A Neural Network Approach to Quality Control Charts

Thomas STÜTZLE
Department of Statistics and O.R.
University Complutense, Madrid

Abstract

In this paper Quality Control Charts without memory and neural networks are compared. Neural networks are trained to decide whether a process is under or out of statistical control. As only the last sample is used to decide about the state of the production process, a comparison to Shewhart-control charts leads automatically to a comparison between statistical tests and neural networks. Here neural networks are compared to control charts for the process mean and the process variability. Next, by a comparison to a combined control chart to control the process mean and the process variability, the kind of classifications of the type of change is considered more exactly.

1 Introduction

Recently there have been some attempts to use neural computing for special tasks in the Statistical Process Control (SPC).[1] Pugh (1991) tried to imitate a Shewhart Control Chart to control the mean of the process. In Smith (1994) a neural network was trained to replace a combined $\bar{x}$-R chart. Neither of these articles considers the different types of errors, so no statements of the possible usefullness of neural networks in this area were made.

Another kind of application of neural networks was investigated in Guo, Dooley (1992). When a shift of the parameters by a combined CUSUM-Mosum scheme[2] was detected, they compared neural networks and a quadratic discriminant function to determine whether the out-of-control signal was actually due to a shift of the process mean or of the process variability. They compared the results obtained with neural networks or the quadratic discriminant function to some obvious heuristics that consider the control chart that first signals a change in the parameters. In this kind of application the control charts are interpreted to aid fault diagnosis by giving an indication for possible causes.[3]

Another possibility is the interpretation of control charts to detect unnatural patterns on the control charts. Some of these unnatural patterns that also indicate an out of control situation are described in Montgomory (1985). The detection of unnatural patterns was investigated by Hwarng, Hubele (1993) and Pham, Oztemel (1994). In Hwarng, Hubele the Backpropagation-algorithm was used, whilst Pham and Oztemel make use of some derivate of the Learning Vector Quantizer (LVQ).

The last two approaches use also past samples to decide about the actual state of the production process. Here the neural networks were used as a possible classification tool for a special problem of pattern recognition. This may be a more convincing application of a nonparametric method like neural networks. Nevertheless in this article neural networks are compared to Shewhart Control Charts to investigate if neural networks can be used as a substitute for this traditional approach to quality control.

In section 2 a short introduction to quality control charts is given and the $\bar{x}$-chart and the s-chart are introduced. Then an interpretation of the statistical tests as a special kind of classification is given in section 3. Section 4 consists of an introduction to neural networks and the backpropagation algorithm for the training of feedforward neural networks is presented. Finally some properties of feed-forward neural nets are stated. In Section 5 the results of the empirical tests are presented. Here first some practical considerations for the experiments are made. Then the results for the comparison of neural networks to a $\bar{x}$-chart, to a s-chart, to a combined $\bar{x}$-s-chart and an a posteriori classification of the kind of shift are given.

[1] See the papers of Pugh (1991), Guo and Dooley (1992), Hwarng and Hubele (1993), Smith (1994) and Pham and Oztemel (1994).

[2] For an introduction to CUSUM-charts see Ryan (1989). The MOSUM-chart is described in Bauer, Hackl (1980).

[3] Another approach with a time series modelling of the process for the evaluation of out-of-control signals is described in Dooley, Kapoor (1990)

2 Control Charts

2.1 Introduction to Control Charts

A process is said to be out of *statistical control* when controlable parameters like the process mean or the process variability have changed. Control Charts are used as an aid to check whether a production process is under or out of (statistical) control. For this purpose samples of the process are drawn at constant intervals of time and some quality characteristic(s) is(are) measured. The quality characteristic that is measured is used to give an indication for the actual state of the process. In this article only one quality characteristic is considered.[4] Here it is supposed that the process can be described in terms of its mean μ_t and of its standard deviation σ_t at instant t. If the process is under control these parameters are assumed to be constant $\mu_t = \mu_0$ and $\sigma_t = \sigma_0$. A change of the process parameters causes a change of the process mean and/or a change of the process standard deviation. To reflect the quantity of the process change, the two variables δ_t and ϵ_t can be defined as:

$$\delta_t = \frac{\mu_t - \mu_0}{\mu_0} \qquad \epsilon_t = \frac{\sigma_t}{\sigma_0} \tag{2.1}$$

In order to run a control chart at constant intervals of time a sample of the process is taken and the realization of a quality characteristic is measured. Let the vector $\mathbf{x_t} = (x_{t1}, x_{t2}, .., x_{tn})'$ denote the vector of these measurements at instant t. This vector can be interpreted as a realization of the random vector $\mathbf{X_t} = (X_{t1}, X_{t2}, .., X_{tn})'$ where $X_{t1}, X_{t2}, .., X_{tn}$ are supposed to be independent random variables. A sample function $g(X_{t1}, X_{t2}, .., X_{tn})$ like the sample mean is evaluated to examine whether the process parameters have changed or not.

The objective of the quality control chart is to detect changes of the process parameters as fast as possible but without making to many false alarms.

The running of a control chart can be considered as a periodically repeated statistical test. The null hypothesis H_0 and the alternative hypothesis can be formulated as:

$\mathbf{H_0}$: The process is under statistical control
$\mathbf{H_1}$: The process is out of statistical control

If $g(x_{t1}, x_{t2}, .., x_{tn})$ falls into the rejection region of the corresponding statistical test, it is supposed that a process parameter has changed and some action is taken. In the opposite case nothing is done. As with every test errors of type 1 and type 2 can be commited. The errors of type 1 are called *false alarms*. The possible errors can be formulated as:

Error of type 1: Some action is taken although the process is under control.
Error of type 2: No action is taken although the process is out of control.

The null hypothesis is chosen this way because any action that is not justified, that means false alarms, should be avoided as an action causes an unjustified stop of the production process and an unnecessary search for possible errors.

To compare different control charts, often the power function $G(\delta_t, \epsilon_t)$ of the statistical test is calculated. Another posibility is to calculate the *run length*. The run length is the number of samples until the chart indicates an out-of-control situation. The run length is a random variable and often their expectation, the *average run length (ARL)* is used to compare different kinds of control charts like Shewhart and CUSUM charts.

In the following it is assumed that the random variables $X_{t1}, X_{t2}, .., X_{tn}$ are independently, identically $N(\mu_t, \sigma_t^2)$-distributed. The Shewhart control charts to control the mean and the variability of the process are introduced. These control charts do not consider past samples and only the most recent sample is used to test whether some parameter has changed.

2.2 The $\bar{x}$-chart

The $\bar{x}$-chart is used to control the process mean. The sample function used for this control chart is the sample mean:

$$\bar{X}_t = \frac{1}{n} \cdot \sum_{i=1}^{n} X_{ti} \tag{2.2}$$

[4]When a number of quality characteristics are to be controlled, one possibility is to use multivariate control charts. For an introduction to multivariate control charts the interested reader is referred e.g. to Chapter 9 in Ryan(1989).

Here the sample variables $X_{t1}, X_{t2}, .., X_{tn}$ are assumed to be independently $N(\mu_t, \sigma_0^2)$-distributed random variables. From this follows that $\bar{X}_t$ is distributed as $N(\mu_t, \sigma_0^2/n)$. Observe that here it is supposed that the process variability is constant σ_0^2.

The corresponding statistical test can be formulated as:

$$H_0: \ \mu_t = \mu_0 \qquad H_1: \ \mu_t \neq \mu_0$$

From this the upper and the lower control limits (UCL and LCL) are easily obtained. For a given niveau α the limits are:

$$LCL = \ \mu_0 - \tau_{1-\alpha/2} \cdot \frac{\sigma_0}{\sqrt{n}} \qquad UCL = \ \mu_0 + \tau_{1-\alpha/2} \cdot \frac{\sigma_0}{\sqrt{n}} \tag{2.3}$$

Here $\tau_{1-\alpha/2}$ denotes the $(1-\alpha/2)$-quantil of the standard normal distribution. If the sample mean lies in the rejection region of the test, it is supposed that the process mean has changed and the process is stopped to search for the error. In Europe often a level of $\alpha = 0.01$ is chosen, whereas in America traditionally 3-*sigma* limits are chosen that are equivalent to a level of $\alpha = 0.0027$.

Using the dimensionless variable δ_t defined in (2.1), the power function evaluates to:

$$G_{\bar{X}}(\delta_t) = 1 - \Phi(\tau_{1-\alpha/2} - \delta_t \cdot \sqrt{n}) + \Phi(-\tau_{1-\alpha/2} - \delta_t \cdot \sqrt{n}) \tag{2.4}$$

The average run length is the expectation of a geometric random variable:

$$ARL = \frac{1}{G_{\bar{X}}(\delta_t)} \tag{2.5}$$

Although the $\bar{x}$-charts generally detect large shifts of the process mean quite fast, they have the disadvantage that small shifts can take a long time to be detected. This means that the average run length for small shifts is relatively high. To make the Shewhart-Charts more sensible to small shifts, additional *run rules* can be considered. They have the effect to detect small shifts more rapidely but also give a higher probability of errors of type 1. Another possibility is to use control charts like the CUSUM-chart, that use past samples and are able to detect small shifts of the mean quite fast but usually take longer time to detect larger shifts.

2.3 The s-chart

The standard deviation control chart (s-chart) is used to control the process variability. The sample function $g(X_{t1}, X_{t2}, .., X_{tn})$ is the sample standard deviation S:

$$S = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_{ti} - \bar{X}_t)^2}$$

If one only wants to detect an increase of the process variability the test can then be formulated as[5]:

$$H_0: \ \sigma_t \leq \sigma_0 \qquad H_1: \ \sigma_t > \sigma_0$$

For the assumptions made, $\frac{(n-1)\cdot S^2}{\sigma_0^2}$ is $\chi^2_{n-1}-$ distributed. For a given level α the lower and the upper control limit are obtained as:

$$LCL = 0 \qquad UCL = \sqrt{\frac{\chi^2_{n-1;\alpha}}{n-1}} \cdot \sigma_0 \tag{2.6}$$

The power function for the standard deviation chart, using the variable ϵ_t defined in (2.1), is:

$$G_S(\epsilon_t) = 1 - Chi_{n-1}(\frac{\chi^2_{n-1;\alpha}}{\epsilon_t^2}) \tag{2.7}$$

Here $Chi_{n-1}(\cdot)$ denotes the distribution function of a χ^2_{n-1}-distributed random variable. The average run length can be obtained like in (2.5) substituting $G_S(\epsilon_t)$ for $G_{\bar{X}}(\delta_t)$.

[5] This is the usual formulation of the test. If one also wants to detect a decrease of the process variability the hypothesis could easily be modified for this purpose.

2.4 The combined $\bar{x}$-s-chart

To control the mean and the variability of a production process at the same time, one possibility is to run two Shewhart control charts for the mean and the standard deviation. Now the possibility of a change of the process variability has to be considered to derive the power function for the $\bar{x}$-chart. The parameter σ_t now cannot be chosen constant $\sigma_t = \sigma_0$. The power function for the $\bar{x}$-chart is now a function of μ_t and σ_t. On the contrary, the power function for the standard deviation chart need not consider a change in the process mean as the distribution of the sample standard deviation is independent of the sample mean $\bar{X}$.

To derive the power function of the combined $\bar{x}$-s-chart, one can use the independence of S and $\bar{X}$. No action is taken if neither the $\bar{x}$-chart nor the s-chart indicate an out of control situation. The probability that no out-of-control signal is given is:

$$1 - G_{\bar{X}S}(\delta_t, \epsilon_t) = (1 - G_{\bar{X}}(\delta_t, \epsilon_t)) \cdot (1 - G_S(\epsilon_t)) \tag{2.8}$$

For the power function of the standard deviation chart the expression (2.7) can be used whereas the power function of the mean chart has to be derived considering the possible changes of the process variability. The power function of the mean chart can be shown to be:

$$G_{\bar{X}}(\delta_t, \epsilon_t) = 1 - \Phi(\frac{\tau_{1-\alpha/2}}{\epsilon_t} - \frac{\delta_t}{\epsilon_t} \cdot \sqrt{n}) + \Phi(-\frac{\tau_{1-\alpha/2}}{\epsilon_t} - \frac{\delta_t}{\epsilon_t} \cdot \sqrt{n}) \tag{2.9}$$

Then the power function for the combined $\bar{x}$-s-chart is:

$$G_{\bar{X}S}(\delta_t, \epsilon_t) = 1 - [\Phi(\frac{\tau_{1-\alpha/2}}{\epsilon_t} - \frac{\delta_t}{\epsilon_t} \cdot \sqrt{n}) + \Phi(-\frac{\tau_{1-\alpha/2}}{\epsilon_t} - \frac{\delta_t}{\epsilon_t} \cdot \sqrt{n})] \cdot [Chi_{n-1}(\frac{\chi^2_{n-1;\alpha}}{\epsilon_t^2})] \tag{2.10}$$

When the process is under statistical control, the probability for a false alarm is $1 - (1 - \alpha)^2$ for the combined control chart using a niveau α for each test. The combined scheme can easily be adjusted to give a probability of α of false alarms, choosing a niveau $\alpha' = 1 - \sqrt{1 - \alpha}$ for each chart.

The interesting point for a combined control chart is, that a sole increase of the process variability has an effect on the mean chart but not vice versa. If e.g. the process standard deviation is doubled, that means $\epsilon_t = 2$, then $G_{\bar{X}}(\delta_t = 0, \epsilon_t = 2) \approx 0.197$, independently of the sample size n. Furthermore it is interesting to calculate the probability that a sole shift of the standard deviation is indicated first on the $\bar{x}$-chart. Let A be the event that the sample mean lies inside the control limits of the $\bar{x}$-chart and $\bar{A}$ the event that the sample mean lies outside the control limits. Let the event B and $\bar{B}$ be defined in the same way for the s-chart. The i-th sample indicates an out-of-control signal only on the mean chart, when $i - 1$ times before there has been no sample mean outside the limits of the $\bar{x}$-chart and i times the sample standard deviation has not been outside the control limits of the s-chart. Then the probability that an out-of-control signal is first indicated on the $\bar{x}$-chart, can be calculated:

$$P(Shift\ first\ indicated\ on\ \bar{x} - chart) = \sum_{i=1}^{\infty} P(\bar{A}) \cdot P(A)^{i-1} \cdot P(B)^i = \frac{P(\bar{A})}{P(A)} \cdot \left(\frac{1}{1 - P(A) \cdot P(B)} - 1\right)$$

In a similar manner the probabilities, that a shift is first shown on the s-chart or is shown on both charts at the same time can be obtained. These probabilities are:

$$P(Shift\ first\ indicated\ on\ s - chart) = \frac{P(\bar{B})}{P(B)} \cdot \left(\frac{1}{1 - P(A) \cdot P(B)} - 1\right)$$

$$P(Shift\ indicated\ on\ both\ charts\ at\ the\ same\ time) = \frac{P(\bar{A}) \cdot P(\bar{B})}{P(A) \cdot P(B)} \cdot \left(\frac{1}{1 - P(A) \cdot P(B)} - 1\right)$$

As an example suppose a sample size $n = 6$, a level of $\alpha = 0.01$ and a shift only of the standard deviation of $\epsilon_t = 1.59$. This results in $G_{\bar{X}}(\delta_t = 0; \epsilon_t = 1.59) = P(\bar{A}) \approx 0.105$ and $G_S(\epsilon_t = 1.59) = P(\bar{B}) \approx 0.25$. So the probability, that the mean chart signals first an out-of-control situation, is 0.2395, that an out of control situation is only signalled on the s-chart is 0.6806 and the probability, that both charts signal an out of control situation at the same time, is 0.0799. This means, that in this example a shift only of the standard deviation may be indicated first on the mean chart with a quite high probability. Therefore, the probability of a false classification of the kind of change may be relatively high, if only the last sample is considered. False classifications are to be avoided because the kind of shift may give an indication of the reason of the change and may facilitate the search for possible faults. So by a false classification the search for the faults may take longer. In practice, the influence of a change of the process variability is considered by the rule, that before looking at the mean-chart one should consider the standard deviation chart. If the control charts should be evaluated by a computer, adequate algorithms have to be designed. Rules like *'if more than five consecutive samples lie above the middle-line of the standard deviation chart, a shift in variability is considered, although the mean chart signalled first an out of control situation* may be considered for this aim. Another possibility is to use past samples as an input to some classification procedures and to use them to distinguish the different kinds of change.[6]

3 Classification Properties of Control Charts

In this paper neural networks, that are often used for classification tasks, are compared to control charts. To give some kind of framework to this comparison, the control charts are interpreted here as some kind of classification procedure. This interpretation will be explained first at the example of the $\bar{x}$-chart.

The aim of classification is, to assign a feature vector $\mathbf{x} = (x_1, x_2, .., x_n)'$ to one of c possible classes $\theta_1, \theta_2, .., \theta_c$. One possible way of interpreting the different classes for a $\bar{x}$-chart is to assume the three classes θ_1 : *under statistical control*, θ_2 : *increase of the process mean* and θ_3 : *decrease of the process mean.* This interpretation may be justified since there are an upper and a lower control limit, that is two class boundaries, to distinguish three different classes. The function, used to distinguish between the different classes, is the sample mean. If the realization of the sample mean falls below the lower control limit, it is supposed that the sample belongs to class θ_3 or when the sample mean passes over the upper control limit, it is supposed that it belongs to class θ_2. Here the different importance of the errors of type 1 and type 2 has to be considered.

With a statistical test, the null hypothesis and the alternative hypothesis may be interpreted as the different classes. A statistical test divides the sample space into different regions. According to the region into which a sample falls, a decision is made, i.e a specific class is assigned. This leads to the interpretation of any statistical test as some kind of classification.[7]

With this interpretation a statisticalk test distinguishes only two classes. In terms of control charts the two classes can be interpreted as θ_1' : *The process is under statistical control* and θ_2' : *The process is out of statistical control* which would be the same as the formulation of the null hypothesis and the alternative in section 2.1. The advantage of a formulation with three classes is that one can explicitely distinguish in this interpretation, whether the process mean has decreased or has increased without further interpretation of the sample realization.

If one uses a combined $\bar{x}$-s-chart, there may be other possible classes that are to be distinguished. The different classes may be interpreted as:

θ_0: The process is under statistical control
θ_1: A change of the process mean occurred
θ_2: A change of the process variability occurred

A distinction between the different types of change may be useful because it can indicate different types of error. So an increase of the process variability may be an indication of tool wear, whereas a change of the process mean may indicate a change of the machine adjustments. As a further class θ_3 : *process mean and process variability changed* may be considered, since it may also occur that both parameters change.

[6] See Dooley, Kapoor (1990) or Guo, Dooley (1992).
[7] Or classification may be interpreted as some special kind of statistical test.

4 Neural Networks

In this chapter some basics of Neural Networks are introduced. As only the Backpropagation algorithm for the training of neural nets is described more extensively, the terms introduced in this chapter do not describe neural networks in the most general possible way. Then the Backpropagation algorithm is presented. At last some properties of neural network models are introduced.

4.1 Introduction to Artificial Neural Networks

An *Artificial Neural Network* [8] consists of a set of computational units (also called cells or neurons) and a set of weighted, directed connections between these units. At certain times a unit examines their input signals and computes a number as its output that is send to other units to which it has an outgoing connection. The weights of the connections determine the strenght of the influence, that one unit has on other units. In the following, the architecture, the cell properties, the dynamic properties and the learning properties of neural networks are described with more detail.

4.1.1 Architecture of neural networks

One possibility to describe the architecture of neural networks in a more formal way, is to interpret it as a weighted, directed graph $G = (V, K, W)$. Here V is the finite set of nodes, where node i will be denoted v_i, K is the set of directed edges (arcs) $< v_j, v_i >$ and W is the set of weights w_{ij}. Every arc $< v_j, v_i >$ has assigned a single weight w_{ij}. Every node of the graph corresponds to a cell, the directed edges correspond to the connections and the weights w_{ij} correspond to the strength of the connection from node v_j to node v_i.

The *input units* are only used to distribute the network input to the other cells of the neural network. At the *output units* the output of the network can be obtained. The input and the output units form the part of the neural network that is accessible for the outside world. Units that are neither input units nor output units are called *hidden units.*

Networks whose corresponding graph has a cycle are called *recurrent networks*, networks that do not have a cycle are called *feedforward nets.* In a feedforward net the signal flow only goes into one direction and the nodes can be numbered in a topological order. Feedforward nets can usually be divided into different layers. A *k-layer net* is a net that can be divided into $k + 1$ disjunct subsets $V_0, V_1, V_2, .., V_k$, so that a neuron v_i of a layer V_b has only incoming connections from neurons v_j of a layer V_a, if $a < b$. The layer V_0 consists of all input units and is called *input layer.* In a feedforward neural net the layer V_k comprises normally the output neurons and will be called *output layer.*[9] The other layers are called *hidden layers.* An example for a feedforward net, that can be divided into different layers, is the Multilayer-Perzeptron[10] This type of neural networks will be discussed further in section 4.2 and 4.3.

4.1.2 Cell properties

Every neuron v_i calculates at certain times its output u_i as a function of the inputs it receives. If neuron v_i is not an input neuron, its output is calculated as a function of the outputs of other neurons that have a connection to neuron v_i. First the *propagation function* is calculated. Often the propagation function for neuron v_i is the weighted sum S_i of the neuron inputs and is given by:

$$S_i = \sum_j w_{ij} \cdot u_j + w_{i0} \tag{4.1}$$

S_i is also often called *potential.* The weight w_{i0} is called *bias.* This weight is modelled usually by adding an additional cell v_0 to the neural network, that has a connection to all other cells but the input cells and always has the output $u_0 = 1$.

In the next step the activation of each cell is evaluated by applying the *activation function* $f(S_i)$. This function is a usually the same for all neurons of the network. Finally the output u_i of a cell is calculated as a function of the activation. In the neural networks described in this paper the output of a cell will be always the activation, so here the values u_i and $f(S_i)$ are used synonymously. There are many different

[8] The term artificial will be omitted from now on.

[9] Actually not all output neurons have to be in the last layer, but the neural nets considered here, can be arranged that way, that all the output neurons are in the k–th layer.

[10] Often for these kind of neural networks also the name Backpropagation-net is used.

possible activation functions of which some are presented here. One such activation function is a binary activation function with only two activation values. This activation function is given by:

$$f(S_i) = u_i = \begin{cases} 1 & if\ S_i > 0 \\ 0 & otherwise \end{cases} \tag{4.2}$$

Units that use this kind of activation function are also called *linear threshold units.* This activation function is used by the historically interesting *Perzeptron*[11], that essentially implements a linear discriminant function.

A type of functions that is often used as activation function in the context of the Backpropagation algorithm are the *Sigmoid functions.* One possibility is:

$$f(S_i) = u_i = \frac{1}{1 + e^{-k \cdot S_i}} \tag{4.3}$$

Another activation function, that has a similar shape, is e.g. $u_i = \tanh(k \cdot S_i)$. These kinds of functions are also called *squashing functions* because they map their inputs monitonically into $(0, 1)$ or $(-1, 1)$ respectively.

The activation functions introduced so far are deterministic functions. There exist also stochastic activation functions like:

$$f(S_i) = u_i = \begin{cases} 1 & with\ probability\ \frac{1}{1+e^{\frac{-k \cdot S_i}{T}}} \\ 0 & with\ probability\ 1 - \frac{1}{1+e^{\frac{-k \cdot S_i}{T}}} else \end{cases} \tag{4.4}$$

This kind of activation function is used e.g. in the Boltzman machine.[12]

4.1.3 Dynamical properties

One has to specifiy the order in which the cells of a neural network compute their activation and output values. A basic distinction can be made between *synchronous* and *asynchronous* models.

In synchronous models the activation values are computed in a fixed order. In feedforward nets, like the Multilayer Perzeptron, the activation values are usually computed layerwise, beginning with the input layer and then updating the outputs of all the units of one layer after another up to the output layer.

On the other hand in asynchronous models the activation values are updated in a non–deterministic order. An example for such a network is the Hopfield-net.[13] In asynchronous models usually it is tried to achieve a steady state of the network.

4.1.4 Learning properties

Every neural network has associated a certain kind of *training algorithm*, also called *learning rule.* The objective of the training is to determine the weights of the connections in a proper way. To do this, a set of training examples $\{\mathbf{x}_i\}_{i=1,2..,m}$ is presented to the network. The training examples are given as inputs to the neural net and are used to adjust the weights between the units to minimize some objective function, which in most cases is a function of the error that is made by the network as the examples are presented.

In *supervised learning* the correct response $\mathbf{y}_i$ for every training example $\mathbf{x}_i$ is known, i.e. the training set consists of pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,2..,m}$. The aim for the network is to learn the mapping from the inputs to the outputs. Sometimes we will refer to the output of the network as a function value $F(\mathbf{x}_i, \mathbf{w})$, where $\mathbf{x}_i$ is the input vector of the training example and $\mathbf{w}$ is the vector of all network weights. The objective is then to minimize some kind of error measure $E(\mathbf{y}_i, F(\mathbf{x}_i, \mathbf{w}))$ for the network.

For *unsupervised learning* no correct output is available. Here often a similarity measure between the inputs is defined and the aim is to group similar inputs. In the literature to neural network this kind of learning is also often called a *self organization process.*

A further distinction can be made between *free network* problems and *fixed network* problems. In fixed network problems the architecture of the network is fixed at the beginning of the learning process and only the weights between the units can be adjusted. In free network problems units may be added or

[11] See Duda, Hart (1973).
[12] see e.g. Rojas (1993), chapter 14.
[13] See e.g. Rojas (1993), chapter 12 or Hecht-Nielson (1991), Page 96ff.

other changes of the network architecture may be made. An example for a constructive algorithm where units are added to the net as long as necessary is the casscade correlation algorithm.[14]

4.2 The Backpropagation Algorithm

In this section the Backpropagation algorithm as one possible learning algorithm for neural networks is introduced. It is probably the most popular learning algorithm for neural networks. It can be used as a supervised learning algorithm for feedforward networks.[15] The neural networks, which can be trained with the Backpropagation algorithm are often called *Multilayer Perzeptron* or *Backpropapagtion Nets.* These nets have typically one input layer, at least one hidden layer and an output layer that gives the overall network output. The neurons usually have a nonlinear, differentiable activation functions like (4.3) or $\tanh(k \cdot S_i)$.

The objective of the algorithm is to adjust the weights of the connections between the neurons to minimize a certain error function. The inputs are presented to the input layer and the outputs of the neurons are calculated layer by layer up to the output layer. As it is an algorithm for supervised learning, the output of the neural network $F(\mathbf{x}_i, \mathbf{w})$ can be compared to the correct output $\mathbf{y}_i$ and the error can be calculated.

An often used error measure is the mean squared error between the network output and the desirable output. If one consideres several output nodes, the mean squared error is defined by:

$$E = \frac{1}{m} \cdot \sum_{i=1}^{m} \sum_{j \in output\ units} \left(u_{ji} - y_{ji}\right)^2 \tag{4.5}$$

where y_{ji} is the correct output for the j–th output unit for the i–th trainings example and $u_{ji} = f(S_{ji}) = F_j(\mathbf{x}_i, \mathbf{w})$ is the corresponding network output. In the following the indices for the i–th training example will be suppressed. The objective of the algorithm is to minimize this error function, so the gradient $\nabla_{\mathbf{w}} E$ has to be evaluated and the weights are adjusted proportionally to $-\partial E / \partial w_{kj}$.

For the k-th neuron the partial derivative $\partial E / \partial w_{kj}$ can be expressed as:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial S_k} \cdot \frac{\partial S_k}{\partial w_{kj}} \tag{4.6}$$

Using equation (4.1) we obtain

$$\frac{\partial S_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \cdot \sum_h w_{kh} \cdot u_h = u_j \tag{4.7}$$

Now let δ_k be defined as follows:

$$\delta_k = \frac{\partial E}{\partial S_k} \tag{4.8}$$

To determine the derivative, the δ_k for all nodes have to be obtained. Calculating δ_k, it can be shown[16] that the values of δ_k for any output neuron are:

$$\delta_k = (u_k - y_k) \cdot f'(S_k) \tag{4.9}$$

For a hidden neuron one obtains:

$$\delta_k = f'(S_k) \cdot \sum_h \delta_h \cdot w_{hk} \tag{4.10}$$

This last equation says, that δ_k for a hidden neuron can be calculated recursively from the δ_h of the neurons of an upper layer, to which it has an outgoing connection. With this so called *generalized δ-rule* the error defined at the output units is propagated backwards. The calculation of the errors has to be started at the output layer propagating it backwards through the hidden layers. Basically the Backpropagation algorithm uses a gradient descent method to minimize the error measure.

---

[14] See Fahlman, Lebiere (1989).

[15] A recurrent network can be turned artificially into a multilayer feedforward network and be trained with the Backpropagation algorithm, see Rumelhart, McClelland (1986), Chapter 8.

[16] See e.g Rumelhart, McClelland (1986).

In summary the backpropagation algorithm consists basically of two steps. In the first step the inputs for the neural network are propagated forward up to the output neurons and the network error is calculated. In the second step, the network error is propagated backwards until the errors for all neurons are known and the weights of the connections are updated. The changes Δw_{kj} for the weights are then:

$$w_{kj}^{l+1} - w_{kj}^{l} = \Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = -\eta \cdot \delta_k \cdot u_j \tag{4.11}$$

The constant η is called the learning rate and is often chosen a small value and the weights initially are generated as small random numbers. Then the term w_{kj}^{l} then refers to the weight w_{kj} in the l-th iteration of the algorithm. In the derivation of the gradient it can be seen, that the activation functions of the neurons have to be differentiable, since $f'(S_k)$ must exist. Therefore a binary function like (4.2) cannot be used as an activation function. A linear function like $u_k = const \cdot S_i$ should not be used as an activation function for the hidden units since a network without any hidden layer can easily be shown to give the same output. The nonlinear activation function, that are commonly used, give a nonlinear mapping from the inputs to the outputs.

Other error functions as the quadratic error function can be used. One possibility is e.g. the *cross-entropie error function*, defined by:

$$E = \sum_{i=1}^{P} \sum_{j} \big(y_{ji} \cdot \log u_{ji} + (1 - y_{ji}) \cdot \log(1 - u_{ji}) \big) \tag{4.12}$$

There are different posibilities when to update the weights of the neural network. For an exact implementation of a gradient descent algorithm, the update of the weights should be made after a cycle through the hole training set. This kind of network training is called *batch- or off-line-training.* Another possibility is to update the weights after every presentation of an training example, that is called *on-line training.*

One problem associated with the backpropagation algorithm is, that it takes a long time to train the network. Several efforts have been made to accelerate the training algorithm by using for example the second derivatives of the objective function.[17] Another simple possibility to speed up the convergence of the algorithm is the use of the so called *Momentum-Term.* The use of a Momentum gives the modified update-rule for the network weights:

$$w_{kj}^{l+1} = w_{kj}^{l} - \eta \cdot \delta_k \cdot u_j + \alpha \cdot \Delta w_{kj}^{l} \tag{4.13}$$

where $\Delta w_{kj}^{l} = w_{kj}^{l} - w_{kj}^{l-1}$ and α is the Momentum.[18]

4.3 Properties of neural networks

In this section some properties of feedforward neural networks are introduced shortly. First the approximation of functions is mentioned. Next an interpretation of neural networks from a statistical point of view is given and finally the relationship between neural networks and posterior probabilities, when the neural nets are used for classification tasks, is explained.

4.3.1 Approximation of functions with neural networks

For the capacity of neural networks the kind of functions they can approximate is decisive. The aim of neural networks is to approximate a function $g : \Re^n \rightarrow \Re^c$. Here the most important result is, that feedforward networks with only one hidden layer and any type of a squashing function as an activation function of the hidden layer can approximate any continuous function g up to a desired accuracy, provided that a sufficently large number of hidden neurons is used.[19] These theoretical results give no hint on how many hidden units are needed for a specific problem, so that the determination of an adequate architecture of a neural net is difficult and is often done by trial and error. Nevertheless these results are a justification for the sometimes very good results obtained with neural networks. Any deficiencies in the learning of the relation between the inputs and the outputs may be cause of inadequate learning,

[17] An overview over different methods for the acceleration of the network training can be found e.g. in Battiti (1992).
[18] See Rummelhart, McClelland (1986).
[19] See e.g Hornik et. al. (1989).

an inadequate number of hidden neurons or a statistical instead of a deterministical relation between the inputs and the outputs.[20]

4.3.2 Interpretation of neural networks from a statistical point of view

Often the training examples for the training of neural networks are drawn from a statistical environment and can be interpreted as realizations of random vectors $\mathbf{X}$ and $\mathbf{Y}$.[21] A deterministic relationship may then be considered as a special case of a statistical relationship. The aim of statistical procedures like regression analysis is to determine the relationship that exists between the two random vectors. The joint behaviour of $\mathbf{X}$ and $\mathbf{Y}$ can be described by the joint density function $f_{x,y}(\mathbf{x},\mathbf{y})$, that can be decomposed into $f_x(\mathbf{x}) \cdot f_{y|x}(\mathbf{y}|\mathbf{x})$ where $f_{y|x}(\mathbf{y}|\mathbf{x})$ is the conditional density function that describes the behaviour of $\mathbf{Y}$ given $\mathbf{X}$.

In this section especially the conditional expectation of $\mathbf{Y}$ given $\mathbf{X}$, denoted $E[\mathbf{Y}|\mathbf{X}] = g(\mathbf{X})$, is considered. The conditional expectation gives the value of $\mathbf{Y}$ that will occur on average given a realization of $\mathbf{X}$. A random 'expectational error' $\epsilon \equiv \mathbf{Y} - E[\mathbf{Y}|\mathbf{X}]$ can be defined. Then $\mathbf{Y}$ can be written as

$$\mathbf{Y} = E[\mathbf{Y}|\mathbf{X}] + \epsilon \tag{4.14}$$

where it can be easily verified that $E[\epsilon|\mathbf{X}] = 0$.

The objective of network training is to minimize a certain error or performance measure π. If the target values $\mathbf{y}$ for the network training are given, the performance measure can be written as a function $\pi(\mathbf{y}, F(\mathbf{x},\mathbf{w})$. Often it is desirable to perform well over a range of different values for $\mathbf{y}$ and $\mathbf{x}$. One way to precise this, is to require a good performance 'on average', what can be expressed by the expectation of the random variable $\pi(\mathbf{Y}, F(\mathbf{X},\mathbf{w})$ as

$$h(\mathbf{w}) \equiv E\big[\pi(\mathbf{Y}, F(\mathbf{X},\mathbf{w})\big] = \int \pi(\mathbf{y}, F(\mathbf{x},\mathbf{w}))f(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} \tag{4.15}$$

where $h(w)$ is only a function of the network weights. The objective of learning in neural networks then can be expressed as to give the best average performance, that is to find a solution to the problem:

$$\min_{\mathbf{w}\in\mathbf{W}} h(\mathbf{w}) \tag{4.16}$$

A solution to this problem that need not be unique will be denoted w^*.

Now the special case of a quadratic error measure is discussed more extensively. The expectation of the performance measure can now be written as:

$$h(\mathbf{w}) = E\big[(\mathbf{Y} - F(\mathbf{X},\mathbf{w}))^2\big] \tag{4.17}$$

It can be shown, that this error function can written in terms of $E[\mathbf{Y}|\mathbf{X}]$ as:

$$h(\mathbf{w}) = E\big[(\mathbf{Y} - E[\mathbf{Y}|\mathbf{X}])^2\big] + E\big[(E[\mathbf{Y}|\mathbf{X}] - F(\mathbf{X},\mathbf{w}))^2\big] \tag{4.18}$$

This example shows, that the choice of a special performance measure is strongly related to the statistical relationship between $\mathbf{X}$ and $\mathbf{Y}$ that is of interest. The performance measure $h(\mathbf{w})$ is minimized if $E[\mathbf{Y}|\mathbf{X}] = F(\mathbf{X},\mathbf{w})$ as $E\big[(\mathbf{Y} - E[\mathbf{Y}|\mathbf{X}])^2\big]$ is a constant, independent of the weight vector $\mathbf{w}$ and the term $E\big[(E[\mathbf{Y}|\mathbf{X}] - F(\mathbf{X},\mathbf{w}))^2\big]$ is minimized if $E[\mathbf{Y}|\mathbf{X}] = F(\mathbf{X},\mathbf{w})$. In this case it can be seen, that a weight vector $\mathbf{w}^*$ minimizing $h(\mathbf{w})$, gives a solution $F(\mathbf{X},\mathbf{w}^*)$, that is a mean-squared error minimizing approximation to the conditional expectation $E[\mathbf{Y}|\mathbf{X}]$.

In this context, the density function for the network inputs $f_x(\mathbf{x})$ plays an essential role to determine the optimal weights $\mathbf{w}^*$. The optimal weights give on average small errors for values of $\mathbf{X}$ that are likely to occur and larger errors on average for values of $\mathbf{X}$ that are very unlikely. The density function $f_x(\mathbf{x})$ can be regarded as a kind of environment measure that describes the way in which the input variables are generated. The dependence of the optimality of $\mathbf{w}^*$ on $f_x(\mathbf{x})$ means also, that in a different environment these weights will not give the optimal performance. The objective function also plays an essential role.

[20] See Hornik et. al. (1989).

[21] A more detailed description of the statistical interpretation of neural networks can be found in White (1989), on that the exposure in this section is based. In Cheng, Titterington (1994) different models of neural networks are related to statistical methodology. A comparison between statistical methods for classification and neural networks can be found in Ripley(1993).

The optimality of some weights $\mathbf{w}^*$ depends strongly on the performance measure chosen. So the question of the selection of an adequate performance measure should also be considered.

Another notion, that is often mentioned, is *generalization.* Generalization may be seen as the ability of a system, that is trained with a set of Input-Output pairs from a certain Input-Output space, to determine the output of inputs, that do not belong to the training set, in a good way.[22] In the statistical framework considered here, generalization can be interpreted in this way: As long as the inputs and outputs are governed by the density function $f_{x,y}(\mathbf{x}, \mathbf{y})$, the weights $\mathbf{w}^*$ will give the minimal average performance.

It should be noticed, that the results that are presented here do not depend on a special architecture of neural networks nor is it necessary that a neural network model is used at all. Here a neural network is only a special case of an input-output mapping.

Here the neural networks are interpreted in a theoretical point of view. One condition for the results to hold is, that actually the error function can be minimized by a specific neural network. The problem is, that the approximation of the network output $F(\mathbf{X}, \mathbf{w})$ to the desired output may be bad because a not sufficiently large number of hidden neurons is used, the number of the training examples is very limited or the network has got stuck in a local minimum of the error function.

4.3.3 Classification with neural networks and the relation to a posteriori Probabilities

In section 4.3.2 neural networks were discussed from a more general point of view with respect to statistical procedures. Now the special case of classification is considered.[23]

The objective of classification is to determine one of a finite number of classes to which a certain feature vector $\mathbf{x}$ belongs. Here the random variable Θ specifies the class membership, i.e. $\Theta = \theta_i$, $i = 1, 2, .., c$ denotes the membership to the ith class. To show the relationship with neural networks the random vector $\mathbf{Y} \in \Re^c$ is considered, that takes only discrete values corresponding to the basis vectors $e_i = (0, 0, .., 1, 0, ..)'$. If $\mathbf{Y} = e_i$, then the feature vector $\mathbf{x}$ belongs to the class θ_i. Therefore, the random vector $\mathbf{Y}$ and the random variable Θ relate the same information.

As already mentioned in section 4.3.2, the network output $F(\mathbf{x}, \mathbf{w})$ approximates the conditional expectation $E[\mathbf{Y}|\mathbf{x}]$. Let Y_i denote the ith component of this vector, then $F_i(\mathbf{x}, \mathbf{w})$, the ith component of the network output, is

$$F_i(\mathbf{x}, \mathbf{w}) = E[Y_i|\mathbf{x}] = \sum_{y_i \in \{0;1\}} y_i \cdot P(y_i|\mathbf{x}) = P(Y_i = 1|\mathbf{x}) = P(\theta_i|\mathbf{x})$$

Here $P(\theta_i|\mathbf{x})$ is the posterior probability that the feature vector $\mathbf{x}$ pertains to class θ_i. This means, that minimizing a quadratic error function in the case of classification leads to an approximation of the posterior class probabilities.[24] The outputs of the neural network may be interpreted as posterior probabilities, if every class is represented by one output neuron and the correct output is coded as a basis vector.

In theory, the posterior probabilities can be used to construct a classifier that minimizes the probability of error. In this case the class with the highest posterior probability should be chosen. Also minimum risk classifiers could be constructed. The interesting point here is, that neural networks approximate the posterior probabilities that are necessary to construct a bayesian classifier. These classifiers have the normative property that no other classifier can have a lower error rate or a lower risk *on average.* This gives a theoretical justification of the use of neural networks for classification as they approximate these posterior probabilities. Yet the actual probability of error or the expected loss depend on the accuracy of the estimation of the posterior probablilities by the neural network.

One direct implication of this result is, that the sum of the network outputs should approximate 1 since the sum of the posterior probabilities must be also 1. But the sum of the actual network outputs may differ from 1. One possibility to solve this problem is to norm the network outputs by

$$\hat{u}_j = \frac{u_j}{\sum_i u_i} \qquad j = 1, 2, .., c \tag{4.19}$$

where u_j is the output of the jth output neuron.

[22] See Wolpert (1990).

[23] See Wan (1990) for a more detailed description of the following. The results that are presented here are also described e.g. in Richard, Lippman (1991), Ruck et.al. (1990).

[24] It can be shown, that minimizing the cross-entropy function gives also an approximation of the posterior class probabilities, see Richard, Lippman (1991).

5 Comparison of Neural Networks to Control Charts

In this section the practical experiments that were carried out with neural networks to compare them to quality control charts without memory are presented. First the basic assumptions for the comparison and some practical considerations for the network training are made. Then the results of the comparisons to a $\bar{x}$-chart, a standard deviation control chart and a combined $\bar{x}$-s-chart are discussed.

5.1 Assumptions for the comparison

For the comparisons a sample size of $n = 6$ was chosen. The samples are supposed to be realizations of independent identical $N(\mu_t, \sigma_t^2)$-distributed random variables X_t. If the process is under statistical control the parameters are chosen $\mu_t = \mu_0 = 0$ and $\sigma_t = \sigma_0 = 1$. If a change occurs, either the parameter μ_t or σ_t change. For the different control charts a niveau $\alpha = 0,01$ is chosen.

For the comparisons to the $\bar{x}$-chart, a change in the process mean of an additive constant δ was assumed, so that the new mean is given by $\mu_t = \mu_0 + \delta = \delta$. The process standard deviation was supposed to be constant. The control limits of the $\bar{x}$ chart are then

$$UCL = \tau_{0,995} \cdot \frac{1}{\sqrt{6}} = 1,0516 \qquad LCL = -\tau_{0,995} \cdot \frac{1}{\sqrt{6}} = -1,0516 \tag{5.1}$$

For the comparisons to the s-chart, a change of a multiplicative factor $\epsilon > 1$ was assumed. The changed standard deviation is then $\sigma_t = \epsilon \cdot \sigma_0 = \epsilon$. As only an increase of the standard deviation was considered, the control limits for the s-chart are given by

$$UCL = \sqrt{\frac{\chi^2_{5;0,99}}{n-1}} = 1,737 \qquad LCL = 0 \tag{5.2}$$

For the comparisons to the combined $\bar{x}$-s chart, the two single charts were used with the control limits as given in (5.1) and (5.2). This leads to a probability of false alarms of $\alpha = 0,0199$.

The samples were generated by Monte-Carlo Simulation. When the random samples were generated, the correct output for the training of the neural networks can be determined and additional sample statistics like the sample mean and the sample standard deviation were calculated.

5.2 Some Practical considerations for the training of the neural networks

5.2.1 Partition of the samples into sets

In total three disjunctive sets of samples were generated. The first set, the training set, was used for the training of the neural networks, i.e. to determine the corresponding network weights. The second set of samples was used to determine the interpretation limits for the network outputs. These interpretation limits have to be determined in a certain way, to observe the different weight of errors of type 1 and type 2.[25] The third set was used as a test set, to obtain the numbers of errors for the comparison between the neural networks and the control charts. The tabels of results are based on the evaluation of the test set.

5.2.2 Preprocessing of the samples

It was also one objective of this investigation to examine the effect of different possibilities of preprocessing of the samples. In the discussion of the results differences caused by the preprocessing will be discussed only, when they give obviously different results. Now the different possibilities for the preprocessing considered here are explained.

One possibility is, not to preprocess the sample vector and take it as input for the neural network.

Additionaly sample statistics like the sample mean and the sample standard deviation could be calculated and used as additional inputs for the neural network. This should actually not be necessary, since the neural networks should be able, at least in this case, to learn the relationships to imitate a control chart without sample statistics. Furthermore e.g. only the sample mean should suffice to imitate a $\bar{x}$chart. Therefore, the sample statistics should be redundant.

Another possibility is to order every sample e.g. in increasing order, so that $x_{(1)} < x_{(2)} < .. < x_{(n)}$, where $x_{(j)}$ is the j-largest sample value. This gives the sample vector $(x_{(1)}, x_{(2)}, .., x_{(n)})'$. Such

[25] The determination of these limits will be explained in more detail in section 5.2.3 and section 5.2.4.

a preprocessing has the advantage that the jth input neuron always gets as input the j-largest sample value.

A further possibility is to consider only the upper part of an control chart and to 'reflect' the sample realizations that fall beneath the middle line of the $\bar{x}$ control chart at this middle line. So only one control limit has to be considered. This would be also a way to influence the architecture that is necessary for a neural network to implement the a certain function for the problems that are considered here.

Often a standardization of the input data is made, so that the inputs take values from $[0;1]$ or $[-1;1]$. Ths might be necessary if the input variables are measured in different scales. Because of the selection of the parameters μ_0 and σ_0, no standardization of the data was considered.

5.2.3 Determination of correct outputs

Two different methods were considered to determine the correct outputs needed for the training of neural networks.

One possibility is to choose one output neuron for each class that has to be considered, what corresponds to a vectorial codification of the correct outputs. In this case the class θ_j is represented by a basis vector whose jth component is 1, that is by $e_j = (0,..,0,1,0..0)'$. The advantage of this kind of representation is that the theoretical results presented in section 4.3.3 can be used, since the outputs of the neural networks should approximate the posterior class probabilities of the input. Anyway all classes should be represented by an output neuron, as it is not justified to set the output for the not considered class i to $u_i = 1 - \sum_{j \neq i} u_j$ since the approximation of the posterior probabilities may be poor.

To interprete the outputs, one way is to assign the input to the class that is represented by the output neuron with the highest output value. In this way one implicitly minimizes the probability of error, since one selects the class with the maximum posterior probability. Another way is to multiply the network outputs by the loss associated with a false decision and to assign the class with the smallest estimated posterior loss value. If a class is assigned only if the corresponding output crosses a certain limit, implicitly a certain loss is assigned to the different classes. In this paper another possibility to determine the limits for the interpretation of the outputs that is described in section 5.2.4 is used.

Another way of representing the outputs is to take one output neuron and to assign each class a certain value as the correct output. This can be considered a scalar codification of the outputs. Since the output values will not be obtained exactly as the outputs of the neural network, the possible range of output values has to be divided into intervals. According to the interval, into which the output of the neuron falls, a class is assigned. One disadvantage of this way of determining the correct outputs is, that the theoretical results of section 4.3.3 will not hold. An advantage of this way to choose the outputs is, that classes like *very good*,*good*, *bad* and *very bad* can be represented by increasing values and the error made at the output neuron has a direct interpretation. In a vector representation only the fact that an input is classified into the false class is of concern whereas here a bigger error is made if an input of class *very good* is classified as *very bad* as if it were classified as *good*.

5.2.4 Classification considering errors of type 1

One characteristic of the control charts is, that they should obey a certain rate of error of type 1. The reason for this is, that the process should not be interrupted to many times unnecessarily. For the comparison between neural networks and quality control charts it was considered essential, that also with the neural networks a certain rate of false alarms can be observed. If the outputs are codified as a vector, limits for the output nodes that indicate a change of the process parameters have to be sought. If the output of such a neuron exceeds this limit, it is supposed that the corresponding parameter has changed. In the case of a scalar codification, values have to be determined to divide the output adequately into intervals.

It was considered not to be enough to fix a certain limit for the interpretation of the outputs of the neural networks and then to calculate the corresponding number of errors of type 1 and type 2. An independent set of samples was used to determine these values for the interpretation of the outputs. The number of errors of type 1 and type 2 made by the corresponding control chart was compared to the number of errors of type 1 and type 2 made by the neural network for different values for the output limits to determine adequate interpretation limits. The values for the output limits were examined over a fixed range of values with a stepsize of 0.01 or 0.02.

5.2.5 Training of neural networks

One problem of the training of neural networks is to decide when to stop the training. The training should be stopped if the generalization ability of the neural net is good. One problem that occurs with the Backpropagation algorithm is the so called *overtraining.* This means that a neural net with a complex architecture and a huge number of adjustable weights adapts very well to the training set, but if examples not contained in the training set are presented to the network, the interpolation ability is quite small. One possibility to avoid overtraining is to use a *validation set.*[26] The validation set contains examples that are not used for the determination of the network weights and that are presented to the network after a number of presentations of the training set to the neural network. If the network error reaches a minimum error for the validation set, the network training may be stopped.

For the training of neural networks also the architecture of a special neural net has to be determined. The choice of the right architecture is most often solved in a trial and error manner, so that different architectures have to be tried. By the special classification problem only the input and the output layer are fixed. The selection of the right architecture involves then the selection of the number of hidden layers and the of number of neurons in each hidden layer. One often used rule is to take the smallest network, that describes the data well. Smaller networks are also more likely to give a good generalization, but they are more sensitive to the initialization or the selection of the parameter values and it may be also possible that they are not able to implement the input–output relation well. Another possibility is to begin with a more complex architecture and to reduce it adequately. Methods that change the architecture of a neural network or change the error function are called *Prune–methods.* For an overview over different prune–methods the reader is referred to Reed (1993).

Other possible parameter choices, that are not discussed here, involve e.g. the choice of an appropriate learning rate, the activation function and other more technical parameters.

5.3 Discussion of the experimental results

In this section the results of the practical experiments are presented. First the theoretical results of section 4.3.3 are tested considering an one-sided test for the mean, where the posterior probabilities can be calculated explicitly. Next comparisons to a $\bar{x}$-chart, a s-chart and a combined $\bar{x}$-s-chart are presented. At last the classification of the kind of change is considered.

5.3.1 Test of the approximation of the posterior probabilities

In order to check the theoretical results of section 4.3.3 and some implications for the outputs of neural networks, a test with $\delta = 1$ is considered. The sample size is chosen according to the assumptions as $n = 6$ and $X_1, X_2, .., X_6 \sim N(\mu, 1)$ where μ can take the two values $\mu_0 = 0$ and $\mu_1 = 1$. The corresponding test is:

$$H_0: \ \mu_0 = 0 \qquad H_1: \ \mu_1 = 1$$

The test was chosen in this way, because the posterior probabilities of the parameters can be calculated explicitely and can be compared to the outputs of a neural network. The test can be formulated as a Neyman-Pearson test or as a bayesian test. The both test formulations can be shown to be equivalent in this case and given one of them, the other can be formulated, so that both tests give the same decision. Here the case is considered, that a level α is given for the Neyman-Pearson test. Then the posterior probability for μ_1 can be calculated, that must be exceeded to give an equivalent bayesian test. The interesting point for this comparison is, that a neural network should approximate the posterior probabilities needed for the bayesian test.

For a given level α the Neyman-Pearson test is given by

$$decide \begin{cases} \mu_1 & if\ f_{\mu_1}(\mathbf{x}) > k \cdot f_{\mu_0}(\mathbf{x}) \\ \mu_0 & otherwise \end{cases} \tag{5.3}$$

Here $f_\mu(\mathbf{x})$ is the density function given the parameter μ.

For the formulation of the bayesian test, a loss function has to be given. This function can be defined here by $L(d_0, \mu_0) = L(d_1, \mu_1) = 0$, $L(d_0, \mu_1) = b > 0$ and $L(d_1, \mu_0) = a > 0$. The decision d_0 refers to the acceptance of H_0 and the decision d_1 to a rejection of H_0. So e.g. $L(d_0, \mu_1)$ is the loss made if H_0

[26] See e.g. Hecht-Nielson (1991). There also the use of a validation set is described in more detail.

is accepted but μ_1 is actually the correct parameter. This means that a right decision causes no loss, whereas a wrong decision causes a loss of $L(d_0, \mu_1)$ or $L(d_1, \mu_0)$.

The bayes test is given by

$$decide \begin{cases} d_1 & if\ f_{\mu_1}(\mathbf{x}) \cdot P(\mu_1) \cdot b > f_{\mu_0}(\mathbf{x}) \cdot P(\mu_0) \cdot a \\ d_0 & otherwise \end{cases} \tag{5.4}$$

Additionally the prior probabilities of the parameters have to be determined. In this section it is assumed that $P(\mu_0) = P(\mu_1) = 0.5$. The samples for the training of the neural nets were generated according to this assumption so that for exactly one half of the samples parameter μ_1 was chosen.

One can also calculate the posterior probability for parameter μ_1 and choose a value for this posterior probability from that one should decide d_1 to get a bayesian test equivalent to a Neyman-Pearson test with a fixed level α. Here, with $\alpha = 0,01$, this probability can be shown to be $P(\mu_1|\mathbf{x}) = 0.937$. As the neural networks approximate the posterior probabilities, the value chosen for the interpretation limit of the output, should be close to this value.

The posterior probabilities of parameter μ_0 and μ_1 can be calculated easily. According to the assumptions made in this chapter the posterior probability for μ_0 is:

$$P(\mu_0|\mathbf{x}) = \frac{1}{1 + \exp(\sum_{i=6}^{6} x_i - 3)} \tag{5.5}$$

The posterior probability depends only on the sample mean, since $\sum_{i=6}^{6} x_i - 3 = n \cdot \bar{x}$. This exact posterior probability can be compared to the estimates made by the neural networks.

To test the theoretical results and its implications for the neural networks, different neural networks were trained to imitate the statistical test. The correct outputs were coded like a vector in order to test the approximation of the posterior probabilities. This means that two output neurons were used to represent the two different parameters. It was also tested whether the sum of the outputs approximates 1 as it should, when posterior probabilities are estimated. The neural networks were trained with a training set of 1500 samples. Nets with and without a hidden layer were trained, to see whether with hidden neurons better results are obtained. The limits for the interpretation of the outputs were determined in the way described in section 5.2.4. For the interpretation of the outputs only the node that represents parameter μ_1 had to be considered. If the so fixed limit is exceeded, the parameter μ_1 is chosen. The results given in the table are based on the evaluation of the independent test set.

In table 5.1 first the architecture of the neural networks is given. Here e.g. 6-3-2 refers to a net with 6 input neurons, 3 hidden neurons and 2 output neurons. Next the errors of type 1 and type 2 are given. The third column contains the interpretation limits. As further tests for the approximation to the posterior probabilities, the maximum and the minimum of the summed outputs and the mean absolute deviation of the correct posterior probability and the output of the neuron that represents parameter μ_0 were calculated.

architecture	errors of type 1/type 2	interpretation limit	maximum of summed outputs	minimum of summed outputs	mean absolute deviation from posterior probabilities
control chart	5 / 354	/	/	/	/
6–2	6 / 354	0.92	1.00002	0.99998	0.020986
6–3–2	6 / 343	0.91	1.01231	0.98492	0.030631
6–5–2	6 / 385	0.93	1.0264	0.97846	0.026766
6–7–2	7 / 341	0.90	1.00354	0.99674	0.02784

Table 5.1 Experimental results for the approximation of posterior probabilities and comparison to a statistical test.

The rejection region of the statistical test is given by $\tau_{0.99} \cdot 1/\sqrt{6} = 0.9497$. The results show, that a neural net without hidden layer (network 6–2) suffices to give a good approximation to the posterior probabilities. With this network nearly the same results as with the statistical test are obtained. Only one error of type 1 more is made. The nets with additional hidden units give similar results. Net 6–3–2 gives even less errors of type 2 as net 6–2.

With respect to the explicit approximation to the posterior probabilities the net with architecture 6–2 gives the best results. The maximum and the minimum of the sum of the posterior probabilities is nearly 1. Here the nets that use hidden units give quite large values compared to the net 6–2, but these values do not differ very much from one if one considers that it is the maximum or the minimum with respect to 1500 test samples.[27]

Regarding the mean deviation of the actual posterior probabilities from the outputs of the neural networks, the net without hidden layer gives the smallest value. The other networks with hidden layer give larger deviations. The limits for the interpretation of the output are all close to the theoretical value of 0.937. This should be so, as with a good approximation of the posterior probabilities, the interpretation limits should also be close to the theoretical value. It should be also clear, that the results with respect to errors of type 1 and type 2 should also be near the statistical test, as with the posterior probabilities a bayesian test equivalent to a Neyman-Pearson test can be formulated and this bayesian test depends on these posterior probabilities. This means, that neural networks can be trained to imitate some special statistical tests also with respect to a certain rate of error of type 1 that has to be considered.

5.3.2 Comparison to a $\bar{x}$-chart

Here a more general comparison to a $\bar{x}$ chart is carried out, so that positive and negative shifts of different magnitudes were considered. The samples were generated in that way, that approximately for one third of the samples no shift of the mean occurred, for approximately one third of the data a negative shift uniformly distributed in $[-3\sigma_0; -0.5\sigma_0]$ and for the last third a positive shift uniformly distributed in $[0.5\sigma_0; 3\sigma_0]$ was supposed.[28] As it was assigned randomly, whether a shift had occurred or not, the number of samples without shift need not be exactly one third of the total number of samples generated. For the comparisons only the number of errors of type 1 and type 2 were considered. Another possibility, that was not considered here, would have been to estimate the average run length or the power function for some selected shift δ.

One objective of the experiments was also to investigate the effect of different kind of preprocessing of samples. The preprocessing and the determination of correct outputs can be chosen in such a form, that theoretically no hidden layer is necessary to imitate a $\bar{x}$-chart. Only for one representation of the inputs and the outputs a hidden layer has to be used. This representation contains the sample vector without any preprocessing and a correct output that is one, if a shift of the mean occurred and cero if no shift occurs. A hidden layer has to be used, because to indicate a change of the mean, the output should be big, whereas if no change occurred the value at the output node should be small. This cannot be achieved without hidden layer, as the output gives only a monotone transformation of the inputs and positive as well as negative shifts are considered.

Here the different possible kinds of preprocessing and two different kinds of the choice of the correct outputs were examined. As only very small differences between the different representations could be found, the influence of the preprocessing and representation of outputs will not be discussed here extensively.

First the evaluation of the three sets of samples with a $\bar{x}$-chart is given in table 5.2.

set	number of samples	samples without shift	errors of type 1	errors of type 2
training set	1500	489	4	225
set for det. limits	1500	497	4	200
test set	1500	501	5	203

Table 5.2 Evaluation of the samples for the Shewhart control chart

In the next table some results of the neural networks for different types of preprocessing of the data are presented. Here *reflected* refers to the use of only one side of the control chart, i.e. a sample with a sample mean smaller than cero was multiplied with -1. *Ordered* refers to the ordering of every sample in increasing order. At last with codification the presentation of the correct output is meant. The different number of output nodes can be explained by the different types of preprocessing of the data and the representation of the correct outputs.

[27] It is interesting to compare the shape of equation (5.5) to equation (4.3).

[28] Here a special interpretation of the prior class probabilities is made. This interpretation corresponds to three classes, that have equal prior probabilities. Another possibility would have been to choose for one half of the samples a shift that can be positive or negative and for the other half no shift at all.

architecture	errors of type 1	errors of type 2	remarks
6–2	5	206	reflected, ordered data, vectorial codification
6–2	5	200	not reflected, ordered data, vectorial codification
6–1	5	196	not reflected, ordered data, scalar codification
6–3	5	207	not reflected, not ordered, vectorial codification

Table 5.3 Results for the neural networks for the comparison to a shewhart control chart for the process mean

Generally it can be said, that neural networks are capable to imitate a $\bar{x}$-chart and may obtain quite the same results with respect to errors of type 1 and type 2. This can be seen comparing the results for the neural networks given in table 5.3 to the results of the test set in table 5.2. Accidentially all the neural nets give 5 errors of type 1 with little differences in the number of errors of type 2. The kind of preprocessing of the samples had only very little influence on the results obtained with the neural nets.[29] With different initializations of the weights, the results with respect to errors of type 1 and type 2 for the neural networks were very similar to those in table 5.3.

Another result is, that no hidden layer is necessary to obtain results, that are comparable with the results for the control charts. But this is only the case if really no hidden units are necessary from theoretical considerations as described before. In the case were hidden units are necessary, at least two hidden units were necessary.[30] Besides it should be mentioned, that the results were obtained without using additional inputs like the sample mean.[31] Actually neural networks trained with additional inputs or with a hidden layer gave no improvements over the results presented here.

Finally it should be noticed that the results presented here are according to what someone expects. One reason for this is that the problem considered here is basically a linear problem. So the sample mean, that is a sufficient statistic for the parameter μ of a normal distribution, is a linear function. With the propagation function S_i also a linear function is calculated that is subject to a nonlinear monotone transformation. The problem considered here could also be treated as a regression problem. It should be noticed here, that neural nets, that use the backpropagation algorithm do something equivalent to nonlinear regression.[32] The learning procedure used with the backpropagation algorithm can also be identified as a stochastic approximation procedure.[33]

5.3.3 Comparison to a s chart

In this section a comparison between a standard deviation chart and neural networks was made. Here only an increase of the process variability was considered.

For the generation of the samples it was supposed, that approximately the half of the samples were subject to a shift of the standard deviation, that was uniformely distributed in $[1.2\sigma_0; 3\sigma_0]$ with $\sigma_0 = 1$.

As only an increase of the process variability was considered, there was only one reasonable representation of correct outputs. Here a 1 indicated a shift of the standard deviation and a 0 that no shift had occurred. A second output node was used to test, whether the outputs summed 1. In the comparison between the s-chart and neural nets a strong relation between the type of preprocessing and the results of the neural networks could be detected.

First the evaluation of the different sets of data with a standard deviation chart is given. The results of the neural networks have to be compared with those of the s-chart for the test set.

set	number of samples	samples without shift	errors of type 1	errors of type 2
training set	1500	725	9	319
set for det. limits	1500	757	11	317
test set	1500	742	7	325

Table 5.4 Results of the data sets for the s-chart

In the table 5.5 now the results of the neural networks for the test set are given. Here only "reflected" samples were considered. It was checked, that this kind of preprocessing had no noticeable influence on

[29] As the samples were all simulated starting with the same seed for the random number generator, the results of the control charts are independent of the preprocessing.

[30] For a justification of this result, the problem considered here should be compared to the XOR-Problem, described e.g. in Rojas (1993) and the Perzeptron that can implement this function.

[31] See also remarks made in section 5.2.2.

[32] See Bing Cheng and Titterington, (1994) or Ripley, (1993).

[33] See White, (1989).

the results. Here rather the difference of the results obtained with ordered samples and with samples that were not ordered should be observed.

architecture	errors of type 1	errors of type 2	remarks
6–2	8	297	reflected, ordered data
6–9–2	9	278	reflected, ordered data
6–15–2	9	398	reflected, not ordered data
6–13–5–2	11	397	reflected, not ordered data

Table 5.5 Results for the neural networks for the comparison to a shewhart control chart for the process variability

The most salient result is, that the neural networks can give comparable results to a s-chart, if the ordered samples are taken as inputs for the neural networks. Here even a network without hidden layer suffices. If the samples, that are not ordered, are taken as input, a considerably higher number of errors of type 2 is given.

The sample standard deviation is a nonlinear function of the sample. If no preprocessing is made, hidden units should be necessary to approximate this function. The bad results of the neural nets, that take samples without ordering as inputs, may be interpreted, as that the neural networks were not able to give a good approximation of the function under concern. There may be two main reasons for this. One reason may be, that not enough training examples were used as inputs for the neural networks. Another reason is, that the architecture of the net is not sufficiently complex to approximate the desired function. Here no experiments were made, to further investigate this problem.

If the sample standard deviation or the sample range are used as additional inputs to the not ordered samples, results similar to a s-chart are obtained. This should be so as the sample standard deviation is also the statistic used by the s-chart and the neural network only has to 'learn' to interpret the sample standard deviation in a proper way.

With the ordered samples, similar results to a s-chart could be obtained. A statement, whether the neural nets give 'better' results as the s-chart, is not easy to make. Anyway the term 'better' may be difficult to interpret here, as the different errors have to be rated in different ways. Also it makes no sense to speak of an overall error rate, if the two different kinds of errors are not considered. We will limit us here to state, that the neural nets and the s-chart give approximately the same result, if the samples are ordered.

The reason for the good performance of the neural nets with ordered inputs may be explained. The largest and the smallest sample value are enough to calculate the sample range, that is also a linear function, once these both values are given. With the sample range also the process variability can be controlled, as the sample range has a direct relation to the process variability.[34] Here the input for the neural networks is the vector whose components are the order statistics of the sample. If no hidden unit is used, the output neurons basically give a monotone transformation of the weighted sum of the order statistics. An examination of the weights obtained by the training of the networks showed, that the smallest and the largest sample value actually had associated the largest weights.

5.3.4 Comparison to a combined $\bar{x}$-s chart

Now the comparison between neural networks and a combined $\bar{x}$-s-chart is described. As a consequence of the results obtained in the last two sections, the samples were ordered and reflected at the middle line of the $\bar{x}$-chart, so that no sample mean is smaller than cero. For the correct outputs a vectorial representation and a scalar representation was chosen. Here the samples were simulated so, that for approximately one third no shift occurred, for approximately the second third a shift of the parameter μ, that were supposed to be uniformly distributed in $[-3\sigma_0; -0.5\sigma_0] \cup [0.5\sigma_0; 3\sigma_0]$, and for the last part a shift of the standard deviation, that was uniformely distributed in $[1.2\sigma_0; 3\sigma_0]$, was considered. This means, that three different classes as mentioned in section 3 have to be distinguished. Here another possibility would be to admit shifts of μ and of σ at the same time, but this case was not considered here.

In table 5.6 the evaluation of the test set with a combined $\bar{x} - s$-chart is given and explained below.

[34] There exist also control charts that use the sample range, see e.g. Montgomery (1985).

evaluation with $\bar{x}$-s-chart	samples without shift	samples with shift of μ	samples with shift of σ
on both charts inside control limits	495	99	175
only outside the control limits on the $\bar{x}$-chart	5	387	28
only outside the control limits on the s chart	6	1	232
Outside the control limits of both charts	0	2	70
sum	506	489	505

Table 5.6 Results of the test data set for a combined control chart. Errors of type 1: 11, Errors of type 2: 274, false classifications: 31.

In table 5.6 the results for a combined control chart for all possible combinations are given. E.g. a shift of μ may not be shown, be indicated on the $\bar{x}$-chart, on the s-chart or on both charts at the same time. A shift of a parameter is given, if at least one chart gives an out of control signal. No error is made only if a shift of σ is indicated on the s-chart, a shift of μ is indicated on the $\bar{x}$-chart and if no shift occurred, both charts do not indicate an out of control situation.

Here in total $11(=5+6)$ errors of type 1 and $274(=99+175)$ errors of type 2 were made. Another kind of error is made, if the kind of shift is classified in a wrong way. It is striking here, that a lot of times (70) a shift only of σ is indicated on both charts. This is due to the possible values for the shifts of the standard deviation up to $3\sigma_0$. To determine the number of false classification the results of the combined chart should be interpreted. As indicated in section 2.4, the $\bar{x}$-chart does react to an increase of the process variability with a higher probability of false alarms whereas the s-chart is not sensible to changes of μ. If both charts give an out of control situation, this should therefore be interpreted actually as a shift of σ. Considering this, the total number of false classifications amounts to 31.

Next an exemplary result of a neural net, that was trained to distinguish between the three classes is given. This result is representative for the results that were obtained with other architectures of neural networks that were trained for this problem.[35] With other architectures similar numbers of errors were obtained.

evaluation with network 6-5-3	samples without shift	samples with shift of μ	samples with shift of σ
no change indicated	491	120	194
change of μ indicated	8	367	14
change of σ indicated	7	2	297
change of μ and σ indicated	0	0	0
sum	506	489	505

Table 5.7 Results of the test data set for a neural network with architecture 6–5–3. Errors of type 1: 15, Errors of type 2: 314, false classifications: 16.

For the neural network the number of errors of type 1 and type 2 is higher as for the combined control chart. Only the number of false classifications is smaller.[36] In general the neural nets could not reach the low numbers of errors of type 1 and type 2 of the combined control charts. They only made less false classifications. Nevertheless the false classifications are mainly of that kind, that a shift of σ is interpreted as a shift of μ. It is also striking, that it never occurred, that a shift is indicated as a shift of μ and σ.

It should be stated here, that the disadvantage for the neural nets is, that they do not reach the number of errors of type 1 or type 2 of the control chart. Although the neural nets offer an advantage with respect to a lower number of false classification of the kind of shift, this should be no reason to use neural nets to replace a combined control chart, as the lower numbers of errors of type 1 or type 2 of the combined control chart are more important. The false classifications could also be handeled considering a further interpretation of the control charts considering past samples. One reason, that neural networks do not reach this numbers of errors of type 1 or type 2 may be, that they also minimize explicitely false classification because of the objective function they use.

[35] More results are given in Stützle (1994).

[36] This holds in general for all other neural nets, that were trained to solve this problem.

5.3.5 Classification of the kind of change with neural networks

In section 5.3.4 the number of false classifications for the neural networks is smaller as with the combined control charts. This leads to the idea, that the neural networks may only be used as a classification procedure to classify the type of change, once such a change is indicated by a combined control chart. For the classification of the kind of change here only the last sample is considered.

To check the idea whether a combined control chart – neural network scheme could give some improvements with respect to wrong classifications, 2000 samples were generated that had only shifts of μ or σ. The shifts of μ were generated as uniformely distributed in $[\mu_0 + 0.5\sigma_0; \mu_0 + 2\sigma_0]$ and the shifts of σ were generated as uniformely distributed in $[1.2\sigma_0; 2\sigma_0]$. These samples were used to train a neural net to distinguish between these two different kinds of shifts. Then a test set of 2000 samples was first evaluated by a combined $\bar{x}$-s-chart. For the samples, for that an out of control situation was indicated, the kind of shift was classified by a neural network.

First the evaluation of the test data with a combined $\bar{x}$-s-chart is given in table 5.8.

evaluation with $\bar{x}$-s-chart	samples with shift of μ	samples with shift of σ
on both charts inside control limits	706	688
only outside the control limits on the $\bar{x}$-chart	256	64
only outside the control limits on the s-chart	9	240
Outside the control limits of both charts	7	30
sum	978	1022

Table 5.8 Results of the test data for a combined control chart. False classifications: 80.

The combined control chart would commit a total number of $80(= 64 + 9 + 7)$ false classifications. In table 5.9 now the results of the classification of the kind of shift with the neural network is given. Here only the samples that give an out of control signal are considered.

test of the classification with neural networks	neural net shows shift of σ	neural net shows shift of μ
on both charts outside control limits, actually it is a shift of σ	30	0
on both charts outside control limits, actually it is a shift of μ	2	5
outside the control limits on the $\bar{x}$ chart, actually it is a shift of σ	38	26
outside the control limits on the $\bar{x}$ chart, actually it is a shift of μ	4	252
outside the control limits on the s chart, actually it is a shift of σ	238	2
outside the control limits on the s chart, actually it is a shift of μ	5	4

Table 5.9 Results of the a posteriori test of the kind of classification with neural nets. False classifications: 39.

With the neural nets the number of false classifications could be clearly reduced. This seems to indicate, that by using the neural nets for only this special task, an improvement, that is a lower number of false classifications, can be achieved. This improvement may disappear, if past samples are included to interprete the control charts and to classify the possible kind of change that occurred to the process. Anyway, past samples could also have been considered as inputs for the neural networks to further improve the classification of the kind of shift. This was done e.g. in Guo, Dooley (1992) using as control charts a combined CUSUM-MOSUM scheme, that detects out of control situations faster then the here used $\bar{x}$-s-chart.

6 Summary and final comments

In this paper it was investigated whether the neural networks can imitate a certain model, like it is assumed for the control charts. Here this was done for the special case of a statistical test.

Generally it can be pointed out, that neural networks can be trained to imitate control charts. This holds for the $\bar{x}$-chart as well as for the s-chart. The neural nets can be trained to consider also the different kinds of error of type 1 and type 2. In the case of a comparison to the $\bar{x}$-chart, a neural network without hidden layer is enough to obtain similar results to the control chart. For the comparison to a s-chart the neural networks can reach similar rates of errors of type 1 and type 2, only when the ordered samples are given as inputs. If both kinds of changes are considered, the results of neural networks with respect to type 1 as well as type 2 errors are worse than those of a combined $\bar{x}$-s-chart. Only for the number of false classifications an improvement could be found. So it seems to be more reasonable to use neural networks for special tasks such as classification of the kind of shift. Such tasks are important with respect to the still increasing use of computers in industry and also in quality control. If one wants to evaluate the control charts automatically, one has to use algorithms to do this. Here neural networks are one possible classification procedure to accomplish such tasks.

The main disadvantage of the problems treated here is, that past samples are not considered. One possibility here would be to compare neural networks to control charts like the CUSUM-chart or the Shewhart charts with additional run rules that also use past samples. Considering the patterns of the past samples on the control charts, one could use neural networks to detect unnatural patterns on these like mixtures or a gradual increase of the mean or the process variability. Here neural networks can be used as one special method of pattern recognition. Another possibility is, that different patterns may indicate certain errors that occurred during the machine operation. So the detection of unnatural patterns on the control charts may help the diagnosis of possible faults. The information on the control charts could also be used in addition with sensor data of the machine operation to obtain a better evaluation of the product quality.

References

Bauer, P. and Hackl, P. *An Extension of the MOSUM Technique for Quality Control,* Technometrics, Vol. 22, February 1980, 1–7.

Battiti, R. *First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method,* Neural Computation, Vol. 4, 1992, 141–166.

Bratley, P.; Fox, B.L. and Schrage, L.E. *A Guide to Simulation,* Springer Verlag, New York, 1986.

Bing Cheng, Titterington, D.M. *Neural Networks: A Review from a Statistical Perspective,* Statistical Science, Vol. 9, 1994, 2–54.

Dooley, K.J., Kapoor, S.G. *An Enhanced Quality Evaluation System for Continuous Manufacturing Processes, Part 1: Theory,* Journal of Engineering for Industry, February 1990, 57–62.

Dooley, K.J., Kapoor, S.G. *An Enhanced Quality Evaluation System for Continuous Manufacturing Processes, Part 2: Application,* Journal of Engineering for Industry, February 1990, 63–68.

Duda, R. and Hart, P. *Pattern Classification and Scene Analysis,* John Wiley & Sons, New York, 1973.

Fahlman, S.E. and Lebiere, C. *The Casscade Correlation Learning Architecture,* in Touretzky, D.S. (editor): 'Advances in Neural Processing Systens II', Morgan Kaufmann, Denver, 1989. Gallant, S.I. *Neural Network Learning and Expert Systems,* MIT Press, Cambridge, 1993.

Guo, Y. and Dooley, K.J. *Identification of Change Structure in Statistical Process Control,* International Journal of Production Research, Vol.30, 1992, 1655–1669.

Hecht-Nielson, R. *Neurocomputing*, Addison-Wesley Publishing Company, 1990.

Hornik, K., Stinchcombe, M. and White, H. *Multilayer Feedforwared Networks are Universal Approximators,* Neural Networks, Vol.2, 1989, 359–366.

Hwarng, H.B. and Hubele, N.F. *Back Propagation Pattern Recognizers for X Control Charts: Methodology and Performance,* Computers & Industrial Engineering, Vol. 24, 1993, 219–235.

Kohonen, T. *Self-organisation and associative Memory,* Springer Verlag, Berlin, 1989.

Masson, E. and Wang, Y.-J. *Introduction to Computation and Learning in Artificial Neural Networks,* European Journal of Operations Research, Vol. 47, November 1989, 1–28.

Montgomery, D.C. *Introduction to Satistical Quality Control,* John Wiley & Sons, 1985.

Pham, D.T. and Oztemel, E. *Control Chart Pattern Recognition Using Learning Vector Quantization Networks* International Journal of Production Research, Vol.32, 1994, 721–729.

Pugh, G.A. *A Comparison of neural Networks to SPC Charts,* Computers & Industrial Engineering, Vol.21, 1991, 253–255

Reed, R. *Pruning Algorithms - A survey,* IEEE Transactions on Neural Networks, September 1993, 740–747.

Richard, M.D. and Lippman, M.P. *Neural Network Classifiers Estimate Bayesian a posteriori Probabilities,* Neural Computation, Vol.3, 1991, 461–483.

Rinne, H. and Mittag, H.J. *Statistische Methoden der Qualitätssicherung,* Carl Hanser Verlag, München, 1992.

Ripley, B.D. *Neural Networks and Related Methods for Classification,* Journal of the Royal Statistical Society B, Vol. 56, 1994, 409–456.

Rojas, R. *Theorie der neuronalen Netze,* Springer Verlag, Berlin, 1993.

Rummelhart, D.E. and McClelland, J.L. *Parallel Distributed Processing,* MIT Press, Massachusetts, 1986.

Ryan, T.P. *Statistical Methods for Quality Improvement,* John Wiley & Sons, New York, 1989.

Smith, A.E. *X-bar and R control Chart Interpretation Using Neural Computing,* International Journal of Production Research, Vol.32, 1994, 309–320.

Stützle, T. *Vergleich Neuronaler Netze zu Qualitätsregelkarten ohne Gedächtnis,* M.S. Thesis at the University of Karlsruhe, Faculty of Economics, Karlsruhe, 1994.

Wan, E.A. *Neural Network Classification: A Bayesian Interpretation,* IEEE Transactions on Neural Networks, Vol. 1, December 1990, 303–305

White, H. *Some Asymptotic Results for Learning in Single Hidden–Layer Feedforward Network Models,* Journal of the American Statistical Association, December 1989, 1003–1013.

White, H. *Learning in Artificial Neural Networks: A Statistical Perspective,* Neural Computation, Vol.1, 1989, 425–464.