FEATURE MAPS BASED WEIGHT VECTORS FOR SPATIOTEMPORAL PATTERN RECOGNITION WITH NEURAL NETS

Matthew M. Yen
California State University – Fresno
Fresno, California 93740-0009

Michael R. Blackburn
Hoa G. Nguyen
Naval Ocean Systems Center, Code 943
San Diego, California 92152-5000

ABSTRACT

A neural network algorithm is used to generate the spatial pattern classes for Spatiotemporal Pattern Recognition (SPR). This algorithm is known as Kohonen Feature Maps. Training vectors are presented to the network one at a time. The connection strength between the input and output nodes are adaptively updated. The adaptation process is associated with a decay of the adaptation rate as well as a shrinkage of the neighborhood for updating. The final values of connection strength represent the centroid of clusters of training patterns. The algorithm was tested with hypothetical data as well as hydrophone data. Functional forms and constants for the decay and the shrinkage were empirically determined. The algorithm performs well with broadband data than with narrow band data. Also, the algorithm works better with smaller number of pattern classes.

1 INTRODUCTION

The classification of spatiotemporal patterns such as waterfall display-type data is a common subject to many feature recognition problems. An example of waterfall data is the Fast Fourier Transformed speech data. The data are buffered in a two- dimensional array with time represented on one axis and the frequency bins on the other. For each new time instant, a new row is added to the buffer at one end, and another row scrolls off the opposite end of the buffer. Relevant applications include : recovery of communication symbols, radar waveforms, sonar signals, speech recognition and vision systems [1]. Biological vision is a spatiotemporal pattern recognition process involving the integration of many pattern fragments resulting from sequences of eye movements(saccades) [2]. In the case of radar and sonar signals, large numbers, in the range of thousands to millions, of basic template patterns have to be classified [3]. This area has been traditionally a research theme of mathematical statistics. The advent of the neurocomputer technology and its potential for direct implementation has inspired the design of many neural network architectures to perform such a pattern recognition task [3].

Hecht-Nielsen devised a matched filter bank neural network architecture based on Grossberg's avalanche structure [3,4 and 5]. To facilitate the implementation of matched filter architecture, Hecht-Nielsen [3] suggested to automate the pattern classes generation with self-organizing feature maps [6 and 7]. The required self-organizing tasks are: 1) the determination of the spatial weight vectors and 2) the organization of the temporal template.

The first task was accomplished via Kohonen's feature maps and the second was implemented with a heuristic learning rule. The arrangement of our spatiotemporal pattern classifier is depicted in Figure 1. The spatial weight vectors quantization is described in Section 2. Section 3 summarizes experimental data. Section 4 discusses the experimental results and their implications. The heuristic learning rule as well as the general performance of the avalanche matched filter will be presented in a future paper.

2 Learning Weight Vectors

In the avalanche matched filter architecture, the **z** vectors are the spatial component of the spatiotemporal reference patterns $P_1, P_2, \ldots, P_n$. They are the spatial weight vectors distributed over the pattern space and sufficiently describe the pattern environment. These weight vectors can be manually chosen based on the training set patterns in an *a priori* manner or they could be generated by the processing elements based on the self-organization principle. Kohonen described a learning algorithm that can efficiently perform vector quantization of the input pattern space, i.e.: the classification of all the input vectors. The Kohonen neural network clustering algorithm [6, 7 and 8] is repeated here for completeness:

1. Given a neural network of size $n \times m$, where n is the size of the input vector, i.e. the number of elements of each input vector; m is the number of processing elements in the avalanche matched filter bank. Randomly assign small values to z_{ij}'s. And normalize $\mathbf{z_i}$ to 1.
2. Let $\mathbf{Q}(t) = \{q_1(t), q_2(t), ..., q_n(t)\}, t = 1, \ldots, k$, represent the training vector. Present a new vector $Q(t)$ to the input nodes. Note **Q**'s are normalized so that $|\mathbf{Q}| = 1$.
3. Compute the distance d_j between the input vector pattern and the current weight vector, i.e.: $d_j = \sum_{i=0}^{n}(q_i(t) - z_{ij}(t))^2$.
4. Select the processing element j^* with minimum Euclidean distance d_j as the center, i.e. $c = j^*$. Find a neighborhood of c, $N_c(t)$, by choosing the processing elements whose Euclidean distances d are less than $R(t)$ from j^*. Update weight vectors within $N_c(t)$ by $z_{ij}(t+1) = z_{ij}(t) + a(t) * (q_i(t) - z_{ij}(t))$.
5. Go to step 2 and repeat step 3 and 4 until the weight vectors stop changing their values.

The adaptation parameter $a(t)$ governs the converging speed toward the asymtotic values of z_j's. R(t) determines the influence range of each weight vector z_j. As suggested by Kohonen, both $a(t)$ and $R(t)$ should decrease in time monotonically [8]. In our experiment $a(t)$ is given by $a(t) = K_a e^{(-t/T_a)}$, $K_a < 1$, where K_a is a constant for maximum amount of adaptation and T_a is the constant governing the decreasing rate of $a(t)$ during the presentation session. The size of $N_c(t)$ is determined by the empirical function $R(t)$, which is defined as, $R(t) = R_0 + K_r e^{(-t/T_r)}$, where R_0 and K_r are constants and T_r governing the shrinking rate.

3 Experimental Results

Only synthesized data testings are presented in this section. Tests results with hydrophone data are not included in here due to the nature of data. However, we will draw some general conclusions of the tests in section 5. Eight vectors are used for experiment. Each vector has four elements. (see Table 1 and Figure 1). Note the fourth vector and the last vector are the same. This choice is purposely made to test the performance of Kohonen's feature map technique on categorizing data. Even though the data set is parsimonious , it revealed some interesting characteristics of Kohonen Feature Maps technique. A series of experiments were conducted to perform the *vector quantization* as described in [7]. The experiments are to determine a set of appropriate values for the parameters used in Section 3 as well as to evaluate the performance of Kohonen Feature Maps on small data sets. We hypothesized that:

1. An ideal vector quantizer should produce a set of weight vectors which is identical to the training vectors if the number of categories is the same as the number of the training vectors ;
2. For the same vectors , the vector quantizer should coalesce them into one category.

As pointed out by Kohonen [6, p.132], the form of the algorithm used is a choice for mathematical simplicity. Therefore, we do not expect to find the 'optimum' values of the parameters. The values used in these experiments are purely empirical. Following are the parameters used: $a = K_a * e^{(-t/T_a)}$ and

$R = 1 + (m-2) * e^{(-t/T_r)}$, where m is the number of training vectors; K_a in most cases are 0.9; T_a is the decay constant of the learning rate, which ranges from 1000 to 10000 and T_r is the shrink constant for the updating neighborhood. Its magnitude is about 100.

Table 2 shows the weight vectors generated for different numbers of training vectors. Note that the generated weight vectors do not match with the training vectors except n = 3. Table 3 are the eight training vectors classified into categories of various coarseness. Note that the input vectors could be grouped visually into 4 classes: 1, 5 and 6; 3 and 7; 4 and 8; and 2 by itself. Kohonen's algorithm seems perform rather well with such classifications (see Table 3 and Figure 2). Also note that Kohonen's algorithm was designed to classify large amount of image data rather than simplified data such as in our experiments. Nonetheless, simple data tests provide an effective means to evaluate the performance of algorithm.

Results in Tables 2 and 3 were obtained after a large number of trials by adjusting the parameters. During the experimentation, we have attempted linearly decrease the learning rate as well as the neighborhood shrink rate. Neither performed as good as the exponentially decreased rates. We also experimented with assigning different initial values to the weight vectors. The algorithm generate consistant weight vectors regardless the initial values provided the number of presentation is sufficiently large. However, the weight vectors are sensitive to the presentation sequence of the training vectors as shown in Table 4.

4 Discussions and Conclusions

The values of constants were obtained through many iterations. Generally, the neighborhood size should start with one neuron less than the number of classes; the shrink constant T_r should be less than 50% of the planned presentation time and the decay constant T_a is about 10% of the shrink constant T_r.

Our experimental results have shown some interesting comparisions with that reported by Nasrabadi and Feng [7] on image compression:

1. Our learning rate constants are much higher than that used in image compression, 0.9 vs. 0.1. Our conjecture is that this learning constant is inversely proportional to the size of training data. This might be because faster adaptation can 'freeze' the weight vectors into suboptimal values. Such phenomenon is especially true with a large number of training data.

2. The time constant for the decreased learning rate can be as low as 500 as while in image compression a value of 10,000 is typical. Again, we contend this constant is size dependent;

3. The time constants for the shrink rate of the affected neighborhood in both cases are about 100. Note this value is somewhat size independent.

4. The neighborhood constant $m-2$, where m is the number of elements, reflects the adjustment on the size of updating neighborhood as the number of training vectors changes.

In Table 2 (a), seven patterns classified into eight weight vectors which are different. This is not a problem if out of all the eight vectors there are seven vectors match with the input (or training vectors). Generally, if one tries to overclassify the input vectors, one may be ended up with redundant vectors or vectors which do not belong to any class. The case that the number of weight vectors exceedes the number of categories has no practical applications.

In conclusion, the less the number of classes, the better the algorithm performs. The hypothetical data tests provide an effective means to evaluate the algorithm. The outcome of weight vectors are more sensitive to the neighborhood parameters, T_r and K_r, than the parameters used in the adaptation process, i.e. T_a and K_a. The fact that a wide range of values can be assigned to the adaptation parameters implies that there is no unique set of parametric values for optimum weight vectors. Nasrabadi and Feng [7] suggested that the optimum weight vectors might be obtained if the adaptation and the neighborhood parameters were decreased very slowly. However our experiments indicate that slowly decreasing those parameters do not warrant the optimum weight vectors for even small training set with only four patterns. It would be more

so in the case with a large number of training patterns. Such limitation can be attributed to the simplified formalism as pointed out by Kohonen [6]. Nonetheless the fact that optimum weight vectors can not be readily determined does not limit the uses of the Kohonen's Feature Maps for vector quantization so long as the algorithm is used for high level features extraction or categorization.

As for the testing with hydrophone data, we concluded that the algorithm classifies the broadband data better than it does for the narrow band data. Although the data are not presented here, one can easily verify this by examing the classification between the vectors with and without zero elements. Another experience with using Kohonen Feature Maps for hydrophone data testing is that: it is better not to normalize the training vectors for clustering. Patterns tend to lose their features once they are normalized. This is because normalization is often a process of variance attenuation. However, the weight vector **z** has to be normalized prior to their uses in the SPR procedures. The fact that the weight vectors are affected by the sequence of presentation has significant implication on SPR. Inconsistant weight vectors will be generated for differrent training events of the same class. Such inconsistancy could eventually affect the overall performance of spatiotemporal pattern recognition.

5 Acknowledgements

This project was partially sponsord by the 1989 Navy-ASEE Summer Faculty Research Program. The authors acknowledge the contribution of Leon Bodzin, Research Scientist at Code 943, for setting up the computing equipment for this project; Jason Durham, Computer Scientist Code 943, who provided assistances on algorithm and programming; John Zamora, Research Assistant at California State University, Fresno, who helped on the diagrams, word processing, and software modularization. The principle author also extends acknowledgement to both the American Society of Engineering Education and the sponsors at the Naval Ocean Systems Center, San Diego, CA.

6 References

1. R. Hecht-Nielsen, *Neural Analog Processing* Proc. Soc. Photo-Opt. Instrum. Eng. 360,180(1982).

2. M. R. Blackburn and H. G. Nguyen, *Biological Model of Vision for an Artificial System that Learns to Perceive Its Enviornment.* 1st International Joint Conference of Neural Networks, 1989.

3. R. Hecht-Nielsen, *Nearest Matched filter Classification of Spatiotemporal Patterns.* Applied Optics 26, May 1987, pp. 1892-1899.

4. G. A. Carpenter and S. Grossberg, *A Massively Parallel Architecture for a Self-Organizing Neural Pattern recognition Machine.* Computer Vision, Graphics & Image Processing, 1987, 37, pp. 54-115, 1987.

5. G. A. Carpenter and S. Grossberg, *ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns.* Applied Optics: special issue on neural networks, 1987.

6. T. Kohonen, *Self-Organization and Associative Memory.* Spring- Verlag, 1988.

7. N. M. Nasrabadi and Y. Feng, *Vector Quantization of Images Based Upon the Kohonen Self-Organizing Feature Maps.* ICNN Vol. 1, 1988, I. pp. 101-105.

8. T. Kohonen, *The "Neural" Phonetic Typewriter.* IEEE Computer Magazine, pp. 11-22, March, 1988.

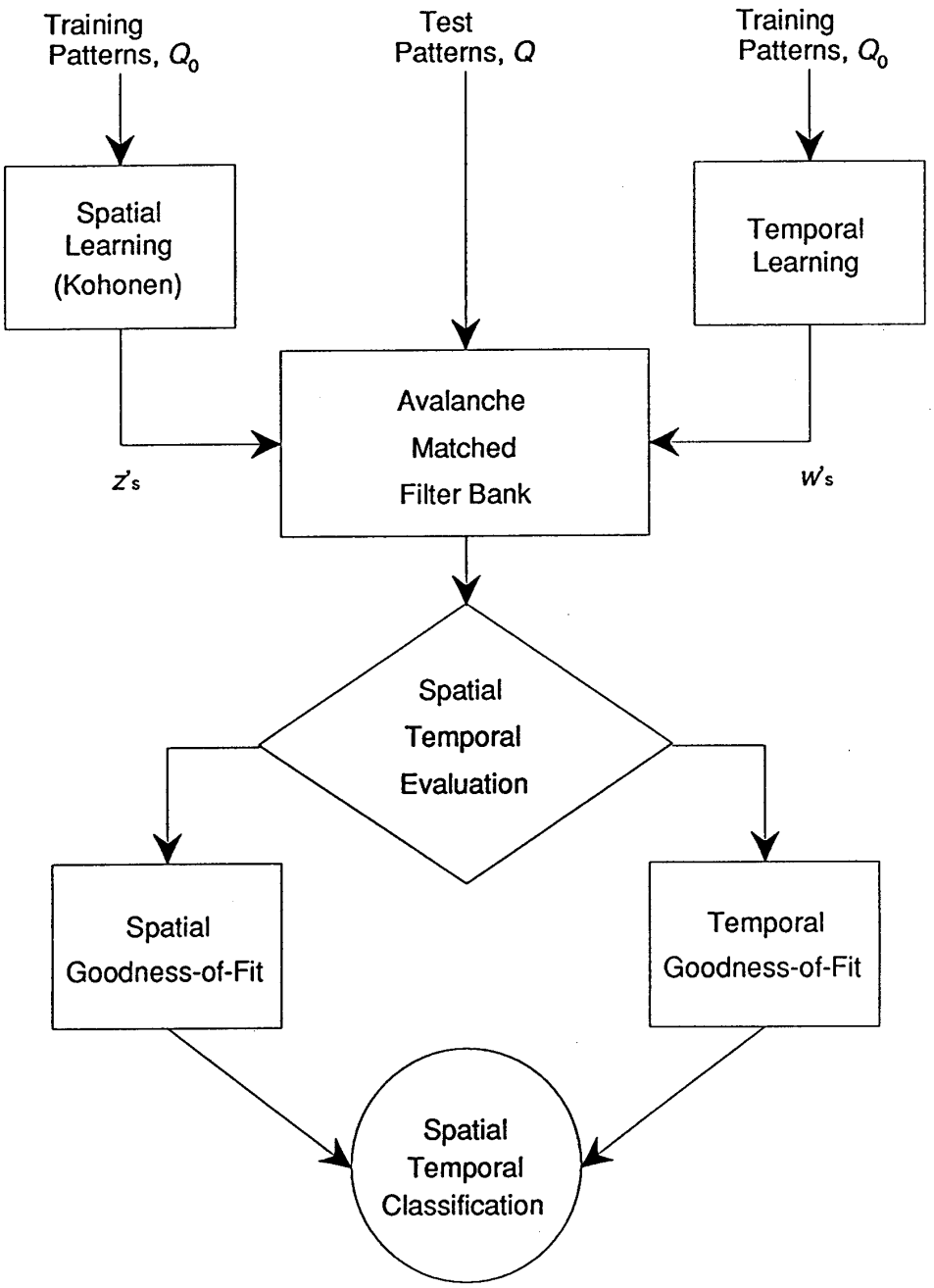

Figure 1

Spatiotemporal Pattern Recognition Flow Diagram

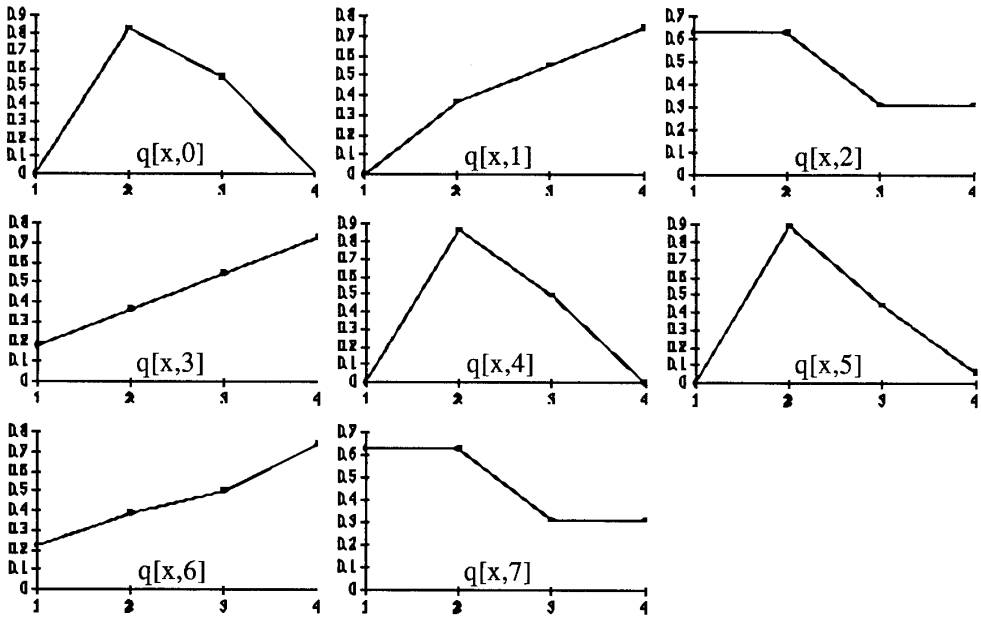

Figure 2 Normalized Training Patterns

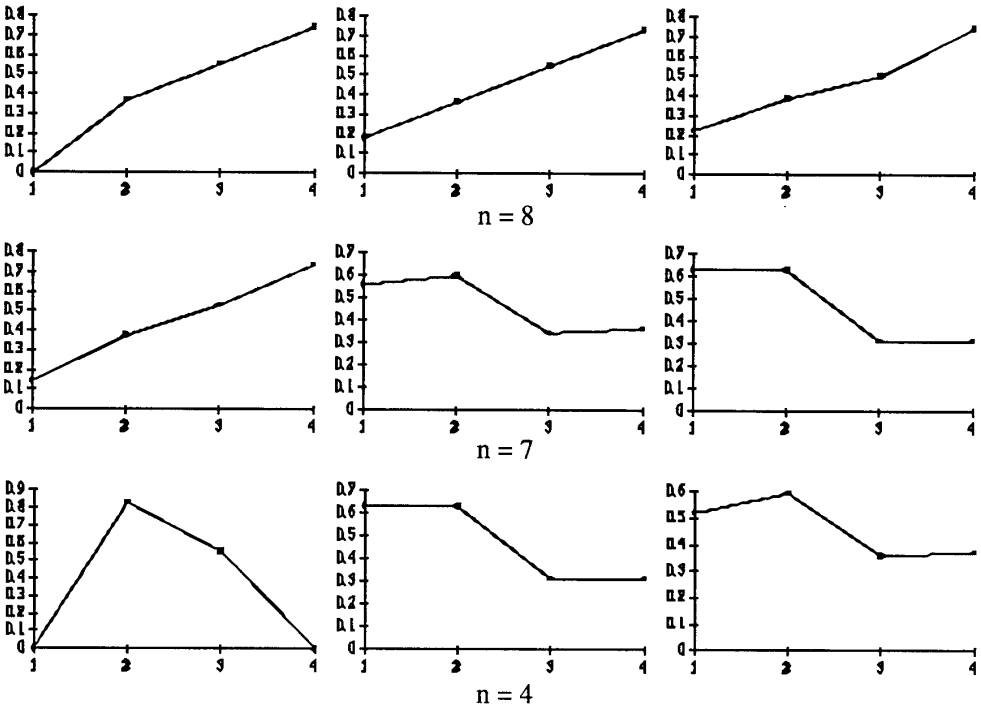

Figure 3 Weight vectors generated by Kohonen's Feature maps

(a) Unnormalized

qt[0,0] =0.00	qt[1,0] =3.00	qt[2,0] =2.00	qt[3,0] =0.00
qt[0,1] =0.00	qt[1,1] =4.00	qt[2,1] =6.00	qt[3,1] =8.00
qt[0,2] =2.00	qt[1,2] =2.00	qt[2,2] =1.00	qt[3,2] =1.00
qt[0,3] =1.00	qt[1,3] =2.00	qt[2,3] =3.00	qt[3,3] =4.00
qt[0,4] =0.00	qt[1,4] =3.50	qt[2,4] =2.00	qt[3,4] =0.00
qt[0,5] =0.00	qt[1,5] =3.00	qt[2,5] =1.50	qt[3,5] =0.20
qt[0,6] =1.20	qt[1,6] =2.10	qt[2,6] =2.70	qt[3,6] =4.00
qt[0,7] =2.00	qt[1,7] =2.00	qt[2,7] =1.00	qt[3,7] =1.00

Table 1. Training patterns.

(a) n = 8

z[0,0]=0.000	z[1,0]=0.371	z[2,0]=0.557	z[3,0]=0.743
z[0,1]=0.183	z[1,1]=0.365	z[2,1]=0.548	z[3,1]=0.730
z[0,2]=0.222	z[1,2]=0.389	z[2,2]=0.500	z[3,2]=0.741
z[0,3]=0.632	z[1,3]=0.632	z[2,3]=0.316	z[3,3]=0.316
z[0,4]=0.008	z[1,4]=0.887	z[2,4]=0.450	z[3,4]=0.056
z[0,5]=0.008	z[1,5]=0.887	z[2,5]=0.450	z[3,5]=0.056
z[0,6]=0.000	z[1,6]=0.851	z[2,6]=0.524	z[3,6]=0.000
z[0,7]=0.000	z[1,7]=0.893	z[2,7]=0.446	z[3,7]=0.060

(b) n = 3

z[0,0]=0.000	z[1,0]=0.371	z[2,0]=0.557	z[3,0]=0.743
z[0,1]=0.632	z[1,1]=0.632	z[2,1]=0.316	z[3,1]=0.316
z[0,2]=0.000	z[1,2]=0.832	z[2,2]=0.555	z[3,2]=0.000

Table 2. Weight vectors generated by Kohonen's Feature Maps for the first n training vectors in Table 1, n is reduced from eight to three. K_a = 0.90, T_a = 1000 , T_r = 100 Note the number of classes is the same number of the training vectors. Note also the weight vectors generated are not identical to the training vectors except n = 3. The redundant vectors in (a) are due to the over-classification of the input vectors.

(a) c = 7

z[0,0]=0.000	z[1,0]=0.371	z[2,0]=0.557	z[3,0]=0.743
z[0,1]=0.183	z[1,1]=0.365	z[2,1]=0.548	z[3,1]=0.730
z[0,2]=0.222	z[1,2]=0.389	z[2,2]=0.500	z[3,2]=0.741
z[0,3]=0.632	z[1,3]=0.632	z[2,3]=0.316	z[3,3]=0.316
z[0,4]=0.008	z[1,4]=0.887	z[2,4]=0.450	z[3,4]=0.056
z[0,5]=0.000	z[1,5]=0.851	z[2,5]=0.524	z[3,5]=0.000
z[0,6]=0.000	z[1,6]=0.893	z[2,6]=0.446	z[3,6]=0.060

(b) c = 3

z[0,0]=0.157	z[1,0]=0.377	z[2,0]=0.529	z[3,0]=0.738
z[0,1]=0.632	z[1,1]=0.632	z[2,1]=0.316	z[3,1]=0.316
z[0,2]=0.000	z[1,2]=0.871	z[2,2]=0.488	z[3,2]=0.026

Table 3. Weight vectors generated by Kohonen's Feature Maps for all eight input vectors in Table 1 clustered into different number of classes, c . Here the number of classes is less than the number of the training vectors, n. K_a = 0.90, T_a = 10000 , T_r = 100 Note that the weight vectors are not indentical to the training vectors when c = n = 8, however, it does appear to extract the features of the as the number of classes, c , is reduced.

q[0]=0.000	q[1]=0.832	q[2]=0.555	q[3]=0.000
q[0]=0.000	q[1]=0.371	q[2]=0.557	q[3]=0.743
q[0]=0.183	q[1]=0.365	q[2]=0.548	q[3]=0.730
q[0]=0.632	q[1]=0.632	q[2]=0.316	q[3]=0.316
q[0]=0.000	q[1]=0.866	q[2]=0.496	q[3]=0.000
q[0]=0.000	q[1]=0.893	q[2]=0.446	q[3]=0.060
q[0]=0.222	q[1]=0.389	q[2]=0.500	q[3]=0.741
q[0]=0.632	q[1]=0.632	q[2]=0.316	q[3]=0.316

(a) c = 7

z[0,0]=0.632	z[1,0]=0.632	z[2,0]=0.316	z[3,0]=0.316
z[0,1]=0.567	z[1,1]=0.605	z[2,1]=0.341	z[3,1]=0.361
z[0,2]=0.000	z[1,2]=0.371	z[2,2]=0.557	z[3,2]=0.743
z[0,3]=0.183	z[1,3]=0.365	z[2,3]=0.548	z[3,3]=0.730
z[0,4]=0.222	z[1,4]=0.389	z[2,4]=0.500	z[3,4]=0.741
z[0,5]=0.196	z[1,5]=0.447	z[2,5]=0.495	z[3,5]=0.661
z[0,6]=0.000	z[1,6]=0.867	z[2,6]=0.494	z[3,6]=0.022

(b) c = 3

z[0,0]=0.632	z[1,0]=0.632	z[2,0]=0.316	z[3,0]=0.316
z[0,1]=0.144	z[1,1]=0.376	z[2,1]=0.533	z[3,1]=0.738
z[0,2]=0.000	z[1,2]=0.867	z[2,2]=0.494	z[3,2]=0.022

Table 4 Same Training patterns except that the third and the fourth vectors are reversed during the training session. K_a = 0.90, T_a = 10000 , T_r = 100 Note that the weight vectors are different from the ones generated in Table 3 even with the same parameter set.