Transactions on Information and Communications Technologies vol 20, © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

A novel firing rule for training Kohonen self-organising maps

D. T. Pham & A. B. Chan
Manufacturing Engineering Centre, School of Engineering, University of Wales Cardiff, P.O. Box 688, Queen's Buildings, Newport Road, Cardiff CF2 3TE, Great Britain
Email: *PhamDT@cardiff.ac.uk   ChanAB@cardiff.ac.uk*

Abstract

Statistical Process Control Charts can exhibit six principal types of patterns: Normal, Cyclic, Increasing Trend, Decreasing Trend, Upward Shift and Downward Shift. All except Normal patterns indicate abnormalities in the process that must be corrected. Accurate and speedy detection of such patterns is important to achieving tight control of the process and ensuring good product quality. This paper describes an implementation of the Kohonen self-organising map which employs the Euclidean Distance as the firing rule for control chart pattern recognition. First, the structure of the network is outlined and the equations which govern its dynamics are given. Then the learning mechanism of the network is explained. The effects of different combinations of network parameters on classification accuracy are discussed. A novel firing rule for the Kohonen self-organising map is proposed. This rule involves component-by-component comparison between the input pattern and the established class templates. When an input vector is presented, it is compared with the class templates in all the neurons in turn. The neuron containing the class template that best matches the input vector will subsequently fire. This approach is intended to enhance the generalisation capability and accuracy of the Kohonen self-organising map. The paper gives a comparison of the results obtained using the Euclidean Distance and the proposed firing rule.

1 Introduction

Control charts are employed in Statistical Process Control (SPC) to assess whether a process is functioning correctly. A control chart can exhibit six main types of patterns: *Normal*, *Cyclic*, *Increasing Trend*, *Decreasing Trend*, *Upward Shift* and *Downward Shift* [1-2]. Figures 1a-f depict these 6 pattern types. Correct identification of these patterns is important to achieving early detection of potential problems and maintaining the quality of the process under observation.

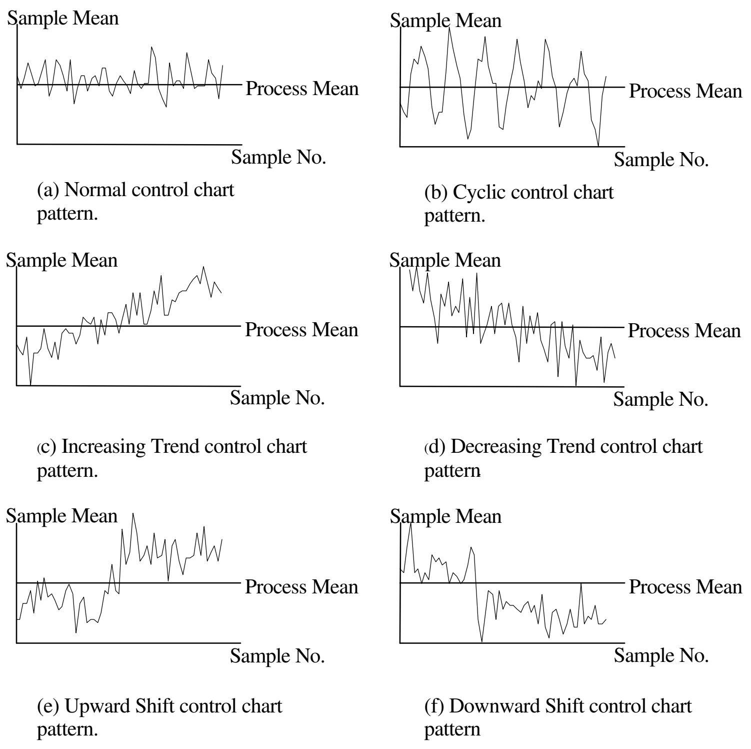


Figure 1: Main types of control chart patterns.

When the Kohonen self-organising map was first proposed, it was intended to be a simplified construction for explaining the ability of the human brain to form a model of its complicated environment. Despite this initial intention, the Kohonen self-organising map has since been employed to solve a variety of classification problems. This is due to the fact that in forming a miniature model of a much larger input space, a Kohonen self-organising map compresses similar feature vectors from the input space into one single neuron. This clustering capability qualifies the Kohonen self-organising map as a classifier as well as a memory model.

In this paper, the results of applying the Kohonen self-organising map to control chart pattern recognition are detailed. Two different firing rules have been used in the training of the maps created for this specific problem. They are the the Euclidean Distance firing rule and a novel firing rule which involves comparisons of corresponding individual components in the input vector and the class templates. The ways in which these firing rules work are explained and a comparison of their performances is made. The optimal network parameter combination for each of the two firing rules is determined from the simulation results.

2 Training and testing of a Kohonen self-organising map

It is apparent that memory plays an important role in human intelligence. Most of the intellectual activities performed by man like reasoning and planning do involve the process of extracting useful and relevant information from memory. If artificial neural networks were to achieve brain-like functionality, the question as to how a memory unit can be implemented in the form of an artificial neural network must be addressed. The Kohonen self-organising map was proposed as a simplified answer. It does not purport to explain all the complexities associated with human memory. Rather, it is meant to demonstrate that it is possible to devise an artificial memory model which carries some important characteristics in common with human memory.

2.1 Background

The Kohonen self-organising map is inspired by the phenomenon that 2-dimensional somatosensory maps are formed on the surface of the human brain in response to senses from different parts of the human body and that different parts of the visual field are projected onto a surface in the visual cortex [3]. The major capability of the Kohonen self-organising map is that it can self-organise to preserve the topology of the training data set [4]. Usually the training data set of a Kohonen self-organising map has much more data than contained in the map. In effect, after the training has stabilised, the Kohonen self-organising map becomes a miniature model of its much larger training data set.

Since the Kohonen self-organising map requires no external control signals, only the input signals, it is regarded as an unsupervised neural network paradigm.

The topology-preserving ability of the Kohonen self-organising map is achieved by implementing a training procedure where the neurons residing close to the firing neuron are updated as well as the firing neuron. Gradually, the neurons in a neighbourhood will develop similar weights. Experiments performed by Kohonen have shown that this simple updating scheme does lead to topology preservation and, in most cases, stability [5].

The updating of neurons close to the firing neuron follows the pattern of the Mexican hat function (shown in Figure 2) which can be observed in biological neural networks [5][6]. Figure 2 depicts that the magnitude of the stimulation that a firing neuron sends to its neighbours reduces as the physical distance between the neighbouring neuron and the firing neuron grows. Within a distance range, the stimulation actually is an inhibitory one. Computer simulations showed that applying an input constantly to a network implementing the Mexican hat function will cause the output of a particular neuron in the network to peak above the others after a certain period [5][6]. This demonstrates the ability of the network to produce a localised response when a particular input is applied.

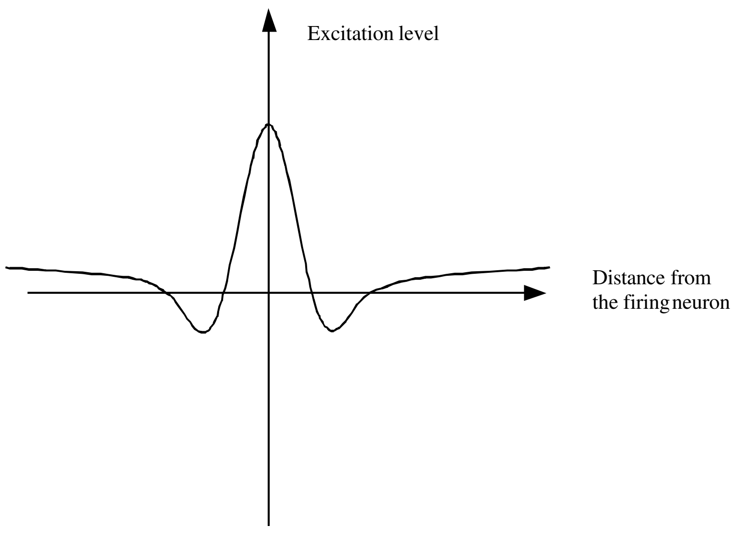


Figure 2: The Mexican hat function.

As mentioned above, although the Kohonen self-organising map was designed as a simple tool to form a miniature topology-preserving model of its training data set, it is possible to apply it as a classifier in some cases. This is because, in the course of training, similar input vectors are clustered into the same neurons. However, the classification capability of the Kohonen self-organising map is coincidental rather than intentional. Apart from classification, this clustering property is also sometimes utilised for the purpose of data compression.

2.2 Training procedure

In this section, the firing rule used will not be defined explicitly. It will be denoted by a general function. This is because the main purpose of this section (section 2) is to outline the training and test procedures used in the computer simulations reported in this paper. The two different specific firing rules will be introduced in sections 3 and 4.

Because all the simulated maps discussed in this paper are 2-dimensional, the training and test procedures described here apply only to 2-dimensional maps which are most commonly adopted. A typical 2-dimensional Kohonen self-organising map is like the one shown in Figure 3. The number of rows is denoted by M and the number of columns by N. Any neuron in the map has co-ordinates, (i, j), with the top left corner defined as (0,0). Each neuron contains a weight vector $\underline{\mathbf{w}}_{ij}$ which has the same dimensionality as any input vector $\underline{\mathbf{x}}$. For example, if $\underline{\mathbf{x}} = [x_1, x_2, x_3, \cdots\cdots, x_k]^T$ then $\underline{\mathbf{w}}_{ij} = [w_1, w_2, w_3, \cdots\cdots, w_k]^T$, i.e. both of them have k elements. Before training begins, all the neuron weights are initialised with random numbers. Then, each weight vector will be normalised using the following equation:

$$\hat{\underline{\mathbf{w}}}_{ij} = \underline{\mathbf{w}}_{ij} / \left\| \underline{\mathbf{w}}_{ij} \right\| \qquad (1)$$

where

$$\|\underline{\mathbf{w}}_{ij}\| = \sqrt{\sum_{i=1}^{k} w_i^2} \qquad (2)$$

The same normalisation process will also be applied to all training vectors as follows:

$$\hat{\underline{\mathbf{x}}} = \underline{\mathbf{x}} / \|\underline{\mathbf{x}}\| \qquad (3)$$

$$\|\underline{\mathbf{x}}\| = \sqrt{\sum_{i=1}^{k} x_i^2} \qquad (4)$$

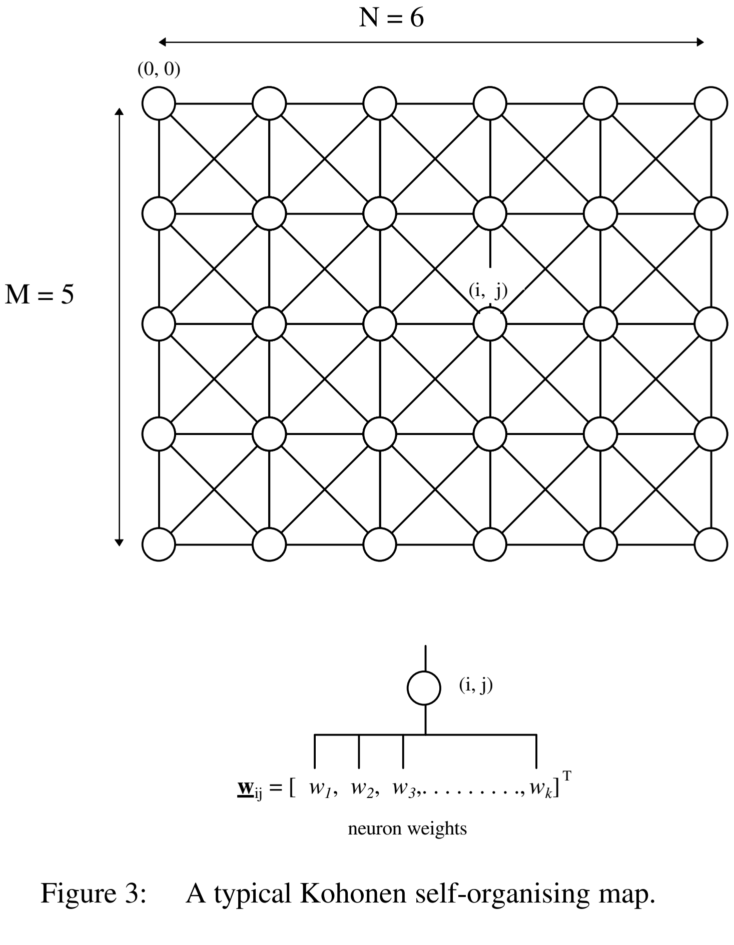

N = 6
(0, 0)
M = 5
(i, j)
(i, j)
$\underline{\mathbf{w}}_{ij} = [\, w_1,\; w_2,\; w_3, \ldots\ldots\ldots, w_k]^T$
neuron weights

Figure 3: A typical Kohonen self-organising map.

When an input vector $\underline{\hat{\mathbf{x}}}$ is presented, a firing rule will decide which neuron in the map should fire. This firing rule can be represented by a general function $\mathbf{F}(\underline{\hat{\mathbf{x}}})$ which returns the co-ordinates, (i, j), of the firing neuron. This firing neuron and the other neurons which lie within its specified neighbourhood will be updated with the following equation:

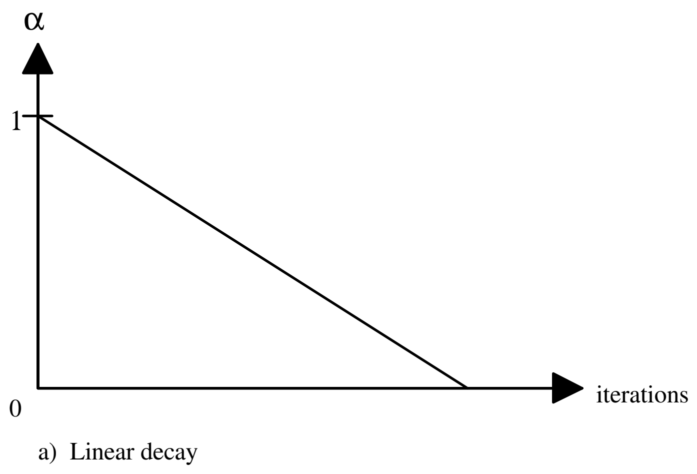
$$\underline{\hat{\mathbf{w}}}_{ij}(t+1) = \underline{\hat{\mathbf{w}}}_{ij}(t) + \alpha[\underline{\hat{\mathbf{x}}} - \underline{\hat{\mathbf{w}}}_{ij}(t)] \qquad (5)$$

The updating parameter α determines the magnitude by which the weight vector is modified towards the input vector $\underline{\hat{\mathbf{x}}}$. For practical reasons, α should be smaller than 1. If α equals 1, the weights in the neurons will oscillate excessively, being set to any new vector presented; on the other hand, if α is larger than 1, it will cause the weights of the firing neuron to "over-learn" the input vector more than necessary in order for them to cover the difference. This could result in the firing neuron losing information acquired from input vectors that previously fired it. It is apparent that α must be larger than 0 if any training is going to take effect upon the weights in the neurons. Thus, it can be expressed that

$$1 > \alpha > 0 \qquad (6)$$

To ensure stability in a Kohonen self-organising map, it is essential that α decreases during training. This follows the logic that the more iterations the map has undergone, the less the neuron weights need modifying. One iteration is defined to be one complete presentation of the training data set to the map. The total number of iterations is determined by the user before the training commences.

There are two common ways in which α reduces: it can decrease either linearly or exponentially with respect to the number of iterations that have taken place. These two ways are illustrated in Figure 4. In both schemes, α starts off with a value close to 1 and gradually decreases to 0. The interpretation of this is that the neuron weights are updated drastically initially and updating activities become minimal towards the end of training, thus ensuring the overall stability of the map. Both schemes have been adopted in this work.



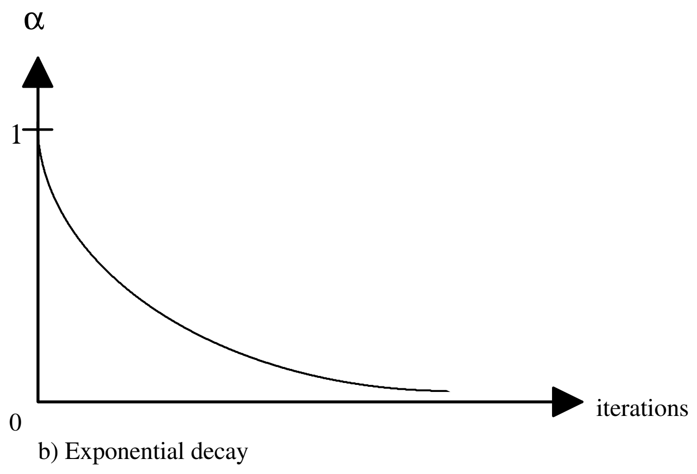


Figure 4: Two types of α decays.

Apart from neuron (i, j) returned by the firing rule $\mathbf{F}(\hat{\underline{\mathbf{x}}})$, the other neurons, which lie within a specific neighbourhood of (i, j), will also be updated according to equation (5). The simulated Kohonen self-organising maps in this work were trained with two different types of neighbourhood schemes: a fixed 3×3 neighbourhood throughout the training session or a neighbourhood which would shrink in the course of training starting with a size of 5×5. Figure 5 illustrates these two schemes. For the second neighbourhood scheme (see Figure 5b), throughout the first quarter of the user-specified iterations, the neighbourhood size is 5×5. Then, for the middle two-quarters of the iterations, the neighbourhood size is reduced to 3×3. Eventually, for the last quarter of the iterations, it becomes 1×1 (that is, it consists of only the firing neuron).

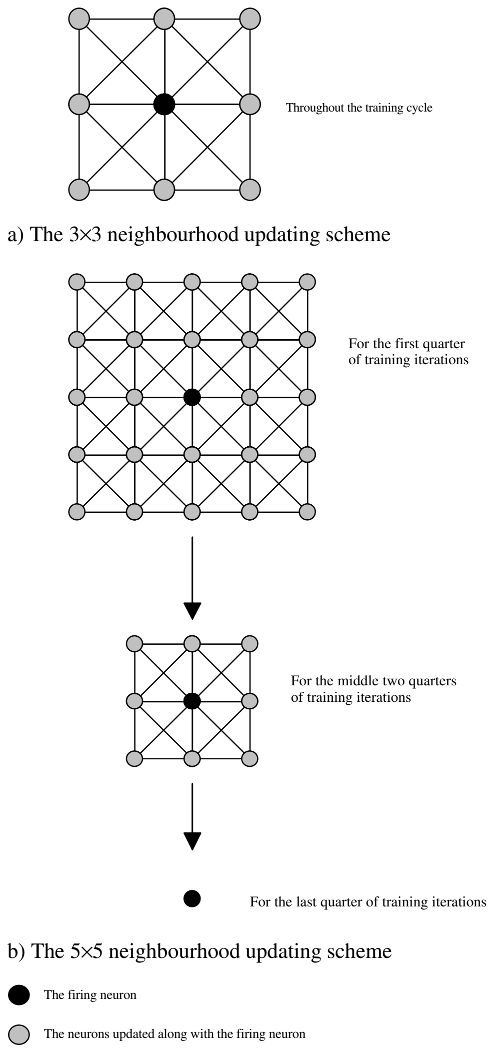

Throughout the training cycle
a) The 3×3 neighbourhood updating scheme
For the first quarter of training iterations
For the middle two quarters of training iterations
For the last quarter of training iterations
b) The 5×5 neighbourhood updating scheme
The firing neuron
The neurons updated along with the firing neuron

Figure 5: Two different neighbourhood updating schemes.

2.3 Test procedure

When training is accomplished, the resulting Kohonen self-organising map should have achieved the same topological order as that in the training data set. Before it can be used as a classifier for the test data set, its neurons have to be labelled to represent different classes. This is done using the training data set.

The way in which a neuron is labelled is as follows. Suppose that there are only 3 classes for both the training and the test data sets and that they are named Class A, Class B and Class C. To decide which class is represented by a neuron (i, j) in the map, all the training vectors which have been clustered into this particular neuron will have to be considered. The class which has the majority among all the training vectors classified into neuron (i, j) will become the class label for it. For example, if 70% of the training vectors classified into neuron (i, j) belong to Class A, 20% to Class B and 10% to Class C, then neuron (i, j) will represent Class A. Consequently, all the vectors not belonging to Class A but classified into neuron (i, j) will be regarded as mis-classified. Once all the neurons in the map have been labelled in this manner, testing can proceed. Any test vector belonging to Class A that fires neuron (i, j) will be counted as a correct classification; any other vectors belonging to other classes that trigger this neuron will be regarded as mis-classifications.

3 Euclidean Distance firing rule

3.1 Firing rule

Euclidean Distance is a measure of the physical distance between two position vectors in n-dimensional space. It is widely employed in clustering algorithms for unsupervised learning classification techniques. In the case of a Kohonen self-organising map, when an input vector $\underline{\hat{x}}$ = [x_1, x_2, x_3,· · · · ·, x_k] is presented, its Euclidean Distance to the weight vector $\underline{\hat{w}}_{ij}$ = [w_1, w_2, w_3, · · · · · ·, w_k] of neuron (i, j) can be calculated using the following equation:

$$\mathbf{Eucli}(\underline{\hat{\mathbf{X}}}, \underline{\hat{\mathbf{W}}}_{ij}) = \sqrt{\sum_{i=1}^{k} \left(x_i - w_i\right)^2} \qquad (7)$$

When an input vector $\underline{\hat{x}}$ is presented, the same operation will be performed with the weight vectors of all neurons in the Kohonen self-organising map. The neuron having the shortest Euclidean Distance from the input vector will be the firing neuron. It and its neighbours will be updated in the training process in accordance with equation (5).

3.2 Classification results

Table 1 records the simulation results for Kohonen self-organising maps trained with the Euclidean Distance firing rule. Map 19 had the best performance (96.42%). It was a 10×10 map trained with an exponentially decaying α and a shrinking neighbourhood. The total number of iterations specified was 10. The best among the 7×7 maps was Map 10 (91.62%). This also used an exponentially decaying α and took 10 iterations to train. However, for Map 10 a fixed neighbourhood was employed rather than a shrinking one. The difference in performance between Map 10 and Map 19 is 4.8%.

The overall average accuracy of the 7×7 maps was 85.26% and that of the 10×10 maps was 93.02%. This considerable difference can be attributed to the particularly poor performance of Map 3 which reduced the overall average accuracy of the 7×7 maps.

The maps trained with a linearly decaying α had an overall average accuracy of 88.16% compared with the 90.12% for those trained with an exponentially decaying α.

The fixed neighbourhood produced an overall average accuracy of 89.86%. On the other hand, the shrinking neighbourhood scheme gave 88.43%. The difference in this case is not very significant.

Those maps trained in 10 iterations did in general perform better than those trained in 30 and 50 iterations. The respective overall accuracies for these different numbers of iterations are 90.68%, 88.30% and 88.45%.

4 A novel firing rule

4.1 Firing rule

The proposed firing rule involves an element-by-element comparison between the input vector and the weight vectors of a Kohonen self-organising map. When an input vector $\underline{\mathbf{x}} = [x_1, x_2, x_3, \cdot\cdot, x_i, \cdot\cdot\cdot, x_k]^T$ is presented to a Kohonen self-organising network, it will be compared with the weight vector $\underline{\mathbf{W}}_{\mathbf{ij}} = [w_1, w_2, w_3, \cdot\cdot, w_i, \cdot\cdot\cdot\cdot, w_k]^T$ of neuron (i, j) in a component-to-component manner. For every corresponding pair of components (x_i, w_i) the following decision concerning x_i will be made:

IF

$$\frac{|x_i - w_i|}{w_i} \leq \beta$$

THEN

x_i **IS REGARDED AS A SUCCESSFUL COMPONENT**

ELSE

x_i **IS REGARDED AS A FAILED COMPONENT**

β, a user-supplied coefficient between 0 and 1, determines the range within which x_i is permitted to vary from w_i before it is considered to be a failed component. For example, if β is set as 0.2 and w_i is equal to 1, x_i can be any value between 0.8 to 1.2 (inclusive) without being rejected as a failed component. After all the components in $\underline{\mathbf{x}}$ have been checked according to this procedure, the total number of successful components in $\underline{\mathbf{x}}$ will be recorded for neuron (i, j). The same procedure is performed on all the neurons in the Kohonen self-organising map. Finally, the neuron with the maximum number of successful components in $\underline{\mathbf{x}}$ will be the firing neuron.

4.2 Classification results

Table 2 shows the simulation results for Kohonen self-organising maps trained with the proposed firing rule. For each map, β was set to 0.2. The best classification accuracy was achieved by Map 23 which had dimensions 10×10. It was trained with an exponentially decreasing α and a fixed neighbourhood scheme in 30 iterations. The average accuracy, 98.15%, of Map 23 is significantly better than the best performing 7×7 map, Map 10, which had an average accuracy of 92.42%.

The overall average accuracy of the 7×7 maps, 84.48%, is considerably lower than that of the 10×10 maps (93.74%). In terms of the updating parameter α, the linearly decaying strategy appears to be inferior to the exponentially decaying alternative because the overall average accuracy of all the maps adopting a linearly decaying α is 86.70% compared to 91.53% of those using the exponentially decaying α. The fixed neighbourhood scheme seems to be as efficient as the shrinking neighbourhood scheme since their average accuracies are nearly equal (88.66% and 89.56% respectively). The overall average accuracy of all the maps trained in 10 iterations is 90.50%. This is higher than the accuracy of those trained in 30 iterations (88.30%) and 50 iterations (88.54%).

Tables 3 and 4 are included to show the effect on the network performance when β is altered. Theoretically, high accuracy can be produced by keeping β at a low value. This is because a low valued β means a stricter classification criterion. The input pattern must resemble the class template to a large extent before it is classified as a member of that class. This assumption is demonstrated by the results in Table 4. The best performing map in Table 3, where β=0.1, has an average accuracy of 98.53% which is higher than that of Map 23 in Table 2 (98.15%). Table 4 shows that when β was set to 0.3, which meant each component was allowed to vary within the range of ± 30% of its counterpart in the class template, the classification criterion became too lax. This resulted in generally low classification accuracy.

5 Conclusions

In terms of the best performing map, Map 23 in Table 2, trained with the proposed firing rule with β = 0.2, was better than the maps trained with the Euclidean Distance firing rule.

Whether α followed a linear decay function or an exponential decay function was of limited importance with the Euclidean Distance firing rule. However, the overall average accuracy was increased by almost 5% when an exponential decay function was adopted with the proposed firing rule.

As for neighbourhood schemes, having a fixed neighbourhood size throughout or having a shrinking one did not affect the average overall accuracy greatly. It can be reasonably assumed that for the control chart pattern recognition problem, these two neighbourhood updating schemes would always yield very similar results.

From the results of using the two different firing rules, it can be claimed that for the control chart pattern recognition problem, it is not helpful to use a large number of iterations. The best performing maps, trained with the Euclidean Distance firing rule were trained in 10 iterations. Although the best performing map trained with the proposed firing rule required 30 iterations, the second-best map (Map 19 in Table 2), having been trained in 10 iterations, only had 0.14% less accuracy.

In this paper, classification results have shown that the proposed firing rule is superior to the Euclidean Distance firing rule. For good classification accuracy, it is necessary to choose β with a suitably low value.

References

[1] Pham, D. T. and Oztemel, E. (1996). *Intelligent Quality Systems*. Springer-Verlag, London.

[2] Pham, D. T. and Chan, A. B. (1997). *Control Chart Pattern Recognition Using a New Type of Self-Organising Neural Network*. Technical Report, University of Wales, Cardiff, Britain.

[3] Knudsen, E. I., du Lac, S. and Esterly, S. D. (1987). "Computational maps in the brain." Annual Review of Neuroscienve, 10:41-65.

[4] Kohonen, T. (1987). "Representation of sensory information in self-organising feature maps, and relation of these maps to distributed memory networks." *Optical and Hybrid Computing*. Szu (Ed.), SPIE Vol. 634, Bellingham.

[5] Kohonen, T. (1982). "Self-organised formation of topologically correct feature maps." *Biological Cybernetics* 43, 59-69.

[6] Kohonen, T. (1988). "An introduction to neural computing," *Neural Networks*, 1, 3-16.

Table 1. The Euclidean Distance firing rule simulation results.

	Map dimensions		Updating parameter		Neighbourhood option		Number of	Classification accuracy		
Map	x-dim	y-dim	linear	exponential	fixed	shrinking	iterations	trn.dat (%)	tes.dat (%)	average (%)
1		7	♦			♦	10	83.22	84.24	83.73
2	7	7	♦			♦	30	79.40	84.24	81.82
3	7	7	♦			♦	50	79.67	78.94	79.31
4	7	7	♦		♦		10	86.23	87.27	86.75
5	7	7	♦		♦		30	81.58	90.30	85.94
6	7	7	♦		♦		50	82.95	81.97	82.46
7	7	7		♦		♦	10	87.49	87.42	87.46
8	7	7		♦		♦	30	84.75	86.67	85.71
9	7	7		♦		♦	50	88.58	85.00	86.79
10	7	7		♦	♦		10	92.96	90.27	91.62
11	7	7		♦	♦		30	84.24	83.48	83.86
12	7	7		♦	♦		50	88.85	86.52	87.69
13	10	10	♦			♦	10	94.25	95.87	95.06
14	10	10	♦			♦	30	91.15	94.85	93.00
15	10	10	♦			♦	50	93.33	92.58	92.96
16	10	10	♦		♦		10	90.60	92.12	91.36
17	10	10	♦		♦		30	94.15	95.61	94.88
18	10	10	♦		♦		50	89.51	91.82	90.67
19	10	10		♦		♦	10	95.97	96.87	96.42
20	10	10		♦		♦	30	86.50	85.01	85.76
21	10	10		♦		♦	50	90.60	95.64	93.12
22	10	10		♦	♦		10	92.98	93.15	93.07
23	10	10		♦	♦		30	95.25	95.61	95.43
24	10	10		♦	♦		50	95.79	93.33	94.56

x-dim: the x dimension of the Kohonen self-organising map
y_dim: the y dimension of the Kohonen self-organising map
trn.dat: classification results using the training data set as inputs
tes.dat: classification results using the test data set as inputs
average: the average classification results using the training and test data set
training data set: 366 vectors; test data set: 132 vectors

Table 2. Simulation results for the proposed firing rule when β=0.2.

β = 0.2

	Map dimensions		Updating parameter		Neighbourhood option		Number of	Classification accuracy		
Map	x-dim	y-dim	linear	exponential	fixed	shrinking	iterations	trn.dat (%)	tes.dat (%)	average (%)
1	7	7	♦			♦	10	80.60	82.58	81.59
2	7	7	♦			♦	30	82.24	81.82	82.03
3	7	7	♦			♦	50	78.42	84.09	81.26
4	7	7	♦		♦		10	82.79	83.33	83.04
5	7	7	♦		♦		30	78.96	80.30	79.63
6	7	7	♦		♦		50	80.60	78.03	79.32
7	7	7		♦		♦	10	91.26	92.42	91.84
8	7	7		♦		♦	30	82.79	81.06	81.93
9	7	7		♦		♦	50	84.43	87.12	85.78
10	7	7		♦	♦		10	93.17	91.67	92.42
11	7	7		♦	♦		30	84.75	85.15	84.95
12	7	7		♦	♦		50	89.07	90.91	89.99
13	10	10	♦			♦	10	96.17	93.94	95.06
14	10	10	♦			♦	30	92.62	92.94	92.78
15	10	10	♦			♦	50	92.35	93.94	93.15
16	10	10	♦		♦		10	89.62	92.42	91.02
17	10	10	♦		♦		30	91.26	91.67	91.47
18	10	10	♦		♦		50	89.89	90.15	90.02
19	10	10		♦		♦	10	97.54	98.48	98.01
20	10	10		♦		♦	30	96.21	94.70	95.46
21	10	10		♦		♦	50	95.36	96.21	95.79
22	10	10		♦	♦		10	90.77	91.21	90.99
23	10	10		♦	♦		30	97.81	98.48	98.15
24	10	10		♦	♦		50	92.44	93.60	93.02

x-dim: the x dimension of the Kohonen self-organising map
y_dim: the y dimension of the Kohonen self-organising map
trn.dat: classification results using the training data set as inputs
tes.dat: classification results using the test data set as inputs
average: the average classification results using the training and test data sets
training data set: 366 vectors; test data set: 132 vectors

Table 3. Simulations results for the proposed firing rule when β=0.1.

$\beta = 0.1$

	Map dimensions		Updating parameter		Neighbourhood option		Number of	Classification accuracy		
Map	x-dim	y-dim	linear	exponentia	fixed	shrinking	iterations	trn.dat (%)	tes.dat (%)	average (%)
1	7	7	♦			♦	10	84.97	81.06	83.01
2	7	7	♦			♦	30	84.97	88.64	86.81
3	7	7	♦			♦	50	84.97	86.36	85.67
4	7	7	♦		♦		10	86.89	92.42	89.66
5	7	7	♦		♦		30	82.79	84.07	83.43
6	7	7	♦		♦		50	88.80	84.85	86.83
7	7	7		♦		♦	10	90.71	92.42	91.57
8	7	7		♦		♦	30	87.98	89.39	88.69
9	7	7		♦		♦	50	84.70	87.12	85.91
10	7	7		♦	♦		10	93.99	93.18	93.59
11	7	7		♦	♦		30	88.52	87.88	88.20
12	7	7		♦	♦		50	94.54	93.94	94.24
13	10	10	♦			♦	10	95.90	96.21	96.06
14	10	10	♦			♦	30	96.72	93.84	95.28
15	10	10	♦			♦	50	93.17	96.21	94.69
16	10	10	♦		♦		10	94.81	97.73	96.27
17	10	10	♦		♦		30	93.17	96.21	94.69
18	10	10	♦		♦		50	93.72	94.70	94.21
19	10	10		♦		♦	10	97.81	99.24	98.53
20	10	10		♦		♦	30	93.99	94.70	94.35
21	10	10		♦		♦	50	92.35	93.18	92.77
22	10	10		♦	♦		10	96.45	96.97	96.71
23	10	10		♦	♦		30	95.90	96.97	96.44
24	10	10		♦	♦		50	91.67	93.22	92.45

x-dim: the x dimension of the Kohonen self-organising map
y_dim: the y dimension of the Kohonen self-organising map
trn.dat: classification results using the training data set as inputs
tes.dat: classification results using the test data set as inputs
average: the average classification results using the training and test data sets
training data set: 366 vectors; test data set: 132 vectors

Table 4. Simulation results for the proposed firing rule when β= 0.3

β = 0.3

	Map dimensions		Updating parameter		Neighbourhood option		Number of	Classification accuracy		
Map	x-dim	y-dim	linear	exponent	fixed	shrinking	iterations	trn.dat (%)	tes.dat (%)	average (%)
1	7	7	♦			♦	10	82.13	81.21	81.67
2	7	7	♦			♦	30	79.13	75.15	77.14
3	7	7	♦			♦	50	71.75	78.18	74.97
4	7	7	♦		♦		10	72.57	73.64	73.11
5	7	7	♦		♦		30	75.30	76.67	75.99
6	7	7	♦		♦		50	71.75	72.88	72.32
7	7	7		♦		♦	10	77.21	78.94	78.08
8	7	7		♦		♦	30	76.01	73.79	74.90
9	7	7		♦		♦	50	72.02	74.39	73.21
10	7	7		♦	♦		10	74.10	70.76	72.43
11	7	7		♦	♦		30	74.48	71.36	72.92
12	7	7		♦	♦		50	76.93	72.88	74.91
13	10	10	♦			♦	10	78.31	78.94	78.63
14	10	10	♦			♦	30	82.13	79.70	80.92
15	10	10	♦			♦	50	81.04	82.73	81.89
16	10	10	♦		♦		10	84.37	88.33	86.35
17	10	10	♦		♦		30	85.03	83.64	84.34
18	10	10	♦		♦		50	83.50	81.20	82.35
19	10	10		♦		♦	10	88.03	87.42	87.73
20	10	10		♦		♦	30	82.57	84.39	83.48
21	10	10		♦		♦	50	83.66	84.39	84.03
22	10	10		♦	♦		10	89.84	82.12	85.98
23	10	10		♦	♦		30	86.67	87.42	87.04
24	10	10		♦	♦		50	84.47	82.40	83.44

x-dim: the x dimension of the Kohonen self-organising map
y_dim: the y dimension of the Kohonen self-organising map
trn.dat: classification results using the training data set as inputs
tes.dat: classification results using the test data set as inputs
average: the average classification results using the training and test data sets
training data set: 366 vectors; test data set: 132 vectors