Transactions on Information and Communications Technologies vol 16, © 1996 WIT Press, www.witpress.com, ISSN 1743-3517

The effect of learning rate to an unsupervised neural network for inspection process

C.K. Lee, C.H. Chung

Department of Electronic Engineering, Hong Kong Polytechnic, Hung Hom, Hong Kong.

Email:encklee@hkpucc.polyu.edu.hk

Abstract

In this paper, we shall investigate the effect of *learning rate* to an unsupervised neural network when applied to an inspection process. The network we use is the *Learning by Experience* (LBE)[1]. Here, we analyse the learning rate for classifying parts which are produced from a machine with a varying parameter. Experiment results using IC leadframes are included.

1. Introduction

In the manufacturing process of any kinds of products, there are some specifications which the produced parts have to follow. Hence, we shall base on some kinds of distributions to classify which product is good or defective. Usually, two parameters are used to describe these distributions: the mean and the standard deviation. In order to distinguish defective parts from the good ones, one must set the tolerance level as an acceptance criterion. The tolerance will be determined from the standard deviation of the distribution, and the mean is the desired output of the process. For example, in a factory that produces screws, the machine may produce screws with a mean length of 3mm. with a tolerance as ±0.1mm. Therefore, screws of length ranging from 2.9mm to 3.1 mm will be classified as good ones; while others outside this range will be classified as defective. If in one day, the settings or parameters of the machine have been changed, and the screws produced drift to a mean length of 3.05mm. It seems to be no problem at all since this value is still within the tolerance. In fact, if the mean of the machine has drifted to 3.05mm, the chance of producing faulty parts will be increased, provided that the standard deviation of the machine remains the same. Here, we propose to apply the learning process of an unsupervised neural network to model this situation.

In the learning process of an unsupervised neural network, memory is used to store the learned patterns inside the network. Commonly, it is represented by the weight vector to the output neuron. Hence, this weight vector will act as the centre of the cluster, and the boundary of this cluster will determine the class that is denoted by that particular output neuron[2]. In one way, the radius of the cluster comes from the threshold of the network. Throughout the learning process, the weight vector to the output neuron will be updated. The level of learning at each iteration is governed by the learning rate. A large learning rate means that the internal weights adapt to the input pattern quickly. That is, a learning rate equal to 1 will change the internal weight equal to the new pattern just learned while a zero learning rate will not change it at all. Therefore, the learning capability of the network is minimized. Hence, the learning rate will be set between 0 and 1. As the internal weight vector being updated, the cluster centre of the output neuron drifts accordingly, and the extent of drifting is proportional to the learning rate. This phenomenon can be explained graphically as follows. Fig. 2 shows an output neuron centred at W_0 with the cluster boundary C_0, which is the radius equal to the threshold. A new input pattern X_1 which lies within the cluster boundary, will be classified by that neuron, and the internal weight of that neuron will be updated. The new cluster centre is then located at W_1, which is represented by the dashed circle. The change of the distance, D, will then depend on the learning rate. Therefore, the centre of the output neuron drifts as the network learns. Thus for an inspection process, the centre will act as the mean of the length of the screw, and the threshold will behave as the tolerance.

As the network continues to learn, the internal weights will drift continuously. A large learning rate will change them rapidly. If the cluster centre is too close to a pattern that is originally belonging to another cluster, then after a rapid change, this particular pattern will be grouped to it. Therefore, it may increase the chance of classifying defective parts as good ones. On the other hand, a small learning rate will slow down the learning process, which in turn will decrease the learning capability of the network. We shall demonstrate this effect in Section 3.

The layout of the paper is as follows. In Section 2, we shall describe the structure of the LBE network and its learning algorithm. In Section 3, we will investigate the effect of the learning rate to the LBE network. In Section 4, we shall describe the fuzzy inference engine and the fuzzy variables used. In Section 5, the application of the LBE network to inspect industrial images (IC leadframes) will be presented. A conclusion will appear in Section 6.

2. Learning By Experience (LBE) Network

2.1 Structure of the LBE network

Unlike other unsupervised learning neural networks, the main idea of LBE network is not the way of updating the weights of the winner, but instead, is to use memory to let output neurons know whether the applied input pattern belonging to the stored patterns or not. In fact, it employs the competitive learning as the updating rule for the winner.

First, a memory is added to each output neuron to store the strength, SS_j. This strength is used to compare the current input pattern with the stored patterns. As analog signals are applied to the network, the strength function is represented by the mean square error between the input pattern and the weight. Therefore, the lower the strength, the closer is the 2 compared patterns. The structure of the LBE network is shown in Fig. 1.

It is a two-layer, feed forward type network. The lower layer contains the input neurons and the upper layer consists of the output neurons. The connections among the lower layer and the upper layer store the weights, W_{ij}, which in turn store the learnt patterns. The connections within the upper layer indicate the output neurons, each receiving SS_j as the inhibitory input, from all the other output neurons. Also, each output neuron has a memory to store its experience. Once the output neuron has become the winner, the current strength SS_j will be stored into its memory. This memory is used to determine whether any applied pattern is associated to this output neuron.

When an input pattern is applied to the network, X_i will be equal to P_i, the input pattern. Then S_j will be built up through the network connections. When S_j is obtained, it will be compared with the strength stored in its internal memory M_j to work out SS_j. Finally the output neuron which has the minimum strength will become the winner, and its internal memory and the weights will be updated only.

2.2 The Learning Algorithm

2.2.1 Step 1 - the initialization phase

All weights, W_{ij}, are set to 0. All memory locations, M_j, are set to $(N + 1)$, where N is the number of output neurons. The *learning rate* can be set to any value from 0 to 1. The *mean square error* can be set to any value from 0 to 1. This is the maximum allowable error between two input patterns to be classified to the same class.

2.2.2 Step 2 - the weight updating phase

Apply the pattern to the input neuron X_i of the network, then S_j is calculated as

$$S_j = \sum_{i=0}^{M} \frac{(W_{ij} - X_i)^2}{N} \tag{2}$$

where i = index of the input neurons,

j = index of the output neurons,

M = the no. of input neurons.

The final strength, SS_j, for competition within the upper layer is obtained as

$$SS_j = \begin{cases} S_j & \text{if } S_j \le M_j + \mathit{mean\ square\ error} \\ \infty & \mathit{otherwise} \end{cases} \tag{3}$$

The winner is searched among all the output neurons, and it is the one which has the minimum strength. Two cases will occur. They are: (1) *normal case* -- the winner with SS_j not equal to ∞. This means the input pattern is recognized; (2) *memory full case* -- the winner with SS_j equal to ∞. This means that there is no pattern stored in this network matched with this pattern and the network has no more unused output neuron to store this new pattern, which is a memory catastrophe. For normal case, the internal memory and the weights of the winner are updated as:

If $M_j = (N + 1)$ then

$$W_{ij} = X_i, \tag{4}$$

else

$$W_{ij} = W_{ij} + L \times (X_i - W_{ij}) \tag{5}$$

where j = index of the winner

L = learning rate

Next, we compute the S_j again and use these new weights, W_{ij}, to update the memory M_j as

$$M_j = S_j \tag{6}$$

and the output is given as

$$Y_j = \begin{cases} 1 & \text{if } j = \mathit{index\ of\ winner} \\ 0 & \mathit{otherwise} \end{cases} \tag{7}$$

3. Effect of the Learning Rate

In the LBE network, each output neuron has a memory to indicate its free state or the Euclidean distance of the learned weight to the input vector. The advantage of using this memory is that all the output neurons can be fully utilized. Also, there is a single threshold value to determine an input vector to the current class, a new class or the memory full. Also a large value in the learning rate will guide the internal weights to adapt to the input vector quickly. While a zero value keeps the weights at the current value, independent of the input vector. During operation, if the created classes of input vectors have adequate separations, there is no wrong classification. However, it is not always the case. In Fig. 2 (for simplicity, two dimensional input vectors are used), an existing class C_0 is trained. The internal weight is at W_0 with a set threshold. An input pattern X_1 which the Euclidean distance from W_0, D, is less than the threshold of the class C_0 appears. Hence, X_1 is grouped into class C_0 and the internal weight after learning becomes W_1. The dotted line shows the new boundary of class C_0.

Suppose that we continue to apply an input pattern with the Euclidean distance D from the current weight. The boundary of the class is drifted away from the original boundary but the class label remains unchanged as C_0. When the drifting distance is greater than two times of the threshold, none of the patterns in the original class can be recognized. Input vectors within the boundary of the original class C_0 will be classified as a new class if an unused output neuron is available. Therefore, a wrong decision is made; and the previously learned pattern is erased.

There is a potential hazard for the existing memories, if its content lies within the pattern change path. This effect must be prevented from occurrence. In the following, we are going to discuss the effect of the threshold to the LBE network.

3.1 Threshold

Based on the learning algorithm, a fixed Euclidean distance between the input vectors and a fixed threshold, the weights of the winner can be expressed as

$$W_{ij} = \frac{[(1-L)^n - (1-nL)]D}{L} + D \tag{8}$$

Memory wash-away [5] happens if the current threshold boundary after n iterations is always equal to or greater than the next input vector at iteration $n+1$. Thus, Equ. 9 is obtained as

$$W_{ij} + Threshold \geq (n+1)D \tag{9}$$

By substituting Equ. 8 into Equ. 9, we get.

$$Threshold \geq \frac{D}{L}[1-(1-L)^n] \tag{10}$$

If n approaches ∞, we get

$$Threshold \geq \frac{D}{L} \tag{11}$$

The relation between the number of iterations, learning rate and the threshold are plotted in Fig. 3. The Euclidean distance between each input vector, D, is set as 0.01. The learning rate is plotted from 0.05 to 0.5 in steps of 0.05. The top curve has the minimum learning rate, 0.05, and the bottom curve denotes the maximum learning rate, 0.5.

Fig. 3 provides a guideline for selecting the threshold for specific applications. For a system having a large learning rate, it should use a small threshold value. We can release the threshold for a slow learning system. Beside this, there are

Transactions on Information and Communications Technologies vol 16, © 1996 WIT Press, www.witpress.com, ISSN 1743-3517

several factors we must consider for choosing the threshold value. The number of output neurons determines how many decision classes have to be partitioned in the pattern space. The similarity of input vectors affects the value of the threshold. To differentiate classes having similar features, we have to tighten the threshold to be used.

For a large number of iterations, Equ. 10 reduces to Equ. 11. This is the criterion for the occurrence of memory wash-away. For the assumed input sequence and the threshold greater than the ratio of the Euclidean distance of two consecutive input vectors and the learning rate, the new learning may wash away the memories of prior learning.

Now, we further elaborate the effect of learning rate on the learning process of the LBE network. We start the simulations as we apply 6 2-D input patterns to it. We first fix one of the inputs, and then vary the other inputs. In other words, the pattern will lie on the same straight line on the 2-dimensional space. Each input pattern is at a distance 0.05 apart except the last pattern, which is 0.1 away from the fifth pattern. The first pattern is at (0, 0.1), while the fifth and sixth pattern are at (0, 0.3) and (0, 0.4) respectively. These six patterns are shown in Fig. 4 where $d = 0.05$.

As the network learns these input patterns, the internal weights will drift in the direction as shown in Fig. 4. A large learning rate will lead the internal weight to drift towards the sixth input pattern more quickly, while a small learning rate will move the internal weights towards the sixth input pattern slowly. Therefore, assuming that the sixth input pattern is faulty, with a large learning rate the network will classify the faulty pattern as a good one eventually. Consequently, the chance of wrong classification will be larger.

Now, we vary the learning rate from 0.002 to 0.01, and take notice of the number of iterations that the sixth pattern is classified to the output neuron. The threshold is set as 0.015 and the number of iterations as 800. The network consists of 2 input neurons and 1 output neuron.

Fig. 5 plots the number of iterations when the sixth pattern is successfully classified. From it, we observe that as the learning rate increases, the number of iterations required to classify the sixth pattern is decreased. Also, the internal weights drift towards the sixth pattern more quickly. When the learning rate is set as 0.001, the network cannot classify the sixth pattern within 800 iterations.

The sum of weights against the number of iterations is shown in Fig. 6. The learning rate used for this simulation is 0.01. Since one input has set to 0, the sum of weights is equal to the weight vector with one of the elements equal to 0. The weights are increased as iterations proceed and they will reflect the status of the learning process. There is a jump in weights at iteration 91 because at this point, the current value is 0.2020213 and the sixth pattern is successfully classified, so the weights have been changed to reflect this learning. Since the learning rate is not equal to 1, therefore, the weight continues to increase as the learning process continues. At iteration 290, the weights stop increasing and equal 0.23501. At that moment, the learning process stops.

4. Fuzzification of the variables

As described in above, if the learning rate is set too low, the learning capability of the network is reduced. On the contrary, if the learning rate is set too large, the chance of wrongly classifying defective parts as good ones will be increased. Therefore, we need to control the learning rate in order to reduce this chance. Since we cannot find a good mathematical model to describe this relationship, we use fuzzy logic [3] here to control the learning rate. But first, we list the abbreviations used as belows:

ZE: ZERO	SM: SMALL	ME: MEDIUM
LA: LARGE	VL: VERY LARGE	
NS: NEGATIVE SMALL	NM: NEGATIVE MEDIUM	
NL: NEGATIVE LARGE	NVL: NEGATIVE VERY LARGE	
PS: POSITIVE SMALL	PM: POSITIVE MEDIUM	
PL: POSITIVE LARGE	PVL: POSITIVE VERY LARGE	

There are 3 factors that we need to monitor in order to control the learning rate. They are:- (1) the current mean square error, (2) the image distance, and (3) the stability of the network. Now, we start to construct the fuzzy membership graphs of these 3 factors.

4.1 Current Mean Square Error

The current mean square error ranges from 0 to 1, and we expect a linear relationship between the current mean square error and the learning rate. A large mean square error reveals that an input pattern deviates much from the stored pattern, so we can change the learning rate to a greater extent. As a result, we divide the mean square error into 5 fuzzy set values evenly, namely:- ZE, SM, ME, LA, and VL. Fig. 7 shows the membership function of the current mean square error.

4.2 Image Distance

The amount of learning rate that can be increased is related to the difference between the input pattern and the pattern stored in the network. Therefore, we need to know the difference between the input pattern and patterns stored. Their difference can be found out from the Euclidean Distance between the patterns. Each input pattern can be viewed as an position vector *X*, in the *M*-dimensional space, where *X* is defined by $X = \{ x_{1,} ..., x_{i,} ..., x_M \}$. The Euclidean distance, *D*, between pattern X and X' is given by

$$D^2 = \sum_{i=1}^{M} (x_i - x'_i)^2 \tag{12}$$

Since the pattern stored in the network is represented by the weight vector to the output neuron, we have to find out the Euclidean distance between the input patterns and the weight vector. In order to reduce the chance of wrong classification, we need to monitor those patterns that have not been classified yet. Here, we define the image distance as the distance between any unclassified input pattern and the output neuron, which is shown in Fig. 8. A large image distance indicates that the unclassified input pattern is different from the pattern stored to a greater extent. As stated in Section 3, we need to reduce the learning rate for a large difference. On the contrary, if the image distance is small, it means that the unclassified input pattern is very close to the pattern stored in that particular output neuron. Hence, we can have a larger learning rate. Therefore, we expect the learning rate is inversely proportional to the image distance. Thus, the image distance is divided into 5 fuzzy subsets as shown in Fig. 9.

4.3 Instability

The most important variable is the stability of the network. As the weight vector of the output neuron will be updated during learning, there is a chance that two output neurons will be grouped together. Therefore, we need to monitor the stability of the network. We use an array of memory to keep track with the classification of the input patterns. Whenever the network learns a new pattern, the new assignment of the input patterns will be checked with the array to determine if there is any previously stored patterns wrongly classified into other output neuron. If there is any changes, the network is considered as being unstable. Thus, the network has to be restored to the previous state and the learning rate should be reduced. The learning process should be started over again from that state. Here the instability ranges from 0 to the number of input patterns. It is normalised by dividing the number of stored pattern that are mapped to different classes by the number of previously stored patterns. A 0 means no stored patterns is re-mapped to another class, whereas an 1 indicates that all stored patterns are re-mapped to other classes. Again the instability is divided into 5 fuzzy subsets as shown in Fig. 10.

4.4 Percentage change in learning rate

From the above 3 input fuzzy variables, the inference engine can determine the new learning rate by finding the percentage change in the learning rate. We divide it into 9 fuzzy subsets and Fig. 11 shows its membership functions.

Now we start to build the rule matrix of the fuzzy inference engine. Since the stability is the most important factor of the network, we need to restore the network to a previous stable state and reduce the learning rate. Hence if the instability is not zero, the percentage change in the learning rate can be worked out as Table 1.

Instability	SM	ME	LA	VL
% Change in Learning Rate	NS	NM	NL	NVL

Table 1. The rule matrix when instability is not zero

When the instability is ZERO, then the percentage change in the learning rate can be obtained from the two variables, the mean square error and the image distance. We tabulate the rule matrix as in Table 2. There are total 16 fuzzy rules. When the current mean square error is small, it reveals that there is a small error between the stored patterns, then we need only a small learning rate; whereas for a large mean square error, there is a large error between the stored patterns, so we need a larger learning rate. We need a smaller learning rate for a larger image distance and a larger learning rate for a smaller image distance as explained above. With these rules in Table 1 and 2, the inference engine first finds out the scalar activation value w_i for each rule. From these scalar activation values, we can then work out the output fit vector, and the value of percentage change in the learning rate is found out by using a defuzzification algorithm - the centroid of the output fit vector.

% Change in Learning Rate		Current Mean Square Error			
		SM	ME	LA	VL
Image Distance	SM	SM	ME	LA	VL
	ME	SM	ME	LA	LA
	LA	SM	ME	ME	ME
	VL	SM	SM	SM	SM

Table 2. The rule matrix when instability is zero

5. Leadframe image classification

In this section, we will perform the inspection of leadframes using the LBE network. The I.C. leadframe is used to provide the mechanical support of the I.C. die and connections between the die and the pins of the integrated circuit. If there is any defects on the leadframe, the support of the integrated circuit or the connections will be weak or faulty. Therefore, we need to inspect the leadframe. There are different types of defects, which can be roughly classified as 2-dimensional and 3-dimensional ones. A complete inspection of all of them is time consuming and not practical. Therefore, common and critical defects are chosen here. Moreover, to increase the production efficiency, an automatic inspection system is needed for this task. Here, we will use an unsupervised neural network, the LBE network, to perform the inspection. In this paper we will concentrate on 2-dimensional shape distortion defects, which include planar rotation and width mismatches. First of all, the image of a whole leadframe is captured. There is a pair of guide pins on the leadframe, and with them, the leadframe can be accurately positioned. From the whole leadframe image, small portions are then extracted. Those locations that defects commonly occur will be chosen for the extraction.

Fig. 12 shows a whole leadframe image, and one small portion is chosen to extract from the leadframe for inspection. Apart from this portion, 7 more images that different from it are also used. Each small portion is 24×24 pixels in size. All these 7 images deviate from the original image by rotating a certain angle clockwise. Fig. 13 shows these 8 different images. Image 1 deviates from image 0 by rotating clockwise by 2^o; whereas image 2 by 5^o clockwise. Images 3, 4, 5, and 6 deviate from image 0 by rotating clockwise 10, 15, 20, 25^o respectively, while image 7 is different from image 0 by rotating 45^o clockwise.

Now we apply these 8 images to the network for inspection. As observed from the data, the differences between the images increase gradually. The LBE network are configured to have only 1 output neuron and 576 input neurons. The initial learning rate is set to 0.01. The threshold is set to 0.011 and the number of iterations is 200. Fig. 14 plots the learning rate against the number of iterations, and Fig. 15 depicts the sum of weights to the output neuron against the number of iterations. The sum of weights is an indication of the learning process, and it can be viewed as the cluster centre of the output neuron which holds the learned pattern. Therefore, it may be increased or decreased as the network learns the input patterns. As observed in Fig. 14, the learning rate is increased at the beginning. The output neuron groups image 0 to 6 together and learns these 7 images according to the learning rate. Fig. 15 reveals that the sum of weight decreases as the network learns these 7 images. However, at iteration 150, the learning rate stops decreasing and then bounces up and down thereafter; whereas the sum of weights increases after iteration 150. This is because at this point, the centre of the cluster corresponds to the output neuron has drifted to a point that image 7, which deviates much from the original (good), has been classified by the output neuron. But image 0, the original one, has been rejected by the output neuron. In another word, the output neuron has forgot the original image. This is because the threshold is set too low (0.011) such that the original image falls outside the cluster boundary of the output neuron. Since there is a change in the classification of the stored pattern inside the network, the network is considered to be unstable. The fuzzy inference engine has to restore the network to a previous stable state. Hence, it decreases the learning rate and retries again. As a result, there is a drop in the learning rate and an increase in the sum of weights. As the network retries the learning, the cluster centre of the output neuron drifts again which causes the network to be unstable again. Consequently, the learning rate bounces up and down after iteration 150 and onwards.

The network forgets the original image as it learns the 7th image. It occurs as a small threshold will not allow the cluster boundary to hold all the images. Therefore, we can increase the cluster boundary by increasing the threshold to 0.0112 and apply the images to the network again. This time, the network learns all the images at iteration 130. Fig. 16 shows the learning rate against the number of iterations and Fig. 17 shows the sum of weights. As observed in Fig. 16, the learning rate is increased to 0.145519 at iteration 120 and is kept unchanged. In Fig. 17, the sum of weights decreases to 194.77142 at iteration 130 and is kept unchanged. Both figures indicate that the network learns the last image at iteration 130 and then all the parameters are being kept unchanged, or there is no need to change it. Therefore, we can increase the learning capability by increasing the threshold. If image 7 is considered to be a defective one, we do not want to increase the threshold to allow the network to classify it. On the contrary, if image 7 is accepted, we need to increase the threshold. Therefore, we still need human supervision on the determination of the acceptance level of the images.

The same simulation has been applied to a LBE network without the learning rate control. When the threshold is set as 0.011 and the initial learning rate set as 0.22, the network groups image 7 together and rejects the original image at iteration 3. With threshold set as 0.0112, image 7 is grouped to the output neuron at iteration 4. The reason why image 7 has been learned by the network so fast is that we have set a large learning rate at the very beginning. Therefore, if image 7 is considered as defective, the chance of wrong classification will be increased as we do not know what should be the initial learning rate. Therefore, the fuzzy inference engine provides a good guess at the initial learning rate -- we just set a low initial learning rate and let the fuzzy inference engine to control it.

Another leadframe is used. Fig. 18 shows a whole leadframe image, and 10 small portions are extracted from the leadframe for inspection of defects. Images 0, 3, 6, 9, 12, 15, 18, 21, 24, and 27 form the ten basic image groups. Apart from these ten groups, 20 more images are extracted from different leadframes. Some of them deviate from these basic images by rotating a certain angle clockwise or counter-clockwise. Fig. 19 shows these 30 different images.

If we set a tolerance of $\pm 20^{o}$ of rotation, image 2, 11, and 14 will become defective. In addition, we will expect image 23 will be defective since it is thicker than the original one too much. Now we apply these 30 images to the network for classification. As expected, there should be totally 11 groups of images as tabulated in Table 3. Here, group 11 is the defective part. Thus, the LBE network is configured to have 10 output neurons and 576 input neurons. The threshold is set to 0.014 and the number of iterations is 50. The results of classification are tabulated as Table 4. As observed in Table 4, there are many images being unclassified (denoted by the symbol *). This is because the threshold is low such that the network does not allow a large error between patterns. Therefore, we

Class	1	2	3	4	5	6	7	8	9	10	Defective parts
Image	0, 1	3, 4, 5	6, 7, 8	9, 10	12, 13	15, 16 17	18, 19 20	21, 22	24, 25 26	27, 28, 29	2, 11, 14, 23

Table 3. Classification of the images

change the threshold to 0.015, and the result of classification are tabulated as in Table 5. As observed, images 7, 8 are classified to output neuron 3 and images 19, 20 are being classified to output neuron 7. Therefore, the increase in threshold will lead to an increase in learning capability of the network. Although the learning

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Output Neuron	1	1	*	2	2	2	3	*	*	4	*	*	5	*	*	6	*	*	7	*	*	8	8	*	9	*	*	10	10	10

Table 4. Assignment of images with threshold = 0.014

capability has been increase, there are still some images that cannot be classified. If we further increase the threshold, some of the defective images will be classified as good ones.

6. Conclusion

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Output Neuron	1	1	*	2	2	2	3	3	3	4	*	*	5	*	*	6	*	*	7	7	7	8	8	*	9	*	*	10	10	10

Table 5. Assignment of images with threshold = 0.015

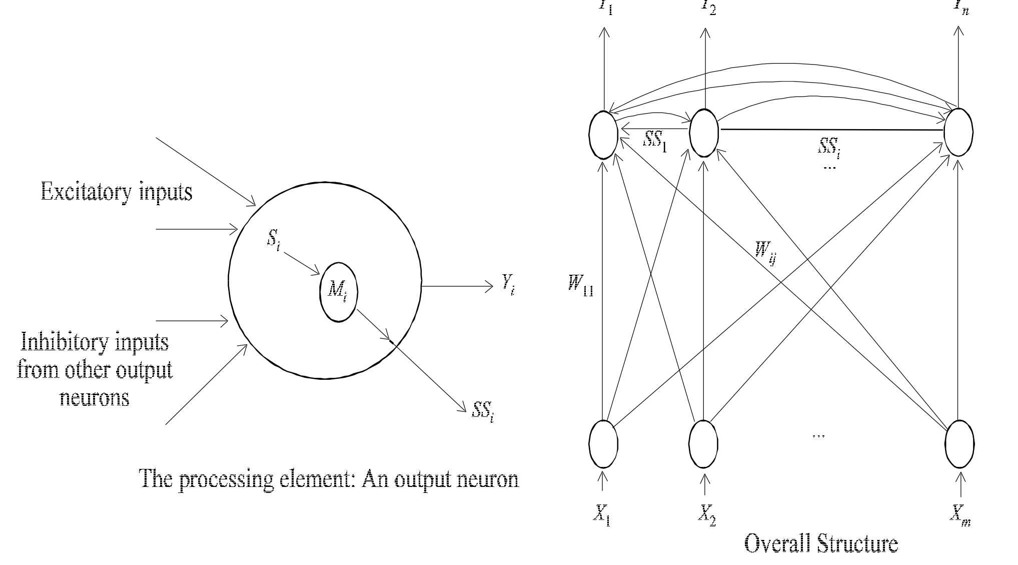
In this paper, we have illustrated a relationship between the learning rate and the learning process. As the learning rate increases, the chance of wrong classification will be increased. A small learning rate will reduce the learning capability of the network since the weights are kept unchanged. Although we can increase the learning capability of an unsupervised neural network by increasing the threshold[4], we need to set an optimal learning rate in order to control the learning process. Therefore, we have to find a method to control the learning rate and the stochastic approach may be the next step[6,7]. The drift of the weights depends on the variation of input patterns, so we need to predict the variation of the input patterns in order to set the required learning rate. Therefore, we incorporate an fuzzy inference engine in the LBE network to serve this purpose. Moreover, the industrial application for IC leadframe inspection using the LBE network has been described.

References

1. C. K. Lee, C. H. Chung, "An unsupervised neural network with a memory replacement effect," *IEEE ICNN'94*, June 26 - July 2, 1994, vol. 2, pp. 675-680, Orlando, USA.
2. C. W. Therrien, *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics.* John Wiley & Sons, 1989.

3. B. Kosko, *Neural Networks and Fuzzy Systems, A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, 1992.
4. C. K. Lee, C. H. Chung, "An unsupervised neural network for machine part recognition with constraint release," *IEEE ETFA'94*, 6-10 Nov., 1994, pp. 166-173, Tokyo, Japan.
5. G.A. Carpenter, S. Grossberg, " The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," *IEEE trans. Computer*, March 1988, pp 77-88.
6. B. Kosko, "Stochastic Competitive Learning," *IEEE Trans. Neural Networks*, vol. 2, No.5, pp. 522-529, Sept 1991.
7. S.G. Kong, B. Kosko, "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," *IEEE Trans. Neural Networks*, vol. 2, No.1, pp. 118-124, Jan 1991.

Fig. 1 The structure of the neural network



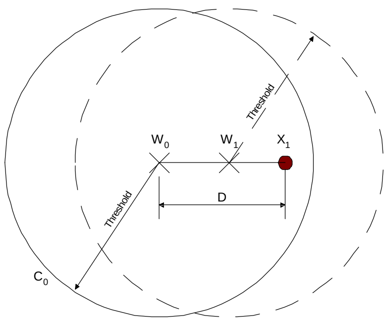


Fig. 2 The drifting of weights as the input pattern changes.

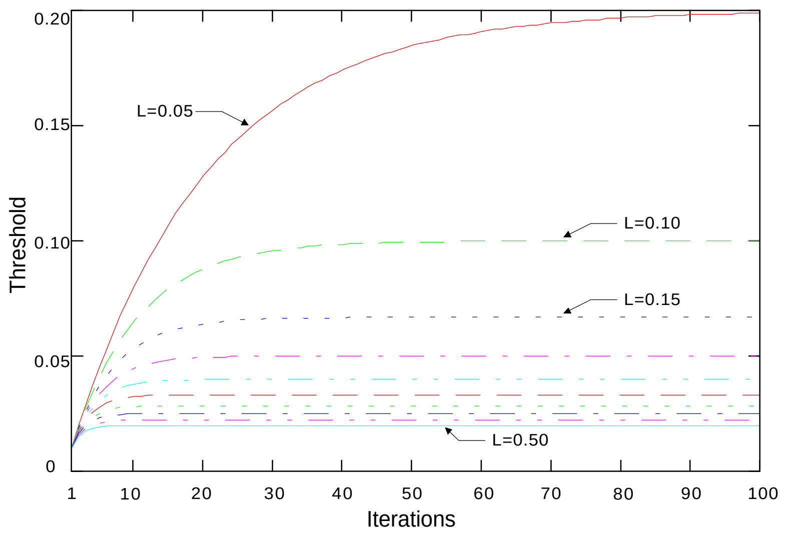


Fig. 3 Threshold vs. no. of iterations.

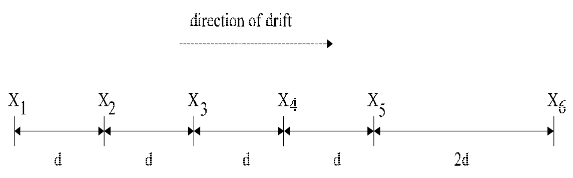


Fig. 4 Placement of input patterns.

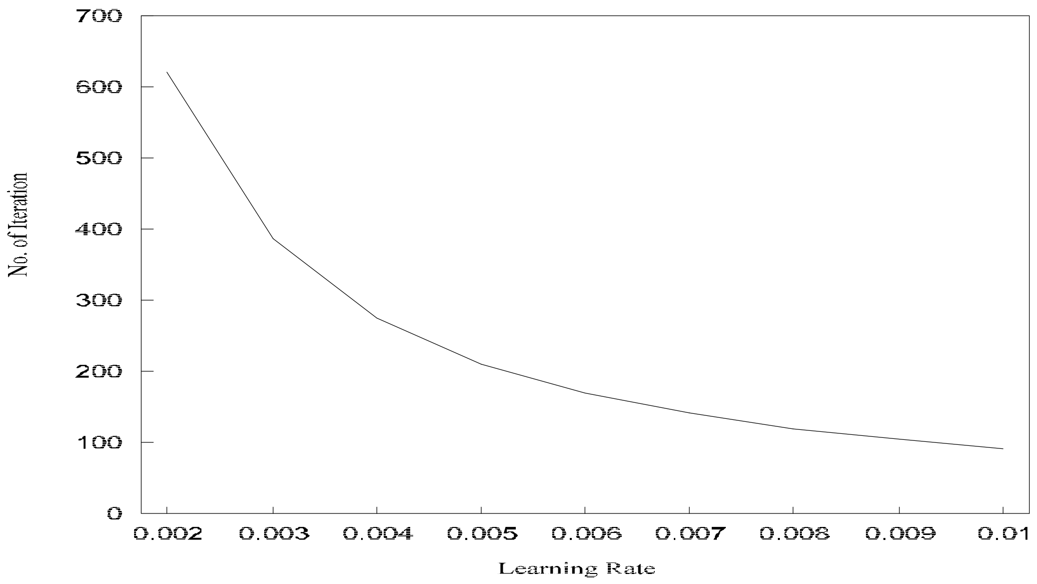


Fig. 5 Number of iterations vs. the learning rate.

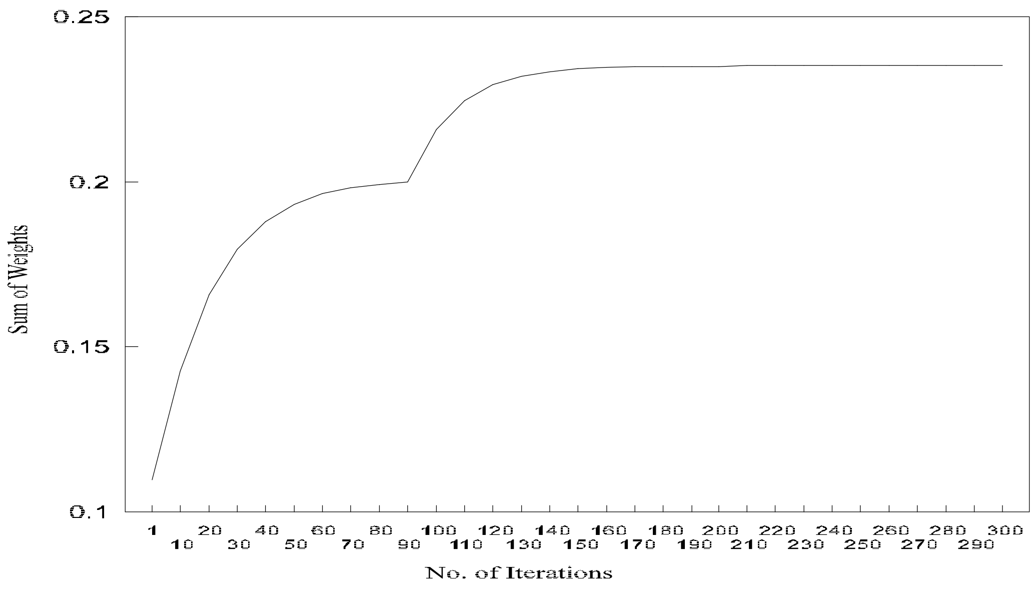


Fig. 6 Sum of weights vs. no. of iterations.

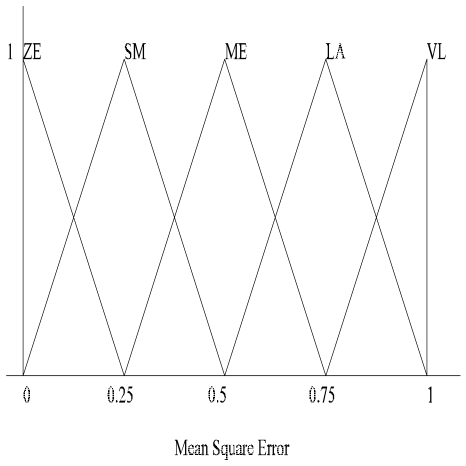


Fig. 7 Membership graph for the mean square error

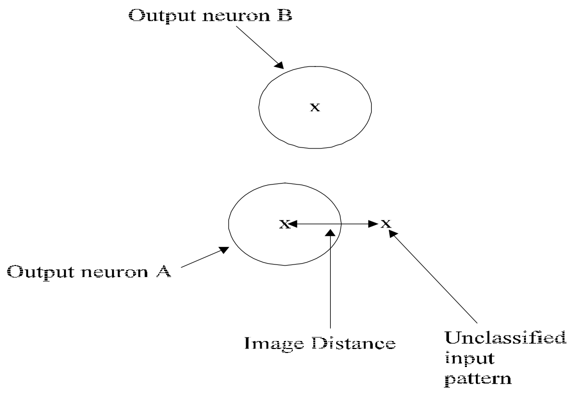


Fig. 8 Illustration of the image distance.

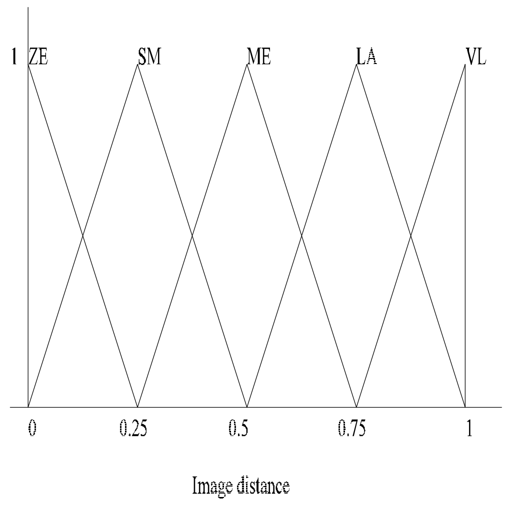


Fig. 9 Membership graph for the image distance

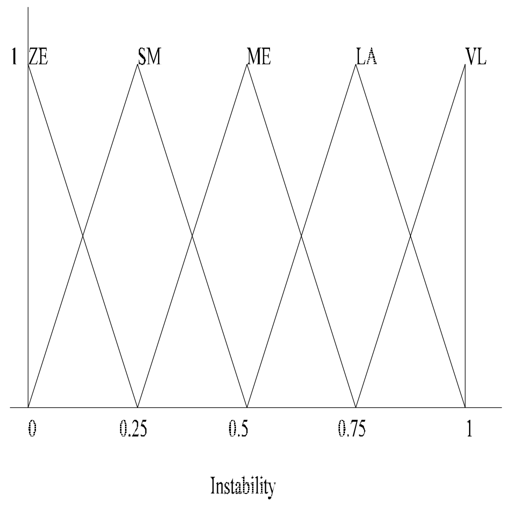


Fig. 10 Membership graph for instability

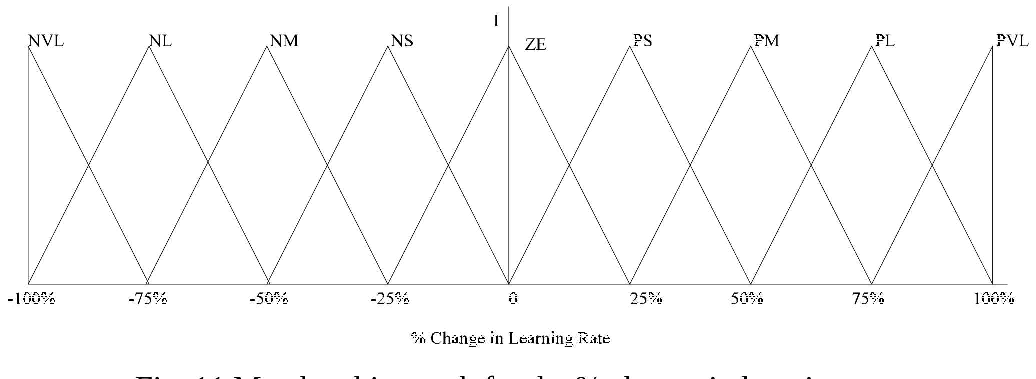


Fig. 11 Membership graph for the % change in learning rate

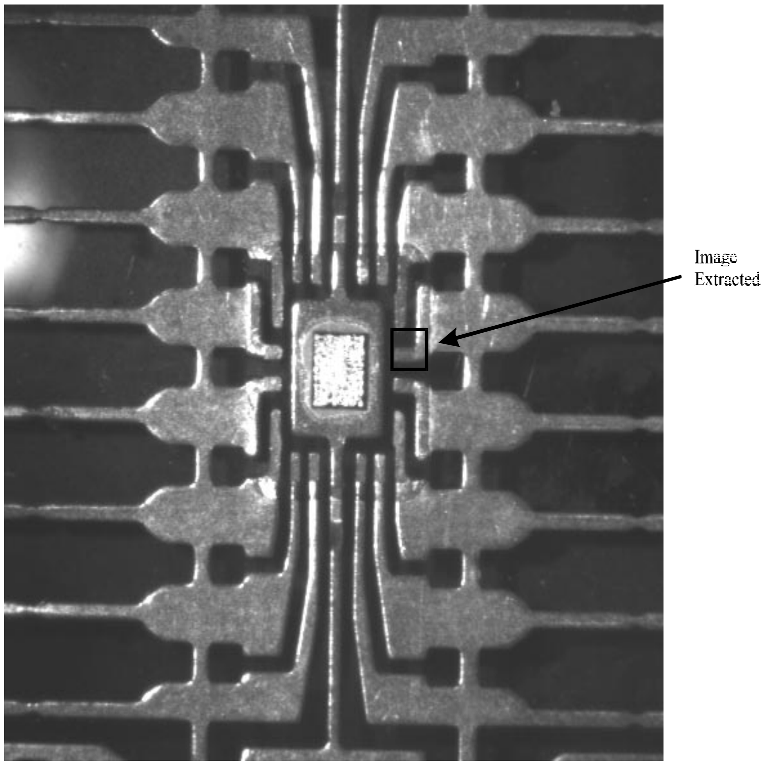


Fig. 12 A leadframe.

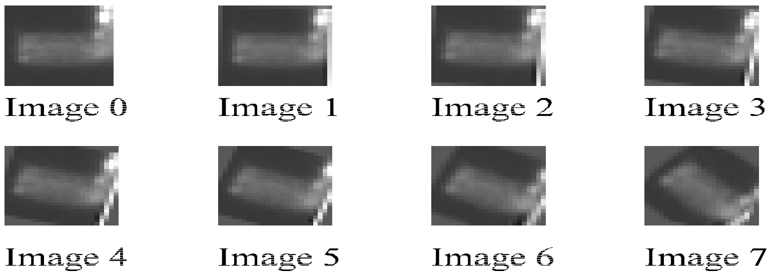


Fig. 13 Variations of the small portion.

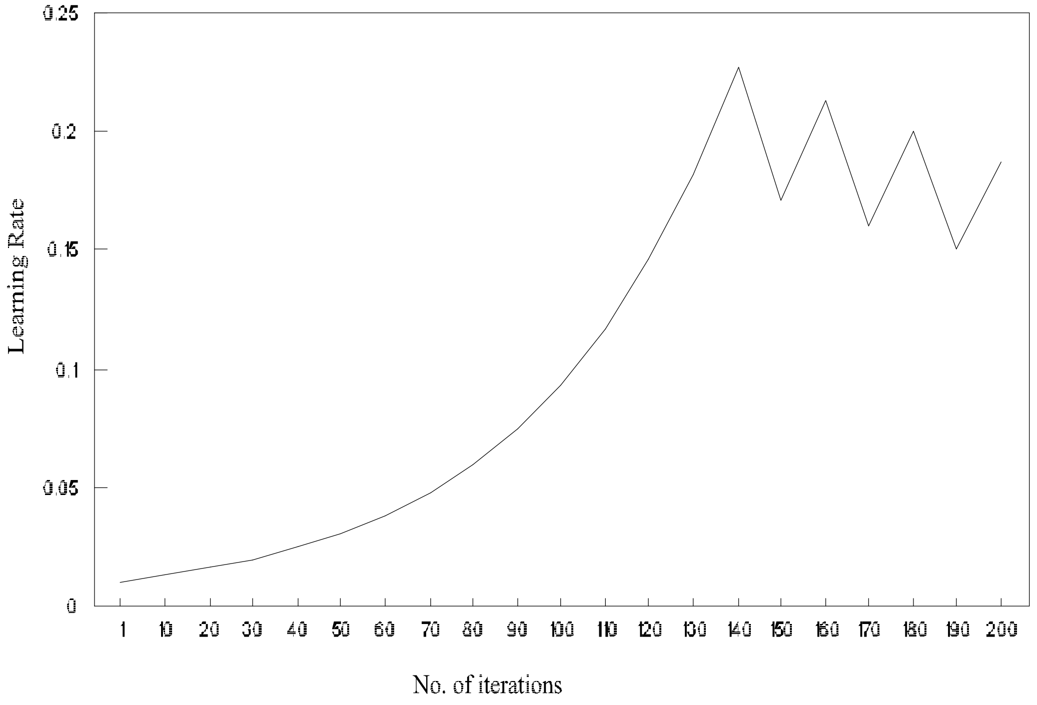


Fig. 14 Learning rate vs. no. of iterations (threshold = 0.011)

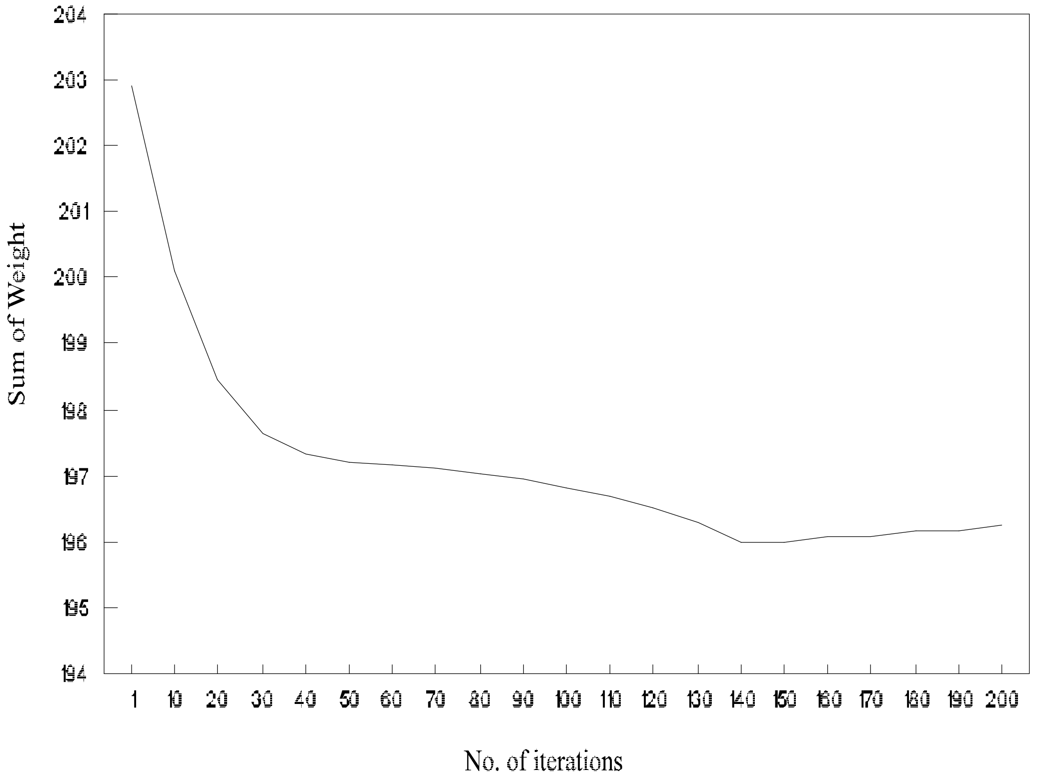


Fig. 15 Sum of weights vs. no. of iterations (threshold = 0.011)

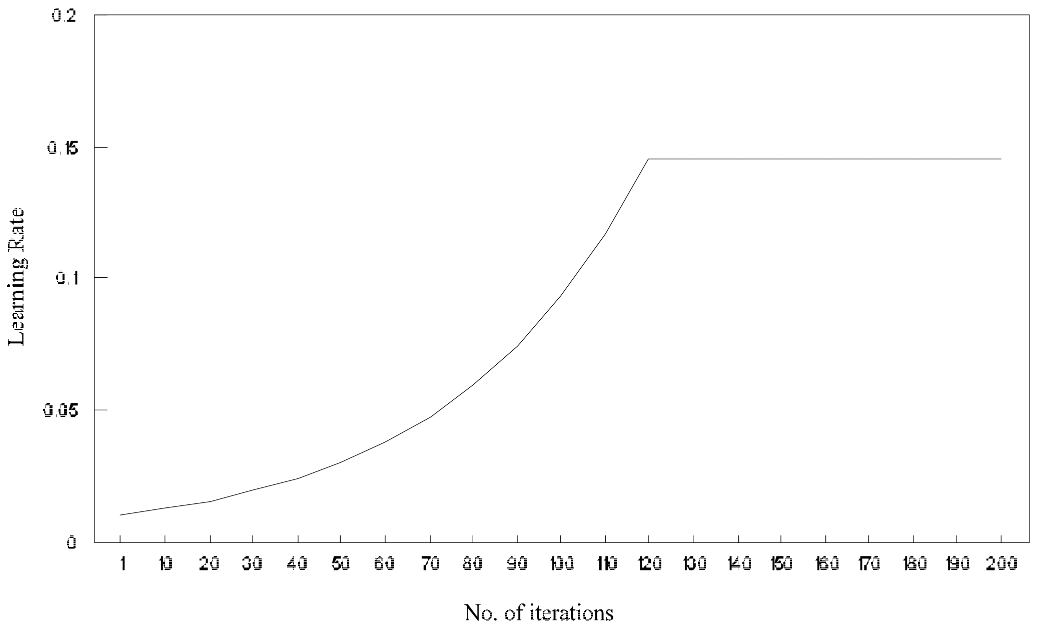


Fig. 16 Learning rate vs. no. of iterations (threshold = 0.0112)

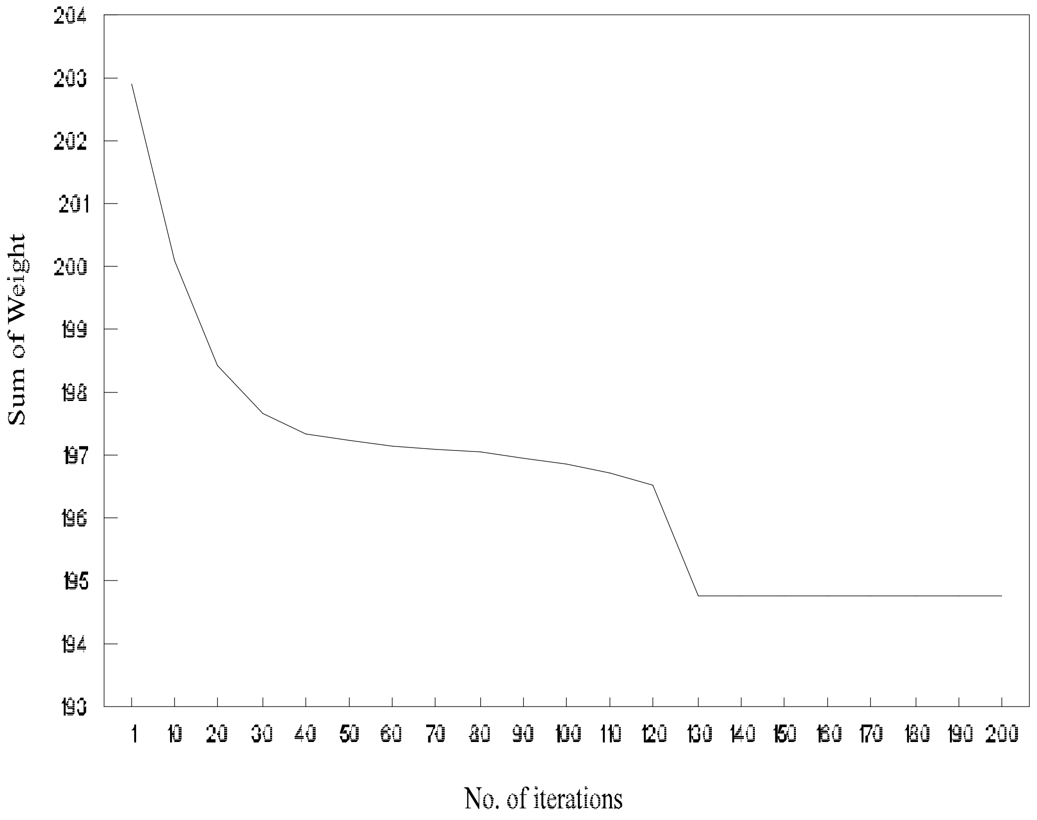


Fig. 17 Sum of weights vs. no. of iterations (threshold = 0.0112)

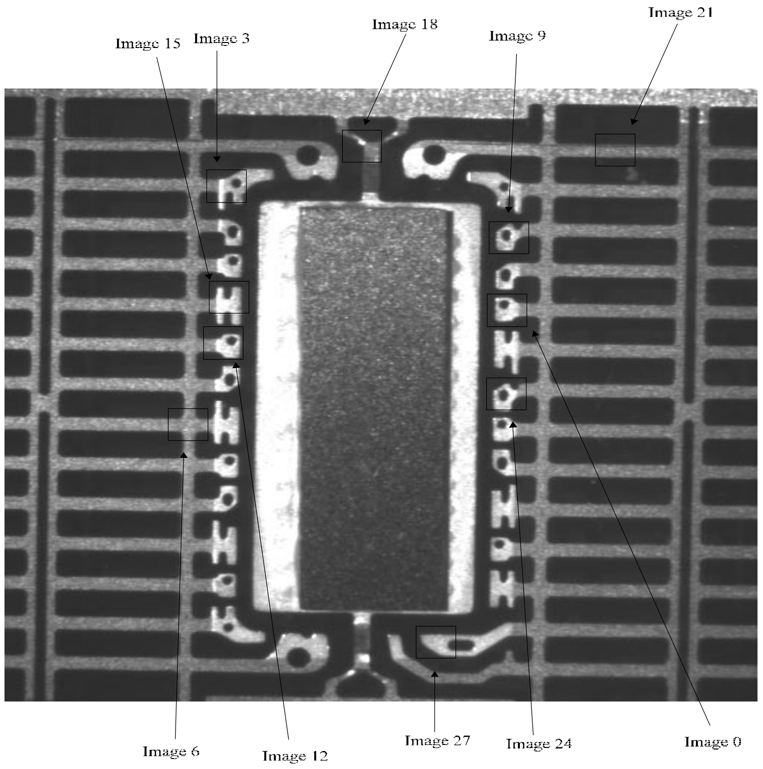


Fig. 18 Another leadframe

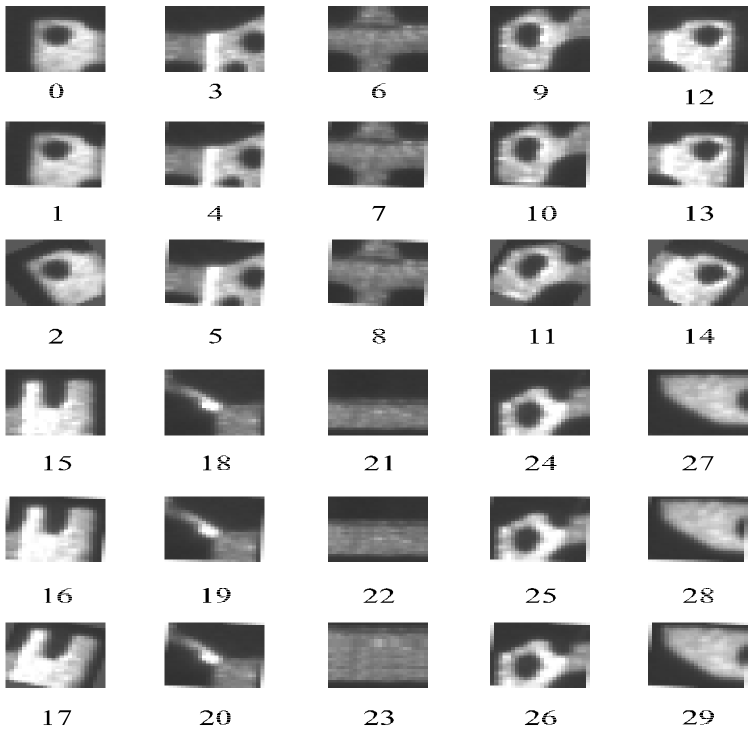


Fig. 19 Portions of the leadframe for inspection.