Incremental Mixture Learning for Clustering Discrete Data

Konstantinos Blekas and Aristidis Likas

Department of Computer Science, University of Ioannina, 45110 Ioannina, Greece
{kblekas,arly}@cs.uoi.gr
(submitted to the 3^{rd} Hellenic Conference on Artificial Intelligence, SETN 2004)

Abstract. This paper elaborates on an efficient approach for clustering discrete data by incrementally building multinomial mixture models through likelihood maximization using the Expectation-Maximization (EM) algorithm. The method adds sequentially at each step a new multinomial component to a mixture model based on a combined scheme of global and local search in order to deal with the initialization problem of the EM algorithm. In the global search phase several initial values are examined for the parameters of the multinomial component. These values are selected from an appropriately defined set of initialization candidates. Two methods are proposed here to specify the elements of this set based on the agglomerative and the kd-tree clustering algorithms. We investigate the performance of the incremental learning technique on a synthetic and a real dataset and also provide comparative results with the standard EM-based multinomial mixture model.

1 Introduction

Clustering of discrete (or categorical) data is an important problem with many significant applications [1–4]. Although several methods have been proposed for clustering continuous (real) data, the clustering of discrete data seems to be more difficult mainly due to the nature of the discrete data: discrete values cannot be ordered, it is not straightforward to define 'distance' measures and it is also more difficult to specify appropriate differentiable objective functions and apply continuous optimization methods to adjust the clustering parameters.

Nevertheless, several techniques have been proposed for clustering discrete data [1–3]. Some of them transform the discrete features into continuous using some type of encoding, most of them 1-of-K encoding for a feature assuming K discrete values [2]. A disadvantage of such methods is that the dimensionality of the input space becomes very large. Other techniques are simply based on the definition of a distance measure (e.g. Hamming distance) which is exploited to construct hierarchical clustering solutions (e.g. agglomerative) [1, 3].

In this work we focus on *statistical model-based* methods for clustering discrete data [5, 3]. Such methods are based on the *generative model* paradigm and assume that the data have been generated by an appropriate mixture model

whose parameters can be identified through the maximization of a likelihood function.

More specifically we consider a *mixture of multinomials* model and assume that each data point has been generated through sampling from some multinomial component of the mixture model [3]. It is well-known that the EM algorithm can be employed to adjust the parameters of the model. Once the model has been trained, a data point is assigned to the cluster (multinomial component) with the highest posterior probability. An additional advantage of this approach is that it allows for soft clustering solutions based on the values of the posterior probabilities.

The main problem with EM is the dependence on the initial parameter values. An effective incremental solution has been recently proposed for the multinomial mixture model, which has been successfully applied in a bioinformatics context [4]. This method starts with one component and each time attempts to optimally add a new component to the current mixture through the appropriate use of global and local search procedures. As it will be described later, the application of the incremental approach requires the specification of set $\mathcal{C}$ of candidate parameter vectors for the new component to be added at each step. In [4] the set $\mathcal{C}$ was considered to contain as many elements as the training set. In this work we propose and evaluate two other methods for constructing the set $\mathcal{C}$ of initialization candidates, based on the methods *kd-tree* and *agglomerative clustering*. Comparative experimental results indicate that the integration of the agglomerative clustering approach into the incremental multinomial mixture learning method leads to a very powerful method for clustering discrete data.

2 An incremental scheme for multinomial mixture models

2.1 The mixture of multinomials model

Consider a dataset $X = \{\mathbf{x}^1 \dots \mathbf{x}^N\}$, where each data point $\mathbf{x} = x_1, \dots, x_d$ contains features with discrete values. More specifically, we assume that each feature x_i $(i = 1, \dots, d)$ can take values from a finite set Ω_i of L_i discrete values, i.e. $x_i \in \Omega_i = \{\alpha_1^i, \dots, \alpha_{L_i}^i\}$. We also assume that each feature x_i can be modeled with a multinomial distribution

$$P(x_i = \alpha_l^i) = p_{il} \text{ , with } |\mathbf{p}_i| = \sum_{l=1}^{L_i} p_{il} = 1 \text{ .} \tag{1}$$

The probabilistic vectors $\mathbf{p}_i$ $(i = 1, \dots, d)$ define a multinomial parameter vector θ, i.e. $\theta = [\mathbf{p}_1 \dots \mathbf{p}_d]$. Assuming that the features are independent, the density function $\phi(\mathbf{x}^n|\theta)$ for an arbitrary observation $\mathbf{x}^n = (x_1^n \dots x_d^n)$ is given by

$$\phi(\mathbf{x}^n|\theta) = \prod_{i=1}^{d} \prod_{l=1}^{L_i} p_{il}^{\mathbf{I}(x_i^n, l)} \text{ ,} \tag{2}$$

where $\mathbf{I}(x_i^n, l)$ is a binary indicator function such that $\mathbf{I}(x_i^n, l) = 1$ if $x_i^n = \alpha_l^i$ and 0 otherwise.

A *mixture of multinomials* model $f(\mathbf{x}^n|\Psi_k)$ with k components is defined as:

$$f(\mathbf{x}^n|\Psi_k) = \sum_{j=1}^{k} \pi_j \phi_j(\mathbf{x}^n|\theta^j), \tag{3}$$

where Ψ_k is the vector of all unknown parameters, i.e. $\Psi_k = [\pi_1 \dots \pi_k, \theta^1 \dots \theta^k]$ and the mixing proportion π_j ($\pi_j \geq 0$) satisfy that $\sum_{j=1}^{k} \pi_j = 1$.

The log-likelihood of the dataset $X = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ given the above model is

$$\mathcal{L}(\Psi_k) = \sum_{n=1}^{N} \log f(\mathbf{x}^n|\Psi_k). \tag{4}$$

The EM algorithm provides a straightforward, convenient approach for maximum likelihood (ML) estimation of the parameters of the component densities based on the iterative application of the following update equations for each component $j = 1, \dots, k$ [6, 3, 4]:

$$z_j^{n^{(t+1)}} = \frac{\pi_j^{(t)} \phi_j(\mathbf{x}^n|\theta^{j^{(t)}})}{\sum_{j=1}^{k} \pi_j^{(t)} \phi_j(\mathbf{x}^n|\theta^{j^{(t)}})}, \tag{5}$$

$$\pi_j^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} z_j^{n^{(t+1)}}, \tag{6}$$

$$p_{il}^{j^{(t+1)}} = \frac{\sum_{n=1}^{N} z_j^{n^{(t+1)}} \mathbf{I}(x_i^n, l)}{\sum_{n=1}^{N} \sum_{r=1}^{L_i} z_j^{n^{(t+1)}} \mathbf{I}(x_i^n, r)} \ . \tag{7}$$

After training the multinomial mixture model, we can assign to a data point $\mathbf{x}^n$ a cluster label corresponding to the highest posterior probability value z_j^n ($j = 1, \dots, k$).

It is well-known that the quality of the solutions provided by the EM algorithm depend highly on the initialization of the model parameters. To overcome the problem of poor initialization for the multinomial mixture model, an incremental learning scheme [4] has been proposed based on an appropriate adaptation of the greedy-EM algorithm for Gaussian mixtures [7].

2.2 Incremental mixture learning

Assume that a new component $k+1$ with density $\phi_{k+1}(\mathbf{x}^n|\theta^{k+1})$ is added to a k-component mixture model $f(\mathbf{x}^n|\Psi_k)$. This new component corresponds to a new cluster in the discrete domain modeled by a parameter vector θ^{k+1} containing the multinomial parameters. The resulting mixture with $k+1$ components can be represented as

$$f(\mathbf{x}^n|\Psi_{k+1}) = (1-a) f(\mathbf{x}^n|\Psi_k) + a\phi_{k+1}(\mathbf{x}^n|\theta^{k+1}), \tag{8}$$

where $a \in (0,1)$. The new parameter vector Ψ_{k+1} consists of the parameter vector Ψ_k of the k-component mixture, the weight a and the vector θ^{k+1}. Then, the log-likelihood for Ψ_{k+1} is given by

$$\mathcal{L}(\Psi_{k+1}) = \sum_{n=1}^{N} \log f(\mathbf{x}^n|\Psi_{k+1}) =$$

$$\sum_{n=1}^{N} \log\{(1-a)f(\mathbf{x}^n|\Psi_k) + a\phi_{k+1}(\mathbf{x}^n|\theta^{k+1})\}. \quad (9)$$

The above formulation proposes a two-component likelihood maximization problem, where the first component is described by the old mixture $f(\mathbf{x}^n|\Psi_k)$ and the second one is the new component with density $\phi_{k+1}(\mathbf{x}^n|\theta^{k+1})$, where $\theta^{k+1} = [\mathbf{p}_1^{k+1} \dots \mathbf{p}_d^{k+1}]$. If we consider that the parameters Ψ_k of $f(\mathbf{x}^n|\Psi_k)$ remain fixed during maximization of $\mathcal{L}(\Psi_{k+1})$, the problem can be treated by applying searching techniques to optimally specify the parameters a and θ^{k+1} which maximize $\mathcal{L}(\Psi_{k+1})$. An efficient technique for the specification of θ^{k+1} and a is presented in [7] that follows a combination of global and local searching.

Global Search: It has been shown that a local maximum of $\mathcal{L}(\Psi_{k+1})$ with respect to a for a given parameter vector $\theta^{k+1} = \vartheta^m$, is given by [7]

$$\hat{\mathcal{L}}(\Psi_{k+1}) = \hat{\mathcal{L}}(\vartheta^m) = \sum_{n=1}^{N} \log \frac{f(\mathbf{x}^n|\Psi_k) + \phi_{k+1}(\mathbf{x}^n|\vartheta^m)}{2} + \frac{1}{2}\frac{[\sum_{n=1}^{N} \delta(\mathbf{x}^n|\vartheta^m)]^2}{\sum_{n=1}^{N} \delta^2(\mathbf{x}^n|\vartheta^m)}, \quad (10)$$

and is obtained for

$$\hat{a} = \frac{1}{2} - \frac{1}{2}\frac{\sum_{n=1}^{N} \delta(\mathbf{x}^n|\vartheta^m)}{\sum_{n=1}^{N} \delta^2(\mathbf{x}^n|\vartheta^m)}, \quad (11)$$

where

$$\delta(\mathbf{x}^n|\vartheta^m) = \frac{f(\mathbf{x}^n|\Psi_k) - \phi_{k+1}(\mathbf{x}^n|\vartheta^m)}{f(\mathbf{x}^n|\Psi_k) + \phi_{k+1}(\mathbf{x}^n|\vartheta^m)}. \quad (12)$$

The above formulation has the benefit of making the problem of likelihood maximization (Equation 9) independent of a. Therefore, it restricts global searching for finding good initial values ϑ^m for the multinomial distribution of the newly inserted component. To this end, the problem is now to define a proper set $\mathcal{C} = \{\vartheta^m, m = 1, \dots, M\}$ of initialization *candidates*. Then, the candidate $\hat{\theta}_{k+1} \in \mathcal{C}$ that maximizes Equation 10 is identified and the corresponding $\hat{a}$ value is computed using Equation 11.

Local Search: The EM algorithm can be used to perform local search for the maximum of the likelihood with respect to parameters a and θ_{k+1} only, starting from the values $\hat{a}$ and $\hat{\theta}_{k+1}$ identified in the global search phase. In analogy to Equations 5-7, the following update equations called *partial EM* can be derived

for maximizing $\mathcal{L}(\Psi_{k+1})$:

$$z_{k+1}^{n^{(t+1)}} = \frac{a^{(t)}\phi_{k+1}(\mathbf{x}^n|\theta^{k+1^{(t)}})}{(1-a^{(t)})f(\mathbf{x}^n|\Psi_k) + a^{(t)}\phi_{k+1}(\mathbf{x}^n|\theta^{k+1^{(t)}})}, \tag{13}$$

$$a^{(t+1)} = \frac{1}{N}\sum_{n=1}^{N} z_{k+1}^{n^{(t+1)}}, \tag{14}$$

$$p_{il}^{k+1^{(t+1)}} = \frac{\sum_{n=1}^{N} z_{k+1}^{n^{(t+1)}}\mathbf{I}(x_i^n, l)}{\sum_{n=1}^{N}\sum_{r=1}^{L_i} z_{k+1}^{n^{(t+1)}}\mathbf{I}(x_i^n, r)} \ . \tag{15}$$

The performance of the above incremental algorithm highly depends on the 'quality' of the initialization candidates included in set $\mathcal{C}$. In the following sections we describe and evaluate several methods for candidate specification.

3 Methods for the specification of initialization candidates

3.1 Exhaustive search over the training set

A reasonable and straightforward strategy to define the set $\mathcal{C}$ of candidates is to consider the whole training set $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, and directly associate each discrete data point $\mathbf{x}^m$ with a multinomial distribution $\vartheta^m = [\rho_1^m \ldots \rho_d^m]$ constructed as follows:

$$\rho_{il}^m = \lambda^{\mathbf{I}(x_i^m, l)}\left(\frac{1-\lambda}{L_i - 1}\right)^{1-\mathbf{I}(x_i^m, l)} \ , \tag{16}$$

It is easy to show that $\sum_{l=1}^{L_i} \rho_{il}^m = 1$ for each feature i $(i = 1, \ldots, d)$. The parameter λ has a fixed value in the range $(0, 1)$, and should satisfy $\lambda \geq 1/L_i$ $(\forall i)$. In such way a set with $M = N$ candidates is created. We will refer to this method as ES (Exhaustive Search).

The drawback of this method is that all the N data points of X must be examined each time a new component has to be inserted. Alternatively, we can use data partitioning schemes that lead to the identification of much less candidates $(M << N)$ and more informative.

3.2 The *k*d-tree algorithm for partitioning discrete data

The first partitioning scheme we have used is based on the notion of kd-trees. Initially, kd-trees [8] were proposed in the case of continuous data, as an attempt to speed-up the execution of nearest neighbor queries. A kd-tree defines a recursive binary partitioning of a k-dimensional dataset, where the root node contains all data. A subset of the original dataset is assigned to each tree node and the tree construction procedure proceeds by partitioning the subset of a node into two

subsets using a hyperplane perpendicular to the direction for which the subset data demonstrate the highest variance.

In order to deal with discrete features we have used the following entropy-based procedure to partition the data points corresponding to a node. In particular, for a node m that contains N^m data points, we first calculate its multinomial parameters $\rho_{il}^m = N_{il}^m / N^m$, based on the sufficient statistics N^m and $N_{il}^m = \sum_{n=1}^{N^m} \mathbf{I}(x_i^n, l)$. Then, the entropy H_i^m of each feature i ($i = 1, \ldots, d$) for the data of node m can be computed as follows

$$H_i^m = -\sum_{l=1}^{L_i} \rho_{il}^m \log \rho_{il}^m \ . \tag{17}$$

Then, we can identify the feature $i^\star$ that exhibits the largest entropy value, i.e. $i^\star = \arg\max_i H_i^m$. The partitioning procedure is based on the values $x_{i^\star}$ of the data points belonging to node m. First the $L_{i^\star}$ different values of the feature $i^\star$ are sorted according to the probabilities $\rho_{i^\star l}^m$ and then each value is marked in turn as *odd* or *even*. In this way, two new nodes m_1 and m_2 are created, where the m_1 (m_2) node contains the node m data for which the value of feature $i^\star$ is marked as odd (even).

One last observation that must be made concerns the selection of the leaf node m that will be partitioned at each step. This is done by selecting among the current leaf nodes of the tree the one that exhibits the minimum likelihood value. Following the definition in Equation 2, the log-likelihood that characterizes a node m is

$$\begin{aligned} \mathcal{L}_m(\vartheta^m) &= \sum_{n=1}^{N_m} \sum_{i=1}^{d} \sum_{l=1}^{L_i} \mathbf{I}(x_i^n, l) \log \rho_{il}^m \\ &= \sum_{i=1}^{d} N^m \sum_{l=1}^{L_i} \rho_{il}^m \log \rho_{il}^m = -\sum_{i=1}^{d} N^m H_i^m \ . \end{aligned} \tag{18}$$

The above top-down procedure builds a tree with several nodes and the partitioning of a leaf node is not allowed when the number of included data points is lower than a fixed value T. The kd-tree construction procedure terminates either when there exist no leaf nodes that can be splitted, or when a predetermined number M of leaf nodes have been constructed.

In our experiments in order to specify a candidate initialization set $\mathcal{C}$ with M vectors ϑ^m, a kd-tree was first constructed with M leafs. Then each vector ϑ^m was determined from the sufficient statistics of the data points assigned to the corresponding leaf m ($m = 1, \ldots, M$). We will refer to this method as KD.

3.3 Agglomerative clustering

In contrast to the kd-tree method, the Agglomerative clustering (AC) is a synthetic clustering scheme [3]. The method starts with a set of N clusters, each

containing one data point $\mathbf{x}^n$. At each step the AC method searches among the set of current clusters to identify the two closest clusters (m_1, m_2) that are subsequently merged into one cluster by assigning all data points of clusters m_1, m_2 to the newly formed cluster $< m_1, m_2 >$. To apply the AC algorithm, an intercluster distance measure $d(m_1, m_2)$ is needed, defined as

$$d(m_1, m_2) = \mathcal{L}_{m_1}(\vartheta^{m_1}) + \mathcal{L}_{m_2}(\vartheta^{m_2}) - \mathcal{L}_{<m_1,m_2>}(\vartheta^{<m_1,m_2>}) \ , \qquad (19)$$

where the $\mathcal{L}_m(\vartheta^m)$ is calculated, as in the kd-tree case, according to Equation 18. The algorithm terminates when a specified number M of clusters have been found. An implementation of the AC algorithm for discrete data is presented in [3] that requires nearly $\mathcal{O}(N^2)$ runtime.

Once the AC algorithm has terminated with M clusters, we specify the elements ϑ^m $(m = 1, \ldots, M)$ of $\mathcal{C}$ using exactly the same procedure with the kd-tree approach described previously.

4 Experimental results

We have conducted a series of experiments to evaluate how the different procedures for candidate specification influence the performance of the incremental algorithm. Using each method (ES, KD, AC) three sets of candidates were specified and the incremental scheme was then applied for initializing the parameters of each added component k. Moreover, two additional standard EM-based multinomial mixture models were created, where their parameters were initialized by the agglomerative and the kd-tree clustering algorithms, respectively. More specifically, we first applied both algorithms until a number of K clusters were created. Then, the statistics of each cluster k were used for initializing the mixture model parameters and the EM algorithm was applied to adjust them. We will refer to these two models as AC-EM and KD-EM, respectively. Two evaluation criteria have been used, the first being the likelihood of each obtained mixture model on a test set and the second the test set classification accuracy (although class labels were not used in training).

4.1 Experiments with a synthetic dataset

In the synthetic dataset used in our experiments each true cluster (called class) j $(j = 1, \ldots, K)$ was associated with a unique *generator* string of length d containing letters from an alphabet $\Omega = \{A, B, \ldots, I\}$, ie. $\Omega_i = \Omega, \forall i = 1, \ldots, d$. In this way, $K = 10$ different data generators with $d = 20$ features were selected, where some of them present high degree of similarity and thus the discrimination among the corresponding clusters is difficult.

The data points of each class j were created as noisy copies of generator string j by randomly deciding whether to mutate each generator feature i with mutation probability $\varrho = 0.4$ $(0 \leq \varrho \leq 1)$, where mutation means that the value of a feature i changes by randomly selecting a value from the corresponding

alphabet Ω_i. In this way, 10000 discrete data points were sampled (1000 for each class) and 3000 of them (300 for each class) were selected for training, while the rest 7000 data were used for testing.

Table 1 presents the results for the synthetic dataset. The AC and KD clustering algorithms were run until $M = 200$ subsets were discovered, which are subsequently used to specify the multinomial parameter vectors included in the set $\mathcal{C}$ of initialization candidates. As the results indicate, the exhaustive search (ES) method for defining the set $\mathcal{C}$, although considering the whole training set, does not provide as good results as the other two approaches AC and KD which illustrate the best (and similar) performance for both criteria.

It is also interesting to note that the proposed incremental schemes compare favorably to both AC-EM [3] and KD-EM, ie. stand-alone EM with the K components, initialized from the statistics of K subsets obtained using either AC or kd-tree. Since the AC-EM approach is considered one of the most effective approaches for clustering discrete data[3], it can be concluded that the proposed methods are very powerful. These conclusions are also supported from the experiments on a real dataset described below.

Table 1. Performance of the incremental approach using three methods for specifying initialization candidates as well as of the AC-EM and KD-EM models on the synthetic dataset.

Performance criteria	*Incremental mixture*			*AC-EM*	*KD-EM*
	Initialization methods				
	AC	*KD*	*AS*		
log-Likelihood (training)	-92612	-92612	-94108	-92715	-93209
log-Likelihood (test)	-220067	-220067	-223676	-220276	-221574
Classification rate (%)	98.21	98.21	72.09	97.33	90.66

4.2 Experiments with the mushrooms dataset

We have also conducted experiments with a real dataset, namely the mushrooms dataset from the UCI Machine Learning repository. It consists of $d = 22$ discrete features that describe physical attributes of mushrooms taking between 2 and 12 values. Totally there are 8124 discrete data labeled as either class 0 (poisonous mushrooms) or class 1 (eligible mushrooms), which were equally divided into a training and testing set (4062 cases in each dataset). In all experiments the class labels were ignored in the training phase. We first applied the AC algorithm to construct a set $\mathcal{C}$ of $M = 200$ initialization candidates for the incremental mixture learning method. The kd-tree algorithm was also applied to create another set $\mathcal{C}'$ of $M = 200$ candidates. The incremental learning scheme using either $\mathcal{C}$ or $\mathcal{C}'$ was compared to AC-EM method.

Figure 1 displays the log-likelihood value $\mathcal{L}(\Psi_K)$ on the test set as a function of the number of clusters (components) K. The superiority of the incremental

approach when initialized with the AC algorithm is clear for all values of K ($2 \leq K \leq 12$). In contrast to the synthetic dataset, the kd-tree algorithm did not offer so good values for initializing the multinomial components, and thus the corresponding model provided worse results for this dataset. Moreover, the superiority of the incremental AC scheme can also be seen in Table 2 providing the class distribution in the clusters obtained for $K = 10$.

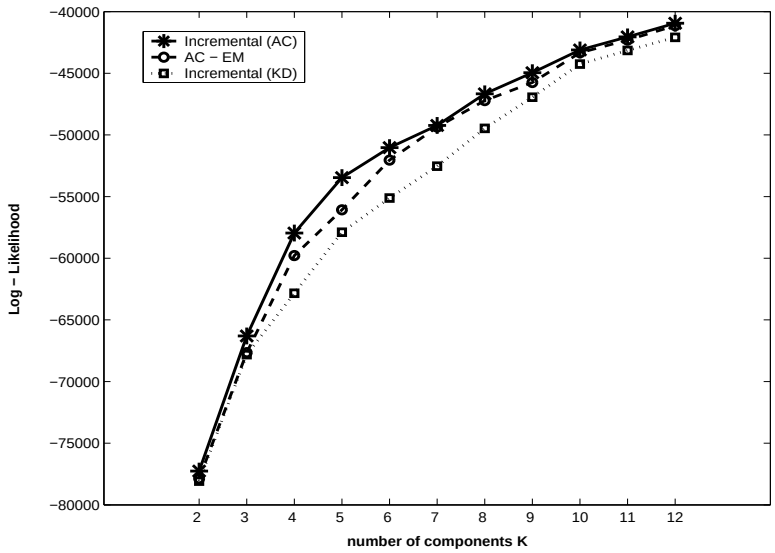

Fig. 1. The test set log likelihood values for several values of the number of multinomial components K.

5 Conclusions

In this paper we elaborated on an incremental scheme for model-based clustering of discrete data, where a multinomial model is constructed by sequentially adding new components. An exploration mechanism based on global and local search ensures the fine tuning of the parameter vector of the added multinomial component. To solve the problem of specifying the set $\mathcal{C}$ of initialization candidates two clustering methods have been examined based on kd-trees and agglomerative clustering. The experiments conducted on a synthetic and a real dataset have shown the incremental training method, coupled with a powerful technique for the specification of initialization candidates, constitutes a very effective approach for clustering discrete data.

In cases of very large datasets, the application of the AC algorithm to the whole training set is time consuming [3]. A solution to this problem is to apply

Table 2. Class distribution in each cluster obtained when running the algorithms with $K = 10$ multinomial components.

Model	*Class*	*Class distribution in every cluster*										*Classification rate (%)*
		1	2	3	4	5	6	7	8	9	10	
Incremental	**0**	655	130	0	0	0	0	870	139	0	185	95.44
mixture (AC)	**1**	0	0	127	105	855	338	0	0	102	556	
AC-EM	**0**	655	130	0	0	0	139	870	101	84	0	92.27
	1	0	0	127	105	855	338	0	91	172	395	
Incremental	**0**	655	130	0	0	0	0	934	240	0	20	86.32
mixture (KD)	**1**	0	0	127	105	855	338	508	48	102	0	

the AC method to a randomly selected portion of the dataset. On the other hand, the size of the dataset does not constitute a problem for the proposed kd-tree algorithm, therefore, the use of kd-tree for the specification of initialization candidates may lead to better results in the case of very large datasets. It is also possible to develop hybrid schemes combining the AC and kd-tree methods. An approach of this type would first divide the dataset into a number of subsets using the kd-tree method and then would apply the AC algorithm to the data of each subset. Finally, it must be noted that in this work we do not address the problem of assessing the optimal number of multinomial components K. This constitutes one of our future research directions and requires the adaptation of several well-known methodologies and criteria for model selection [5].

References

1. Cheeseman P. and Stutz J. Bayesian classification (AutoClass): Theory and resutls. In U. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. CA: AAAI Press, 1995.
2. Bengio Y. and Bengio S. Modeling high-dimensional discrete data with multi-layer neural networks. In S.A. Solla, T.K. Leen, and K.-R. Móller, editors, *Advances in Neural Processing Systems 12*, pages 400–406. MIT Press, 2000.
3. Meilă M. and Hecherman D. An experimental comparison of model-based clustering methods. *Machine Learning*, 42:9–29, 2001.
4. Blekas K., Fotiadis D.I., and Likas A. Greedy mixture learning for multiple motif discovering in biological sequences. *Bioinformatics*, 19(5):607–617, 2003.
5. Chickering D. and Heckerman D. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997.
6. Render R.A. and Walker H.F. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.
7. Vlassis N. and Likas A. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15:77–87, 2002.
8. Bentley J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.