Pergamon

Neural Networks

Neural Networks 11 (1998) 315–322

Contributed article

Encoded pattern classification using constructive learning algorithms based on learning vector quantization

C. N. S. Ganesh Murthy[1], Y. V. Venkatesh*

Computer Vision and Artificial Intelligence Laboratory, Department of Electrical Engineering, Indian Institute of Science, Bangalore–560012, India

Received 13 December 1996; accepted 4 August 1997

Abstract

A novel encoding technique is proposed for the recognition of patterns using four different techniques for training artificial neural networks (ANNs) of the Kohonen type. Each template or model pattern is overlaid on a radial grid of appropriate size, and converted to a two-dimensional feature array which then acts as the training input to the ANN. The first technique employs Kohonen's self-organizing network, each neuron of which is assigned, after training, the label of the model pattern. It is found that a graphical plot of the labels of the neurons exhibits clusters (which means in effect that the feature array pertaining to distorted versions of the same pattern belongs to a specific cluster), thereby justifying the coding strategy used in this paper. When the new, unknown pattern is input to the network, it is classified to have the same label of the neuron whose corresponding model pattern is closest to the given pattern.

In an attempt to reduce the computational time and the size of the network, and simultaneously improve accuracy in recognition, Kohonen's learning vector quantization (LVQ) algorithm is used to train the ANN. To further improve the network's performance and to realize a network of minimum size, two constructive learning algorithms, both based on LVQ, are proposed: (1) multi-step learning vector quantization (MLVQ), and (2) thermal multi-step learning vector quantization (TLVQ). When the proposed algorithms are applied to the classification of noiseless and noisy (and distorted) patterns, the results demonstrate that the pattern encoding strategy and the suggested training techniques for ANNs are efficient and robust. For lack of space, only the most essential results are presented here. For details, see Ganesh Murthy and Venkatesh (1996b). © 1998 Elsevier Science Ltd. All rights reserved.

Keywords: Kohonen network; Learning; Learning vector quantization; Pattern classification; Pattern encoding; Self-organization

1. Introduction

Many models of artificial neural networks (ANNs) have been proposed to solve the problem of automatic recognition of patterns. They seem to offer better performance and distinct computational advantages over other non-neural models in handling this problem. An ANN consists of interconnected elementary processors which interact at discrete time steps, and is specified by (1) its node characteristics (response function) and (2) its dynamics (or the updating of the neurons). A neuron in an ANN performs a weighted sum of its inputs, compares this with an internal threshold value, and turns on (or fires) if this level is exceeded; otherwise it is off. The state of each processor at any instant is determined by the states of other processors connected to it, and is weighted by the strengths of the appropriate links. The ANN is 'taught' the transformation (F) of a given input to the (desired) output by changing the weights of the links according to a well-defined strategy.

The perceptron (Minsky and Pappert, 1969) was the earliest neural architecture, comprising a single layer of units, meant to classify input patterns into two or more classes. But it can 'learn' to recognize only linearly separable patterns. The multilayer perceptron (MLP) (Lippmann, 1987), which overcomes the limitations of the perceptron, contains neurons whose states are real, and the response function is sigmoidal. However, an MLP treats patterns as vectors, and such an architecture cannot satisfactorily recognize shifted/scaled/rotated patterns. For example, consider the pattern in Fig. 1(a) and its distorted versions in Fig. 1(b) and (c). The MLP has been found to be unsatisfactory in analysing such patterns, because it treats each variation of the same pattern as a distinct vector.

* Requests for reprints should be sent to Professor Y. V. Venkatesh. Tel. 0091 80 309 2572, 334 1566; Fax: 0091 80 334 1683, 334 2085.

[1] Present address: Department of Electrical Engineering, University of Victoria, Victoria, BC, Canada

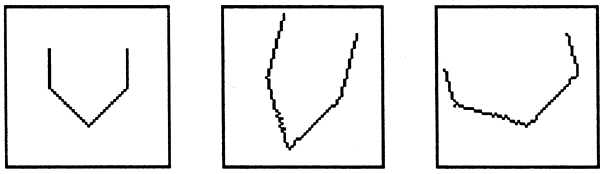
Fig. 1. A pattern and its distorted versions.

In general, ANNs cannot be directly used for distortion-invariant recognition. The patterns first have to be pre-processed to extract invariant features before the ANNs are used to classify. The present paper deals with the encoding of patterns in such a way as to achieve their recognition by the ANN, in spite of changes (to a limited extent) in scale, shift and rotation of the input patterns.

The paper is organized as follows. After a brief survey of recent literature on ANNs in section 2, we present our proposed technique for pattern recognition and illustrate its application. In section 3 we deal with (1) learning vector quantization (LVQ) and its multi-step version, and (2) training the ANN with the thermal perceptron rule. In section 4, we analyse the experimental results obtained by the proposed method.

2. ANNs and pattern recognition

In an attempt to imitate human vision, pattern recognition research has shifted emphasis from (traditional) image analysis-based techniques to ANN's. Yoshida (1993) describes an ANN with a neocognitron type of architecture (see Fukushima, 1991), which is trained to recognize handwritten alphanumeric characters. The ANN of Fukushima (1991) consists of an input layer and four or five intermediate layers before the final output layer. The middle layers are trained using an unsupervised learning techique based on a combination of competitive and Hebbian learning schemes. See Ganesh Murthy and Venkatesh (1991) and Ganesh Murthy and Venkatesh (1996a) for results of some experiments on the neocognitron.

For a small-sized input array, Nakanishi and Fukui (1993) propose a multilayered ANN using hierarchical feature types and their locations for training and recognition. Kojima et al. (1993) describe a handwritten numeric character recognition system, the main classifier of which is a fuzzy vector quantizer, involving an improved version of the vector quantization algorithm. Features are extracted from the pre-processed input pattern, and used for training the network. Zufira (1993) presents some improvements on a neural network structure of the multilayered perceptron, with a pre-processing network to perform translation-, rotation- and scale-invariant pattern recognition.

You and Ford (1994) propose a three-stage network architecture for invariant object recognition and rotation angle estimation. However, no specific features are extracted. Ghorpade and Ray (1993) deal with a new adaptive and connectionist architecture for English alphabetic patterns.

In terms of feature extraction, the work which has some relation to our coding strategy is that of Jian and Chen (1992), who proposed a non-neural system for recognition of handwritten Chinese characters, containing short line segments. The input is a two-dimensional pattern which is pre-processed—mainly thinning and line approximation—to extract the following features: the centre point coordinate, the slope and the relations between the line segment and its neighbouring line segments. Classification of these characters is made in three stages. First, short line segments are extracted, for which a new efficient algorithm is proposed (based on accumulated chain codes) for line approximation. A coarse classification is made using the directions of the short line segments (based on the histogram of their quantized directions). In the second stage, feature extraction, features of each character are computed as indicated above. In the last stage, matching, dynamic programming is first used to calculate the similarity between a segment of each of the input and reference characters, using the feature vectors of the segments. The overall similarity between the two characters is then computed for the next operation of recognition. However, the effects of scaling and rotation are not discussed.

For other published results, including those on invariant pattern recognition using neural networks, see Fukumi et al. (1992), Khotanzad and Lu (1990), Khotanzad and Hong (1990), Srinivasa and Jounch (1992) and Perantonis and Lisboz (1992).

3. A new technique for pattern encoding and recognition

In contrast to the results in the literature, we present a technique to convert a pattern to a feature array in such a way that the encoded version of the pattern is invariant to pattern transformations such as shift, scaling and (moderate) rotation. A distinct departure from the traditional published methods is the use of this array of codes as the training vector to train the self-organizing network, the neurons of which are appropriately labelled. The plot of the labels is then checked for the formation of clusters of the labels of the neurons. With this procedure, we intend to confirm the relevance or otherwise of the feature array to pattern classification. The steps involved in encoding the given pattern are as follows:

- Find the centroid (h,k) of the input pattern P (Fig. 2a).
- Find the furthest point (p,q) of the pattern P from the centroid (h,k) (Fig. 2a).
- Now place the pattern P on the radial grid with (h,k) as the centre and with $d[(h,k),(p,q)]$, the Euclidean distance between the points (h,k) and (p,q), as the radius of the outermost circle (Fig. 2b).

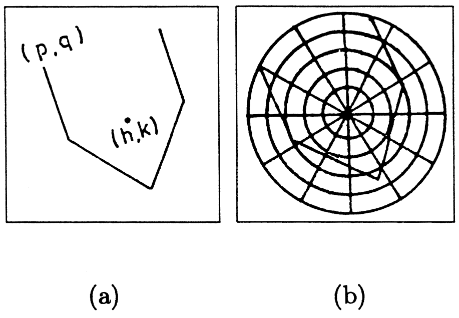

Fig. 2. (a) Pattern with the furthest point (*p*,*q*) and the centroid (*h*,*k*). (b) Pattern in (a) overlaid on the radial grid.

- Count the number of points in each of the radial grids to obtain the feature array, inp[][]. Normalize inp[][] with respect to max(inp[][]).

For details, including the pseudocode for obtaining the input array of codes inp[][] from the input pattern P, see Ganesh Murthy and Venkatesh (1994, 1996b). Fig. 3(a) shows a subset of exemplar patterns, and Fig. 3(b) a subset of the typical random variations of the exemplars, from which the inp[][] are calculated and used for training.

3.1. Training of the self-organizing network

As the first stage in evolving an efficient recognition strategy, the array inp[][] obtained in the manner described above is then fed to a Kohonen-type self-organizing network (Kohonen, 1990a). After training, each neuron is given the label of the pattern to which it responds most, and then the plot of the labels is checked for the formation of clusters. If the plot exhibits clusters, the aptness of the feature extraction procedure is established.

In our simulation studies, inp[][] of the given pattern has been calculated using a grid in which the angular measure has been quantized to 12 levels, and the radial distance to five. Hence the number of weights for each neuron is 12×5. In the actual implementation, the input array inp[][] has been obtained for 1000 variations of 28 different exemplar patterns (i.e. a total of 28,000 patterns).

In the creation of variations of the exemplars, each exemplar (of size 31×31) is subjected to random independent magnifications (in the *x* and *y* directions) and rotation. The value of magnification ranges from 1.0 to 2.0, and the random rotation from $-15°$ to $+15°$. In Fig. 4, typical examples of the variations of the patterns are shown. These 28,000 inp[][] arrays are randomly (and repeatedly) selected to train a self-organizing network with 25×25 neurons. See Ganesh Murthy and Venkatesh (1996b) for details of the training procedure. At the end of maxiter = 2,800,000 iterations, each of the 28,000 inp[][] arrays will have been selected ~100 times. The next step after training the self-organizing network is to label each neuron corresponding to the exemplar which it represents.

3.2. Labelling of the neurons

The neurons are labelled to correspond to the patterns

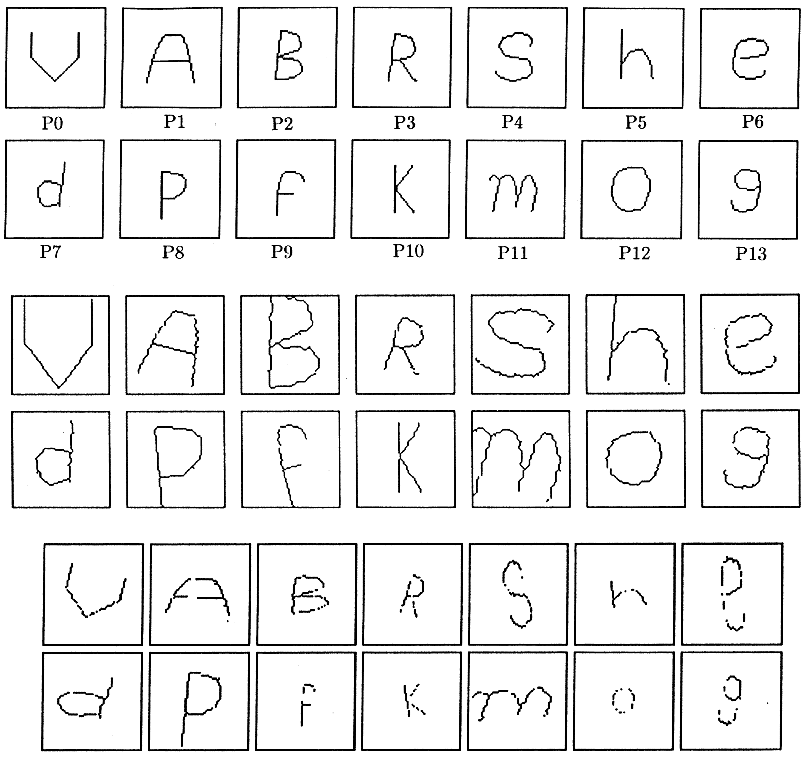

Fig. 3. (a) Subset of the 28 exemplars used in simulation. (b) Subset of the typical variations of the exemplars used for training and testing. (c) Examples of severely distorted patterns used to demonstrate the performance of the proposed techniques.

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

Fig. 4. The labels of the 25 × 25 neurons (after training and labelling), indicating the clusters formed.

which they represent. To this end, each exemplar is subjected to random variations, and the coded input array inp[][] of the distorted exemplar is fed to each neuron of the network in the following sequence. First, consider a specific neuron at location, say, (i,j). Feed it with a set having one distorted version per exemplar pattern. Note the label of the (distorted) exemplar that is closest to the weight vector of the neuron. Next, repeat this process several times with the next set of distorted exemplars. Keep track of the number of times each (distorted) exemplar is closest to the neuron under consideration. Identify the exemplar that gets the maximum count, and label the neuron at location (i,j) correspondingly.

From Fig. 4, it is seen that the labels of the 25 × 25 neuron array do exhibit the formation of clusters, thereby admirably justifying the coding method used. It is also evident from the same figure that the feature arrays of the distorted versions of the same pattern form clusters.

3.3. Discussion of the results

To test the network, an inp[][] array for 1000 random variations of each exemplar is fed to it. This procedure is repeated three times. Typical results are summarized in Table 1. The network is also tested for its performance with noisy input patterns. Noise, which is introduced into the pattern by flipping the pixels with a specified probability, gives rise to breaks and isolated dots in the pattern. However, isolated dots can easily be removed by a threshold scheme, and the breaks can be filled in by using simple morphological techniques Serra (1982), such as dilation followed by thinning. For each exemplar, 1000 variations are used, and the entries denote the number of misclassifications for the cases NL (noiseless), 10NM (10% noisy patterns pre- processed using morphological operations) and 20NM (20% noisy patterns pre-processed using morphological operations).

The first column for each test contains the results of the experiments when noiseless patterns are used. The second and third columns contain the results when 10% and 20% noisy patterns are first pre-processed by morphological techniques and then used. Table 1 indicates that the worst case for noiseless patterns is with respect to pattern P5 where ~150 variations are misclassified out of 1000. The worst case for 10% and 20% noisy patterns (after pre-processing) is found to be ~190/1000 and ~210/1000 respectively. To save space, results for only a subset of patterns (P1 to P14) are shown, whereas the total misclassifications and % misses (the last two rows in Table 1) have been computed on the basis of the results for the full set of 28 patterns.

The percentage misclassification over all patterns for the noiseless case is found to be only ~3.0, which is believed to be very impressive. In contrast, for noisy patterns with 10% and 20% noise, the corresponding number is ~3.8. This clearly shows that the proposed self-organization scheme can classify the patterns quite successfully. Further results of our experiments, with the angular measure and the radial

Table 1
Results of the three tests on the self-organizing network after training

Pattern	Test 1			Test 2			Test 3		
	NL	10NM	20NM	NL	10NM	20NM	NL	10NM	20NM
P1	0	0	0	0	0	1	0	0	0
P2	30	33	20	28	27	38	29	35	31
P3	8	28	30	13	40	33	16	35	34
P4	14	22	20	24	16	21	20	23	22
P5	143	190	204	139	208	216	155	187	208
P6	0	0	0	0	0	0	0	0	2
P7	9	3	11	7	3	10	10	2	9
P8	0	0	0	0	0	0	0	0	2
P9	42	38	57	47	63	65	45	34	49
P10	0	0	1	0	2	1	0	0	2
P11	6	9	10	4	8	4	6	6	11
P12	1	1	3	0	3	5	0	2	5
P13	29	35	31	50	43	39	28	37	42
P14	7	23	29	9	23	26	5	26	30
…	…	…	…	…	…	…	…	…	…
Total	823	1035	1037	856	1022	1139	893	1024	1085
% Misses	2.94	3.70	3.70	3.06	3.65	4.07	3.19	3.66	3.88

distance each quantized to different levels, are given in Ganesh Murthy (1996).

4. Other schemes of learning

In an attempt to improve on the learning algorithm described above, but using the same encoding technique, we now summarize the results obtained from other schemes of learning, along with the corresponding ANNs; for details, see Ganesh Murthy and Venkatesh (1996b):

- Kohonen's learning vector quantization (LVQ)
- First constructive algorithm based on LVQ, called the multi-step learning vector quantization (MLVQ).
- Second constructive algorithm based on LVQ, called the thermal multi-step learning vector quantization (TLVQ).

4.1. Training using the LVQ algorithm

We apply Kohonen's learning vector quantization algorithm (LVQ) (Kohonen, 1990b) to learn recognition of patterns, using the feature arrays described in the previous section. To this end, we train an array of five neurons per pattern with the help of the LVQ algorithm, using 10, 50, 100 and 200 variations of the given patterns, and analyse the results for both noiseless and noisy patterns. For details of the technique, see Ganesh Murthy and Venkatesh (1996b).

It has been found that the LVQ is faster than Kohonen's self-organization scheme, because in each iteration (of the LVQ) only one neuron needs to have its weight updated. Furthermore, the LVQ performs better with fewer neurons than Kohonen's. After 100,000 number of repetitions of step 3, the training is stopped. Typically, training takes ~40 s on an HP-9000/715 workstation to complete the required number of iterations. On the basis of the results obtained by testing the network with noisy patterns and pre-processed noisy patterns, it is found that the performance of the LVQ is superior to that of Kohonen's self-organizing network.

In an attempt to improve the LVQ technique, we propose two new variants for a step-by-step constructive design of the network. The variants are superior to LVQ in terms of training speed and accuracy.

4.2. Multi-step learning vector quantization

In the LVQ, our choice of five neurons per class is apparently ad hoc. We now present a technique to construct systematically a multilayered neuronal network with the required number of neurons to accomplish the classification of patterns. First, we start with a network having only one layer of neurons, with one neuron representing each class. We use this network to classify correctly as many training-set patterns as possible. If there are still any wrongly classified training-set patterns, we add another layer of neurons to the network. The training of this layer is done using only the patterns that are not unambiguously classified in the previous layer. We repeat the process until all the training set patterns are correctly classified.

This technique enables one to develop a network of the required size systematically, thereby avoiding ad hoc assumptions. As before, the network is trained with $T = 10, 50, 100, 200$, and for each case the network is tested with 3000 random variations of the patterns. For a description of the testing scheme, see Ganesh Murthy and Venkatesh (1996b). Further improvement is achieved by a multi-step scheme using the 'thermal perceptron' (TP) learning rule which follows.

Table 2
Performance of LVQ, MLVQ and TLVQ when tested with 3000 variations of each pattern, i.e. a total of 3000 × 28 patterns

Class	Noiseless			10% Noise			10% Noise + morphology		
	LVQ	MLVQ	TLVQ	LVQ	MLVQ	TLVQ	LVQ	MLVQ	TLVQ
P1	11	43	10	18	75	28	8	49	43
P2	84	59	19	100	84	22	108	66	18
P3	134	62	41	143	303	53	175	184	97
P4	192	138	55	262	199	108	240	146	91
P5	142	184	70	251	286	96	239	269	115
P6	140	19	29	131	44	59	153	8	38
P7	53	85	27	115	199	49	32	106	27
P8	59	98	80	75	115	45	61	112	43
P9	18	36	12	84	116	121	48	76	30
P10	1	16	3	29	67	28	7	12	10
P11	64	18	7	100	69	19	78	23	23
P12	2	35	28	14	44	37	16	46	56
P13	335	50	20	441	121	32	364	76	11
P14	37	50	55	117	141	119	59	116	59
…	…	…	…	…	…	…	…	…	…
Total	2191	1721	954	3203	3219	1720	2817	2572	1536
% Misses	2.6	2.0	1.1	3.8	3.8	2.0	3.4	3.1	1.8

4.3. Multi-step scheme trained with the TP learning rule

The second constructive technique uses the TP learning rule (Frean, 1992) to modify the weights instead of the LVQ algorithm. The result is a significant improvement in performance compared with the MLVQ. The TP, which is a simple extension of Rosenblatt's perceptron learning rule for training individual linear threshold units, finds stable weights for linearly separable as well as non-separable pattern sets. The learning rule stabilizes the weights (learns) over a fixed training period, and tries to classify correctly as many patterns as possible.

The perceptron convergence theorem (Minsky and Pappert, 1969) states that if there exist sets of weights for which the classification is perfect, then the PLR converges to one such set after a finite number of pattern presentations. The set of patterns for which such a weight set exists is known to be linearly separable. Clearly, when the pattern set is not linearly separable (i.e. when there is no weight set for which the classification is perfect), the weights do not converge under PLR. In such a case, it would be worth while to come up with a learning rule by which the weights converge while as many patterns as possible are correctly classified.

In the following, the *i*th element of an input pattern is denoted by ξ_i, and the associated weight by W_i. In response to a given pattern, a perceptron goes into one of the two output states given by:

$$o = \begin{cases} 1(\mathrm{ON}) & \text{if } \phi > 0 \\ 0(\mathrm{OFF}) & \text{otherwise} \end{cases}$$

where $\phi = \sum_{i=1}^{N} W_i \xi_i$ and N is the dimension of the pattern vector.

The difficulty with the PLR is that whenever ϕ is too large, then the changes in the weights needed to correct this error often misclassify the other patterns that were correctly classified previously. One solution to overcome this problem is to make the change in weights dependent on ϕ in such a way that the weight changes are biased towards correcting errors for which ϕ is small. The thermal perceptron (TP) learning rule is used as follows:

$$\Delta W_i = \alpha(t^{\mu} - o^{\mu})\xi_i^{\mu} \mathrm{e}^{-|\phi|/\Theta}$$

The parameter Θ determines the strength of attenuation for large ϕ: a high value of Θ means less attenuation, and a low value of Θ more attenuation. If Θ is very close to zero, the weights are frozen at their current values. In practice, Θ is started off with some high value, and gradually reduced to zero with iteration number, thus stabilizing the weights. When used for linearly non-separable patterns, the TP learning rule would give a stable set of weights, with the property that as many patterns as possible would be correctly classified.

The TP learning rule is adapted to develop the thermal multi-step learning vector quantization (TLVQ). When it is applied to linearly non-separable patterns, it gives a stable set of weights, thereby classifying correctly as many patterns as possible. As a consequence, the above modification enables the network to improve its performance significantly. Furthermore, the size of the network is reduced, and in fact on most occasions only one layer of neurons is sufficient to classify the training set patterns completely.

To highlight the nature of the results obtained, only one set of peformance results is presented in Table 2. For other results, see Ganesh Murthy and Venkatesh (1996b). The testing is carried out using noiseless, noisy and pre-processed noisy patterns, and the training with 200 variations of each pattern, i.e. 200 × 28 patterns. The MLVQ

Table 3
Comparison of performance of various techniques for different training set sizes

Method	Number of patterns in the training set 10 × 28			Number of patterns in the training set 50 × 28			Number of patterns in the training set 100 × 28			Number of patterns in the training set 200 × 28		
	NL	10N	10NM	NL	10N	10NM	NL	10N	10NM	NL	10N	10NM
LVQ	10.0	11.6	11.3	4.0	5.1	4.7	4.1	5.2	5.0	2.6	3.8	3.4
MLVQ	15.7	18.0	16.8	6.5	8.1	8.5	3.0	4.6	4.4	2.0	3.8	3.1
TLVQ	12.6	13.1	13.8	4.1	5.0	5.9	2.2	3.3	2.9	1.1	2.0	1.8
MLP	21.07	28.45	26.21	5.26	10.4	9.0	2.5	6.6	5.6	1.21	4.6	3.87

needs five layers to classify the training set completely, but the TLVQ needs only one layer for this. To save space, results for a subset of patterns (P1 to P14) are shown, whereas the total misclassifications and % misses (the last two rows in Table 2) have been computed on the basis of the results for the full set of 28 patterns. The entries indicate the number of misclassifications out of 3000 test inputs for each pattern.

4.4. Discussion of the results

All the three techniques (LVQ, MLVQ and TLVQ) have been extensively tested for their performance with respect to a different number of patterns (typically, 10, 50, 100, 200 variations of each pattern as different cases) used for training. The test set consists of 3000 random variations of each pattern. Both noisy and pre-processed noisy patterns have been used for testing in addition to the noiseless patterns. Their performance has been tested for each of the different training cases, and a typical analysis of the results is presented later below.

The training set consists of 100 variations of each pattern, and the percentage misclassification for the noiseless case for the three techniques is 4.1, 3.0 and 2.2 respectively. Considering the amount of distortion allowed in the patterns (non-uniform scaling with scale factor between 1.0 and 2.0 and rotation between $-15°$ and $+15°$), a training set of 100 variations for each pattern is quite reasonable. In this case the MLVQ needs three layers to classify the training set. In remarkable contrast, the TLVQ can do the same with only one layer. The percentage misclassifications for the noisy test set for the three methods is 5.2, 4.6 and 3.3 respectively, whereas for the pre-processed noisy patterns it is 5.0, 4.4 and 2.9.

The time taken by MLVQ and TLVQ to complete the training depends on the number of layers they require to classify the training set completely. For MLVQ, there are 100,000 presentations of inp[][] for each layer, whereas for TLVQ the number is 300,000. The simulations were carried out on a HP-9000/715 workstation. Typically, for a training set of size 200×28, the MLVQ needs five layers and the time taken for training is 30 s. In contrast, for the TLVQ the corresponding figures are one layer and 37 s.

So far as published results on the application of NNs to OCR are concerned, the best reported (Ganesh Murthy & Venkatesh, 1995b) accuracy of 97.5% is achieved by a probabilistic neural network.

The above discussion demonstrates the efficacy of MLVQ and TLVQ techniques in comparison with the conventional LVQ. Note that the networks generated by MLVQ and TLVQ have to preserve additional information (viz. the distance of the nearest misclassified pattern) with each neuron, but the number of neurons involved is much fewer than that needed by the LVQ. Apart from being constructive techniques for building a network to accomplish the task, the MLVQ and TLVQ yield networks with better performance than that of the LVQ. Finally, comparison of the performance of the MLP with that of LVQ, MLVQ and TLVQ shows that the training time for the former is considerably more than that for the latter. Similar results have been obtained for the characters from the NIST database. See Ganesh Murthy and Venkatesh (1996b) for details. Table 3 summarizes the final results of all the four techniques (LVQ, MLVQ, TLVQ and MLP) for training sets of different sizes (see also Ganesh Murthy and Venkatesh 1995a). The entries in this table are percentage misclassifications on noiseless (NL), noisy (10N, i.e. 10% noise) and preprocessed noisy (10NM, i.e. 10%noise + morphology) test patterns.

5. Conclusions

A new efficient method for encoding patterns has been proposed, to provide a feature array as input to a self-organizing neural network (SONN) for classification of patterns. Encoding of patterns consists of overlaying them on a certain radial grid for conversion to a two-dimensional array to form the input to the SONN. After training, the neurons are labelled appropriately, and the plot of the labels of the array of neurons exhibits clusters, thereby justifying the method of encoding.

When the SONN is tested with both noiseless and noisy variations of the exemplars, it is found in general that accuracy in the recognition of patterns is ~90%. In an attempt to increase the speed and improve the performance of the network, three techniques based on LVQ have been proposed. These three techniques are found to be highly accurate (~95% recognition accuracy) and robust with respect to noise. More details and experimental results are found in Ganesh Murthy and Venkatesh (1996b) and Ganesh Murthy (1996).

Acknowledgements

The authors wish to thank the referees and the editor for their critical comments which have led to an improved, compressed version of the paper.

References

Frean, M. (1992). Thermal perceptron learning rule. *Neural Computation*, *4*, 946–957.

Fukumi, M., Omato, S., Taketo, F., & Kosaka, T. (1992). Rotation invariant pattern recognition. *IEEE Trans. on Neural Networks*, *3*, 272–279.

Fukushima, K. (1991). Hand written alphanumeric character recognition by the neocognitron. *IEEE Trans. on Neural Networks*, *2*, 355–365.

Ganesh Murthy, C. N. S. (1996). Pattern recognition techniques based on self-organization and learning vector quantization. Ph.D. Thesis, Indian Institute of Science, Bangalore.

Ganesh Murthy, C. N. S. & Venkatesh, Y. V. (1995a). A comparison of the performances of the backpropagation network and constructive learning algorithms based on LVQ for classification of encoded patterns. Technical Report, Department of Electrical Engineering, Indian Institute of Science, Bangalore.

Ganesh Murthy, C. N. S. & Venkatesh, Y. V. (1995b). Character recognition using encoded patterns as inputs to neural networks. In *Proc. National Conference on Neural Networks and Fuzzy Systems NCNNFS* (pp. 220–225). Anna University, Madras, India.

Ganesh Murthy, C. N. S., & Venkatesh, Y. V. (1996a). Modified neocognitron for improved 2-D pattern recognition. *IEE Proceedings I, Vision, Image and Signal Processing*, *143*, 31–40.

Ganesh Murthy, C. N. S. & Venkatesh, Y. V. (1996b). Classification of encoded patterns uisng constructive learning algorithms based on learning vector quantization (LVQ). Technical Report, Department of Electrical Engineering, Indian Institute of Science, Bangalore.

Ganesh Murthy, C. N. S., & Venkatesh, Y. V. (1994). Efficient classification by neural networks using encoded patterns. *Electronics Letters*, *31*, 400–403.

Ganesh Murthy, C. N. S. & Venkatesh, Y. V. (1991). Experimental investigations on the performance of neocognitron for 2-D pattern recognition. In *International Conference on Information Processing* (pp. 127–132). Singapore.

Ghorpade, S. W., & Ray, A. K. (1993). Connectionist network for feature extraction and classification of English alphabetic characters. In *ICNN'93 Munich: International Neural Network Conference*, Vol. 4 (pp.1606-1611).

Jian, H., & Chen, B. (1992). Recognition of handwritten Chinese characters via short line segments. *Pattern Recognition*, *25*, 5543–5552.

Khotanzad, A., & Hong, Y.H. (1990). Invariant image recognition by Zernike moments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *12*, 489–497.

Khotanzad, A., & Lu, J. (1990). Classification of invariant image representations using a neural network. *IEEE Trans. ASSP*, *38*, 1028–1038.

Kohonen, T. (1990a). *Self Organization and Associative Memory*. New York: Springer Verlag.

Kohonen, T. (1990b). *Learning Vector Quantization for Pattern Recongition*. Technical Report TKK-F-A602, Helsinki University of Technology, Finland.

Kojima, Y., Yamamoto, H., Kohda, T., Sakaue, S., Maruno, S., Shimeki, Y., Kawakami, K., & Mizutani, M. (1993). Recognition of handwritten numeric characters using neural networks designed on approximate reasoning architecture. *Proc. IJCNN*, *3*, 2161–2164.

Lippmann, R.P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, *4*, 4–22.

Minsky, M. & Pappert, S. (1969). *Perceptrons*. MIT Press: Cambridge, MA.

Nakanishi, I., & Fukui, Y. (1993). Pattern recognition using hierarchical feature type and location. *Proc. IJCNN*, *3*, 2165–2168.

Perantonis, S.J., & Lisboz, P.J.G. (1992). Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Trans. on Neural Networks*, *3*, 241–251.

Serra, J. (1982). *Image Analysis and Mathematical Morphology*. New York: Academic Press.

Srinivasa, N., & Jounch, M. (1992). A neural network model for invariant pattern recognition. *IEEE Trans. on Signal Processing*, *4*, 1595–1599.

Yoshida, T. (1993). Construction of a feature set for character recognition. *Proc. International Conference on Neural Networks*, *3*, 2153–2156.

You, S.D., & Ford, G.E. (1994). Network model for invariant object recognition. *Pattern Recognition Letters*, *15*, 761–768.

Zufira, P. (1993). Extended BP for invariant pattern recognition neural network. *Proc. IJCNN*, *3*, 2097–2100