Fast and Accuracy Control Chart Pattern Recognition using a New cluster-k-Nearest Neighbor

Samir Brahim Belhaouari

Abstract—By taking advantage of both k-NN which is highly accurate and K-means cluster which is able to reduce the time of classification, we can introduce Cluster-k-Nearest Neighbor as "variable k"-NN dealing with the centroid or mean point of all subclasses generated by clustering algorithm. In general the algorithm of k-means cluster is not stable, in term of accuracy, for that reason we develop another algorithm for clustering our space which gives a higher accuracy than K-means cluster, less subclass number, stability and bounded time of classification with respect to the variable data size. We find between 96% and 99.7% of accuracy in the classification of 6 different types of Time series by using K-means cluster algorithm and we find 99.7% by using the new clustering algorithm.

Index Terms—Pattern recognition, Time series, k-Nearest neighbor, k-means cluster, Gaussian Mixture Model, Classification.

I. INTRODUCTION

Pattern recognition is an information-reduction process which aims to classify patterns based on a priori knowledge or based on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations coming from process or event.

A complete pattern recognition system consists of a sensor that interacts the observations to classify, a feature extraction mechanism that computes numeric or symbolic information from the observations, and a classification that classify patterns.

The classification is usually based on the availability of a set of patterns that have already been classified. This set of patterns is termed training set, and the resulting learning strategy is characterized as supervised. Learning can also be unsupervised, in the sense that the system is not given a priori labeling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.

The most perfect classification would verify at least the three conditions below:

1. Highly accurate,
2. Minimum classification time,
3. Smaller size of training data.

Classification using Gaussian Mixture model (GMM)[14] or k-Nearest Neighbor (k-NN) [5] are almost the same in the sense that they consider the neighbor data of the new pixel vector \( x \), which will be classified and will be more accurate compared to NN. This is because they are more efficient in the overlapping area as these methods take more consideration to samples of training data that are less frequent.

To reduce the classification time for k-NN, we need to cluster our space (training data) into subclasses, each subclass will be represented by one data (we can choose two or more representatives according to the number of subclasses), after that we use classification algorithm of NN (or k-NN) to classify by using representative data. Each subclass contains a random number of data, which are relatively close to each others. We call this manner of classification Cluster-k-NN (C-k-NN) which is similar to 'variable k'-NN.

The classification time in NN and k-NN are of the order \( O(N^2) \), i.e., \( O(N^2) \), where \( N \) is the training data size, to reduce this time we need to cluster our space, in fact the time of classification will depend just on the number of subclasses \( m_i \), with \( m_i \) the number of subclasses in the class \( C_i \), and that what do C-k-NN.

In general \( m_i \) is a small number and doesn’t depend on the training data size (the number of Gaussian functions to estimate a probability density, for GMM method, is in general bounded with respect to the variable \( N \)).

The most widely used method of non-parametric density estimation is k-NN [1]. k-NN rule is a powerful technique, that can be used to generate highly nonlinear classification with limited data. To classify a pattern \( x \), we find the closest \( k \) examples in the dataset and select the predominant class \( C_i \) among those \( k \) neighbors (problem if two or more classes are predominant class!). However one drawback of k-NN is that the training data must be stored, and a large amount of processing power is needed to evaluate the density for a new input pattern, C-k-NN correct those drawbacks points.

The k-NN classifier is generally based on the Euclidean distance between a test sample \( x \) and the specified training samples, in C-k-NN we can introduce other metrics in order to better estimate the density probability. Let \( x_{i,j} \) belongs to the subclass \( j \) of the class \( i \), we note \( C_{i,j} \), and for all positive number \( s \) we can define the following metric

\[
d(x, x_{i,j}) = \frac{d_{\text{euclidean}}(x, \hat{x}_{i,j})}{m_{i,j}}, \quad \forall x \in R^d,
\]

where \( \hat{x}_{i,j} \) is the representant of \( C_{i,j} \) and \( m_{i,j} = \text{card}(C_{i,j}) \) or it can be the variance of the set \( C_{i,j} \).

For parametric method we have Gaussian Mixture Model [14] which is classed as a semi-parametric density estimation method since it defines a very general class of functional forms for the density model. In a mixture model, a probability density function is expressed as a linear combination of basis functions.
A model with \( M \) components is described as mixture distribution:

\[
P(x) = \sum_{j=1}^{M} P(j)P(x|j),
\]

where \( P(j) \) are the mixing coefficients and \( P(x|j) \) are the component density functions. Each mixture component is defined by a Gaussian parametric distribution in \( d \) dimensional space:

\[
P(x|j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp \left\{ - \frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right\}.
\]

The parameter to be estimated are the mixing coefficients \( P(j) \), the covariance matrix \( \Sigma_j \) and the mean vector \( \mu_j \).

In C-k-NN we approximate each Gaussian function, \( P(j)P(x|j) \), by \( \frac{1}{cst+d(x,\mu_j)} \), where \( cst \) is any small number, we add it just to avoid the division by 0. To estimate the number of subclasses and their representatives for C-k-NN (or the number of Gaussian functions, \( M \), and their means \( \mu_j \) for GMM) we can use k-means cluster, or another new similar stable algorithm which will be explained later. The k-means cluster algorithm or the modified one need the number of subclasses as input, so to fixed number of clusters we iterate the number of clusters starting from one and we take the following conditions as stopping criterium:

a. All the representatives or centroid (\( \mu_{i,j} \)) have to be closer to their class \( C_i \) (or \( C_{i,j} \)) than to other classes. This is to decrease the misclassification (i.e., no error in the classification of our own training data).

b. The variance of each class, \( \text{var} \), does’t decrease considerably in comparison to the previous iteration. We define the variance of each class as \( \text{var} = \sum_{i=1}^{n} \text{var}_i \), where \( \text{var}_i \) is the variance of the subclass \( i \) and we can take \( \Delta \text{var} = \frac{\text{var}}{\text{var}_i} \leq \alpha \) as criterion of smooth function \( \text{var} \) with the number of subclasses as variable. In our simulation \( \alpha = 0.9 \) gives the best result.

d. Our dictionary is well defined now, for each class \( C_i \) is represented by \( \{\mu_{i,1}, \mu_{i,2}, ..., \mu_{i,m_i}\} \), \( 1 \leq i \leq cn \), where \( m_i \) is number of subclasses for the class \( C_i \) and \( cn \) is number of classes. To classify a new pattern \( x \) we use NN algorithm or k-NN algorithm with small \( k \) on the dataset \( \{\mu_{i,j} : 1 \leq i \leq cn, 1 \leq j \leq m_i\} \), which means we consider the minimum distance rule, and assign to the class \( C_i \) which is verified

\[
C_i = \arg \left( \min_{1 \leq n \leq m_i} \{d(x,\mu_{i,n})\} \right),
\]

where \( d(x,\mu_{i,j}) = C_i \), for all \( 1 \leq cn \).

The algorithm of k-means cluster is not stable since the result depends on the random choice of \( k \) initial vectors, further we develop another way to initialize this algorithm, which in general gives a better result than the classic k-means cluster e.i.,

\[
\text{Var}_{\text{modified algorithm}} \leq \text{Var}_{\text{classic algorithm}}
\]

For verification of our method we Select Synthetic Control Chart Time Series (SCCTS) from UCI KDD as test time series dataset. This dataset contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos (1999) (D.T. Pham and A.B. Chan, 1998).

There are six different classes of control charts and the classes are organized as follows: (Class one)1-100 Normal, (Class two)101-200 Cyclic, (Class three)201-300 Increasing trend, (Class four)301-400 Decreasing trend, (Class five)401-500 Upward shift, (Class six)501-600 Downward shift. We use the 50 first data for each class for training step and the others for control step.

The test result shows that the proposed approach give a good accuracy with bounded time of classification, independently with the size of data, as shown in the Table I.
A. Near-to-Near algorithm

This algorithm consists of calculating the distance \( d(x_i,n, x_i,m) \) for all \( x_i \in C_i \) and starting to cluster our class to \( N - 1 \) subclasses, where \( \text{card}(C_i) = N_i \). We put the two closest data at the same subclass

\[
C_{i,1} = \{ x_{i,n_0}, x_{i,m_0} \} \quad \text{where} \quad \min_{n \neq m} d(x_i, n, x_i,m) = d(x_i, n_0, x_i,m_0),
\]

and we put each other data at separate subclass,

\[
C_{i,j} = \{ x_{i,j} \} , \forall j \in \{1, ..., N_i \} - \{n_0, m_0\}.
\]

The next step, the following index \( n_1 \), and \( m_1 \) are considered for which

\[
\min_{n \neq m} d(x_i, n, x_i,m) = d(x_i, n_1, x_i,m_1).
\]

If \( x_{i,n_1} \) and \( x_{i,m_1} \) belong to the same subclass \( C_{i,r} \), we split this subclass in to two others subclasses

\[
C_{i,r+1} = C_{i,r} - \{ x_{i,n_1}, x_{i,m_1} \} \quad \quad (1)
\]

\[
C_{i,r} = \{ x_{i,n_1}, x_{i,m_1} \} \quad \quad (2)
\]

\[
\text{but} \quad \text{if} \quad x_{i,n_1} \quad \text{and} \quad x_{i,m_1} \quad \text{belong to two different subclasses} \quad i \neq j, \quad \text{and} \quad C_{i,r_1} \quad \text{and} \quad C_{i,r_2} \quad \text{respectively, in this case we put} \quad x_{i,n_1} \quad \text{in the subclass} \quad C_{i,r_1} \quad \text{if} \quad \text{card}(C_{i,r_1}) > \text{card}(C_{i,r_2}) \quad \text{and} \quad \text{we put} \quad x_{i,m_1} \quad \text{in} \quad \text{the subclass} \quad C_{i,r_2} \quad \text{if} \quad \text{card}(C_{i,r_2}) \leq \text{card}(C_{i,r_1}). \quad \text{Indeed to use} \quad \text{a cardinality of set we can use the distance between the vector} \quad x \quad \text{and} \quad \text{set as} \quad d(\text{vector, mean of set}).
\]

\[
\text{then we obtain} \quad k \quad \text{subclasses, we stop the iteration, and our} \quad k \quad \text{vectors will be the mean of each subclasses.}
\]

B. Near-to-Mean algorithm

This algorithm is almost the same as the Near-to-Near one, but we will deal with the mean of subclass \( C_{i,r} \) in the processing. We start to split our class \( C_i \) into two subclasses

\[
C_{i,1} = \{ x_{i,n_0}, x_{i,m_0} \} \quad \text{where} \quad \min_{n \neq m} d(x_i, n, x_i,m) = d(x_i, n_0, x_i,m_0)
\]

\[
C_{i,2} = \{ x_{i,j} \quad | \quad j \notin \{n_0, m_0\} \}.
\]

We call the new class or set

\[
C_{i}^1 = \{ x_{i,1}, ..., x_{i,n_0-1}, s_0 \times x_{i,n_0+1}, ..., x_{i,m_0-1}, s_0 \times x_{i,m_0+1}, \ldots \},
\]

where \( s_0 = (\sum_{i} x_{i,m_0}) / 2 \).

Next we consider \( x_{i,n_1} \) and \( x_{i,m_1} \) such as

\[
d \left( x_{i,n_1}, x_{i,m_1} \right) = \min \left\{ \left| d \left( x_{i,n}, x_{i,m} \right) \right| \middle| d \left( x_{i,n}, x_{i,m} \right) \neq 0 \right\}.
\]

This algorithm is almost the same as the Near-to-Near one, but we will deal with the mean of subclass \( C_{i,r} \) in the processing. We start to split our class \( C_i \) into two subclasses

\[
C_{i,1} = \{ x_{i,n_0}, x_{i,m_0} \} \quad \text{where} \quad \min_{n \neq m} d(x_i, n, x_i,m) = d(x_i, n_0, x_i,m_0)
\]

\[
C_{i,2} = \{ x_{i,j} \quad | \quad j \notin \{n_0, m_0\} \}.
\]

We call the new class or set

\[
C_{i}^1 = \{ x_{i,1}, ..., x_{i,n_0-1}, s_0 \times x_{i,n_0+1}, ..., x_{i,m_0-1}, s_0 \times x_{i,m_0+1}, \ldots \},
\]

where \( s_0 = (\sum_{i} x_{i,m_0}) / 2 \).

Next we consider \( x_{i,n_1} \) and \( x_{i,m_1} \) such as

\[
d \left( x_{i,n_1}, x_{i,m_1} \right) = \min \left\{ \left| d \left( x_{i,n}, x_{i,m} \right) \right| \middle| d \left( x_{i,n}, x_{i,m} \right) \neq 0 \right\}.
\]

We replace all the data in \( C_{i}^1 \) that are equal to \( x_{i,n_1} \) or \( x_{i,m_1} \) by \( s_1 \), which is the mean of the union of the two subclasses where \( x_{i,n_1} \) and \( x_{i,m_1} \) belong to,

\[
s_1 = \frac{C_{n_1}, \{x_{i,n_1}, x_{i,m_1}\}, \frac{C_{m_1}, \{x_{i,m_1}\}}{C_{n_1} + C_{m_1}}.
\]

Our algorithm of classification doesn’t need to keep all the data but it needs just the average of each subclass, this is one of the powerful point of the clustering. To classify a new data or vector \( x \), we use k-NN algorithm, i.e., we assign \( x \) to the class \( C_i^1 \) for which

\[
\hat{i} = \arg \min_{i,j} d(x, \mu_{i,j}).
\]

Another idea about the k-NN algorithm is indeed to find the closest \( j \)- examples in the dataset and select the predominant class, we can find the smallest closest examples in the dataset and select the predominant class that have exactly \( k \) examples.

III. Experiments and Results

Ideally, sample patterns to simulate a Control Chart Patters (CCP) should be collected from the real world. Usually we need a big number of sample to develop and validate a CCP recognizer and those are not available in general, so simulated data are often used.

Various control chart patterns are generated considering different pattern parameters as shown in the table III. The windows size \((N)\) in taken to be 60 data points. The values of different patterns are varying randomly in a uniform manner between the shown limits. For the generation of training and test data, 50 and 50 time series of standard normal data are used respectively. Since there are six pattern classes considered in this study, a total of 300 (50*6) and 300 (50*6) sample patterns are simulated for training and verification phases respectively.

The test results, shown in table II, tell us that the new approach gives a remarkable accuracy with limited time of classification, independency with the size of the training data set. As conclusion we can say that k-NN has the advantage to not need to estimate the density function. However, its high computational load makes it unsuitable for hyperspatial data classification. Using the idea of k-means cluster we can remove the time drawback.
Fig. 1. Graphs of variance in function of number of subclasses by using k-means cluster, the “+” with Near To Near algorithm, “-“ with Near To Mean algorithm and “*” with aleatory initialization.

**TABLE II**

<table>
<thead>
<tr>
<th>Classification Results Comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>Clustering-k-NN</td>
</tr>
<tr>
<td>ClusteringM-k-NN* with Near to Near</td>
</tr>
<tr>
<td>ClusteringM-k-NN** with Near to Mean</td>
</tr>
<tr>
<td>NN</td>
</tr>
</tbody>
</table>

* for modified algorithm of k-means cluster with $\alpha = 0.9$ for Near to Near algorithm
** for modified algorithm of k-means cluster with $\alpha = 0.2$ for Near to Mean algorithm.

**TABLE III**

<table>
<thead>
<tr>
<th>Parameters for Simulation Control Chart Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control chart patterns</td>
</tr>
<tr>
<td>Normal</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Cyclic</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Increasing trend</td>
</tr>
<tr>
<td>Decreasing trend</td>
</tr>
<tr>
<td>Upward shift</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Downward shift</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

$r_i$ is normally distributed random number.
The Figure 1 shows that if we choose valid initial vectors for k-means cluster we can obtain a small variance. In this case, the variance of Near-to-Near algorithm is smaller than the variance of Near-to-Mean algorithm if the number of subclasses is bigger than 5. The Near-to-Mean algorithm deals better if the number of subclasses is small than the Near-to-Near algorithm.

IV. CONCLUSION

Classification by using Cluster-\(k\)-Nearest Neighbor with Near-To-Near algorithm is almost perfect method of classification in terms of accuracy and bounded time of classification and this by taking advantage from K-Nearest Neighbor, Gaussian Mixture Model and clustering. Near-to-Near algorithm give reasonable set of vectors to initialize K-means cluster algorithm in order to minimize the variance of space data.

REFERENCES


Brahim Belhaouari Samir S. Lecturer E.E. department University Technology of PETRONAS, MALAYSIA. Ph.D. in stochastic processes, EPFL, Switzerland. Master in Telecommunication & Network, INP, France.