J Glob Optim (2007) 37:513–525
DOI 10.1007/s10898-006-9059-3

ORIGINAL ARTICLE

Speed-up for the expectation-maximization algorithm for clustering categorical data

F.-X. Jollois · M. Nadif

Received: 17 June 2006 / Accepted: 23 June 2006 / Published online: 5 August 2006
© Springer Science+Business Media B.V. 2006

Abstract In model-based cluster analysis, the expectation-maximization (EM) algorithm has a number of desirable properties, but in some situations, this algorithm can be slow to converge. Some variants are proposed to speed-up EM in reducing the time spent in the E-step, in the case of Gaussian mixture. The main aims of such methods is first to speed-up convergence of EM, and second to yield same results (or not so far) than EM itself. In this paper, we compare these methods from categorical data, with the latent class model, and we propose a new variant that sustains better results on synthetic and real data sets, in terms of convergence speed-up and number of misclassified objects.

Keywords Mixture model · Expectation-maximization algorithm · Clustering · Acceleration · Categorical data

1 Introduction

Many clustering methods used in practice are based on a distance or a dissimilarity measure. However, basing cluster analysis on mixture models has become a classical and powerful approach. In fitting a mixture model to a data by maximum likelihood via the expectation-maximization (EM) algorithm (Demspter et al. 1977), the partition is derived from the conditional probabilities in using the maximum a posteriori (MAP) principle. Recently the clustering of categorical data has attracted much attention. One example includes the popular tool in the data mining field, Autoclass, developed

F.-X. Jollois (✉)
CRIP5, Université Paris Descartes, 45 rue des Saint-Pères, 75270 Paris Cedex 06, France
e-mail: jollois@univ-paris5.fr

M. Nadif
LITA – UFR MIM, Université Paul Verlaine-Metz, Ile du Saulcy, 57045 Metz Cedex 1, France
e-mail: mohamed.nadif@univ-metz.fr

by Cheeseman and Stutz (1996). Note that, in this case the latent class model is in common use for its simplicity and its performance.

In a mixture component analysis context, classical optimization techniques such as Newton–Raphson or gradient methods can be used but the EM algorithm is commonly employed. Unfortunately, the most documented problem occuring with EM is its possible poor rate of convergence in some situations. Different alternatives have recently been considered that guarantee a monotonic increase in the objective function. These methods can usually be cast as variants of the generalized EM (GEM) algorithm (Demspter et al. 1977), in which the M-step improves rather than maximizes the expected complete data log-likelihood. Such work includes the Expectation Conditional Maximization (ECM) algorithms and generalizations (Meng and van Dyk 1997). Other variants replace the M-step with a faster (conjugate) gradient step (Jamshidian and Jennrich 1993) or (quasi) Newton type step (Meilijson 1989).

These kinds of techniques are unsuited for large quantities of data. The time spent in the E-step is linearly dependent of the number of objects. Consequently, some variants seek to reduce the cost of EM by reducing the time spent in the E-step (Moore 1999; McCallum et al. 2000). These methods do not maintain the convergence guarantees of EM. In this paper, we consider only the algorithms that are guaranteed to converge to a local maximum. We focus here on the variants known as Sparse EM (SpEM) and Lazy EM (LEM). These versions, which are based on a partial E-step, provide encouraging results for Gaussian mixtures (Neal and Hinton 1998; Thiesson et al. 2001). The main objectives of this article are to extend these works to the case of categorical data by using the latent class model and to propose an interesting variant, called *e*LEM, which gives much better performance than these methods. But the major disadvantage of these kinds of methods is that they depend on the number of iterations of the partial E-step and a threshold used to select the objects for this step. Then, we propose an efficient strategy, *e*LEM (*), that permits us to avoid this crucial choice of these parameters, and which yields very good results to speeding-up the EM algorithm.

The outline of the remaining sections is as follows. In Sect. 2, we first introduce the mixture model and cluster analysis concepts. Then, we describe both approaches based on the EM and the Classification EM algorithms. In Sect. 3, which is devoted to the speed-up of EM, we review the SpEM and LEM variants and propose a new version called *e*LEM. To study the behavior of these methods, we present detailed experimental results on synthetic and real data in Sect. 4. Finally, some concluding remarks are made in the last section.

2 Model-based cluster analysis

2.1 Finite mixture model

In the mixture approach, it is assumed that the objects $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to be clustered are from a mixture of s component densities in some unknown proportions $p_1, \ldots, p_s$. That is, each object $\mathbf{x}_i$ is taken to be a realization of the mixture probability density function (pdf.)

$$f(\mathbf{x}_i; \boldsymbol{\theta}) = \sum_{k=1}^{s} p_k \varphi_k(\mathbf{x}_i; \boldsymbol{\alpha}_k),$$

where $\varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k)$ denotes the density of $\mathbf{x}_i$ from the kth component and $\boldsymbol{\alpha}_k$ is the corresponding mixture component parametric description. Here, the vector $\boldsymbol{\theta}$ of unknown parameters consists of the mixing proportions $(p_1,\ldots,p_{s-1}) = \mathbf{p}$ (with the constraint $\sum_{k=1}^{s} p_k = 1$) and the parameters of each component $(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_s) = \boldsymbol{\alpha}$. From the observed data $\mathbf{x} = (\mathbf{x}_1,\ldots,\mathbf{x}_n)$, the log-likelihood is given by

$$L(\mathbf{x},\boldsymbol{\theta}) = \log f(\mathbf{x};\boldsymbol{\theta}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{s} p_k \varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k)\right)$$

assuming that the observations $\mathbf{x}_1,\ldots,\mathbf{x}_n$ are independent. In the clustering context, each $\mathbf{x}_i$ is conceptualized as being generated by one of the components of the mixture model being fitted. Thus, let $\mathbf{z}_i$ be a s-dimensional vector with $z_{ik} = 1$ or 0, according to if $\mathbf{x}_i$ was or was not produced by the kth component. The complete data is therefore declared to be $(\mathbf{x}_1,\mathbf{z}_1),\ldots,(\mathbf{x}_n,\mathbf{z}_n)$. From this formulation the complete data log-likelihood is given by

$$L(\mathbf{x},\mathbf{z};\boldsymbol{\theta}) = \log f(\mathbf{x},\mathbf{z};\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{s} z_{ik} \log\left(p_k \varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k)\right). \tag{1}$$

2.2 Cluster analysis

In order to find an optimal partition in s clusters $\hat{\mathbf{z}} = (\hat{z}_1,\ldots,\hat{z}_s)$, we generally use two approaches: the maximum likelihood (ML) approach (Day 1969) and the classification maximum likelihood (CML) approach (Symons 1981). The ML approach includes estimating the parameters of the mixture and the partition is derived from these parameters using the MAP principle. An iterative solution is provided by the EM algorithm of Demspter et al (1977). The EM algorithm maximizes $L(\boldsymbol{\theta};\mathbf{x})$ iteratively maximizing the conditional expectation of the complete log-likelihood given a previous current estimate $\theta^{(q)}$ and $\mathbf{x}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q)}) = \sum_{i=1}^{n}\sum_{k=1}^{s} t_{ik}^{(q)}\{\log(p_k) + \log\varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k)\}, \tag{2}$$

where

$$t_{ik}^{(q)} = \frac{p_k^{(q)}\varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k^{(q)})}{\sum_{\ell=1}^{s} p_\ell^{(q)}\varphi_\ell(\mathbf{x}_i;\boldsymbol{\alpha}_\ell^{(q)})}$$

denotes the conditional probability, given $\mathbf{x}$ and $\boldsymbol{\theta}^{(q)}$, that $\mathbf{x}_i$ arises from the mixture component with density $\varphi_k(\mathbf{x}_i;\boldsymbol{\alpha}_k)$. Each iteration of EM uses the following steps.

- *E-step*: compute the conditional expectation of the complete log-likelihood. Note that in the mixture case this step reduces to the computation of the conditional density of the $t_{ik}^{(q)}$.
- *M-step*: compute $\boldsymbol{\theta}^{(q+1)}$ maximizing $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q)})$. This leads to $p_k^{(q+1)} = \frac{1}{n}\sum_{i=1}^{n} t_{ik}^{(q)}$ and the exact formulae for the $\boldsymbol{\alpha}_k^{(q+1)}$ will depend on the involved parametric family of distribution probabilities.

The second approach, sometimes called the classification approach, is based on the complete data. With this approach, $\boldsymbol{\theta}$ and the unknown component-indicator vectors $\mathbf{z}_1,\ldots,\mathbf{z}_n$ of the observed data $\mathbf{x}_1,\ldots,\mathbf{x}_n$ are chosen to maximize the complete

data log-likelihood $L(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$. This optimization can be done by the Classification EM (CEM) algorithm (Celeux and Govaert 1992), a variant of EM, which converts the probabilities t_{ik} to a discrete classification in a Classification step before performing the Maximization step.

- *C-step*: Each cluster $z_k^{(q+1)}$ is defined with $z_{ik}^{(q+1)} = \text{argmax}_k \, t_{ik}^{(q)}$.

Note that when the data are continuous, using the Gaussian mixture model, the standard k-means can be shown to be a simple version of the CEM algorithm. Simplicity, fast convergence and the possibility to process large data sets are the major advantages of the CEM algorithm. These virtues suggest to combine CEM and EM in order to obtain a solution quickly. Instead of running EM r times from random centers, it suffices to run CEM r times from random centers, and when it provides no empty cluster partition (the number of obtained clusters is equal to the number of required clusters), the EM algorithm is initialized with the parameter values derived from the best partition. Unfortunately, from an estimation point of view, CEM is not expected to converge to the ML estimate of the parameters and yields inconsistent estimates especially when the mixture components are overlapping or are in disparate proportions (see McLachlan and Peel 2000, Sect. 2.21 and Govaert and Nadif (1996)). Then, in these situations, as it will be observed in Sect. 4, the strategy of applying CEM followed by EM (CEM-EM) can give disappointing results in term of quality of the partition.

2.3 Latent class model

This section provides a focus on categorical data. The model commonly used is the latent class model, which was proposed by Lazarfeld and Henry (1968). With this model, the association between any pair of attributes should disappear once the latent attribute is held constant. This is the basic model of latent class analysis with its fundamental assumption of local independence. This hypothesis is commonly chosen when the data are categorical or binary (Celeux and Govaert 1992; Cheeseman and Stutz 1996). In the following, each variable j has c_j categories, and $x_i^j \in \{1, \ldots, c_j\}$. Then one has $\varphi_k(\mathbf{x}_i, \boldsymbol{\alpha}_k) = \prod_{j=1}^{d} \varphi_k(\mathbf{x}_i^j, \boldsymbol{\alpha}_k^j)$ where $\boldsymbol{\alpha}_k^j = (\alpha_k^{j1}, \ldots, \alpha_k^{jc_j})$. The category e can be associated with a binary variable noted je and defined by $x_i^{je} = 1$ if $x_i^j = e$ and 0 otherwise. Assuming the multinomial distribution for x_i^j, it follows that

$$\varphi_k(\mathbf{x}_i, \boldsymbol{\alpha}_k) = \prod_{j=1}^{d} \prod_{e=1}^{c_j} (\alpha_k^{je})^{x_i^{je}}, \quad \text{with} \sum_{e=1}^{c_j} \alpha_k^{je} = 1, \tag{3}$$

where $\alpha_k^{je} = Pr(x_i^j = e | \mathbf{z}_i = k)$ is the probability that object $\mathbf{x}_i$ belonging to component k has category e in categorical variable j.

The assumption of local independence, sometimes called the naive Bayes, allows one to estimate the parameters separately; this hypothesis greatly simplifies the computing, especially when the number of attributes is large. Although this assumption is clearly false in most real data, naive Bayes often performs clustering very well. This paradox is explained by Domingos and Pazzani (1997).

3 Acceleration

In this section, we are interested in speeding-up the convergence of EM. For that, many variants were proposed and some reduce the computational cost of the M-step. Here, with the latent class models, the M-step is easy to compute, and do not need to be accelerated. Then, we chose to study large database adapted versions of EM, which use a partial E-step instead of a complete one. As pointed by Neal and Hinton (1998), the kind of these methods is justified theoretically by noticing that the formulation of EM by the authors is applicable for an arbitrary and not necessarily an exhaustive set of data blocks, as long as the data are visited regularly. Two versions of this formulation, known as SpEM (Neal and Hinton 1998) and LEM (Thiesson et al. 2001), described hereafter, are very efficient for Gaussian models. We propose to apply them on categorical data and discuss their behavior.

3.1 Sparse and Lazy expectation-maximization

The SpEM algorithm minimizes the computational cost of the E-step by choosing which computations to perform, with a given threshold (th). The SpEM searches very small conditional probabilities (less than th), and does not recompute them until a given number of iteration it. The main idea is that an object, which has a small probability to be in a cluster, has a small chance of growing quickly in only one iteration. Then, these probabilities can be fixed during a certain period, and recomputed during a complete (standard) E-step. Therefore the update concerns only the set of objects and the components with $t_{ik}^{(q)} > \text{th}$, noted y_{sparse}. More precisely, for the ith object, if y_{sparse}^i (the complement of y_{sparse}^i) is not the null set, then update the conditional probabilities is given by

$$\left[\sum_{\ell \in y_{\text{sparse}}^i} t_{i\ell}^{(q-1)}\right] \frac{t_{ik}^{(q)}}{\sum_{\ell \in y_{\text{sparse}}^i} t_{i\ell}^{(q)}}$$

for the components belonging to y_{sparse}^i; otherwise do not update the conditional $t_{ik}^{(q-1)}$.

With the same purpose as SpEM, the LEM algorithm reduces the time spent in the E-step. For that, it attempts to periodically identify significant (important) objects, and focuses attention on them for a given number of iterations. The significance criterion is defined as follows: each object will be said significant if it has all its conditional probabilities t_{ik} less than a given threshold th. Then, if an object has a high conditional probability to be in a cluster, it is not appropriated to assess it in another cluster.

3.2 *e*Lazy expectation-maximization: a new variant of expectation-maximization

In LEM, the significance criterion of an object is reduced to a simple comparison between a conditional probability and a specified threshold. But, this criterion can be expressed from the change (or *evolution*) of its conditional probabilities. Then, if an object has similar probabilities between two following steps, it indicates that this object is not important, and we can fix its probabilities during a certain number of iterations. Thus, instead of focusing only on significant probabilities associated to the components, we consider here all these probabilities. For that, we compute the

differences between the conditional probabilities of iterations before and after the standard E-step and suggest the following criterion

$$1/s \sum_{k=1}^{s} \left| t_{ik}^{(q)} - t_{ik}^{(q-1)} \right| < \text{th}, \tag{4}$$

where th is a threshold to be defined. Thus an object is significant if the change in the average deviation is large. As the threshold must be necessarily small, in our experiments, the tests were performed with different values, and the choice of th less than 0.025 was empirically selected. Our new variant of EM is called *e*LEM.

Let y_{lazy} be the set of significant objects, and y_{lazy} the set of other objects. Each iteration uses a standard (complete) E-step or a *lazy* (partial) E-step, with a standard M-step. The complete E-step computes conditional probabilities for all objects, and sets the list of significant objects. A partial E-step updates only conditional probabilities for objects in y_{lazy}. The steps become

- *Standard E-step*: Compute conditional probabilities $t_{ik}^{(q)}$ and identify y_{lazy} as the set of non-significant objects to ignore in lazy E-step. The formulae is

$$t_{ik}^{(q)} = \frac{p_k^{(q-1)} \varphi_k \left(\mathbf{x}_i; \boldsymbol{\alpha}_k^{(q-1)} \right)}{\sum_{\ell=1}^{s} p_\ell^{(q-1)} \varphi_\ell \left(\mathbf{x}_i; \boldsymbol{\alpha}_\ell^{(q-1)} \right)}.$$

- *lazy E-step*: In this step, compute conditional probabilities $t_{ik}^{(q)}$ for all objects in the block y_{lazy}. For all other objects in y_{lazy}, we have then $t_{ik}^{(q)} = t_{ik}^{(q-1)}$, for $k = 1, \ldots, s$. Only the conditional expectation associated to block y_{lazy} noted as Q_{lazy} is updated. Then, the global quantity that we search to maximize in the M-step will be

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q)}) = Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q-1)}) - Q_{\overline{lazy}}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q-1)}) + Q_{\overline{lazy}}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q)}). \tag{5}$$

- *M-step*: Compute $\boldsymbol{\theta}^{(q+1)}$, which maximizes $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(q)})$. This leads to

$$p_k^{(q+1)} = \frac{\sum_{i=1}^{n} t_{ik}^{(q)}}{n}$$

concerning the updating of $\boldsymbol{\alpha}_k$, it can be seen from (2) that $\boldsymbol{\alpha}_k^{(q+1)}$ is obtained as an appropriate root of

$$\sum_{i=1}^{n} \sum_{k=1}^{s} t_{ik}^{(q)} \frac{\partial \log \varphi_k(\mathbf{x}_i, \boldsymbol{\alpha}_k)}{\partial \boldsymbol{\alpha}_k} = 0. \tag{6}$$

One nice feature of the EM algorithm is that the solution of (6) exists in closed form and each component of $\boldsymbol{\alpha_k}$ is given by

$$(\alpha_k^{je})^{(q+1)} = \frac{\sum_{i=1}^{n} t_{ik}^{(q)} \times x_i^{je}}{\sum_{i=1}^{n} t_{ik}^{(q)}}.$$

This algorithm starts with a standard iteration of EM (with a standard E-step), and computes *it* iterations with a *lazy* E-step before performing the M-step. This process is repeated until convergence. Note that, *e*LEM does not tend to sacrifice the simplicity EM usually enjoys.

4 Numerical experiments

4.1 Synthetic data

In order to efficiently compare the three different methods of speeding-up EM, we study their performances on simulated data sets. Using the latent class model with three components, $n = 5000$, $d = 10$, $\mathbf{p} = (0.5, 0.2, 0.3)$, $c^j = 3, \forall j = 1, \ldots, 10$, we generated data sets with different degrees of overlap. We have randomly generated a finite set of values for $\boldsymbol{\alpha}_1$ and obtained $\boldsymbol{\alpha}_2$ and $\boldsymbol{\alpha}_3$ by adding two constant values to have different level of separation of components. We varied the constant values and by using Monte Carlo simulations, we estimated the degree of overlap by the mean of percentages of misclassified objects computed by comparing the partitions simulated with those obtained by applying a C-step of the CEM algorithm starting from true parameters. Three situations of overlap were considered: well separated clusters (+), moderately separated clusters (++), and ill-separated clusters (+++), corresponding, respectively, to around 5%, 12%, and 20% of misclassified objects. For each situation with $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$, and $\boldsymbol{\alpha}_3$ fixed, we generated 30 data sets.

For each algorithm, we selected the most relevant parameter settings after several tests. For SpEM: it $= 1, 2, 3, 4$ and th $= 0.001, 0.005, 0.010, 0.050$, for LEM: it $=$ $1, 2, 3, 4$, and th $= 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 0.99$, and for *e*LEM: it $= 1, 2, 3, 4$, and th $= 0.001, 0.005, 0.010, 0.015, 0.020$. Starting with the same random parameters (20 times), we have run each method and selected the best solution. We report in Table 1 the means recorded by EM, SpEM, LEM and *e*LEM in total running time (on the 20 runs), speed-up $= \frac{\text{time}_{\text{EM}}}{\text{time}_{\text{method}}}$ and percentage of misclassified objects obtained over the 30 simulated data sets. Then, we observe that

- The *e*LEM is always the fastest method with a speed-up coefficient of three for situation (+), 4.12 for (++) and 4.76 for (+++). To confirm this performance, in Tables 2 and 3. are displayed the best and worst results obtained by SpEM, LEM and *e*LEM by varying it and th. It appears clearly that *e*LEM remains the best in all situations.

Table 1 Total running times (on the 20 runs), speed-up and percentage of misclassified objects: means and standard deviations (in parentheses) over the 30 simulated data sets (*All the experiments are run on a Pentium* 933 MHz *computer*)

Degree of mixture	Type	Time	Speed-up	Percentage of misclassified objects
+	EM	275.62 (±7.62)	1.00 (−)	4.54 (± 0.24)
	SpEM	212.31 (±23.66)	1.31 (±0.13)	4.54 (±0.24)
	LEM	241.13 (±49.49)	1.19 (±0.23)	4.54 (±0.24)
	*e*LEM	91.00 (±6.47)	3.04 (±0.21)	4.66 (±0.23)
++	EM	476.15 (±62.07)	1.00 (−)	13.21 (± 0.67)
	SpEM	403.82 (±46.48)	1.19 (±0.18)	13.21 (±0.67)
	LEM	467.89 (±109.99)	1.07 (±0.27)	13.21 (±0.67)
	*e*LEM	116.19 (±9.17)	4.12 (±0.62)	14.12 (±0.75)
+++	EM	612.97 (±43.37)	1.00 (−)	19.17 (± 0.19)
	SpEM	534.78 (±51.66)	1.16 (±0.13)	22.19 (±8.03)
	LEM	609.76 (±145.48)	1.06 (±0.23)	21.04 (±6.78)
	*e*LEM	129.39 (±9.74)	4.76 (±0.39)	20.69 (±3.53)

Table 2 Total running times (on the 20 runs), speed-up and percentage of misclassified objects : best run among the 30 simulated data sets

Degree of mixture	Type	(th,it) - best run -	Time	Speed-up	Percentage of misclassified objects
+	EM		262.95	–	4.38
	SpEM	(0.001,2)	179.81	1.53	4.46
	LEM	(0.990,2)	154.63	1.78	4.46
	*e*LEM	(0.010,2)	74.77	3.69	4.46
++	EM		417.13	–	12.66
	SpEM	(0.005,1)	333.96	1.43	12.66
	LEM	(0.990,1)	303.62	1.57	12.66
	*e*LEM	(0.001,3)	97.52	4.88	13.16
+++	EM		573.79	–	19.34
	SpEM	(0.005,1)	460.28	1.33	18.84
	LEM	(0.990,2)	419.07	1.46	18.84
	*e*LEM	(0.005,2)	109.35	5.61	20.32

Table 3 Total running times (on the 20 runs), speed-up and percentage of misclassified objects: worst run among the 30 simulated data sets

Degree of mixture	Type	(th,it) - worst run -	Time	Speed-up	Percentage of misclassified objects
+	EM		287.51	–	4.32
	SpEM	(0.050,4)	304.00	0.91	4.32
	LEM	(0.600,4)	406.58	0.68	4.86
	*e*LEM	(0.015,4)	118.99	2.32	4.64
++	EM		648.55	–	13.54
	SpEM	(0.001,2)	564.66	0.84	12.70
	LEM	(0.600,4)	788.41	0.60	12.32
	*e*LEM	(0.010,4)	142.70	3.34	14.90
+++	EM		779.56	–	34.06
	SpEM	(0.001,1)	823.42	0.74	35.28
	LEM	(0.600,4)	1275.25	0.48	50.62
	*e*LEM	(0.020,2)	170.30	3.60	34.58

- In terms of object classification, for (+) and (++), *e*LEM is less effective than EM and the other variants but the loss is marginal (0.12% for (+) and 0.91% for (++)). In the third situation (+++), even if *e*LEM is slightly less effective than EM (20.69% vs 19.17%), it is better than LEM and SpEM.

These results show clearly that *e*LEM outperforms the other variants but the crucial problem of the choice of parameters (number of iterations and threshold for all methods) seems open to debate. Nevertheless, as our main objective is the clustering, our intensive experiments allow us to propose some answers. Then we report in Figs. 1 and 2 the evolution of speed-up and of the percentage of misclassified objects, for each parametrization (th, it), for each situation (+, ++ and +++). From these numerical experiments, the main points arising are the following:

- The number of iterations it equal to 1 provides a low percentage of misclassified and an interesting speed-up.

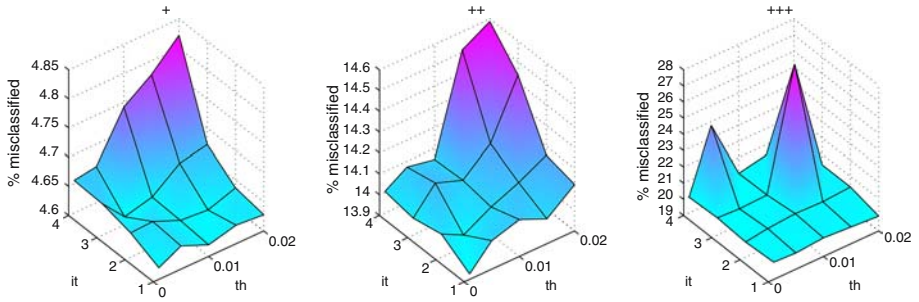

Fig. 1 Evolution of the percentage of misclassified objects for *e*LEM

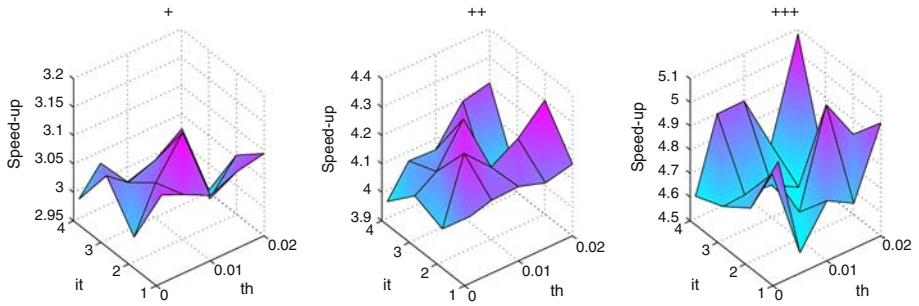

Fig. 2 Evolution of the speed-up for *e*LEM

Table 4 Total running times (on the 20 runs), speed-up and percentage of misclassified objects: means and standard deviations (in parentheses) over the 30 simulated data sets

Degree of mixture	Time	Speed-up	Percentage of misclassified objects
+	89.68 (± 6.46)	3.09 (± 0.22)	4.63 (± 0.21)
++	115.18 (± 7.71)	4.15 (± 0.28)	14.04 (± 0.74)
+++	132.78 (± 10.15)	4.64 (± 0.35)	21.10 (± 3.60)

- The threshold th in {0.001, 0.005, 0.010, 0.020} provides good results for each situation.

From that, we suggest then running *e*LEM 5 times for each threshold in order to have the same global number of runs (equal to 20) than the standard EM, SpEM, LEM, *e*LEM. We call this strategy *e*LEM(*) and the results on simulated data are reported in Table 4. In comparing these results and those of Table 1, it appears clearly *e*LEM(*) is a good way to avoid the problem of tuning parameters. Next, we evaluate all the EM variants on real data.

4.2 Real data

In this section, to illustrate the behaviors of SpEM, LEM, *e*LEM and the strategy previously called as *e*LEM(*), we apply them on three real data sets whose the clusters

are known. These data which are commonly used in the machine learning community are described hereafter. The algorithms SpEM, LEM and *e*LEM are run 20 times. The strategy *e*LEM is the same than before.

The ADN data set[1] consists of 3186 genes, described by 60 DNA sequence elements, called nucleotides or base pairs, with four possible modalities (A, C, G or T). These genes are distributed into three different clusters: 'intron → exon' or ie (sometimes called donors, 767 objects), 'exon → intron' or ei (sometimes called acceptors, 765 objects), and neither, noted as *n* (1,654 objects). From this data set, and according to the indications of the creators of the data, we choose to retain only the attributes 21–40, which represent nucleotides closest to the gene junction.

The Mushroom data set[2] consists of the description of 8,124 mushrooms, by 22 nominal attributes (color, shape, population, habitat, …). There are two clusters of mushrooms: edible (4,208) and poisonous (3,916).

The Congressional Votes data set consists of the votes for each of the U.S. House of Representative Congressmen[3], for 16 key votes, on different subjects (handicap, religion, immigration, army, education, …). For each vote, three answers are possible: yes, nay, and unknown. The individuals are separated into two clusters: democrats (267) and republicans (168).

The Titanic data set[4] gives the values of four categorical attributes for each of the 2,201 people on board the Titanic when it struck an iceberg and sank. The attributes are social class (first class, second class, third class, or crewmember), age (adult or child), sex, and whether or not the person survived.

In Table 5 we report the results of EM, CEM-EM (CEM followed by EM), and from the best parametrization of SpEM, LEM, *e*LEM, and *e*LEM(*). As mentioned before, the CEM-EM is an interesting methodology primarily but only when the clusters are well separated ADN. This performance decreases when the degree of overlap increases (Mushroom, Votes) and can be dramatically disappointing in term of quality of the partition when the clusters are ill-separated (Titanic).

On the contrary, the *e*LEM remains clearly the most performant. We note that in most situations the eLEM algorithm outperforms the EM, SpEM, and LEM algorithms, and the results on real and simulated data are approximatively the same, except for mushroom data for which the speed-up is very high. That is due to the sequence of likelihood values which was trapped at some saddle point (Fig. 3). It is a disadvantage of EM and therefore the obtained speed-up is more due to EM than to eLEM.

Furthermore, even if *e*LEM seems to be concerned only with the cost per iteration, the rate of convergence appears to be a consequence as, we illustrated in Fig. 3. Hence, the number of iterations which depends on the degree of overlap seems to be reduced with *e*LEM. It is possible that when, we focus only on the important objects, we deal with better the overlapping of the classes and therefore the convergence is accelerated. This remark is an empirical result whose internal mechanism deserves further investigation.

[1] ftp://genbank.bio.net

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html

[3] Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985

[4] http://www2.ncsu.edu/ncsu/pams/stat/info/jse/homepage.html

Table 5 Comparison between EM, CEM-EM, SpEM, *e*LEM, and *e*LEM(*) on real data sets: total running times (on the 20 runs), speed-up and percentage of misclassified objects

Data	Methods	(th, it) - best run -	Time (sec)	Speed-up	Percentage of misclassified objects
ADN	EM		485.68	1.00	4.8
	CEM-EM		89.50	5.43	4.8
	SpEM	(0.050,1)	292.20	1.66	4.8
	LEM	(0.99,1)	321.80	1.51	4.8
	*e*LEM	(0.005,1)	141.39	3.44	4.9
	*e*LEM	(*)	155.30	3.13	4.8
Mushroom	EM		19028.05	1.00	10.0
	CEM-EM		1080.89	17.60	12.6
	SpEM	(0.001,1)	475.65	40.00	10.6
	LEM	(0.99,3)	557.04	34.16	10.6
	*e*LEM	(0.005,1)	226.53	84.00	10.6
	*e*LEM	(*)	265.20	71.75	11.0
Votes	EM		25.27	1.00	13.1
	CEM-EM		9.79	2.58	13.1
	SpEM	(0.005,1)	16.28	1.55	13.1
	LEM	(0.60,1)	6.78	3.72	13.1
	*e*LEM	(0.001,3)	3.78	6.69	12.9
	*e*LEM	(*)	4.13	6.12	13.1
Titanic	EM		21.27	1.00	22.4
	CEM-EM		2.73	7.79	54.0
	SpEM	(0.050,3)	13.07	1.63	22.7
	LEM	(0.50,3)	1.65	12.89	22.7
	*e*LEM	(0.010,1)	9.65	2.20	22.7
	*e*LEM	(*)	10.55	2.02	22.7

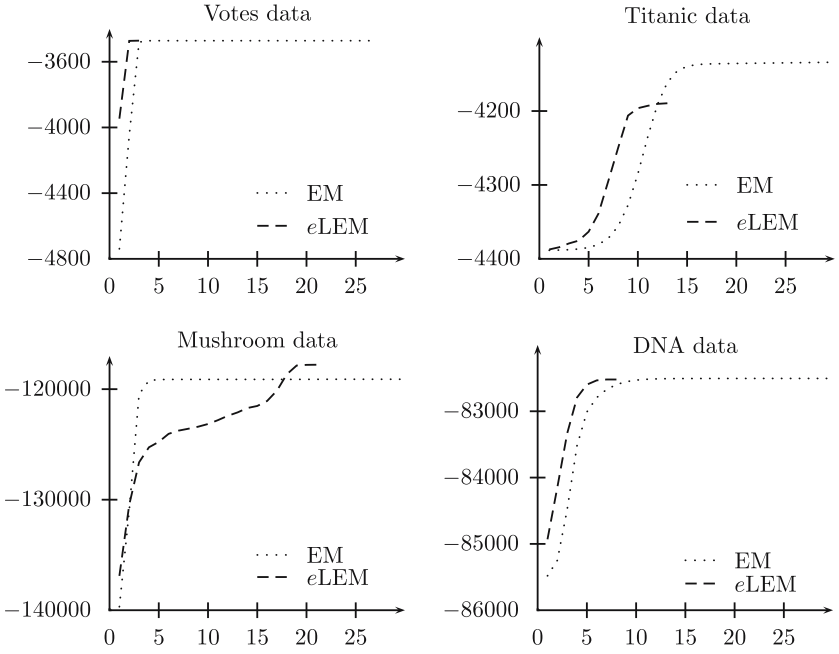

Fig. 3 Evolution of likelihood for the best run of EM and *e*LEM on the four real data sets

Finally, note that from these experiments it is evident that the use of the *e*LEM(*) strategy is beneficial to avoid the crucial choice of the parameters of the method.

5 Conclusion

The most documented problem occurring with EM is its possible low-speed in some situations. In the mixture context, some variants are proposed to accelerate this algorithm by reducing the time spent in the E-step. They consist of starting with a standard iteration of EM (with a standard E-step), and computing it iterations with a partial E-step before performing the standard M-step. This process is repeated until convergence.

From Gaussian mixtures, the SpEM and LEM methods have been studied. In this paper, we have extended these works to categorical data by using a latent class model, and we have presented a new variant of EM, called *e*LEM which preserves the simplicity of implementation of EM in its standard form. From several Monte–Carlo simulations, we have compared these methods and it appears clearly that our method *e*LEM outperforms the other ones. Unfortunately, as for SpEM and LEM, this performance depends primarily on the parameters of the method. Then in order to avoid the choice of these parameters, we have developed a simple and really efficient strategy, *e*LEM (*). This one consists in fixing the number of iterations with partial E-step and taking the threshold belonging to a set of known values.

Acknowledgements The authors would like to thank the anonymous referees for their useful comments to this work.

References

Celeux, G., Govaert, G.: A classification EM algorithm for clustering and two stochastic versions. Comput. Stat. Data Anal. **14**, 315–332 (1992)

Cheeseman, P., Stutz, J.: Bayesian classification (autoclass): theory and results. In: Fayyad, U., Pitesky-Shapiro, G., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 61–83 AAAI Press, CA (1996)

Day, N.E.: Estimating the components of a mixture of normal distributions. Biometrika **56**, 464–474 (1969)

Dempster, A., Laird, N., Rubin, D.: Maximum likelihood for incomplete data via the EM algorithm. J. Roy. Stat. Soc. **39(B)**, 1–38 (1977)

Domingos, P., Pazzani, M.: Beyond independence: Conditions for the optimality of the simple bayesian classifier. Mach. Learn. **29**, 103–130 (1997)

Govaert, G., Nadif, M.: Comparison of the mixture and the classification maximum likelihood in cluster analysis with binary data. Comput. Stat. Data Anal. **23**, 65–81 (1996)

Jamshidian, M., Jennrich, R.: Conjugate gradient acceleration of the EM algorithm. J. Am. Stat. Associ **88**(421), 221–228 (1993)

Lazarfeld, P., Henry, N.: Latent Structure Analysis. Houghton Mifflin, Boston (1968)

McCallum, A., Nigam, K., Ungar, L.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Ramakrishnan, R., Stolfo, S. (eds.) Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 169–178. ACM, NewYork (2000)

McLachlan, G., Peel, D.: Finite Mixture Models. Wiley, New-York (2000)

Meilijson, I.: A fast improvement to the EM algorithm on its own terms. J. Roy. Stat. Soc. Ser B **51**(1), 127–138 (1989)

Meng, X.-L., van Dyk, D.: The EM algorithm – an old folksong sung to a fast new tune (with discussion). J. Royal Stat. Soc. Ser B **59**, 511–567 (1997)

Moore, A.: Very fast EM-based mixture model clustering using multiresolution kd-trees. In: Kearns, M.S., Solla, S.A., Cohn, D.A. (eds.) Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference, pp. 543–549. MIT Press, Cambridge, MA (1999)
Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M. (ed.) Learning in Graphical Models, pp. 355–371. Kluwer Academic Publishers, Dordrecht (1998)
Symons, M.: Clustering criteria by multivariate normal mixtures. Biometrics **37**, 35–43 (1981)
Thiesson, B., Meek, C., Heckerman, D.: Accelerating EM for large databases. Mach. Learn. **45**, 279–299 (2001)