Neural Network and Adaptive Feature Extraction Technique for Pattern Recognition

WAEL M. KHEDR and QAMAR A. AWAD
Mathematical Department, Faculty of Science, Zagazig University (Egypt).
(Received: January 14, 2011; Accepted: February 18, 2011)

ABSTRACT

In this paper, we propose adaptive K-means algorithm upon the principal component analysis PCA feature extraction to pattern recognition by using a neural network model. Adaptive k-means to discriminate among objects belonging to different groups based upon the principal component analysis PCA implemented for statistical feature extraction. The features extracted by PCA consistently reduction dimensional algorithm, thus demonstrating that the suite of structure detectors effectively performs generalized feature extraction. The classification accuracies achieved using feature learning process of backpropagation neural network. A comparison of the proposed adaptive and previous non-adaptive ensemble is the primary goal of the experiments. We evaluated the performance of the clustering ensemble algorithms by matching the detected and the known partitions of the iris dataset. The best possible matching of clusters provides a measure of performance expressed as the misassignment rate.

Key words: Adaptive K-means algorithm, PCA, Neural network, Backpropagation algorithm.

INTRODUCTION

The design of a recognition system requires careful attention to the following issues: definition of pattern classes, pattern representation, feature extraction, cluster analysis, classifier design and learning, selection of training and test samples, and performance evaluation.

Interest in the area of pattern recognition has been renewed recently due to emerging applications which are not only challenging but also computationally more demanding. These applications include data mining, document classification, financial forecasting, organization and retrieval of multimedia databases, and biometrics (personal identification based on various physical attributes such as face and fingerprints).

Clustering analysis is a fundamental but important tool in statistical data analysis. In the past, the clustering techniques have been widely applied in a variety of scientific areas such as pattern recognition, information retrieval, microbiology analysis, and so forth. In the literature, the k-means is a typical clustering algorithm, which aims to partition N inputs (also called data points interchangeably).

The most commonly used family of neural networks for pattern classification tasks is the feed-forward network, which includes multilayer perceptron and Radial-Basis Function (RBF) networks. These networks are organized into layers and have unidirectional connections between the layers. Another popular network is the Self-Organizing Map (SOM), or Kohonen-Network, which is mainly used for data clustering and feature mapping. The learning process involves updating network architecture and connection weights so that a network can efficiently perform a specific classification/clustering task. The increasing
popularity of neural network models to solve pattern recognition problems has been primarily due to their seemingly low dependence on domain-specific knowledge.

Adaptive K-means cluster algorithm

The adaptive k-means algorithm is built on Kullback scheme [7], its details steps as follows:

Step 1

We implement this step by using Frequency Sensitive Competitive Learning [1] because they can achieve the goal as long as the number of seed points is not less than the exact number K of clusters. Here, we suppose the number of clusters is \( k \geq k^* \), and randomly initialize the \( k \) seed points \( m_1, m_2, \ldots, m_k \) in the input data set.

Step 1.1

Randomly pick up a data point \( x_t \) from the input data set, and for \( j = 1, 2, \ldots, k \), let

\[
\eta_j = \begin{cases} 1 & \text{if } j = \arg \min_i \| x_t - m_i \| \\ 0 & \text{otherwise} \end{cases} \tag{1}
\]

where \( \eta_j = \frac{\eta_j}{\sum \eta_j} \), and \( \eta \) is the cumulative number of the occurrences of \( u = 1 \).

Step 1.2

Update the winning seed point \( m_w \) only by

\[
m_w^{new} = m_w^{old} + \eta \left( x_t - m_w^{old} \right) \tag{2}
\]

Steps 1.1 and 1.2 are repeatedly implemented until the \( k \) series of \( \eta_j \) for \( j = 1, 2, \ldots, k \) remain unchanged for all \( x_t \) s. Then go to Step 2.

In the above, we have not included the input covariance information in Eqs. (2) and (3) because this step merely aims to allocate the seed points into some desired regions as stated before, rather than making a precise value estimate of them. Hence, we can simply ignore the covariance information to save the considerable computing cost in the estimate of a covariance matrix.

Step 2

Initialize \( \alpha_j = 1/k \) for \( j = 1, 2, \ldots, k \), and let \( \Sigma_j \) be the covariance matrix of those data points with \( u_j = 1 \). In the following, we adaptively learn \( \alpha_j \), \( m_j \) and \( \Sigma_j \).

Step 2.1

Given a data point \( x_t \), calculate \( I(j \mid x_t) \) s by Eq. (1).

Step 2.2

Update the winning seed point \( m_w \) only by

\[
m_w^{new} = m_w^{old} + \eta \frac{\partial R}{\partial m_w} m_w^{old} \tag{3}
\]

Table 1: Output of Kohonen Map’s network

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Digit</th>
<th>Pattern</th>
<th>Win nod index</th>
<th>Neigh. size</th>
<th>Average distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>56</td>
<td>5 0.107580</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>81</td>
<td>73</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>82</td>
<td>29</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>83</td>
<td>88</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>84</td>
<td>11</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>85</td>
<td>65</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>86</td>
<td>0</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>87</td>
<td>79</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>88</td>
<td>50</td>
<td>5</td>
<td>0.107580</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>89</td>
<td>23</td>
<td>5</td>
<td>0.107580</td>
</tr>
</tbody>
</table>
\[ m_w^{\text{new}} = m_w^{\text{old}} + \eta \Sigma_w^{-1} (x - m_w^{\text{old}}) \quad \ldots(4) \]

or simply by Eq. (3) without considering \( \Sigma_w^{-1} \). In the latter, we actually update \( m_w \) along the direction of \( \Sigma_w^{-1} \frac{\partial R}{\partial m_w} \) that forms an acute angle to the gradient-descent direction. Further, we have to update the parameters \( \alpha_j \) s and \( \Sigma w \) through a constrained optimization algorithm in view of the constraints on \( \alpha_j \) s in Eq. (5). Alternatively, we here let

\[ \alpha_j = \frac{\exp(\beta_j)}{\sum_{r=1}^{k} \exp(\beta_r)}, \quad 1 \leq j \leq k, \quad \ldots(5) \]

where the constraints of \( \alpha_j \) s are automatically satisfied, but the new variables \( \beta_j \) s are totally free. Consequently, instead of \( \beta_j \) s, we can learn \( \beta_{w_{\text{new}}} \) only by

![Diagram of PCA network](image)

Fig. 1: PCA network

![Handwritten digits numbers from 0 to 9, image of (28×28)](image)

Fig. 2: Handwritten digits numbers from 0 to 9, image of (28×28)

![Digits numbers from 0 to 9, image of 7×7](image)

Fig. 3: Digits numbers from 0 to 9, image of 7×7

![Reconstruction Digits from PCA](image)

Fig. 4: Reconstruction Digits from PCA
with the other $\beta\_s$ unchanged. It turns out that $aw$ is exclusively increased while the other $\alpha\_s$ are penalized, i.e., their values are decreased. Here, please note that, although $\beta\_s$ are gradually convergent, Eq. (6) always makes the updating of $\beta$ increase without an upper bound upon the fact the $aw$ is always smaller than 1 in general. To avoid this undesirable situation, one feasible way is to subtract a positive constant $cb$ from all $\beta\_s$ when the largest one of $\beta\_s$ reaches a prespecified positive threshold value.

As for $\Sigma\_w$, we update it with a small step size

$$\Sigma\_w^{new} = (1 - \eta\_s) \Sigma\_w^{old} + \eta\_s z\_sz\_T^T$$

where $z\_t = x\_t - m\_w^{old}$ and $\eta\_s$ is a small positive learning rate, e.g. $\eta\_s = 0.1\eta$. In general, the learning of a covariance matrix is more sensitive to the learning step size than the other parameters.

The simple K-means partitional clustering algorithm described above is computationally efficient and gives surprisingly good results if the clusters are compact, hyper spherical in shape and well-separated in the feature space. If the Mahalanobis distance is used in defining the squared error in (3), then the algorithm is even able to detect hyper ellipsoidal shaped clusters.

![Fig. 5: PCA of Iris dataset](image)

![Fig. 6: Classification of PCA Iris dataset by Backpropagation network](image)
PCA Feature Extraction by Hebbian Algorithm

The principal component analysis (PCA) is a Multivariate Statistical Process Control (MSPC) methods. Feature extraction methods determine an appropriate subspace of dimensionality \( m \) (either in a linear or a nonlinear way) in the original feature space of dimensionality \( n \), such that \( m \) is less than \( n \). Linear transforms, such as principal component analysis, factor analysis, linear discriminant analysis, and projection pursuit have been widely used in pattern recognition for feature extraction and dimensionality reduction. Principal components can be extracted using single-layer feed-forward neural networks.

Hebbian Learning process

There is a close correspondence between the behavior of self-organizing neural networks and the statistical method of principal component analysis. In fact, a self-organizing, fully interconnected 2-layer network with \( i \) inputs and \( j \) outputs (with \( i > j \)) can be used to extract the first \( j \) principal components from the input vector, thus reducing the size of the input vector by \( j-i \) elements as shown in fig.(1). With \( j=1 \), such a network acts as a maximum eigenfilter by extracting the first principal component from the input vector. The algorithm used to train the network is based on Hebb’s postulate of learning.

\[
y_j(n) = \sum_{i=1}^{n} w_{ij}(n) x_i(n), \quad j = 1, 2, 3, \ldots, m \quad \ldots(8)
\]

Hebbian Learning process by eq.(9)

\[
\Delta w_{ij}(n) = [y_j(n)x_i(n) - y_j(n) \sum_{k=1}^{n} w_{jk}(n)x_k(n)], \quad j=1, 2, \ldots, m; i=1, 2, \ldots, l \quad \ldots(9)
\]

Here \( n \) denotes the number of training samples, and the learning rate. The attentive reader will notice that the unconstrained use of this learning algorithm would drive to infinity because the weight would always grow but never be decreased. In order to overcome this problem, some sort of normalization or saturation factor needs to be introduced. A proportional decrease of \( w \) by a normalization term introduces competition among the synapses, which, as a principle of self-organization, is essential for the stabilization of the learning process.

The adjusted weight \( w_{ij}(n+1) \) is calculated as

\[
w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \quad \ldots(10)
\]

Reconstruction pattern from Hebb network by eq(11)

\[
X_{new} = W^f \cdot X_{old} \quad \ldots(11)
\]

In general, the algorithm converges very quickly. There is, however, one serious problem: errors in the input vector may cause some weights to break through the constraint mechanism and grow to infinity, which as a result destroys the PCA filter. The implementation should throw an exception when a particular weight grows above an appropriate upper limit.

Application of PCA on Handwritten

We applied the PCA by Hebbian on a handwritten digits numbers from 0 to 9, which represented by image of (28x28) pixels as shown in figure (). We have been applied a Kohonen’s map network to recognition theses pattern. The result of recognition is shown in table(1)

Application of adaptive K-means and PCA algorithm on Iris dataset

A data set with 150 random samples with nine size dimensional space of flowers from the iris species setosa, versicolor, and virginica collected by Anderson (1935). From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in cm.

We applied adaptive K-means algorithm to classify 150 samples dataset into three clusters and applied PCA algorithm to reduce the dimensional into two size. The classification accuracies achieved using the half of dataset features to learning process of back propagation neural network, the result of classification is shown as in Fig.
REFERENCES


