World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:3, No:1, 2009

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 gathers 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) in to 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 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

$$d(x, x_{i,j}) = \frac{d^s_{euclidean}(x, \widehat{x}_{i,j})}{n_{i,j}}, \quad \forall x \in R^d,$$

where $\widehat{x}_{i,j}$ is the representant of $C_{i,j}$ and $n_{i,j} = card(C_{i,j})$ 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.

Samir Brahim B. is with University technology of PETRONAS.

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 components is defined by a Gaussian parametric distribution in d dimensional space

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

The parameter to be estimated are the mixing coefficients $P(j)$, the covariance Matrix Σ_j and the mean vector μ_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 μ_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, var, does't decrease considerably in comparison to the previous iteration. We define the variance of each class as $var = \sum_{i=1}^{n} var_i$, where var_i is the variance of the subclass i and we can take

$$\frac{\Delta var}{var} \leq \alpha$$

as criterium of smooth function var with the number of subclasses as variable. In our simulation $\alpha = 0.9$ gives the best result.

Our dictionary is well defined now, for each class C_i is represented by $\left\{\mu_{i,1}, \mu_{i,2}, ..., \mu_{i,m_i}\right\}$, $1 \leq i \leq cn$, 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 $\left\{\mu_{i,j} : 1 \leq i \leq cn, 1 \leq j \leq m_i\right\}$, which means we consider the minimum distance rule, and assign to the class $C_{\hat{i}}$ which is verified

$$C_{\hat{i}} = \arg\left(\min_{\substack{1\leq i\leq cn \\ 1\leq j\leq m_i}} \left\{d(x,\mu_{i,j})\right\}\right),$$

where $\arg\left(d(x,\mu_{i,j})\right) = C_i$, 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.,

$$Var_{\textit{modified algorithm}} \leq Var_{\textit{classic algorithm}}.$$

For validation 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

II. Method

Each class, C_i, should be cluster to several subclasses, $C_{i,j}$, with $1 \leq j \leq m_i$, and each subclass will be represented by it means, $\mu_{i,j}$. Thus the cluster analysis seeks to identify a set of groups, which minimize the within-group variation and maximize the between-group variation.
We apply k-means cluster algorithm for each class in order to do the clustering, then we need to define the number of subclasses for each class and the initial k-vectors to initialize the k-means cluster algorithm.
To find the best suitable number of subclasses, we iterate the number of subclasses starting from 1 with two conditions to stop the iteration:

1. All the representative, $\mu_{i,j}$, should be close,in respect to the metric d, to their class C_i i.e., if we classify all the representatives $\mu_{i,j}$ we have to find 100% of accuracy.
   If there are some misclassifications of $\mu_{i,j}$, we have to decrease the parameter α by multiply it by a another factor, α', less than 1.
2. The variance of each class C_i, var_i, doesn't decrease considerably in comparison to the previous iteration.
   We can use $\frac{\triangle var}{var} \leq \alpha$ as a criterium to quantify if there is a decrease or if it is approximately still constant. In certain case, it is better to stop the iteration if the condition, $\frac{\triangle var}{var} \leq \alpha$, was checked twise or more, i.e., after where the variance will be smooth.

For initialization of k-means cluster algorithm in general we choose aleatory k-vectors, which belong to our class data. This way makes the algorithm not stable in sense the final variance,

$$var_{C_i} = \sum_{j=1}^{m_i} var_{C_{i,j}},$$

depends on the initial vectors.
From here, the question "How to choose the initial vectors in order to find a minimal variance?" arises. In this paper we develop two algorithms: near-to-near and near-to-mean, which we can do some modification related to each application.

TABLE I
CLASSIFICATION RESULTS COMPARISONS

	Cluster-k-NN	ClusterM-k-NN*	NN
Accuracy	between 96 and 99.7%	99.7%	96%
Subclass number	[3,7,4,4,7,9]	[2,2,2,2,11,4]	[50,50,50,50,50,50]
Classification time	11% of T	7.6% of T	T=$O(50^2)$

* *for modified algorithm of k means cluster with* $\alpha = 0.9$ *and* $\alpha' = 0.82$.

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 $N_i - 1$ subclasses, where $card(C_i) = N_i$. We put the two closest data at the same subclass

$$C_{i,1} = \left\{x_{i,n_0}, x_{i,m_0}\right\}, \textit{where } \min_{n \neq m} d(x_{i,n}, x_{i,m}) = d(x_{i,n_0}, x_{i,m_0}),$$

and we put each other data at separate subclass,

$$C_{i,j} = \left\{x_{i,j}\right\}, \forall j \in \{1, ..., N_i\} - \{n_0, m_0\}.$$

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

$$\min_{\substack{n \neq m \\ (n,m) \neq (n_o, m_0)}} d(x_{i,n}, x_{i,m}) = d(x_{i,n_1}, x_{i,m_1}).$$

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

$$C_{i,r+1} = C_{i,r} - \left\{x_{i,n_1}, x_{i,m_1}\right\} \quad (1)$$

$$C_{i,r} = \left\{x_{i,n_1}, x_{i,m_1}\right\} \quad (2)$$

but If x_{i,n_1} and x_{i,m_1} belong to two different subclasses C_{i,r_1} and C_{i,r_2} respectively, in this case we put x_{i,n_1} in the subclass C_{i,r_2} if $card(C_{i,r_2}) > card(C_{i,r_1})$ and we put x_{i,m_1} in the subclass C_{i,r_1} if $card(C_{i,r_2}) \leq card(C_{i,r_1})$. Indeed to use the cardinally of set we can use the distance between the vector and the set as $d(\textit{vector}, \textit{mean of set})$.

When we obtain k subclasses, we stop the iteration, and our initial k-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 here 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

$$C_{i,1} = \left\{x_{i,n_0}, x_{i,m_0}\right\}, \textit{where } \min_{n \neq m} d(x_{i,n}, x_{i,m}) = d(x_{i,n_0}, x_{i,m_0})$$

$$C_{i,2} = \left\{x_{i,j} \quad | \quad j \notin \{n_0, m_0\}\right\}.$$

We call the new class or set

$$C_i^1 = \left\{x_{i,1}, ..., x_{i,n_0-1}, s_0, x_{i,n_0+1}, ..., x_{i,m_0-1}, s_0, x_{i,m_0+1}, ...\right\},$$

where $s_0 = \left(x_{i,n_0} + x_{i,m_0}\right)/2$.
Next we consider x_{i,n_1} and x_{i,m_1} such as

$$d\left(x_{i,n_1}, x_{i,m_1}\right) = \min\left\{d\left(x_{i,n}, x_{i,m}\right) \mid d\left(x_{i,n}, x_{i,m}\right) \neq 0\right\}.$$

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,

$$s_1 = \frac{C_{n_1} x_{i,n_1} + C_{m_1} x_{i,m_1}}{C_{n_1} + C_{m_1}},$$

where C_{n_1} is the number of repetition of x_{i,n_1} inside of C_i^1 and C_{m_1} is the number of repetition of x_{i,m_1} inside of C_i^1. Our algorithm stops once the number of distinct vectors inside of C_i^r is equal to k.

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 $C_{\widehat{i}}$ for which

$$\widehat{i} = \arg_i \min_{i,j} d(x, \mu_{i,j}).$$

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 n 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 N. 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.

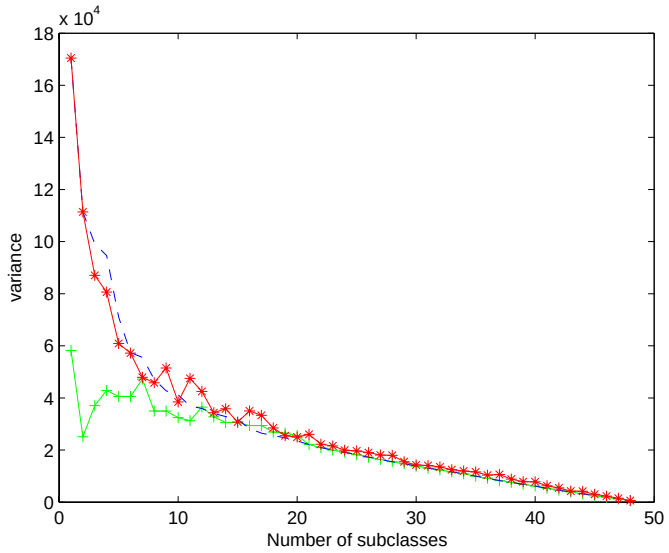

Fig. 1. Graphs of variance in function of number of subclasses by using k-means cluster, the "+" with Near To Near algorithm ,"-" with Near To Mean algorithm and "*" with aleatory initialization.

TABLE II
CLASSIFICATION RESULTS COMPARISONS

	Accuracy	Subclass number	Classification time
Cluster-k-NN	between 96 and 99.7%	[3,7,4,4,7,9]	11% of T
ClusterM-k-NN* with Near to Near	99.7%	[2,2,2,2,11,4]	7.6% of T
ClusterM-k-NN** with Near to Mean	97%	[3,6,5,4,50,50]	22% of T
NN	96%	[50,50,50,50,50,50]	T=$O(50^2)$

$*$ *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
PARAMETERS FOR SIMULATION CONTROL CHART PATTERNS

Control chat patterns	Pattern parameters	Parameters values	Pattern equation
Normal	Mean (μ) Standard deviation (σ)	80 5	$y_i = \mu + r_i\sigma$
Cyclic	Amplitude (a) Period (T)	1.5σ to 2.5 σ	$y_i = \mu + r_i\sigma + a\sin(\frac{2\pi}{T})$
Increasing trend	Gradient (g)	0.05σ to 0.1 σ	$y_i = \mu + r_i\sigma + ig$
Decreasing trend	Gradient (g)	-0.1σ to -0.05 σ	$y_i = \mu + r_i\sigma - ig$
Upward shift	Shift magnitude (s) Shift Position (P)	1.5σ to 2.5 σ 9,17,25	$y_i = \mu + r_i\sigma + ks$ $k = 1$ if $i \geq P$ else $k = 0$
Downward shift	Shift magnitude (s) Shift Position (P)	-2.5σ to -1.5σ 9,17,25	$y_i = \mu + r_i\sigma - ks$ $k = 1$ if $i \geq P$ else $k = 0$

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.

Neat-to-Near algorithm give reasonable set of vectors to initialize K-means cluster algorithm in order to minimize the variance of space data.

References

[1] B. V. Dasarsthy, Ed., *Nearest Neighbor (NN) Norms: NN Pattern classification techniques*. Los Alamitos, CA:IEEE Computer Society Press, 1990.
[2] P. J. Hardin, *Parametric and Nearest Neighbor methods for hybrid classification: A comparaison of pixel assignment accuracy*, *photogramm Eng. Remote Sens.,* Vol. 60 pp. 1439-1448, Dec. 1994 .
[3] T. P. Yunck, A technique to identify NN, *IEEE Trans. Syst., Man, Cybern,*. Vol. SMC-6, no. 10, pp. 678-683, 1976.
[4] X. Jia and J. A. Richards, *Cluster-space representation for hyperspectral data classification, IEEE Trans, Geosci. Remote Sens.,* vol. 40, no. 3, pp. 593-598, Mar. 2002.
[5] X. Jia and J. A. richards, *Fast k-NN classification Using the Cluster-Space Approach, IEEE Trans., Geosci. Remote Sens.,* vol. 2, no. 2, April 2005.
[6] K. wagstaff and S. Rogers, *Constrained k-means Clustering with Background Knowledge, Proc. of the 8th International conference on Machine learning,* p. 577-584, 2001
[7] Pentakalos, O., Menasce, D., and Yesha, Y., *Automated clustering-based workload characterization,* Joint Sixth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies and Fifth IEEE Symposium on Mass Storage Systems, College Park, MD, March 23-26, 1998.
[8] Jain, A., Murty, M., Flynn, P., *Data Clustering: A Review,* http://scgwiki.iam.unibe.ch:8080/SCG/uploads/596/p264-jain.pdf
[9] Eisen, Michael. *Cluster 3.0 Manual.* http://bonsai.ims.u-tokyo.ac.jp/ mdehoon/software/cluster/cluster3.pdf.
[10] Hale. Introductory Statistics, http://espse.ed.psu.edu/statistics.
[11] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd ed. New York: Wiley, 2000.
[12] I. T. Nabney, Netlab, Algorithms for Pattern Recognition. London, U.K.: Springer-Verlag, 2003.
[13] D. J. C. MacKay. (1997). *Gaussian processesA replacement for supervised neural networks,* in Lecture Notes TutorialNIPS . [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/gp.pdf
[14] D. M. Titterington, A. F. M. Smith, and U. E. Makov, Statistical Analysis of Finite Mixture Distributions. John Wiley, New York, 1985.
[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
[16] S. Brahim-Belhouari and A. Bermak, Pattern Recognition Letters. Elsevier Science Publisher, 2005.
[17] S. Brahim-Belhouari,Modified k-means cluster. University technology of PETRONAS, 2008.

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.