Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 57 – 66

ISSN 1896-7094

A Data Clustering Algorithm Based On Single Hidden Markov Model

Md. Rafiul Hassan, Baikunth Nath and Michael Kirley

Computer Science and Software Engineering, The University of Melbourne,
Melbourne 3010, Australia.
{mrhassan, bnath, mkirley}@csse.unimelb.edu.au

Abstract. The ability to cluster data into different groups based on a particular similarity measure has a wide appeal in many domains, including: data mining, image classification, speech recognition, fraud detection and in network traffic anomaly detection. Typically, the clustering algorithm partitions a dataset into a fixed number of clusters supplied by the user. In this paper, we propose a single Hidden Markov Model (HMM) based clustering method, which identifies a suitable number of clusters in a given dataset without using prior knowledge about the number of clusters. Initially, the dataset is partitioned into windows of fixed size based on the HMM log likelihood values. This provides a framework for identifying the most appropriate number of clusters (windows of varying sizes). After determining the number of clusters, the data values are then labeled and allocated to clusters. The algorithm is tested using a number of benchmark datasets. The proposed algorithm for both small and large datasets (KDD 1999 Intrusion Detection dataset) performed significantly better compared to other commonly used clustering algorithms.

Keywords: HMM; Unsupervised clustering; SOM, Fuzzy c-means

1. Introduction

Clustering is the process, where a given collection of unlabeled patterns (dataset), the data items are divided into groups (clusters) based on some measure of similarity [1]. A variety of clustering techniques have been proposed in the machine-learning, pattern recognition, data mining and statistics domains. Well known examples include *k*-means [2], fuzzy c-means [3, 4], Self Organizing Map (SOM) [5, 6], Artificial Neural Networks [7]] and Support Vector Machines [8]. These techniques belong to one of two groups-supervised clustering and unsupervised clustering-depending upon the underlying method used to cluster the data items. In supervised clustering, the algorithm is trained using a proportion of the dataset (the training set) and then this trained algorithm is used to classify an unknown dataset (test set). Typically, the number of clusters for supervised learning is prespecified. Alternatively, in unsupervised clustering a training dataset is not used. Here, assumptions have to be made about the number of clusters to be used and/or the spread of each cluster prior to clustering the data. The algorithm then partitions the data into clusters. For instance, to cluster the dataset using *k*-means, fuzzy c-means and hierarchical clustering [9], the number of clusters must be known a priori. Hidden Markov Models (HMM) [10] can also be used for classifying patterns from an unknown dataset. For example, in speech related literature HMM has been used for classifying speakers [11-14] or speech patterns [15, 16]. Typically, for pattern classifica-

tion, a number of HMM are used in combination with supervised techniques. In this paper, we propose a single HMM-based unsupervised clustering algorithm.

In our model, a single HMM is used to identify the number of clusters in a given dataset. The data items are then labeled and partitioned into the appropriate clusters. Initially, the HMM is used to calculate log-likelihood values for each of the data items. Here, the log-likelihood values on one hand represent how well the data fits the trained HMM and on the other provide a similarity measure between data items. Based on these log-likelihood values the dataset is then partitioned into windows of fixed size. The plot of the log-likelihood values after segmenting into "windows" is used to identify the possible number of clusters in the dataset. If the difference in the sorted log-likelihood values exceeds some threshold, a new cluster is formed. Once the number of clusters has been determined, the sorted log-likelihood values are used to divide the data items into appropriate clusters.

The remainder of the paper is organized as follows. Section 2 briefly reviews commonly used clustering techniques. In section 3, we describe our HMM-based data clustering algorithm, and place particular attention to describing the process of creating "window" introduced above. Section 4 provides the experimental results from the proposed model and compares with other clustering techniques using benchmark data sets. Finally, we conclude this paper with a discussion of the model and its implications in Section 5.

2. Background

In this section, we briefly review the well-known unsupervised clustering techniques reported in the literature. A more comprehensive review can be found in [1].

The *k*-means algorithm is one of the oldest unsupervised clustering algorithms [17]. The idea is to group data into *k* clusters (known a priori) using *k* centroids (one for each cluster). The performance of clusters thus obtained depends on the initial centroid values. The aim of this algorithm is to minimize the Euclidean distance between the data points and the corresponding cluster centroid, which is achieved by minimizing the objective function:

$$\sum_{j=1}^{k}\sum_{i=1}^{n} || x_i^{(j)} - c_j ||^2$$

where $x_i^{(j)}$ is the data point, and c_j is the jth cluster center.

An inherent assumption built into the *k*-means algorithm is that the data points are independent. Consequently, there is degradation in the algorithm effectiveness (accuracy) when the data points are highly dependent on each other. The *k*-means algorithm is not able to find the optimal configuration compared with the global objective function minimum [18].

Fuzzy c-means [3, 19] is an improved version of the crisp *k*-means algorithm. In this model, each data item is associated with every cluster by means of a membership function [20]. In the crisp case, data items belong to one cluster (partition) only. While in the fuzzy c-means algorithm, the fuzzy partition of the N data items into C clusters is obtained by selecting N×C membership matrix U (where an element of the matrix U is the degree of membership of data item to cluster). Then the following objective function

$$\sum_{j=1}^{k}\sum_{i=1}^{n} \mu_{ij}^{m} || x_i^{(j)} - c_j ||^2 \qquad 1 \le m \le \infty$$

is minimized iteratively using the U matrix.

In fuzzy c-means clustering, the introduction of "soft partitioning" reduces problems associated with identifying local minima. However, the algorithm may still converge to local minima by the squared error criterion [1]. In addition, the design and initialization of the membership function also impact on algorithm

performance [3]. A Self Organizing Map (SOM) is an alternative data clustering technique proposed by Kohonen [5]. The SOM is built based on a two dimensional grid of map units which are represented by means of prototype vectors. For unit *i*, the prototype vector is represented as $m_i = [m_{i1}, \ldots, m_{id}]$, where *d* is input vector dimension. Here, a uniform distribution of two or three dimensions represents the map units [21]. The neighborhood relationships among the units are responsible for connecting the adjacent units. The performance of SOM is dependent on the number of map units and the topology of the map. During the training, the mapping of the dataset is done in such a way that the closer data points in the input space are mapped onto closer map units. Using an iterative process, data points in the input space are mapped onto a 2-D grid. The number of clusters can then be visually identified after training. In general, when SOM is used to cluster d-dimensional data, clustering is accomplished by mapping the data to a lower dimension space. The visualization properties of the SOM facilitate qualitative analysis of the dataset [22]. However, performing a quantitative analysis remains a challenge. Furthermore, the accuracy and the generalization capability of SOM are solely dependent on the topology of the map chosen and the map-size.

3. The model

The motivation for the work described in this paper, is to design an effective unsupervised data clustering algorithm. For a given dataset, we do not specify a priori the number of clusters. The model is deemed capable to identify appropriate partitions in the dataset and divide the data items into appropriate clusters.

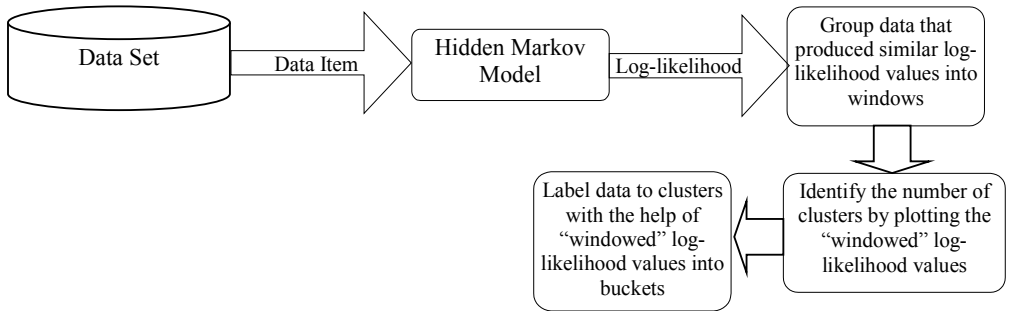

Fig. 1. The HMM based model to cluster multidimensional dataset

There are three steps to implement the proposed method. The first step consists of training the HMM and producing log-likelihood values for each of the data items. The second step is sorting the log-likelihood values of the dataset according to the defined "windows" or "bins". This process helps in identifying the number of clusters. In the third and final step, labeling of data items to their respective clusters is carried out. Figure 1 illustrates the basic functionality of the model. The specific details of the model follow.

3.1. Generating likelihood values using HMM

3.1.1. Hidden Markov Model

Hidden Markov Models (HMM) were first introduced in the 1970s as a tool for speech recognition [23]. Recently, the popularity of HMM has increased in the pattern recognition domain primarily because of its strong

mathematical basis and the ability to adapt to unknown data. This section describes HMM in more detail together with a description of the algorithms used to induce HMM. Further details can be found in [10].

A HMM is a finite automaton with a fixed number of states. A HMM has the following elements:

1) *N,* the number of states,

2) *M* , the number of observation symbols,

3) *A,* matrix of state transition probabilities,

4) *B,* matrix of observation emission probability distribution,

5) π , matrix of prior probabilities.

The parameter set (*A, B,* π), or simply λ represents the overall HMM model. To fit a model for a given observation sequence, the model parameters *A* , *B* and π are chosen in such a way that the model can suitably explain the observed data. A number of algorithms are available for adjusting the parameters of the HMM. The Baum-Welch (B-W) [24] algorithm is the most widely used technique. In the B-W algorithm, the parameters of an HMM are trained to maximize the probability of the observation sequence for a given model. This optimization is known as the maximum likelihood criterion.

In the sequel, the likelihood function is represented by $P(O \mid \lambda)$, where *O* represents the observation sequence, and λ is the given model. The next step is to calculate the likelihood values.

3.1.2. Finding the likelihood value of an observation sequence

Given one observation sequence $O = (O_1, O_2, \ldots, O_T)$, where *T* is the length of the sequence, and the model $\lambda = (A, B, \pi)$, then for a state sequence $Q = (q_1, q_2, \ldots, q_T)$, we have

$$P(O \mid \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda) = b_1(O_1) b_2(O_2) \ldots b_T(O_T)$$

and $$P(Q \mid \lambda) = \pi_1 a_{1,2} a_{2,3} \ldots a_T$$

Now, $$P(O, Q \mid \lambda) = P(O \mid Q, \lambda) P(Q \mid \lambda)$$

Therefore, $$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(Q \mid \lambda) = \sum_{q_1 q_2 \ldots q_T} \pi_1 \prod_{t=1}^{T} a_{t,t+1} b_t(O_t)$$

The complexity of this calculation is $O(TN^T)$ multiplications, which is very complex. The value of $P(O \mid \lambda)$ may however be calculated more efficiently using the forward-backward procedure [11, 24]. Here, for completeness, we describe only the forward procedure.

3.1.3. The forward procedure

In this procedure, a forward variable $\alpha_t(i)$ the probability of the partial observation sequence up to time *T* and the state *i* at time *t* , given the model λ, is defined as

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_T; q_t = 1 \mid \lambda)$$

This forward variable may be computed as follows:

Initialization: $\alpha_1(i) = \pi_i \qquad 1 \le i \le N$.

Induction $\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\right] b_j(O_{t+1}) \qquad 1 \le t \le T-1, \quad 1 \le j \le N$.

Finally $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

This is a much simpler algorithm that requires only $O(N^2 T)$ multiplications.

The likelihood values may be interpreted as follows. Consider two *d*-dimensional data items $d_n = (x_{1n}, x_{2n}, \ldots, x_{dn})$ and $d_m = (x_{1m}, x_{2m}, \ldots, x_{dm})$, where x_{ij} represents the i[th] attribute of the j[th] data item in the dataset. If the log-likelihood values l_n and l_m of the above observations are close, then we may infer that the data items d_n and d_m should belong to the same cluster.

3.2. Number of Clusters in the Dataset

The Baum-Welch algorithm maximizes the probability that each data item fits the model. The log-likelihood values for two data items represent similarity between them [10, 15]. Since in any datasets, typically, there are two or more clusters, it is reasonable to expect that close log-likelihood values will gather into one group, while the remaining log-likelihood values will appear at a distance from this group.

In order to determine the number of clusters in the dataset, we need to process the data based on their corresponding log-likelihood values. Figure 2 displays the log-likelihood values along the vertical axis and the respective data labels (horizontal axis) for a sample dataset. The data label refers to the record number of the data item in the dataset. As it stands, this plot does not provide any useful information. We therefore group the log-likelihood values into several bins (windows) in order of magnitude ranging from minimum to maximum. These bin frequencies for the benchmark Ringnorm data are displayed in Figure 3. A simple pseudo code for distributing log-likelihood values into bins is given below.

Pseudo code for grouping similar data items

1. Get LL values for each of the data item in the dataset;
2. Set: window_size= theta;
3. Set: minimum_LL=minimum LL value;
4. Set : maximum_LL=maximum LL value;
5. Set: window_ptr=minimum_LL and i=1;
6. Repeat steps until window_ptr < maximum_LL
7. Set: window(i)=data items which produces LL value in between window_ptr and window_ptr + window_size;
8. Set: window_ptr=window_ptr+window_size;
9. Increase i by 1;

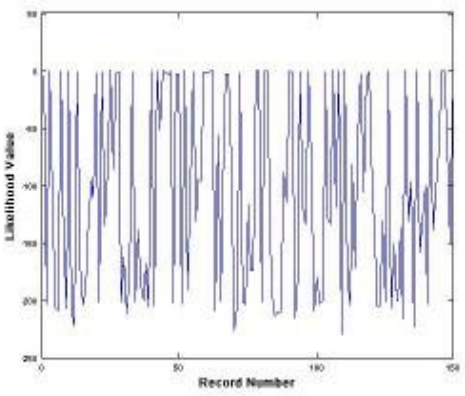

Fig. 2. Log-likelihood values of data items

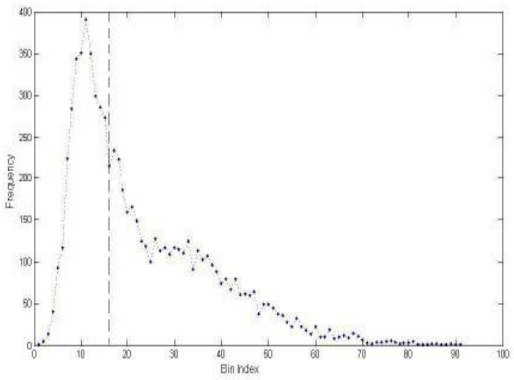

Fig. 3. Ringnorm bin frequency chart

Figures 4 and 5 display the log-likelihood values for another two classical data sets Iris data and Thyroid. An inspection of plotted bin frequencies in figure 3 or log-likelihood values in Figures 4 and 5 suggests that there are distinct breaks or changes in the curves, which leads us to think of dividing lines for possible clusters. Despite the subjective nature of such an inspection and placement of segmentation line(s), when a change in log-likelihood values exceed some threshold (e.g. an inflection point) can identify the resulting partition. Further, we notice in the lower part of figure 5 there are many gaps in the curve that may suggest a number of possible clusters. However, since the number of points in this region is not significant, we may combine all points into one cluster and consider a total of three clusters dividing the data. The experimental results using this procedure are detailed in Section 5.

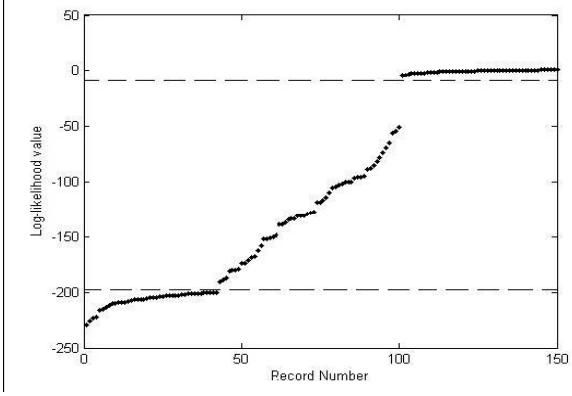

Fig. 4. Number of clusters (Iris Data)

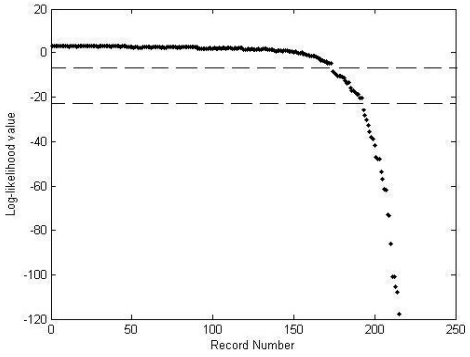

Fig. 5. Number of Clusters (Thyroid Data)

3.3. Labeling Data for Clusters

In Section 3.2, the log-likelihood values were initially partitioned into a window of fixed size (width 1 in this case). The next task is to label the data items into clusters. To do this, we examine the frequency of data items in each of the windows. Initially we use a minimum threshold frequency as a means of identifying the natural partitions in the dataset. Then we merge windows on either side of the partition into a variable-sized window representing a cluster. The aim is to find the minimum number of windows (variable size) such that all the similar data items fall into their respective bins. If there is no empty window, a threshold value (minimum frequency) is subtracted from the frequencies of all windows to generate some empty windows. When the

number of empty windows is more than the number of clusters identified in Section 3.2, some of the windows (empty as well as non-empty) in close neighborhood are merged.

The algorithm below describes the steps for labeling and allocating data to clusters.

Algorithm - Allocating data items to clusters

1. Set : threshold_frequency=
2. Repeat step 3 for i=1 to total_ number_ of_ window
3. Subtract ⊖ from the window(i)
4. Repeat step 5 and 6 until last_window is reached
5. Locate the empty windows and combine subsequent empty windows(if any)
6. Combine subsequent occupied windows
7. (a)Find the maximum gap between two occupied windows and point it as a current_cluster edge
(b)Locate the occupied windows from 1st window to the current_ cluster edge as the first cluster
(c)Collect data items in the windows pointed as cluster in step (b). Label these data item to the cluster
(d)Set: previous_cluster edge=current_cluster edge;
8. Repeat step 9 until the desired number of clusters is found or the last window is reached
9. (a)Find the next maximum window gap and point it as current_cluster edge
(b)Locate the occupied windows from previous cluster_edge to the cluster edge as the next cluster
(c)Collect data items in the windows pointed as cluster in step (b). Label these data item to the cluster.
(d)Set: previous_cluster edge=current_cluster edge;

---

4. Experimental data and results

4.1. Test Data

To evaluate the performance of our model, we considered a number of benchmark datasets including the Fisher's Iris dataset [25], Thyroid dataset [26], Ringnorm dataset [27], the Diabetes dataset [28] and the KDD 1999 Intrusion Detection System (IDS) dataset [31].

Fisher's Iris dataset [25] consists of sepal and petal measurements. The dataset consists of 50 objects from each of three species {Iris-virginica, Iris-versicolor, Iris-setosa}. Each object in the dataset is described by four attributes {sepal length, sepal width, petal length, petal width}. The data is four-dimensional.

Thyroid dataset [26] consists of complete medical record including anamnesis, scan etc. to help diagnosis whether a patient's thyroid belongs to the class euthyroidism, hypothyroidism or hyperthyroidism was first used by Coomans [29]. There are three classes in the dataset: normal , hyperthyroid and hypothyroid and five attributes: T3-resin uptake test, total serum thyroxin, total serum triiodothyronine, basal thyroid-stimulating hormone (TSH) and maximal absolute difference of TSH value followed by some condition. The dataset consists of 215 samples of which 150 are normal, 35 are in hyper class and the rest 30 in the hypo class. In our model, we have considered variables with numerical values only.

The ringnorm dataset is an implementation of Lei Breiman's ringnorm example [27]. The dataset contains 20 attributes and 2 classes. The attributes of each class are found from a multivariate normal distribution while the classes are made different to each other by differing the mean and covariance matrix. Of the 7400 samples, 3736 samples belong to class 1 and the rest to class 2.

Diabetes dataset [28] originated from the National Institute of Diabetes and Digestive and Kidney diseases. The dataset contains 8 attributes useful to recognize the status of diabetes (either positive or negative) of a patient. Out of 768 sample dataset, 500 are negative while the rest 268 are positive.

The KDD 1999 dataset [31] is for detecting and classifying attack/normal network behavior. Each data item has 41 attributes, 34 continuous and 7 categorical. The data are labeled as normal (non-attack) and the rest of the data classified into other classes are attacks. To test our proposed model we used 11 attributes from a total of 41. The chosen 11 continuous attributes were found to contain significantly larger values compared to other continuous attributes. We did not consider the categorical attributes at this stage, as most of the algorithm used in our proposed model can handle only numerical values. The 11 attributes considered are: duration, src_bytes, dst_bytes, count, srv_count, same_srv_count, dst_host_count, dst_host_srv_count, dst_host_same_srv_count, dst_host_same_src_port_rate, dst_host_srv_rerror_rate. The dataset has 4,898,431 records, of which 3,925,651(80.1%) records are labeled as attacks. However, we considered a small part of this dataset consisting of 26,829 data records selected at random, having 25,620 data items labeled as attacks.

4.2. HMM Model parameters

The first step is to set the number of states in the HMM model equal to dimension of the dataset. For example, a 3-dimensional dataset has three states. We have conducted a number of empirical trials using varying number of states; however, there was no significant difference in the results obtained.

The initial HMM parameter A, B and π were initialized randomly. For the prior probability π_i, a random number was chosen and normalized so that

$$\sum_{i=1}^{N} \pi_i = 1 .$$

Since the dataset considered is continuous, the probability of emitting symbols (emission probability B) from a state cannot be calculated. We therefore made use of a single-dimension Gaussian distribution for the observation probability density function. The rationale for this choice was based on the fact that the underlying distribution was not known.

4.3. Experimental Result

We tested our model using each of the benchmark datasets with three other well-known data clustering algorithms, the *k*-means [2], SOM (supervised) [3, 4], and Fuzzy c-means [5-7]. Table 1 displays the performance results (in terms of misclassification rate) for each of the dataset-model combination.

Table 1: Misclassification percentage for some benchmark data

Benchmark Data	Number of predictors	Number of classes	Misclassification percentage			
			HMM	SOM (Supervised)	*k*-means	Fuzzy c-means
Iris Data	4	3	5.33	8.667	15.33	10.67
Thyroid	5	3	12.09	11.1628	22.9♣	26.3♣
Ringnorm	20	2	13.43	21.4189	35.878	23.80
Diabetes	8	2	2.86	27.7344	33.724	31.12
KDD 1999 IDS Dataset	11	2	1.05	1.24	1.27	1.34

♣ Adopted from [30]

For the *k*-means and fuzzy c-means algorithms, we supplied the relevant number of clusters for each of the dataset. For the *k*-means algorithm a 10 iteration stopping rule was chosen to stop the clustering program. For the Iris dataset, the number of clusters was identified from Figure 4.

5. Discussion & conclusion

In this paper, a data clustering algorithm based on single HMM has been proposed to i) identify the number of clusters in a dataset (both visually and algorithmically), and ii) label the data item to its respective clusters. Our model is implemented in three steps. First, using a single HMM the log-likelihood values are calculated for a given dataset. Second, similar log-likelihood values are grouped into windows of fixed size. The windows are then visually scanned to identify logical partitions in the dataset. Finally, labeling of data items to their respective clusters is performed.

Our model provides a means of identifying possible clusters in a given dataset. The plotting of log-likelihood values allows for the identification of partitions in the dataset, thereby providing the necessary information to label data items to clusters. A key component of the model is the examination of similarity measures between data items based on their log-likelihood values. It is the differences in log-likelihood values, which eventually produce clusters of different sizes. The results reported in Table 1, clearly establish that our model outperforms other well-known data clustering models for the benchmark datasets considered.

An important assumption on which the HMM is based, is that there is some relationship (degree of correlation) between the attributes of particular data items in the dataset considered. This in turn provides a measure of similarity (log-likelihood values) between the data items. This is in contrast to most of the existing clustering methods, which assume that each attribute of the data items are independent to each other. The HMM based method can clearly identify the possible number of clusters in the data set, if there is a little or no overlap between two adjacent clusters. This strength of identifying cluster number from scattered data clearly differentiates the proposed method with most of the existing clustering techniques.

Despite the fact that our model has outperformed other data clustering models for the given datasets, we have identified that the performance of the HMM-based technique may get degraded (in terms of misclassification percentage) when the clusters are overlapping (that is, where it is difficult to clearly identify cluster edges). To address this problem, further sensitivity analysis is required in terms of the threshold frequency levels used in the fixed variable sized windows. Having said this, it is worth mentioning that the problem of identifying exact partitioning is an ongoing research challenge for most of the clustering methods.

This algorithm can be used as an online clustering technique, where the similarity level of the online data items is calculated using the trained HMM, and the data items are partitioned into the identified clusters. In future work, we will examine the efficacy of the model using real-world problems (both online and offline) and investigate alternative techniques to deal with overlapping clusters.

References

[1] Jain A. K., Murty, M. N. and Flunn P. J. *Data Clustering: A Review,* ACM Computing Surveys, Vol. 31(3), 264-323, 1999.

[2] Hartigan J. A. and Wong M. A. *Algorithm AS 136: a k-means clustering algorithm,* Applied statistics, **28** , 100-108, 1979.

[3] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Press, 1981.

[4] Bezdek J. C., Ehrlick R. and Full W. *FCM: The fuzzy c-means clustering algorithm,* Comp. Geosci. **10** :2, 191-203, 1984.

[5] Kohonen T. *Self-organization and Associative Memory (3rd edn.),* Springer, 1989.

[6] Kohonen T. *Self-Organizing Maps* Springer Series in Information Sciences, Vol **30**, Springer, 1995.

[7] Begg R., Hassan R., Taylor S. and Palaniswami M. *Artificial Neural Network Models in the Diagnoses of Balance Impairments.* Proceedings of International conference on Intelligent Sensing and Information Processing (`http://www.ieeexplore.org`), pp. 518-522, 2005.

[8] Vapnik V. N. *The Nature of Statistical Learning Theory.* New York: Springer, 1995.

[9] Kaufman L. and Rousseeuw P. J. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, 1990.

[10] Rabiner R. L. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, Vol **77** (2), pp 257-286, 1989.

[11] Sadaoki F. *Speaker Recognition.* `http://cslu.cse.ogi.edu/HLTsurvey/ch1node9.html`

[12] Ajmera J. and Wooters C. A *Robust Speaker clustering algorithm.* IEEE Workshop on Automatic Speech Recognition and Understanding, pp 411-416, 2003.

[13] Che C. W., Qiguang L. and Dong-Suk Y. *An HMM approach to text-prompted speaker verification.* IEEE International Conference on Acoustics, Speech and Signal Processing, pp 673-676, 1996.

[14] Ajmera J., Bourlard H., Lapidot I. and McCowan I. *Unknown-multiple speaker clustering using HMM,* International Conference on Spoken Language Processing, pp. 573-576, 2002.

[15] Huang X., Ariki Y. and Jack M. *Hidden Markov Models for speech recognition,* Edinburgh University Press, 1990.

[16] Xie H., Anreae P., Zhang M. and Warren P. *Learning models for English speech recognition.* Proceedings of the 27th Conference on Australasian Computer Science, pp 323-329, 2004.

[17] MacQueen J. B. *Some Methods for classification and Analysis if Multivariate Observations.* Proceedings of 5-th Berkeley Symposium in Mathematical Statistics and Probability, pp. 281-297, 1967.

[18] *A tutorial on clustering algorithms.*
`http://www.elet.polimi.ot/upload/matteucc/Clustering/tutorial_htm/kmeans.html`

[19] Dunn J. C. *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters,* Journal of Cybernetics, **3** , 32-57, 1973.

[20] Zadeh L. *A. Fuzzy sets.* Info Control, **8**, 338-353, 1965.

[21] Rudhomme E. and Lallich S. *Quality measure based on Kohonen maps for supervised learning of large high dimensional data.* 2005. (`http://asmda2005.enst-bretagne.fr/IMG/pdf/ proceedings/246.pdf`)

[22] Vesanto J. and Alhoniemi E. *Clustering of the Self-Organizing Map.* IEEE Transactions on Neural Networks, Vol. **11** (3), pp. 586-600, 2000.

[23] Hassan M. R. and Nath B. *Stock Market Forecasting using Hidden Markov Model: A new approach.* Proceedings of International Conference on Intelligent Systems Design and Applications, IEEE Computer Society Press, pp. 192-196, 2005.

[24] Baum L. E., Petrie T., Souls G. and Weiss N. *A maximization technique occurring in statistical analysis of probabilistic functions of Markov chains.* Ann Math Stat., Vol **41** (1), pp 164-171, 1970.

[25] Fisher R. A. *The use of multiple measurements in taxonomic problems,* Annual Eugenics **7** , Part II, pp 179-188, 1936.

[26] `www.ics.uci.edu/pub/ml-rwpoa/machine-learning-darabase/` , 2001.

[27] Breiman L. *Bias, variance and arcing classifiers.* Tec. Report 460, Statistics department. University of California. April 1996.

[28] Smith J. W., Everhart J. E., Dickson W. C., Knowler W. C., and Johannes R. S. *Using the ADAP learning algorithm to forecast the onset of diabetes mellitus.* In Proceedings of the Symposium on Computer Applications and Medical Care, IEEE Computer Society Press. pp. 261—265, 1988.

[29] Coomans Broeckaert M. and Brocekaert D.L. *Comparison of multivariate discriminant techniqes for clininical data – application to the thyroid functional state.* Meth. Inform. Med., pp. 93-101, 1983.

[30] Albayrak S. and Amasyali F. *Fuzzy C-means Clustering on Medical Diagnostic Systems.* International XII Turkish Symposium on Artificial Intelligence and Neural Networks, 2003.
`http://www.ce.yildiz.edu.tr/mygetfile.php?id=269`

`[31]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`