ISSN:0975-9646

Automation of Data Clusters based on Layered HMM

G.S.N. Murthy[*], Dr. V. Vijay Kumar[**], K.Krishna Chaitanya[*], Ch. Ravi Kishore[*]
[*]Aditya Institute of Technology And Management(AITAM), India,
[**]Dean-Computer Science & Head-SRRF, GIET, Rajahmundry, India

Abstract
One of the major problems in cluster analysis is the determination of the number of clusters in unlabeled data, which is a basic input for most clustering algorithms. Typically, the clustering algorithm partitions a dataset into a fixed number of clusters supplied by the user, i.e., Given a dataset *O* representing *n* Objects {o_1, o_2… o_n}, clustering aims to partitions data into *c* groups, i.e., C_1, ... C_c, so that $C_i \cap C_j = \emptyset$ and $C_1 \cup C_2 \cup C_3 \cup \ldots\ldots \cup C_c = O$. The present paper propose a novel method ,which is based on Layered Hidden Markov Model(LHMM) to identify a suitable number of clusters in a given unlabeled dataset without using prior knowledge about the number of clusters. For this, the present paper partitions the dataset into windows of fixed/different size based on a novel scheme called log likelihood values of HMM. The proposed scheme works as a framework for identifying the appropriate number of clusters. The proposed method is implemented on Iris dataset. The experimental results indicate the efficacy of the proposed method.

Keywords: LHMM; Unlabeled Dataset; Clusters

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]. Many clustering algorithms require the number of clusters *'c'* as an input parameter, so the quality of the resulting clusters is largely dependent on the estimation of '*c*'. For some applications, users can determine the number of clusters with domain knowledge. However, in many situations, the value of '*c*' is unknown and needs to be estimated from the data themselves. 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) [5] can also be used for classifying patterns from an unknown dataset.

In this paper, a Layered 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 LHMM based data clustering algorithm, and place particular attention to describing the process of creating "window" introduced above.

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 [6]. 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:

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

where $x_i^{(j)}$ is the data point and c_j is the j^{th} 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 [7].

Fuzzy c-means [3, 8] 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 [9]. 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

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

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].

3. THE LAYERED HMM[12]

he layered hidden Markov model (LHMM) is a statistical model derived from the hidden Markov model (HMM). A layered hidden Markov model (LHMM) consists of N levels of HMMs, where the HMMs on level $i+1$ correspond to observation symbols or probability generators at level i. Every level i of the LHMM consists of K_i HMMs running in parallel.

LHMMs are sometimes useful in specific structures because they can facilitate learning and generalization. For example, even though a fully connected HMM could always be used if enough training data were available, it is often useful to constrain the model by not allowing arbitrary state transitions. In the same way it can be beneficial to embed the HMM in a layered structure which, theoretically, may not be able to solve any problems the basic HMM cannot, but can solve some problems more efficiently because less training data is needed.

A layered hidden Markov model (LHMM) consists of N levels of HMMs where the HMMs on level $N+1$ corresponds to observation symbols or probability generators at level N. Every level i of the LHMM consists of K_i HMMs running in parallel.

At any given level L in the LHMM a sequence of T_L observation symbols $O_L=\{O_1,O_2,\ldots..O_{TL}\}$ can be used to classify the input into one of K_L classes, where each class corresponds to each of the K_L HMMs at level L. This classification can then be used to generate a new observation for the level $L-1$ HMMs. At the lowest layer, i.e. level N, primitive observation symbols $O_P=\{O_1,O_2,\ldots O_{TP}\}$would be generated directly from observations of the modeled process. For example in a trajectory tracking task the primitive observation symbols would originate from the quantized sensor values. Thus at each layer in the LHMM the observations originate from the classification of the underlying layer, except for the lowest layer where the observation symbols originate from measurements of the observed process.

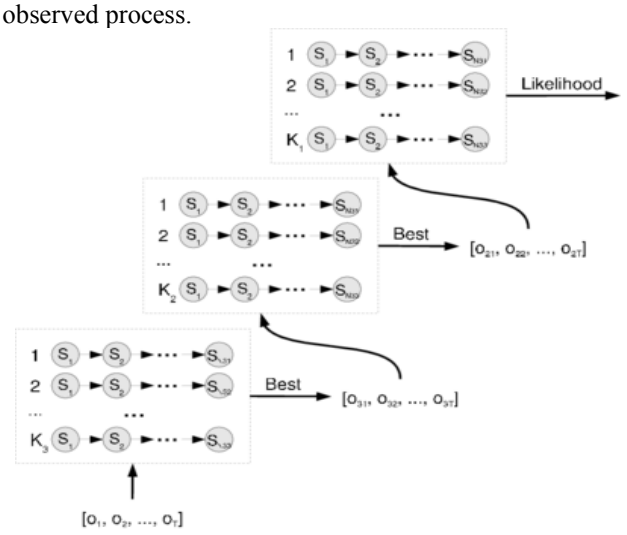

Fig 1: A layered hidden Markov model

It is not necessary to run all levels at the same time granularity. For example it is possible to use windowing at any level in the structure so that the classification takes the average of several classifications into consideration before passing the results up the layers of the LHMM.

Instead of simply using the winning HMM at level $L+1$ as an input symbol for the HMM at level L it is possible to use it as a probability generator by passing the complete probability distribution up the layers of the LHMM. Thus instead of having a "winner takes all" strategy where the most probable HMM is selected as an observation symbol, the likelihood $L(i)$ of observing the i^{th} HMM can be used in the recursion formula of the level L HMM to account for the uncertainty in the classification of the HMMs at level $L+1$. Thus, if the classification of the HMMs at level $n+1$ is uncertain, it is possible to pay more attention to the a-priori information encoded in the HMM at level L.

A LHMM could in practice be transformed into a single layered HMM where all the different models are concatenated together. Some of the advantages that may be expected from using the LHMM over a large single layer HMM is that the LHMM is less likely to suffer from over-fitting since the individual sub-components are trained independently on smaller amounts of data. A consequence of this is that a significantly smaller amount of training data is required for the LHMM to achieve a performance comparable of the HMM. Another advantage is that the layers at the bottom of the LHMM, which are more sensitive to changes in the environment such as the type of sensors, sampling rate etc, can be retrained separately without altering the higher layers of the LHMM. The model is shown below

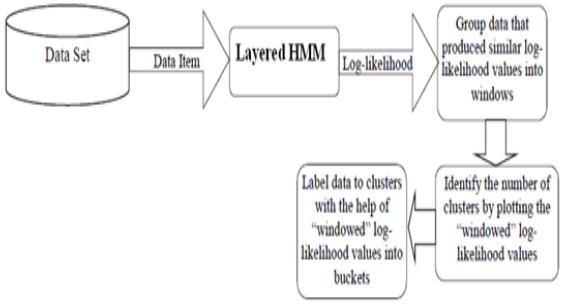

Fig 2: The LHMM based model to cluster multidimensional dataset

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.

3.1. Generating likelihood values

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) $\prod$, matrix of prior probabilities.

The parameter set (A, B, $\prod$), or simply λ represents the overall HMM model. To fit a model for a given observation sequence, the model parameters A , B and $\prod$ 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 (BW) 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.

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 the stste sequence $Q=(q_1,q_2,\ldots\ldots q_T)$, we have

$$P(O|\lambda)=\prod_{t=1}^{T}P(O_t|q_t,\lambda)=b_1(O_1)b_2(O_2),\ldots..b_t(O_t) \text{ and}$$

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

$$\text{Now,} \quad P(O,Q \mid \lambda) = P(O \mid Q\lambda)P(Q \mid \lambda)$$

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[10,11]. Here for completeness, we describe only the forward procedure.

3.2 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

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

This forward variable may be computed as follows:

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

Induction $$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$$

$1\le t\le T-1,\ 1\le j\le N$

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

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

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

3.3. 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 [5,13]. 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 3 displays the log-likelihood values of Iris dataset ,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. Log-likelihood values in Figures 3 suggests that there are distinct breaks or changes in the curves, which leads us to think of dividing lines for possible clusters

3.4. Labeling Data for Clusters

In the section 3.3 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 3.3 , some of the windows (empty as well as nonempty) in close neighborhood are merged.

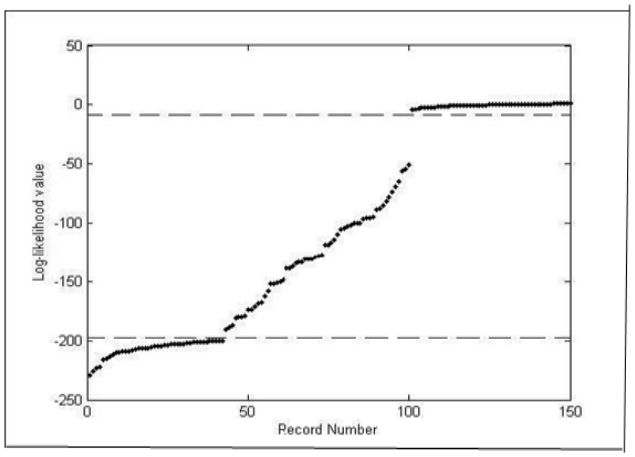

Fig 3: Number of clusters (Iris Dataset)

4. DISCUSSION AND CONCLUSION

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.

A LHMM could in practice be transformed into a single layered HMM where all the different models are concatenated together. Some of the advantages that may be expected from using the LHMM over a large single layer HMM is that the LHMM is less likely to suffer from over-fitting since the individual sub-components are trained independently on smaller amounts of data. A consequence of

this is that a significantly smaller amount of training data is required for the LHMM to achieve a performance comparable of the HMM. Another advantage is that the layers at the bottom of the LHMM, which are more sensitive to changes in the environment such as the type of sensors, sampling rate etc, can be retrained separately without altering the higher layers of the LHMM.It is worth mentioning that the problem of identifying exact partitioning is an ongoing research challenge for most of the clustering methods.

REFERENCES

[1] Jain A. K., Murty, M. N. and Flunn P. J. Data Clustering: A Review, ACM Computing Surveys, Vol. 31(3), 264323, 1999.

[2] Hartigan J. A. and Wong M. A. Algorithm AS 136: a kmeans clustering algorithm, Applied statistics, 28 , 100108, 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 cmeans clustering algorithm, Comp. Geosci. 10 :2, 191203, 1984.

[5] Rabiner R. L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, Vol 77 (2), pp 257286, 1989.

[6] MacQueen J. B. Some Methods for classification and Analysis if Multivariate Observations. Proceedings of 5th Berkeley Symposium in Mathematical Statistics and Probability, pp. 281297, 1967.

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

[8] Dunn J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact WellSeparated Clusters, Journal of Cybernetics, 3 , 3257, 1973.

[9] Zadeh L. A. Fuzzy sets. Info Control, 8, 338353, 1965.

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

[11] 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 164171, 1970.

[12] [N.oliver, A.Garg and E.Horviz,"layered Representations for Learning and inferring Office Acitivity from multiple sensory channels", Computer vision and image understanding,vol.96,p.163-180,2004,]

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

G S N MURTY, M.Tech., M.C.A.,(Ph.D), Presently working as Associate Professor in the dept.of M.C.A in Aditya Institute of Technology And Management.

Dr. V. VIJAY KUMAR, Dean-Computer Science & Head-SRRF, GIET, Rajahmundry.

K.KRISHNA CHAITANYA, M.Tech(IT) GITAM University and M.Sc(Physics) from Acharya Nagarjuna University. Presently working as Assistant Professor in the dept.of Information Technology in Aditya Institute of Technology And Management.

CH. RAVI KISHORE, M.Tech from Birla Institute of Technology (Mesra) and M.Sc(Mathematics) from Andhra University. Presently working as Assistant Professor in the dept.of Information Technology in Aditya Institute of Technology And Management.