alarm system acts as a *classifier*, demarcating an alarm region in the $(\widehat{x}_L(t), \widehat{x}_D(t))$-plane. All the prediction points in the alarm region will give rise to an alarm and those outside the alarm region will not. Table 1 highlights some important aspects of the situation as it appears for the process in Figs. 1 and 2. Unless otherwise noted, we will henceforth assume that the process is stationary and thus we can write $A_{t,k} = A_k$ for simplicity. The operating characteristics, OC, are estimated by

$$\begin{aligned} \widehat{P}(A_k \mid C) &= \frac{\#(\textit{alarm and catastrophe})}{\#(\textit{catastrophe})}\,, \\ \widehat{P}(C \mid A_k) &= \frac{\#(\textit{alarm and catastrophe})}{\#(\textit{alarm})}\,. \end{aligned} \tag{7}$$

3 The Neural Network as Catastrophe Predictor

3.1 The Neural Network Model

This is a summary of the type of network used (cf. Bishop 1995). In the first couple of studies a two layer feed forward network will be used. In such a network (see Figs. 3 and 4 for graphical illustrations of the network structures and the corresponding functional relations) with activation function g, the output at node i is computed

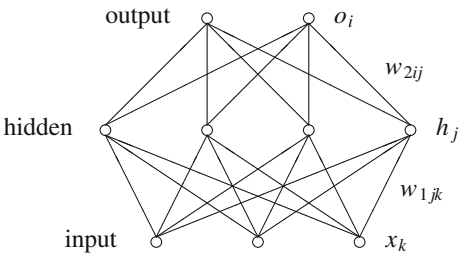

Fig. 3 Neural network with three input nodes, four hidden nodes and two output nodes

as

$$o_i(\mathbf{y}) = g\left(\sum_j w_{2ij}\, g\left(\sum_k w_{1jk} y_k\right)\right),$$

where $\mathbf{y} = (y_1, \ldots, y_n)$ is an input pattern presented to the network, w_{1jk} denotes the weight from input node k to hidden node j and w_{2ij} denotes the weight from hidden node j to output node i. A bias term $b_\cdot$ has been included in the sum as a weight $w_{\cdot 0}$ connected to an input constantly clamped to $+1$. For a "k-step-ahead" predictor of a time series based on n consecutive values of the same series, the input variable at time $t-k$ for prediction of $x(t)$ is

$$\mathbf{y}(t) = (x(t-k-n+1), \ldots, x(t-k))^{\mathsf{T}}.$$

When the network is used as a classifier, as in our case, we want to classify $\mathbf{y}(t)$ as belonging to the alarm region or not, and then we need only one output node that assigns the data $\mathbf{y}(t)$ to one of the two classes $C = C_t$ or $C^* = C_t^*$ (under the stationarity assumption). The network output is then

$$o(\mathbf{y}) = g\left(\sum_j w_{2j}\, g\left(\sum_k w_{1jk} y_k\right)\right). \tag{8}$$

Let $\tau = \tau(t)$ be the target, i.e. $\tau = 1$ or $\tau = 0$, depending on whether a catastrophe occurs at time t or not. Further, let the output be $o = o(\mathbf{y})$ when the information

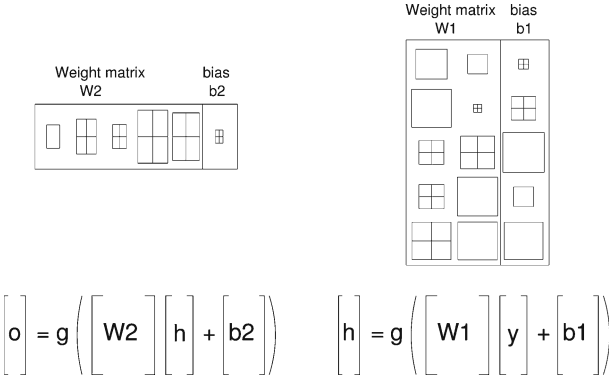

Fig. 4 Graphical representation of network matrices. Each matrix element is represented by a *small rectangle*. The size of each *rectangle* is proportional to the numerical value of the corresponding matrix element (a + in the rectangle means positive value, empty rectangle means negative value). $\mathbf{y}$ = input vector, $\mathbf{o}$ = output vector (cf. (8))

is $\mathbf{y} = \mathbf{y}(t)$, and let T be the random variable corresponding to τ, i.e. the indicator function defined as

$$T_t = \begin{cases} 1 \text{ if } X(t) \in C\,, \\ 0 \text{ if } X(t) \in C^* \end{cases}.$$

To measure the distance between target and output we use the Kullback-Leibler distance, also called the cross entropy function, (for a general reference for the following facts, see e.g. Bishop 1995, Sect. 6.7):

$$\mathcal{F} = -[T \log o(\mathbf{Y}) + (1 - T) \log (1 - o(\mathbf{Y}))]\,,$$

with expectation

$$E[\mathcal{F}] = -E[T \log o(\mathbf{Y}) + (1 - T) \log (1 - o(\mathbf{Y}))]\,,$$

where the expectation is taken over all possible T and all possible $\mathbf{Y}$. This distance is minimized when the output $o(\mathbf{Y})$ is equal to the conditional expectation

$$E[T \mid \mathbf{Y}] = P(C \mid \mathbf{Y})\,.$$

A neural network to compute this conditional probability can be estimated from a long time series. The error function to be minimized is

$$Q_0 = -\sum_p \left(\tau^{(p)} \log o^{(p)} + (1 - \tau^{(p)}) \log (1 - o^{(p)})\right),$$

where $\tau^{(p)}$ is the target under input number p, and correspondingly for the network output $o^{(p)}$. This error function diverges if the output is at the wrong extreme. The importance of this property becomes clear in a situation like ours, where the catastrophes only make up a small part of the data.

3.2 Network Considerations

To complete the model we take the function $g(x) = 1/(1 + e^{-x})$ as activation function and use the back propagation algorithm to update the weights of the network from a series of data. If we, in analogy with Eq. 8, write $h_j(\mathbf{y}) = g(\sum_k w_{1jk} y_k)$ for short, the formulas for the weight changes are given by

$$\Delta w_{2j} = -\eta \sum_p (o^{(p)} - \tau^{(p)}) h_j^{(p)}\,,$$

$$\Delta w_{1jk} = -\eta \sum_p (o^{(p)} - \tau^{(p)}) w_{2j} h_j^{(p)} (1 - h_j^{(p)}) y_k^{(p)}\,,$$

where η is a step length parameter, which has to be adjusted with respect to the direction with the fastest convergence.

Back propagation is well suited for parallel computation, but may not be very efficient on serial computers. It is, however, the algorithm implemented in the Matlab toolbox (Demuth and Beale 1992), which has been used in this work. Unfortunately, there is no guarantee that this method will find the global minimum. The problem of local minima is discussed in Bishop (1995), Demuth and Beale (1992), Ripley (1993, 1994).

Now the question arises: how many parameters shall one choose? As for other methods of function estimation, too many free parameters in the model may result in overfitting of the model to the data presented, leading to poor generalization or prediction ability. When it comes to neural networks it is not only a question of how many weights to use but also to find a suitable architecture—i.e. the number of weights in the input and hidden layers respectively and the degree of connectivity.

One strategy is to begin with a network that is too large for the problem and then successively remove superfluous weights using the method of weight decay, as described in Bishop (1995), and Weigend et al. (1990). If we in each updating step let the weights decay according to

$$w_{ijk}^{new} = (1 - \varepsilon_{ijk}) w_{ijk}^{old},$$

weights with little influence on the output will eventually decrease to zero. The decay terms ε_{ijk} are chosen as

$$\varepsilon_{ijk} = \frac{\eta\gamma}{(1 + (w_{ijk}^{old})^2)^2}, \tag{9}$$

where η is the step length in the gradient descent procedure and γ is the decay rate. The application of weight decay corresponds to adding an extra complexity term to the error function, which in our case will take the form

$$Q = Q_0 + \frac{1}{2}\gamma \sum_{ijk} \frac{w_{ijk}^2}{1 + w_{ijk}^2},$$

where Q_0 is the original error function. For large $|w_{ijk}|$ the cost is $\frac{1}{2}\gamma$, whereas for small weights it is almost zero. The price to pay for the structure simplification for this modification of the error function, is that one looses the nice interpretation of the optimum as an estimate of the conditional expectation.

4 Special Methodological Considerations

4.1 The Example Processes

The specific cases to be studied here are an ARMA(6,4)-process, an ARMA(2,1)-process, both Gaussian, and a Duffing process, which is a highly non-Gaussian process. For the ARMA processes we can compare with the correct optimal alarm region, but for the non-Gaussian process no such comparison is possible. For the ARMA(2,1)-process we shall use two different approaches, one where we use the predicted value and the predicted rate of increase as input to the network, and one where we let the network find the proper input combination.

The ARMA(p, q)-processes in the first three cases are given by

$$\begin{aligned} &X(t) + a_1 X(t-1) + \ldots + a_p X(t-p) \\ &\quad = e(t) + c_1 e(t-1) + \ldots + c_q e(t-q), \end{aligned}$$

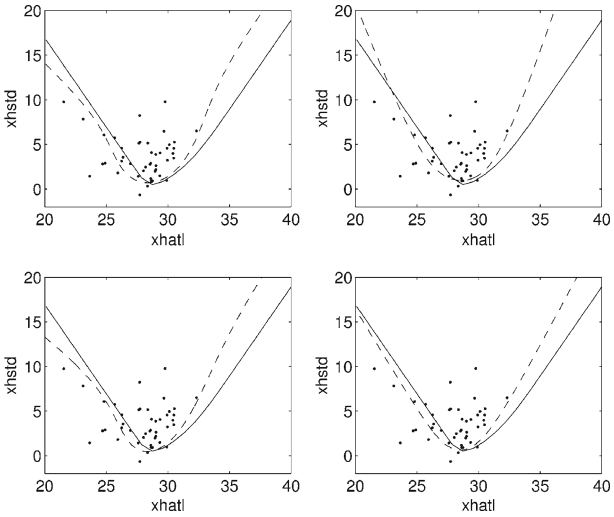

Fig. 5 Illustration of pruning and training for the ARMA(6,4)-process in Section 5.1. *Solid line* = optimal alarm system according to Eq. 5. *Broken line* = neural network solution. *Top left*: Network after 2000 epochs of training with $\gamma = 0.1$. *Bottom left*: Network after pruning and 3000 epochs of further training with $\gamma = 0.1$. *Top right*: Network (with the same starting values as the one to the left) after 2000 epochs of training with $\gamma = 0.25$. *Bottom right*: Network after pruning and 3500 epochs of further training with $\gamma = 0.25$

where $\{e(t)\}$ are independent standard normal variables uncorrelated with $X(t-1), X(t-2), \ldots$. Writing z^{-1} for the shift operator, we define the $A-$ and $C-$polynomials in z^{-1} as

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_p z^{-p} ,$$
$$C(z^{-1}) = 1 + c_1 z^{-1} + \ldots + c_q z^{-q} .$$

4.2 Choice of Input Variables

Our knowledge of the optimal alarm, given by Eq. 5 for Gaussian processes, suggests a hybrid network predictor with the k-step predictions $(\widehat{x}_{L,k}(t), \widehat{x}_{D,k}(t))$, defined by Eq. 6, as input instead of the original data $\mathbf{y}$. Then the network should classify these prediction points as belonging to the optimal region or not, as shown in the example in Fig. 2. In neural network language we should use the predicted value and the (conditionally) expected rate of increase as pre-processed variables.

When the process is Gaussian the k-step predictions $\widehat{x}_{L,k}(t)$ and $\widehat{x}_{D,k}(t)$ are linear functions of available past observations:

$$\widehat{x}_{L,k}(t) = \sum_{j=1}^{n} a_j^L x(t-k-n+j) ,$$
$$\widehat{x}_{D,k}(t) = \sum_{j=1}^{n} a_j^D x(t-k-n+j) . \qquad (10)$$

The coefficients $\{a_j^L\}$ and $\{a_j^D\}$ are calculated by using the orthogonality principle.

In the first two examples we used the Gaussian assumption and the prior knowledge that the optimal level crossing predictor is based in the predicted value and predicted rate of increase. In the third example we shall compare this with a crude approach where the network is fed with the raw sequence of observed values, letting the estimation algorithm find the best alarm predictor.

In connection with tuning of the learning algorithm we would like to stress the importance of scaling all inputs to the same order of magnitude as that of the output

of the nodes in the network. In our case this is necessary for two reasons: Firstly, we have only one step length parameter η, which has to be adjusted with respect to the direction with the fastest convergence. Secondly, weight decay requires that the inputs to all nodes—input nodes as well as hidden and output nodes—are on the same scale.

4.3 Choice of Model Structure—Weight Decay

Before we set out to find the parameters of the network model, there are some parameters pertaining to the optimization method which must be given appropriate values. The choice of these values may be critical to the performance of the method, so they have to be chosen with some care. There are a number of more elaborate methods to find "good" values of these parameters, but we will approach this matter rather pragmatically.

The weight decay term which has been discussed in Section 3.2 has two effects, smoothing (i.e. avoiding over-fitting) and pruning. To set the decay parameter γ in Eq. 9, a series of experiments was made and a weight decay with $\gamma = 0.25$ was found to work well in the given problems. The effect of the value of γ is illustrated in Fig. 5 (where its smoothing influence can be seen—a higher weight decay straightens out the flanks).

To get a first hint of a suitable architecture—i.e. the number of weights in the input and hidden layers respectively and the degree of connectivity—a set of starting matrices was tried, with matrices so large that they could be pruned after a few hundred epochs of training. The pruning was done by hand after inspection of the matrices. This is not such an ambiguous procedure as it may seem, since the performance in each of the cases was quite conclusive. A typical example is shown in Fig. 13, where the matrices are represented graphically.

4.4 Choice of Starting Values

The result of the back propagation algorithm is quite dependent on the starting values of the weights w_{ijk}, so before we venture on a more extensive simulation we will try to find a set of good starting matrices. To get a view of how the training of the network proceeds, a plot of the training error as a function of the number of training epochs is quite useful. To assess the generalization ability of the network, a quite common procedure is to plot the prediction error of the network on some test set. In our case one can get a more direct evaluation by considering the two OC-variables $P(A \mid C)$ and $P(C \mid A)$, estimated by Eq. 7, which are basic to the alarm problem.

However, the OC-plots estimated from small data sets can be a bit tricky to interpret, as the graphs can be rather jagged. Since we are interested in the general performance of the networks and work with simulated data, we will estimate the OC-plots with the help of large datasets. We estimate the OC-variables for 200 test sets, each of size 10000 points, and then compute our final estimates as the mean of the 200 estimates thus obtained. All OC-plots in our examples are computed in this way.

We can now generate random starting matrices for those weights that remain after the preliminary pruning, and train these networks on the same training set. After they have been trained the same number of epochs, we compare the OC-plots. If we can find a network which is uniformly best, the choice is clear, otherwise we will choose

the network that is best on the interval of primary interest. The network chosen in this way will serve as a starting value for further simulations.

To conclude this section, our procedure in the different examples can be summarized like this:

Step 1. Find an appropriate architecture.
Step 2. Train networks with different starting values and pick the best.
Step 3. Do the final training.

5 Four Examples

The results for the following four examples are illustrated by similar figures: one figure illustrates how the OC-function depend on the boundary probability and on the size of the training set (only for the ARMA-examples), two figures show the alarm regions in the $(\widehat{x}_L, \widehat{x}_D)$-plane, and its dependence on the boundary probability c, and the final figure gives an overall picture of the performance.

5.1 An ARMA(6,4)-Process

In the ARMA(6,4)-process used in the following, the $A-$ and $C-$polynomials in the defining relation $A(z^{-1})x_t = C(z^{-1})e_t$, with $z^{-1}x_t = x_{t-1}$, are:

$$\begin{aligned} A(z^{-1}) &= 1 - 3.5244z^{-1} + 5.1926z^{-2} - 4.0972z^{-3} \\ &\quad + 1.8281z^{-4} - 0.4380z^{-5} + 0.0441z^{-6}\,, \\ C(z^{-1}) &= 1 - 0.7831z^{-1} + 0.2333z^{-2} - 0.0313z^{-3} \\ &\quad + 0.0016z^{-4}\,. \end{aligned}$$

The catastrophe level is 30 and 10 consecutive old values are used in the predictions $\widehat{x}_{L,k}$ and $\widehat{x}_{D,k}$, i.e. $n = 10$ in Eq. 10. The prediction horizon is $k = 2$ and the coefficients of the mean-square optimal predictors are,

$$\begin{aligned} \{a_j^L\} = (&-0.0001, 0.0004, 0.0011, 0.0030, -0.0034, \\ &-0.0735, -0.2553, 2.2263, -4.6495, 3.7217)^{\mathrm{T}}\,, \end{aligned}$$

Table 2 Detection and reliability parameters for the ARMA(6,4)-process with alarm regions in Fig. 8

c	Alarms			c	Catastrophes		
	Correct	False	Total		Detected	Undetected	Total
0.2	16 (0.43)	21 (0.57)	37	0.2	16 (0.76)	5 (0.24)	21
0.4	10 (0.50)	10 (0.50)	20	0.4	10 (0.48)	11 (0.52)	21
0.6	8 (0.57)	6 (0.43)	14	0.6	8 (0.38)	13 (0.62)	21

Numbers in parentheses give relative frequencies within each class, c=boundary probability. The data set consists of 2000 points

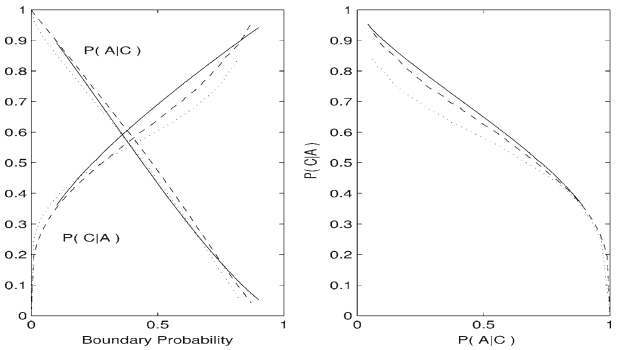

Fig. 6 *Left*: plot of OC-functions vs. boundary probability for ARMA(6,4). *Right*: plot of probability of correct alarm vs. probability of detected catastrophe. *Solid lines* optimal alarm system, *dotted lines* network trained on 5000 data points, *broken lines* network trained on 200000 data points

$$\{a_j^D\} = (-0.0001, 0.0004, 0.0011, 0.0031, -0.0024,$$
$$-0.0696, -0.2584, 2.0049, -3.6737, 1.9608)^{\mathsf{T}},$$

with index $j = 1$ corresponding to the oldest datapoint that is used.

For the training set, containing a total of N patterns, i.e. data points $y(t_1), \ldots, y(t_N)$ at discrete times $t_1,\ldots,t_N$, it is known when catastrophes occur. Then a binary target vector $\boldsymbol{\tau} = (\tau^{(1)}, \ldots, \tau^{(N)})$, with $\tau^{(p)}$ equal to one if there is a catastrophe at time p and zero otherwise, can be associated with the $2 \times N$ input matrix with columns containing the corresponding pairs of $\widehat{x}_{L,2}$ and $\widehat{x}_{D,2}$.

We follow the procedure described in Section 4, and the first step is to find an appropriate network architecture. To this end we begin with three different random starting matrices and train them for 1000 epochs on a training set with 2000 data points. All three networks tend to a 2-3-1-architecture (but with different structures).

The next step in the procedure is to train several networks with different starting values and pick the best. Thus, 15 random matrices with 2-3-1-architecture are

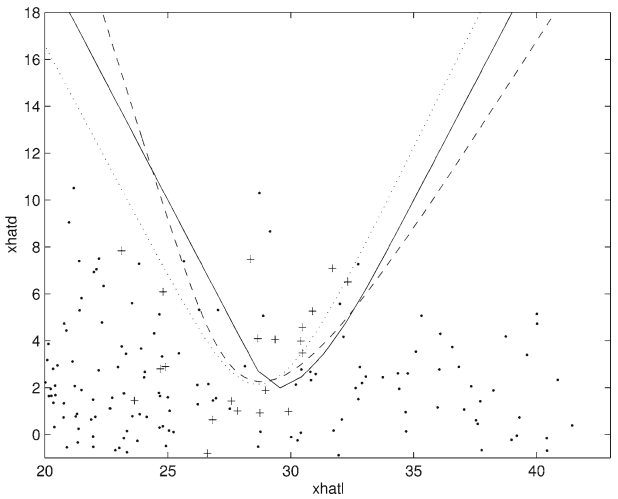

Fig. 7 Alarm regions of the networks in Fig. 6; *plus signs* = catastrophe points, *dots* = non-catastrophe points

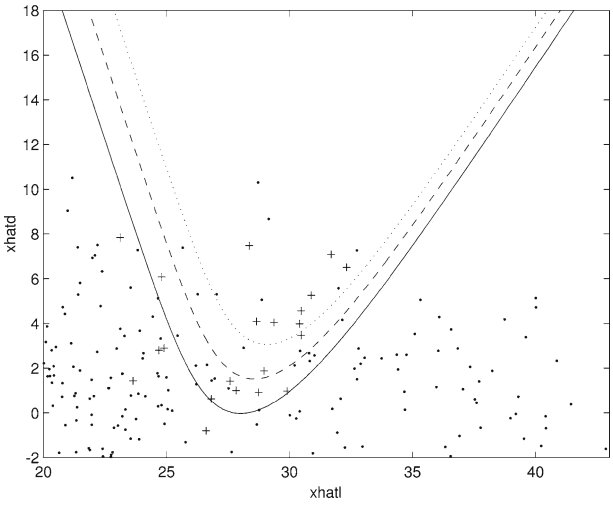

Fig. 8 The influence of the boundary probability c on the alarm region for the ARMA(6,4)-process, summarized in Table 2; *solid line* $c = 0.2$, *broken line* $c = 0.4$, *dotted line* $c = 0.6$; *plus signs* = catastrophe points, *dots* = non-catastrophe points

trained for 1800 epochs on a training set with 1000 data points. Of the resulting networks, six can be classified as almost equally good, while the rest are not as good.

The third and last step in the procedure is the final training. As starting value of the longer simulations we take the best of the fifteen networks above. This network is now trained for 2100 epochs on training sets of size 5000, 20000, and 200000 respectively.

The results for this example are summarized in Table 2 and illustrated in Figs. 6, 7, 8, and 9. Both plots in Fig. 6 show the OC-variables, but the plot to the right is much easier to read: a higher curve means a better model. As we can see, the optimal alarm system, derived theoretically, has the highest curve of all, and it represents the theoretical limit.

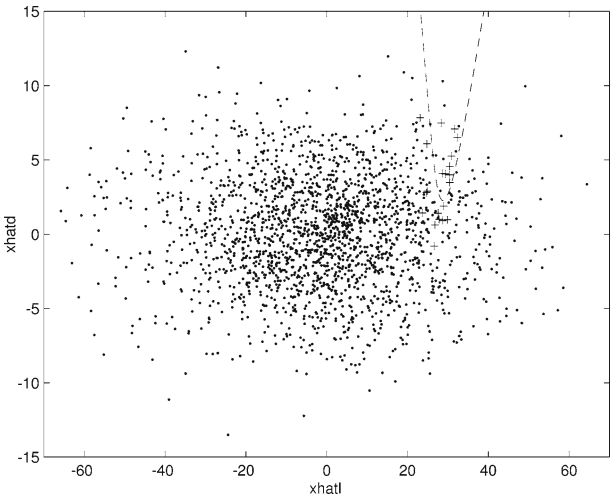

Fig. 9 Simulated predictions from an ARMA(6,4)-process with prediction horizon $k = 2$; *broken line* = boundary of network alarm region (the same network as in Fig. 8 and Table 2 but with boundary probability $c = 0.5$); *plus signs* = catastrophe points, *dots* = non-catastrophe points

Table 3 Detection and reliability parameters for the ARMA(2,1)-process with alarm regions depending on boundary probability c

c	Alarms			c	Catastrophes		
	Correct	False	Total		Detected	Undetected	Total
0.20	29 (0.50)	29 (0.50)	58	0.20	29 (0.88)	4 (0.12)	33
0.32	22 (0.56)	17 (0.44)	39	0.32	22 (0.67)	11 (0.33)	33
0.60	8 (0.89)	1 (0.11)	9	0.60	8 (0.24)	25 (0.76)	33

Numbers in parentheses give relative frequencies within each class. The data set consists of 2000 points

5.2 An ARMA(2,1)-Process

This process is defined by the A- and C-polynomials

$$A(z^{-1}) = 1 - 1.8z^{-1} + 0.9z^{-2} ,$$
$$C(z^{-1}) = 1 + 0.5z^{-1} .$$

The catastrophe level is 16 and 3 old values are used in the predictions $\hat{x}_L$ and $\hat{x}_D$. The prediction horizon is $k = 2$ and the coefficients of the mean-square optimal predictors are:

$$\{a_j^L\} = (0.5377, -2.3123, 2.6403)^\top ,$$

$$\{a_j^D\} = (0.3073, -1.3213, 0.8659)^\top .$$

We begin with three different random starting matrices and train them for 1000 epochs on a training set with 2000 data points. All three networks tend to a 2-3-1-architecture. Next, 15 random starting matrices with 2-3-1-architecture are trained for 1000 epochs on a training set with 1000 data points. Of the resulting networks, nine can be classified as almost equally good, while the rest are not as good. As starting value of the longer simulations we take the best of the 15 networks and train it on training sets of size 5000, 20000, and 200000 respectively. It is trained for 3000 epochs in each case. Table 3 and Figs. 10, 11, 12 summarize the performance.

5.3 A Direct Approach for the ARMA(2,1)-Process

The process generating the data is the same ARMA(2,1)-process as in the previous example, but we train the network on the "raw" data instead of on the predicted value and slope. If we use n old values of the process, the input vector has the form $\mathbf{y}(t) = (x_{t-k-n+1}, \ldots, x_{t-k})^\top$, with prediction horizon $k = 2$ as before. The target vector is constructed as in the previous cases and the input matrix data is scaled by division by $a = \max_i |x_i|$.

Three sets of starting matrices are trained for 500 epochs on a set of 1000 data points. All three networks tend to a 5-4-1-architecture (but with different structure). One of these networks can be seen in Fig. 13.

Next, five random sets of starting matrices with 5-4-1-architecture are trained for 1000 epochs on a training set with 1000 data points. The best network is picked out for further study, which implies further 1000 epochs of training on a data set of 200000 points.

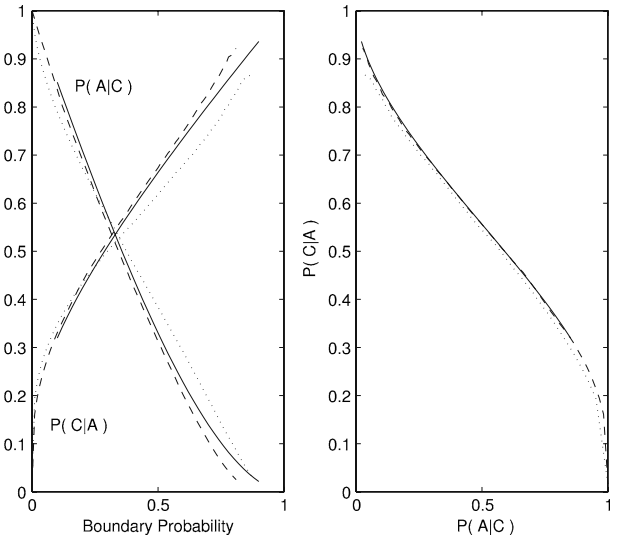

Fig. 10 *Left*: Plot of OC-variables vs. boundary probability for the ARMA(2,1)-process. *Right*: plot of probability of correct alarm vs. probability of detected catastrophe. *Solid lines* optimal alarm system, *dotted lines* network trained on 5000 data points, *broken lines* network trained on 200000 data points

Figure 14 indicates that the data reduction to the $(\hat{x}_L, \hat{x}_D)$-plane is quite powerful, and this is as it should be according to the theory in Lindgren (1985). It also seems to confirm that neural networks are sensitive to transformations of the input. It ought to be mentioned, though, that the comparison is a bit unfair, since the network trained on transformed data is picked as the best out of 15 candidates, while the network trained on "raw" data is the best out of 5. Anyway, it is clear that the direct neural network without too much effort performs much better than the naive catastrophe predictor of Eq. 1.

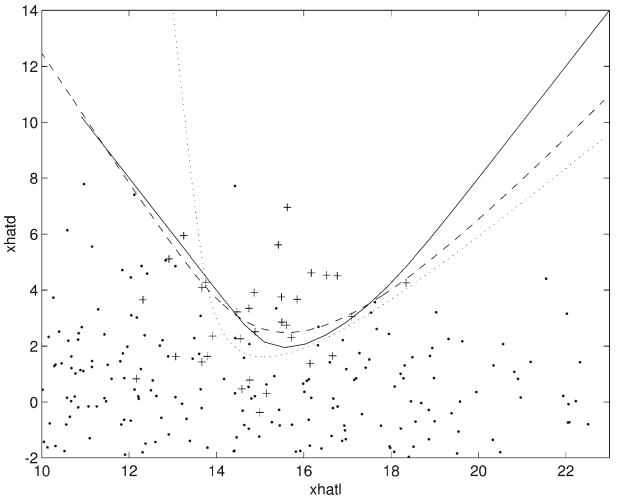

Fig. 11 Alarm regions of the networks in Fig. 10; *plus signs* = catastrophe points, *dots* = non-catastrophe points. In regions where the distribution of points is sparse the alarm regions differ more than one would expect from the OC-plots

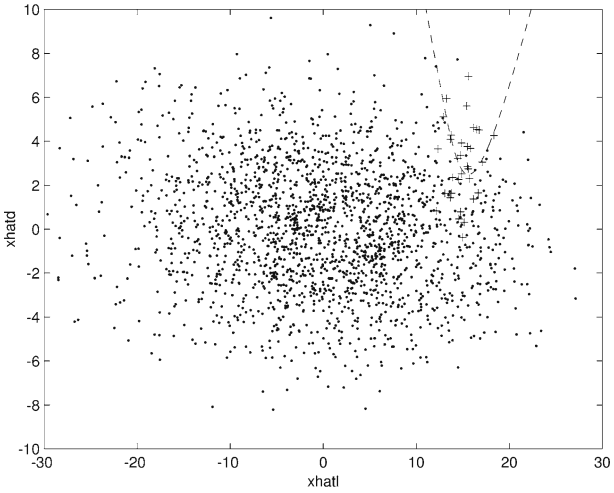

Fig. 12 Simulated predictions from an ARMA(2,1)-process with prediction horizon $k = 2$; *broken line* = boundary of network alarm region with boundary probability $c = 0.5$; *plus signs* = catastrophe points, *dots* = non-catastrophe points

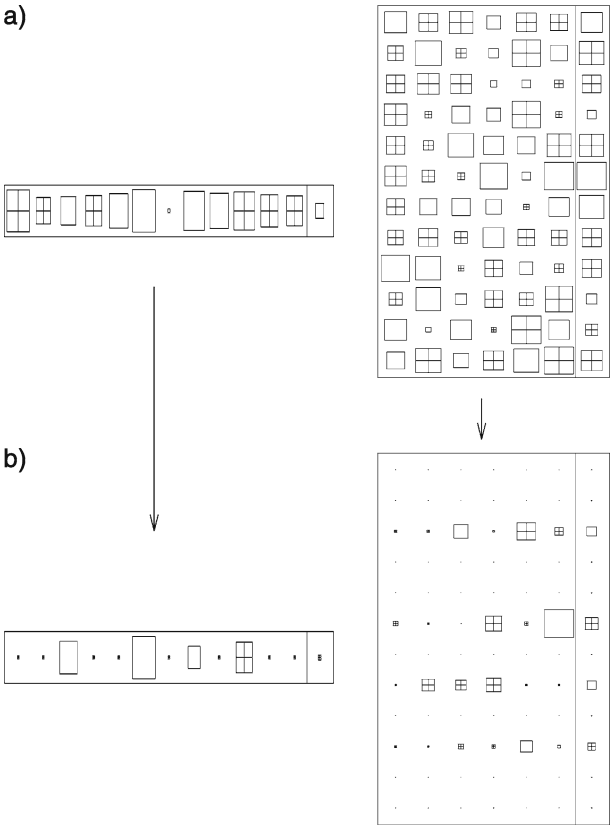

Fig. 13 *Top*: random starting matrices of network for the ARMA(2,1)-process (direct approach). *Bottom*: matrices of network after 1000 epochs of training with weight decay

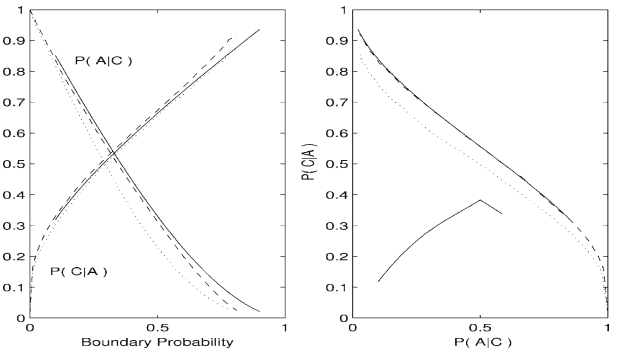

Fig. 14 *Left*: plot of OC-variables vs. boundary probability for the ARMA(2,1)-process. *Right*: plot of probability of correct alarm vs. probability of detected catastrophe, *solid line* (*bottom*) = naive alarm system (according to Eq. 1); *solid line* = optimal alarm system, *broken lines* = network trained on 200000 data points (with $\hat{x}_L$ and $\hat{x}_D$ as input), *dotted lines* = network trained on 200000 data points (direct approach)

5.4 A Duffing Process

In this case the process generating the data is given by a Duffing-type equation,

$$x_{m+1} = x_m + h\dot{x}_m$$
$$\dot{x}_{m+1} = \dot{x}_m + h(x_m - ax_m^3 - b\,\dot{x}_m) + \varepsilon_m\sqrt{2bh}\,,$$

$m = 0, 1, 2, \ldots$, where $\{\varepsilon_m\}$ are independent standard normal variables and the initial values are

$$x_0 = \varepsilon_0/\sqrt{a}, \quad \dot{x}_0 = \varepsilon_0\,.$$

In a typical realization of this process, see Fig. 15, the process is sampled with a period of $T_m = 50$. With this new time scale we get

$$x(t) = x_{50t}, \quad \dot{x}(t) = \dot{x}_{50t}, \quad t = 0, 1, 2, \ldots$$

To predict the process we use the following equations:

$$\hat{x}_{m+1} = \hat{x}_m + h\hat{\dot{x}}_m,$$
$$\hat{\dot{x}}_{m+1} = \hat{\dot{x}}_m + h(\hat{x}_m - a\hat{x}_m^3 - b\,\hat{\dot{x}}_m)\,.$$

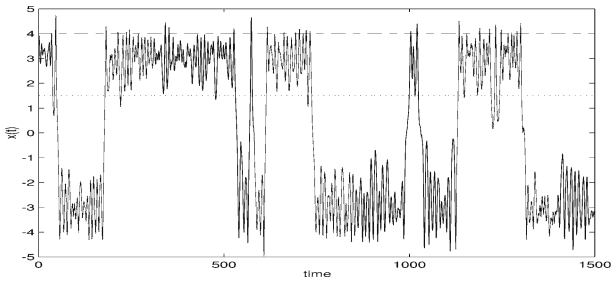

Fig. 15 A sample of the Duffing process; catastrophe levels: *broken line* $u = 4.0$, *dotted line* $u = 1.5$

Table 4 Detection and reliability parameters for the Duffing process (lover level) with alarm regions depending on boundary probability c

c	Alarms			c	Catastrophes		
	Correct	False	Total		Detected	Undetected	Total
0.20	48 (0.53)	43 (0.47)	91	0.20	48 (0.92)	4 (0.08)	52
0.44	36 (0.69)	16 (0.31)	52	0.44	36 (0.69)	16 (0.31)	52
0.60	28 (0.70)	12 (0.30)	40	0.60	28 (0.54)	24 (0.46)	52

Numbers in parentheses give relative frequencies within each class. The data set consists of 2000 points

To calculate $\hat{x}(t+i)$ and $\hat{\dot{x}}(t+i)$ we have to iterate these $50i$ times with starting values

$$\hat{x}_0 = x(t), \quad \hat{\dot{x}}_0 = \dot{x}(t) \ ,$$

assuming that we know $x(t)$ and $\dot{x}(t)$. The process may be considered as stationary if we discard the initial transient phase.

In our simulation of the process we have set the parameters to $h = 0.01$, $a = 0.1$ and $b = 0.2$. The prediction horizon in t-scale is $k_t = 2$ and the catastrophe levels are $u = 1.5$ and $u = 4.0$ respectively (before rescaling).

5.4.1 Catastrophe Level Below the Upper Equilibrium

The catastrophe level is set at 1.5, and we begin with three different random starting matrices and train them for 1000 epochs on a training set with 2000 data points. All three networks tend to a 2-3-1-architecture (but with different structure).

Next, 15 random matrices with 2-3-1-architecture are trained for 1000 epochs on a training set with 2000 data points. Of the resulting networks, 13 are almost equally good. As starting value of the longer simulations we take the best of the 15 networks and train it for 1500 epochs in each case on training sets of size 5000, 20000, and 200000, respectively.

In this case we have no theoretical results for an optimal alarm system to compare with. What we have are the operating characteristics (as in Table 4) and the plots in Figs. 16, 17, and 18. The behaviour of the network is the same as we have witnessed in the previous cases with the Gaussian processes.

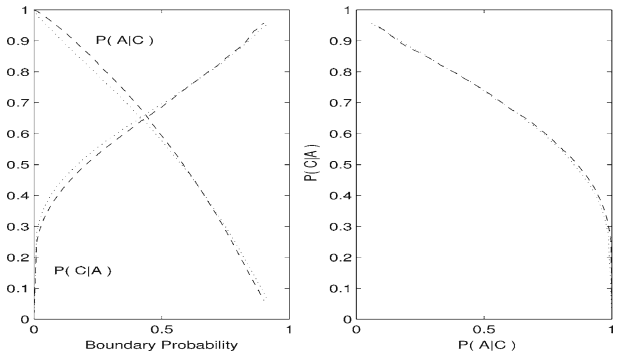

Fig. 16 *Left*: plot of OC-variables vs. boundary probability for the Duffing process (lower level). *Right*: plot of probability of correct alarm vs. probability of detected catastrophe. *Dotted line* = network trained on 5000 data points, *broken line* = network trained on 200000 data points

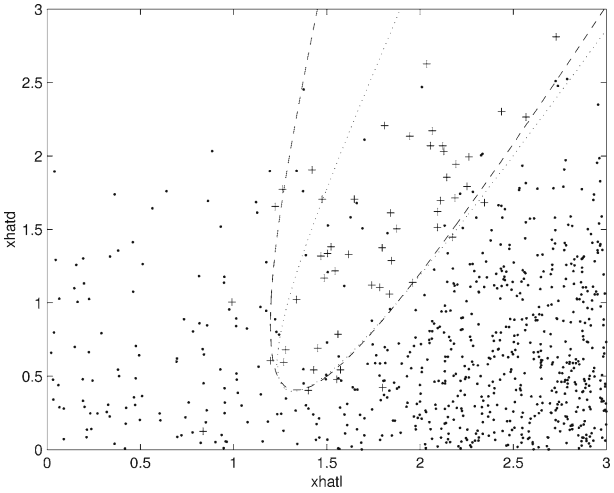

Fig. 17 Alarm regions of the networks in Fig. 16; *plus signs* = catastrophe points, *dots* = non-catastrophe points

5.4.2 Catastrophe Level Above the Upper Equilibrium

The catastrophe level is now set to 4.0, which is above the upper equilibrium of the process (see Fig. 15). The architecture and starting values were chosen in the same way as in the previous example. After steps one and two the network is trained for 1500 epochs on training sets of size 5000, 20000, and 200000 data points, respectively.

In Figs. 19, 20, and 21 the picture is not quite as clear as in the earlier cases. The crossing of the $P(A \mid C)$- and $P(C \mid A)$-curves for the network trained on the larger data set is at a lower boundary probability than the crossing of the curves for the network trained on the smaller set. From the previous examples one would expect the opposite. Possible explanations for this could be that the large dataset demands more training or perhaps has got stuck in a local minimum.

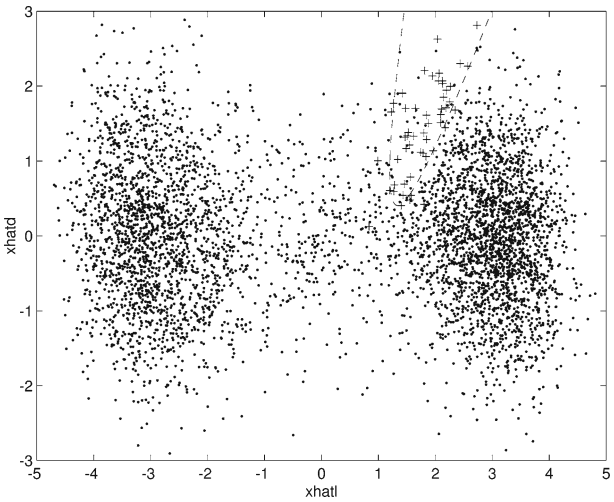

Fig. 18 Simulated predictions for a Duffing-type process with low catastrophe level and prediction horizon $k = 2$; *broken line* = boundary of network alarm region with boundary probability $c = 0.3$, *plus signs* = catastrophe points, *dots* = non-catastrophe points

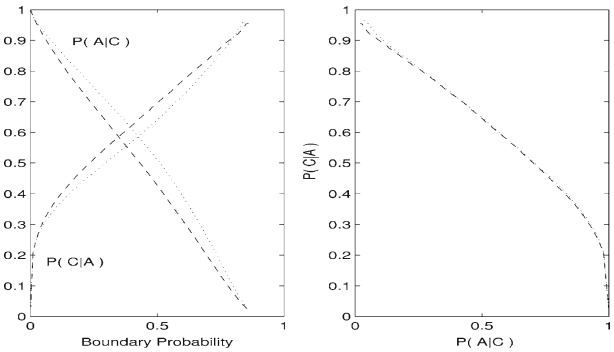

Fig. 19 *Left*: plot of OC-variables vs. boundary probability. *Right*: plot of probability of correct alarm vs. probability of detected catastrophe. *Dotted lines* = network trained on 5000 data points, *broken lines* = network trained on 200000 data points

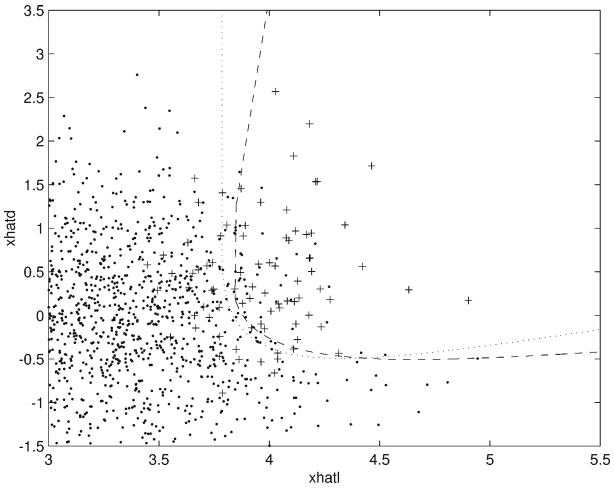

Fig. 20 Alarm regions for the networks in Fig. 19

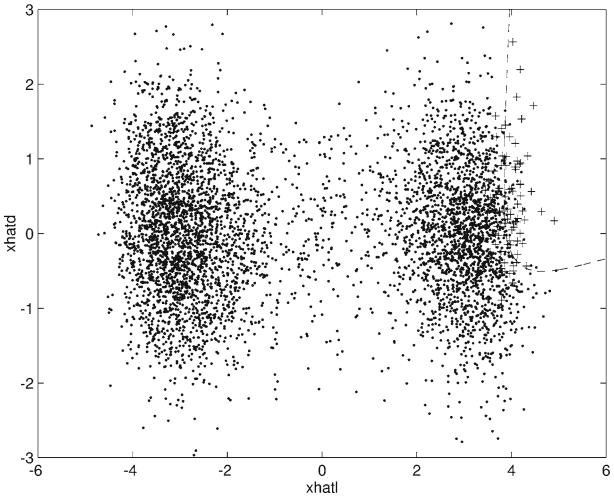

Fig. 21 Simulated predictions from a Duffing-type process with high catastrophe level and prediction horizon $k = 2$; *broken lines* = boundary of network alarm region with boundary probability $c = 0.3$, *plus signs* = catastrophe points, *dots* = non-catastrophe points

Table 5 Detection and reliability parameters for the Duffing process (upper level) with alarm regions depending on boundary probability c

c	Alarms			c	Catastrophes		
	Correct	False	Total		Detected	Undetected	Total
0.20	55 (0.46)	65 (0.54)	120	0.20	55 (0.67)	27 (0.33)	82
0.35	43 (0.63)	25 (0.37)	68	0.35	43 (0.53)	39 (0.47)	82
0.60	24 (0.83)	5 (0.17)	29	0.60	24 (0.29)	58 (0.71)	82

Numbers in parentheses give relative frequencies within each class, The data set consists of 2000 points

The network solutions to the two examples from the non-Gaussian Duffing-type process show that the data reduction to the $(\hat{x}_L, \hat{x}_D)$-plane still is quite powerful. We do not know what the optimal alarm system looks like in this case, but the data in Tables 4 and 5 are comparable to those for the ARMA-processes.

6 Conclusion

In this article we have shown that a feed-forward neural network can approximate an optimal alarm system. We have applied the network to Gaussian as well as non-Gaussian stochastic processes. In the Gaussian case we could compare the network model with the theoretically derived optimal alarm system. To get a picture of the dependence of the network approximation on the size of the training set, the networks were trained on training sets with increasing size. All simulation experiments on neural networks were performed in three steps:

1. choice of architecture,
2. choice of starting values,
3. final simulations.

The technique of weight elimination was used to prune the networks in step number one.

The back-propagation algorithm, although it may not be the most effective algorithm, was chosen mainly because it was already implemented in Matlab. Because the back-propagation algorithm is quite dependent on the starting values of the weights, we have to treat the choice of starting values very carefully. Here consideration of OC-functions was more helpful than measurements in terms of training or validation error.

Step number three delivered some information on generalization and consistency of the approximated optimal alarm system. We did, however, not succeed in finding a satisfactory stopping criterion for the training algorithm.

Two network models with different input representation were compared with the naive as well as with the optimal catastrophe predictor. In all cases the network models were much better than the naive predictor, even if they were not quite as good as the optimal predictor.

Central to the simulation studies is the interpretation of the network output as an estimate of the conditional probability of a catastrophe given the information. This interpretation, which is given in the theoretical part of the article, holds under the assumption of stationarity of the process.

Acknowledgement Part of this work was carried out in the framework of the EU project SEAMOCS (contract MRTN−−CT−−2005−−019374).

References

Beckman S-I, Holst J, Lindgren G (1990) Alarm characteristics for a flood warning system with deterministic components. J Time Ser Anal 11:1–18

Bishop MC (1995) Neural networks for pattern recognition. Oxford University Press, Oxford

Cawley G, Janacek G, Haylock M, Dorling S (2007) Predictive uncertainty in environmental modelling. Neural Netw 20:537–549

de Maré J (1980) Optimal prediction of catastrophes with application to Gaussian processes. Ann Probab 8(4):841–850

Demuth H, Beale, M (1992) Neural network toolbox user's guide. The MathWorks, Natick

Dutot A-L, Rynkiewicz J, Steiner F, Rude J (2007) A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions. Environ Model Softw 22:1261–1269

Lindgren G (1975) Prediction of catastrophes and high level crossings. Bull Intern Statist Inst 46(Book 2):225–240

Lindgren G (1985) Optimal prediction of level crossings in Gaussian processes and sequences. Ann Probab 13(3):804–824

Nunnari G (2006) An improved back propagation algorithm topredict episodes of poor air quality. Soft Comput 10(2):132–139

Ripley BD (1993) Statistical aspects of neural networks. In: Barndorff-Nielsen OE, Jensen JL, Kendall WS (eds) Networks and chaos—statistical and probabilistic aspects. Chapman and Hall, London, pp 40–123

Ripley BD (1994) Neural networks and related methods for classification. J R Stat Soc Ser B 56(3):409–456

Svensson A, Holst J, Lindqvist R, Lindgren G (1996) Optimal prediction of catastrophes in autoregressive moving average processes. J Time Ser Anal 17:511–531

Weigend AS, Huberman BA, Rumelhart DE (1990) Predicting the future: a connectionist approach. Int J Neural Sys 1(3):193–209