Efficient Feature Extraction Method Applied to the OCR of Persian Digits

Farnad Laleh and Ahmad R. Mirzai

Aria Technology Group
&
SECOMP Research Lab., Dept. of Computer Eng., IUST

ABSTRACT

In this paper, a method will be described for reduction of an n-dimensional feature vector into a 2-dimensional feature vector. Reaching for this goal, a structure is introduced, referred to as the chaining structure (CS), which is generated from the initial n-dimensional feature vector. The proposed technique can be thought as a feature extraction method. The simplicity and the consistency of the technique beside the fact that the resulted feature set is of 2-dimension, are the main advantages of the proposed method. It will also be illustrated how a specially designed neural network can be used to implement the proposed method. The efficiency of the proposed feature extraction algorithm will be illustrated by applying the method to the OCR of handwritten Persian digits. Finally, it will be compared with principal component analysis.

Keywords: pattern recognition, feature extraction, data projection, neural networks

1. INTRODUCTION

A large number of numeric methods have been proposed for feature extraction and data projection. These methods are based on space reduction or space conversion, e.g., Sammon's nonlinear projection [1, 2], principal component analysis [3, 4] and linear discriminant analysis [1, 5, 6]. The efficiency of these methods has been studied by J. Mao and A. Jain [7].

In this paper, a method for space reduction with two interesting properties will be proposed. The first property is the maximum reduction, i.e., it can convert an n-dimensional feature vector into a 2-dimensional feature vector. The second property is the simplicity of the method so that its computational time is notably smaller than the previous methods. It will also be illustrated how the efficiency of the method can be increased by constraining a mapping algorithm in order to achieve maximum separation between the classes in the resulted 2-dimensional space. Finally, a specially designed neural network will be introduced for implementing the proposed method.

In section 2, the concept of the chaining structure, as used in this paper, will be introduced and the feature extraction algorithm will be explained. Next in section 3, a method will be described to improve the efficiency of the proposed algorithm in order to get maximum separation in the resulting 2-D feature space. Section 4 will present a specially designed neural net for the implementation of the method. Finally in sections 5 and 6, the efficiency of the method will be studied when the proposed method is applied to an OCR problem and then it will be compared with the principal component analysis in a cardiogram signal clustering problem.

Further author(s) information:
<Farnad Laleh> lalehf@ariatg.com; http://www.ariatg.com

2. CHAINING STRUCTURES

Most of the pattern recognition algorithms operate on a set of features, which have been extracted from the patterns to be recognized. In statistical pattern recognition algorithms, these features are real numbers and the features that are used to describe a pattern can be considered as a feature vector. In most applications these vectors have a large dimension and for simplifying the pattern recognition algorithms it is very desirable to reduce this dimension. In this section, a method is described that can be used to reduce an n-D feature vector into a 2-D feature vector.

In the proposed method a chaining structure (CS) is used which is a 2-D graphical representation of a feature vector. In the 2-D feature space, a CS is constructed using n arrows, where n is the number of features in the original feature vector, as shown in figure 1. The arrows are all of unit length and the angle of each arrow to the x-axis is calculated according to the following steps. Suppose $A = [a_1, a_2 ... a_n]$ is an n-D feature vector of real numbers.

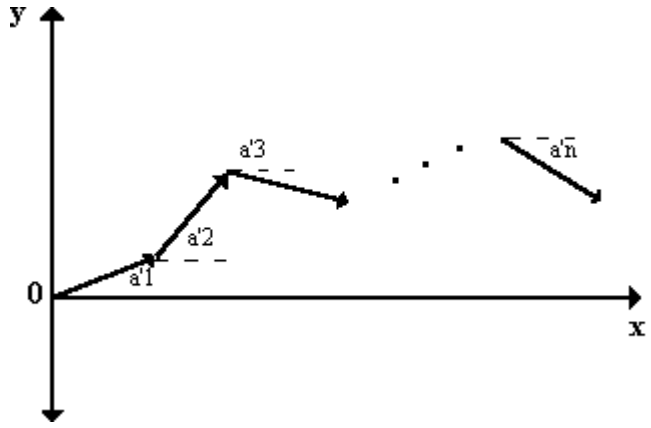

Fig. 1- General form of the CS

Step 1: Normalize the feature vector, i.e., divide each feature by the length of the n-D vector as indicated below:

$$a'_i = \frac{a_i}{sqrt(\sum a_k^2)} \qquad (k=1 \text{ to } n) \qquad (1)$$

After this step, a normalized vector $A' = [a'_1, a'_2 ... a'_n]$ is obtained with features between -1 and 1. If all of the features of original feature vector are between -1 and 1, there is no need to the normalization.

Step 2: Construct the CS of the normalized feature vector by using the value of the a'_i, i.e., i^{th} feature in the normalized feature vector, in radian as the angle between the i^{th} arrow and the x-axis.

As an example, in Figure 2, two CS's are shown that are obtained from the following feature vectors:
A = [6, 2.3, -1, 6] & B = [-5, 2.5, -2, 1].
These vectors are normalized. The normalized vectors are:
A' = [0.678, 0.256, -0.113, 0.678] & B' = [-0.830, 0.415, -0.332, 0.166].

The proposed method of normalization has one major problem. For two original feature vectors with the same direction in an n-D space, the normalized vectors are equivalent. In order to avoid this problem, the normalizing method can be modified. One way is to consider the length of the vector as an additional feature in the original feature vector.

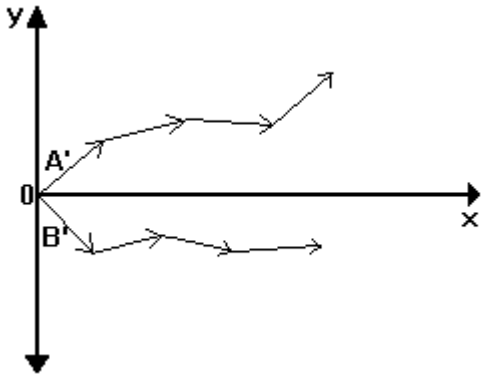

Fig. 2- Two CS's that are constructed from 4-D feature vectors

It is quite evident that this mapping is a one-to-one mapping, i.e., a given feature vector will generate a unique CS and if two feature vectors generate one CS they must be same.

The resulted CS's can be represented by a point in different ways in the 2-D space. Since they all start from the origin of the 2-D space, the beginning of the CS's can not be used. Also since there may be CS's, which would end at the same place in the 2-D space, the end points of the CS's can not be used to represent them in the 2-D space. One simple approach would be to calculate the center of mass of the CS. The center of mass is a 2-D numeric feature of the CS.

There are two main problems in using the center of mass. First, there may be situations when two CS's will result a same center of mass. Secondly, the classes may overlap when transferred from the n-D space into the 2-D space. To overcome these problems a weighting skim can be introduced where each arrow will be weighted so that the center of mass for every CS is unique. Also the inner distances between the centers of mass belonging to one class are minimized when the distances between different classes are maximized. The center of mass of each arrow is considered in the middle of it.

Equations 2 and 3 display the formula of the center of mass for x-axis and y-axis [8], respectively, where a'_i is the i^{th} component of the feature vector and W_i is the weight corresponds to the i^{th} arrow. In these equations, the length of the arrows is considered equal to 2. As can be seen, the denominator of these equations does not depend on the input feature vector. Therefore the denominator of the equations can be disregarded.

$$X_{CM} = \frac{\cos(a'_1)W_1 + (2\cos(a'_1)+\cos(a'_2))W_2 + \ldots + (2\cos(a'_1)+2\cos(a'_2) + \ldots + \cos(a'_n))W_n}{\sum W_i} \quad (2)$$

$$Y_{CM} = \frac{\sin(a'_1)W_1 + (2\sin(a'_1)+\sin(a'_2))W_2 + \ldots + (2\sin(a'_1)+2\sin(a'_2) + \ldots + \sin(a'_n))W_n}{\sum W_i} \quad (3)$$

In equations 4 and 5, the denominator has been eliminated. In this paper, we use equations 4 and 5 as the center of mass of the CS.

$$X_{CM} = \cos(a'_1)W_1 + (2\cos(a'_1)+ \cos(a'_2))W_2 + \ldots + (2\cos(a'_1)+ 2\cos(a'_2) + \ldots + \cos(a'_n))W_n \quad (4)$$

$$Y_{CM} = \sin(a'_1)W1 + (2\sin(a'_1)+ \sin(a'_2))W_2 + \ldots + (2\sin(a'_1)+ 2\sin(a'_2) + \ldots + \sin(a'_n))W_n \quad (5)$$

As mentioned before, a weight has been assigned to each of the arrows in the CS. These weights can be represented as a vector that its length is equal to the length of original feature vector. This vector is named *weight vector*. An important question rises here is whether the weighting skim can guarantee a one-to-one mapping, i.e., each of the resulted 2-D features just corresponds to one of the original n-D vectors. The following corollary discusses about this subject.

Corollary: Suppose that n feature vectors are given. Each feature vector has m real-number features. In fact, a matrix with n row and m column exists. This matrix is referred to as the data set. Each of these feature vectors is converted to its CS model, then its center of mass is determined. It is proved that there is a weight vector, which can guarantee a one-to-one mapping.

Proof: Suppose that there is no weight vector that satisfies above condition, i.e., for every arbitrary weight vector at least two feature vectors have a same center of mass. Therefore all weight vectors are portioned at most into $C(n, 2)$ or $\frac{n \times (n - 1)}{2}$ (the number of double combinations of feature vectors) classes by the formula of the center of mass.

A weight vector that belongs to a specific class, causes a couple of feature vectors have the same center of mass. If we can demonstrate that there is at least one weight vector that belongs to none of these classes, then the corollary will be proved. Because, there is no couple of vectors with the same center of mass for this weight vector. Reaching for this goal, we must find the relation between the weight vectors in each of the classes. Considering the formula of the center of mass, it is easily found that in each class the following relation exists:

$\forall\ i, j \in N,\ W_j = \alpha_{ij} \times W_i$, where W_i and W_j are two weight vectors and α_{ij} is a real number and '$\times$' is scalar multiplier.

This relation demonstrates all of the weight vectors, which belong to a class have a same direction. There are many (infinite) weight vectors, which have not the same direction with the weight vectors of these classes. Choosing one of them, a weight vector is selected that does not belong to these classes. Therefore the corollary is proved.

3. THE CONVERGENCE ALGORITHM

After proving the existence of weight vectors that guarantee one-to-one mapping, it is important to find the best weight vector. It means we must find a weight vector, which causes the maximum separation between different classes in a given data set. There is no exact equation for finding such a weight vector. In this case, the most popular approach would be to use an algorithm for approximating the proposed weight vector. Reaching for this goal, different types of optimization algorithms can be used to generate a proper weight vector. In this paper, an algorithm based on *Delta rule* (gradient descent) [9] is designed, which is called *fragmentation minimizing algorithm (FMA).*

The FMA determines the weights so that the centers of mass of the CS's belonging to one class will be close together while the outer distances between the classes will be maximized. Note that the weights may be negative numbers. The center of mass of the CS has two components, i.e., X (horizontal) and Y (vertical) components in the Cartesian plane. There are some situations in which for a specific weight vector the horizontal components of different classes in the data set may overlap, while the vertical components may not or vice versa. Avoiding this problem and reaching for the maximum separation on both of the components, a couple of weight vectors is considered for the CS, one for the x-axis and the other for y-axis. However it is not necessary to consider two weight vectors instead of one, but it is offered for more efficiency. In this paper, we use this strategy. The FMA is as the following steps.

1. Assign a random value to each arrow as a weight.
2. Determine the average value of the centers of mass for each of the classes.
3. For each sample of a class modify the Wi ($1 \le i \le n$) using Delta rule [9] (see equations 6 & 7) so that the Euclidean distance between the center of mass of the sample and the average value of the proposed class will be minimized.

$$W_{xi}(t+1) = W_{xi}(t) + \alpha \times \Delta_x \times (2\cos(a'_1) + 2\cos(a'_2) + \ldots + \cos(a'_i)) \quad (6)$$

$$W_{yi}(t+1) = W_{yi}(t) + \alpha \times \Delta_y \times (2\sin(a'_1) + 2\sin(a'_2) + \ldots + \sin(a'_i)) \quad (7)$$

In these equations Δ is the error, i.e., the difference between the primary average value of the proposed class and the current value of the center of mass for the given sample and α is a positive small constant (less than one). Equations 6 and 7 are used for x-axis and y-axis, respectively.

4. Repeat step 3 for all of the classes.

5. If the error values for all of the classes are smaller than a threshold value, then terminate the algorithm, else continue with step 4 (Steps 4 and 5 are referred to as the *convergence or fragmenting minimizing* process).

In step 2, if the average values of the center of mass for two or more classes are close together, after using the FMA, these classes will overlap. To avoid this problem, in step 2, a supervisor can modify the average value of the classes so that an acceptable distance exists between each pair of the classes.

The time complexity of the proposed algorithm is $O(n^2)$, where n is the length of feature vectors. It is regardless of the number of feature vectors in the data set and the number of iterations required for the convergence process. It is very desirable to reduce the computational time of the proposed algorithm. For this purpose, we rewrite equations 4 and 5 in terms of the angles rather than the weights (equations 4, 5) as follows:

$$X_{CM} = (W_1 + 2W_2 + 2W_3 + \ldots + 2W_n)\cos(a'_1) + (W_2 + 2W_3 + 2W_4 + \ldots + 2W_n)\cos(a'_2) + \ldots + (W_n)\cos(a'_n) \quad (8)$$

$$Y_{CM} = (W_1 + 2W_2 + 2W_3 + \ldots + 2W_n)\sin(a'_1) + (W_2 + 2W_3 + 2W_4 + \ldots + 2W_n)\sin(a'_2) + \ldots + (W_n)\sin(a'_n) \quad (9)$$

Next, a new weight vector W' is considered which is obtained from the previous weight vector W. The relation between W and W' is represented as follows:

$$W'_1 = W_1 + 2W_2 + 2W_3 + \ldots + 2W_n$$
$$W'_2 = W_2 + 2W_3 + 2W_4 + \ldots + 2W_n$$
...

$$W'_n = W_n$$

If we use W' instead of W, then the equations 6 and 7 are modified into the following forms:

$$W'_{xi}(t+1) = W'_{xi}(t) + \alpha \times \Delta_x \times \cos(a'_i) \quad (10)$$

$$W'_{yi}(t+1) = W'_{yi}(t) + \alpha \times \Delta_y \times \sin(a'_i) \quad (11)$$

Using equations 10 and 11 in the FMA, the time complexity of the proposed algorithm will be reduced to $O(n)$. As mentioned before, it does not include the number of iteration required for the convergence process. It just expresses the required time for analyzing a feature vector in the data set. So the number of iterations required for the convergence process and the number of classes, only depend on the given data set. Using the FMA, a proper weight vector with desired properties will be generated. It guarantees our desired mapping from the n-D into a 2-D space.

4. SINUSOIDAL NEURAL NETWORK

In many practical applications of pattern recognition, the length of the original feature vectors is very long (up to 200) and the proposed data set may be complex. In these cases, a direct mapping from an n-D into a 2-D may lose some information. To overcome this problem, an approach will be to implement the mapping from the n-D into a 2-D in a distributed form, i.e. using several FMA's, simultaneously. Neural networks are popular tools for the implementation of feature extraction and data projection algorithms [7] and have been known as a general model of parallel distributed systems [10]. In the meantime, the optimization method, which is used in the previous section uses one of the major rules in learning neural nets. These facts motivate us to designing a special neural network for the implementation of the proposed mapping.

The proposed neural net includes a special type of neuron, which is called *sinusoidal neuron*. This type of neuron is always used in coupled form with another neuron of the similar type. The inputs of the coupled neurons are equal. It is supposed that the input vectors are normalized by the method mentioned in section 2, before feeding to the network. One of the coupled neurons calculates its output using equation 8 and the other neuron uses equation 9. The first neuron is named *cos-neuron* and the second is named *sin-neuron*. The outputs of coupled neurons are X_{CM} and Y_{CM}, respectively. The

weight vectors of a coupled sin-neuron and cos-neuron are not equal. In each layer of the network, the number of neurons is an even number.

The learning process of a layer in the sinusoidal network is independent from the other layers. The sinusoidal network is a one-layer network. For more efficiency, we cascade several layers to each other as a multi-layer network, so that the final layer has just a pair of coupled neurons. Figure 3 shows the general form of the SNN.

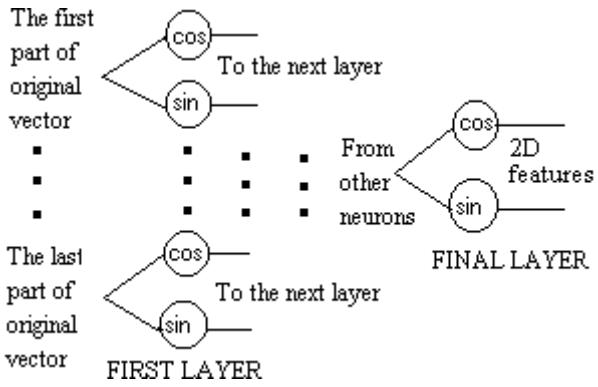

Fig. 3- General form of the SNN

As seen in figure 3, the input feature vector is divided into several sub-vectors. Each of the sub-vectors is fed to a pair of coupled neurons in the first layer. The lengths of the sub-vectors are not necessarily equal and must be determined according to the specific application. The output of the first layer is fed to the second layer as it is done for the first layer. At the final layer, a pair of coupled neurons is used that is fed from its previous layer. The output of these neurons is the output of the network. Note that the number of layers and the number of neurons in each layer is determined experimentally, according to the specific application. There is no mathematical formula to determine the number of neurons and layers.

The learning process begins from the first layer. Each of the coupled neurons, is learned using the FMA. After learning the neurons in the first layer, the output of the first layer is calculated for all of the input feature vectors. These output values are the input feature vectors for the second layer. These new feature vectors are normalized and are fed to the second layer. The learning process will be done for the second layer as it is done for the first layer. This work is done for all of the layers. The learning process of the network will be completed, when the final layer is learned.

5. THE OCR APPLICATION

One of the popular applications of image processing systems is the optical character recognition (OCR). There are many different approaches and algorithms to the solution of OCR problems, each of them is appropriate for a special kind of characters. An important stage in every OCR problem is feature extraction, specially when the characters are handwritten, where the features are varied substantially. To demonstrate the ability of the CS and SNN, we applied them to the OCR of handwritten Persian digits, where the conventional approaches such as neural nets, fuzzy systems and structural pattern recognition algorithms cannot satisfy the problem well [11]. The following figure shows a sample of handwritten Persian digits.

In this problem, the original feature vectors are made of the contour of shapes and represented by chain codes, which is a kind of boundary extraction [12]. In chain coding the direction between successive boundary pixels are encoded. In our proposed OCR problem, the original feature vectors, which are generated by chain coding, have a maximum length of 200. As can be seen in figure 4, there are nine digits without considering zero.

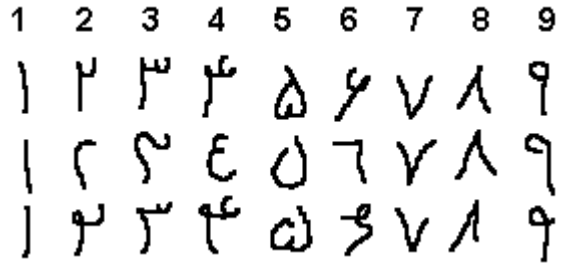

Fig. 4- A sample of handwritten Persian digits

In this problem, our proposed data set contains 45 samples, 5 samples for each digit. The employed SNN just contains a pair of coupled neurons. The FMA, which is used for learning the network, uses random value between -100 and 100 at its first step. The required iterations for learning the proposed data set is 5000 steps with α = 0.000001. Figure 5 displays the output of the network before learning. As can be seen, the 2-D features belonging to different classes have messed together. In Figure 6, the output of the SNN is shown after learning, where the outer distances between the different classes are maximized. In these figures, the area of each class is surrounded by a broken line. The number that indicates the area of each class is written at the first of each broken line.

6. COMPARISON

The main advantages of the proposed neural network (the SNN) are its fast learning algorithm, i.e. the FMA, its reduced number of neurons with respect to the previous methods [7, 13] and its simple interconnections. As mentioned before, the number of neurons in the OCR problem is just two neurons that is exciting for such a complex problem. Also this network gives the maximum space reduction, because it maps all of the vectors into the 2-D space. The SNN can be considered as a general method for space reduction and data projection. So it is required that a comparison between the resulted features of the SNN and the resulted features of the previous methods will be given. For this purpose, we compare the SNN with the principal component analysis by an example.

We use a 4-D data set, which has been extracted from a cardiogram. In this paper, we do not care to the method of extracting the original 4-D feature vectors. It is required to reduce the proposed data set into a 2-D data set with maximum separation between the different classes. There are three classes in the proposed data set and it includes 75 feature vectors, 25 vectors for each class. The proposed data set has been applied to a PCA network. The resulted 2-D space is shown in figure 7. Three different classes are shown by '×', '+' and 'O', respectively. The resulted 2-D features are aligned and the 2nd class has overlapped with the 3rd class in a small area.

The proposed data set has been applied to a two-neuron SNN. The SNN has been learned in 600 step with α = 0.005. Figure 8 shows the resulted 2-D space by the SNN. As can be seen in figure 8, the resulted 2-D features are not aligned same as the resulted features by the PCA net. But the features belonging to one class have nearly circulated around the average value of the class. The 2nd and 3rd classes have overlapped, but the overlapped area is considerably smaller than the similar area in the PCA method.

7. DISCUSSION

In this paper, we studied a method for reducing the dimension of feature vectors from an n-D space into a 2-D space. An especially designed neural network was employed to implement this method which is called SNN (sinusoidal neural network). An important note must be mentioned, is that the SNN must not be confused with the other types of neural nets such as *functional link networks (FLN).*

In the FLN [9] for speeding up the learning process of network and reducing the number of layers, the input vector passes through a functional link before distributing to the units. In some cases, the functional links use “sin” and “cos” functions. One must note that this method is inherently different to the SNN.

In the future works, two major fields of research are proposed. At the first approach, one can optimize the FMA (fragmenting minimizing algorithm) so that the number of required iterations will be decreased. At the second approach, the convergence algorithm can be changed substantially, i.e., another optimization method such as genetic algorithms or simulated annealing can be used in order to find an appropriate weight vector. Afterward, a comparison between the different optimization methods should be given. Also a complete comparison between the SNN and all of the other data projection methods would be useful.

ACKNOWLEDGMENTS

This work was done as a part of Aria Technology SCS (structural computing systems) project. The authors appreciate M. Kamali and M.A. Zia for their useful comments and also thank to SECOMP lab’s members specially M. Afshari for the cardiogram data set and H. Moqasemi for the OCR data set.

REFERENCES

1. J. Mao and A. K. Jain, “Discriminant analysis neural networks,” in *Proc. IEEE Int. Conf. Neural Networks,* San Francisco, CA, March 1993, vol. 1, pp. 300-305.
2. J. W. Sammon, Jr., “A nonlinear mapping for data structure analysis,” *IEEE Trans. Comput.,* vol. C-18, pp .401-409, 1969.
3. J. H. Friedman and J. W. Tukey, “A projection pursuit algorithm for exploratory data analysis,” *IEEE Trans. Comput.,* vol. C-23, pp. 881-890, Sept. 1974.
4. P. J. Huber, “Projrction pursuit,” *Ann. Statist.*, vol. 13, pp. 435-475, 1985.
5. D. H Foley and J. W. Sammon, “An optimal set of discriminant vectors,” *IEEE Trans. Comput.,* vol. C-24, no. 3, pp. 281-289, March 1975.
6. T. Okada and S. Tomita, “An optimal orthonormal system for discriminant analysis,” *Patt. Recognition,* vol. 18, pp. 139-144, 1985.
7. J. Mao and A. K. Jain, “Artificial neural network for feature extraction and multivariate data projection,” *IEEE Trans. Neural Networks,* vol. 6, no. 2, March 1995.
8. D. Halliday and R. Resnick, *Physics*, John Wiley & Sons, NY, 1978.
9. J. A. Freeman, *Simulating Neural Networks with Mathematica*, First Ed. Addison Wesley, MA, 1994.
10. D.E. Rumelhart and J.L. MCclelland, *Parallel Distributed Processing,* MIT Press, MA, 1986.
11. M. Moatar and M. Vahedi, “Online OCR for handwritten Persion digits: comparison between different feature extraction methods,” *B.Sc project, SECOMP Lab., IUST,* 1996.
12. Anil k. Jain, *Fundamental of Digital Image Processing,* First Ed. Prentice-Hall, NJ, 1989.
13. J. Rubner and P. Tavan, “A self-organizing network for principal component analysis,” *Europhysics Lett. vol. 10, pp. 693-698, 1989.*

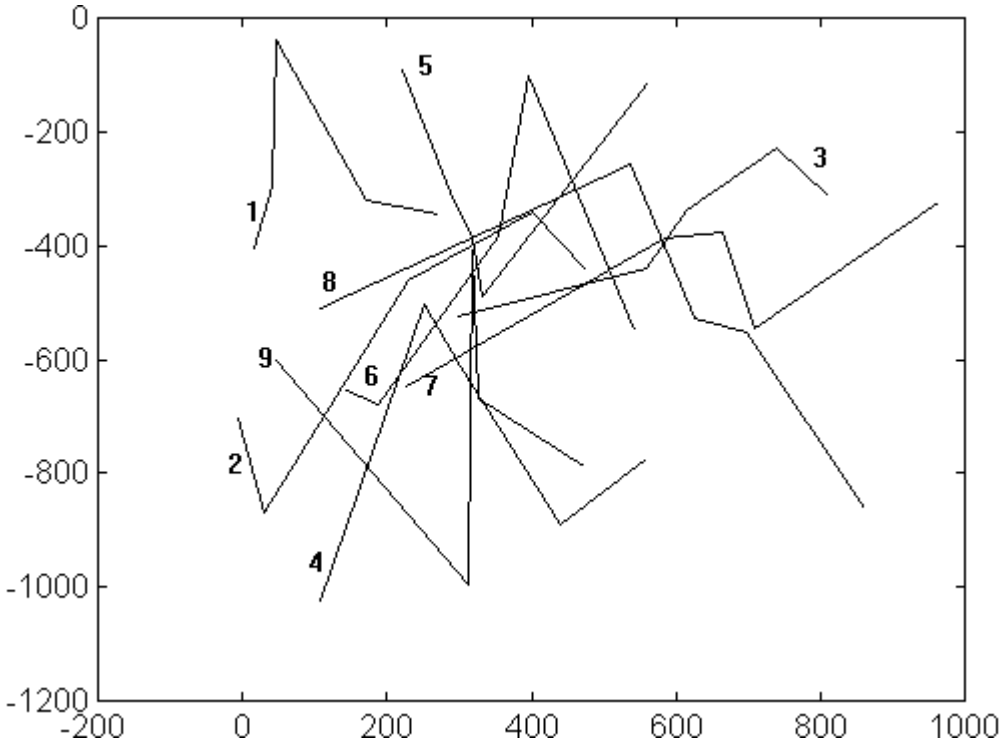

Fig. 5- Output of the SNN before learning in the proposed OCR problem

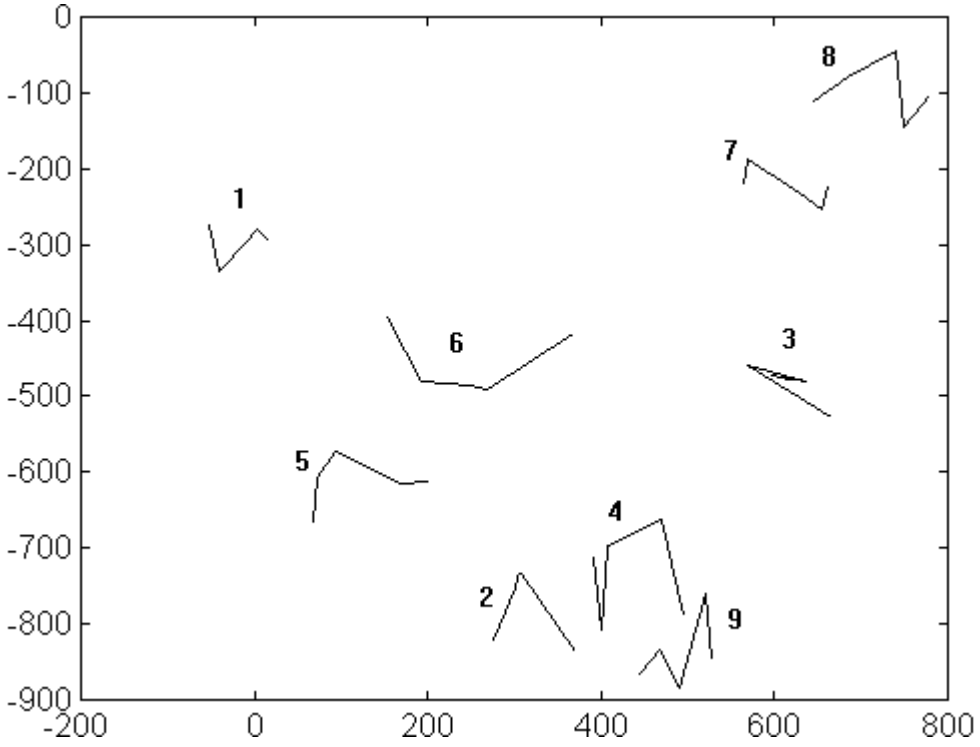

Fig. 6- Output of the SNN after learning in the proposed OCR application

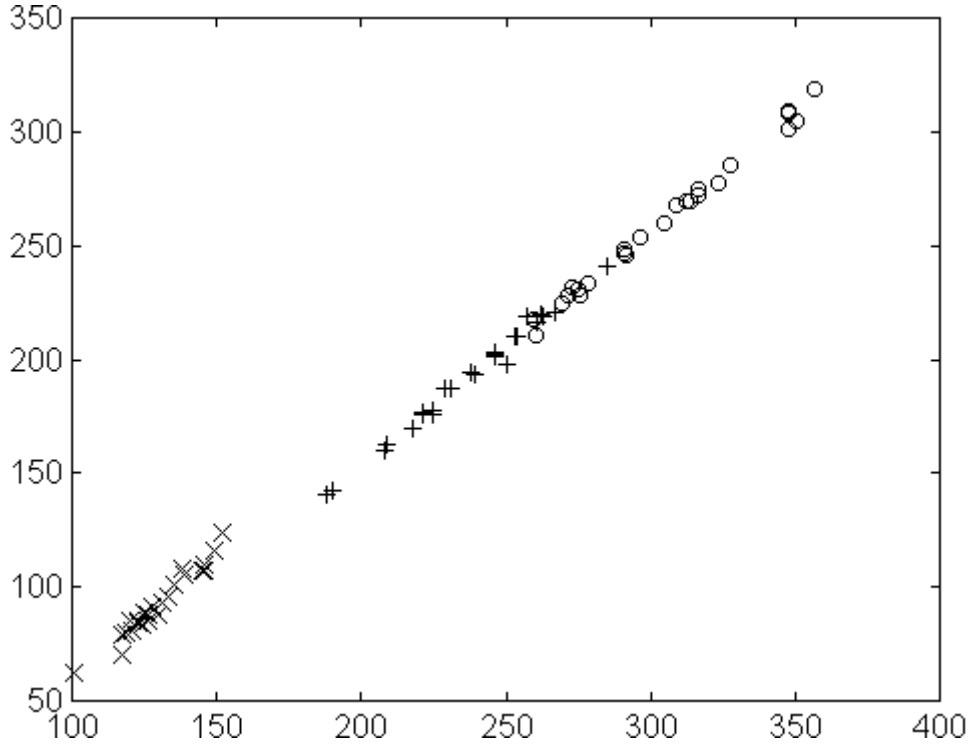

Fig. 7- 2-D feature space for the cardiogram data set resulted by the PCA net

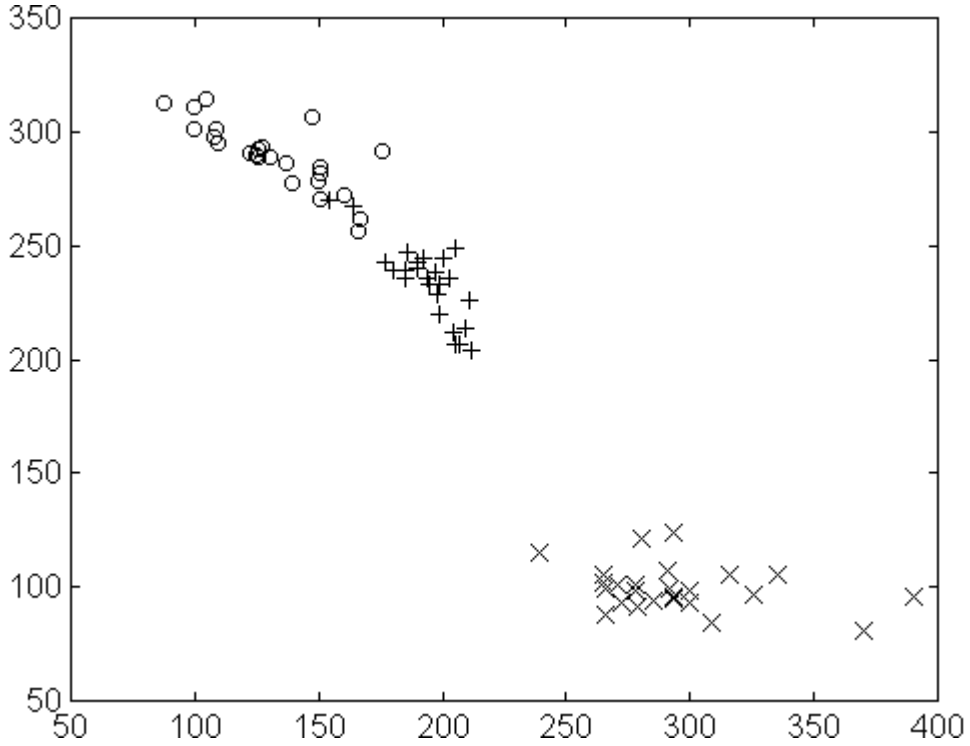

Fig. 8- 2-D feature space for the cardiogram data set resulted by the SNN