Accélération de EM pour données qualitatives : étude comparative de différentes versions

Mohamed Nadif, François-Xavier Jollois

LITA - IUT de METZ, Université de Metz,
Ile du Saulcy, 57045 METZ Cedex, France
{jollois,nadif}@iut.univ-metz.fr,

Résumé. L'algorithme EM est très populaire et très efficace pour l'estimation de paramètres d'un modèle de mélange. L'inconvénient majeur de cet algorithme est la lenteur de sa convergence. Son application sur des tableaux de grande taille pourrait ainsi prendre énormément de temps. Afin de remédier à ce problème, nous étudions ici le comportement de plusieurs variantes connus de EM, ainsi qu'une nouvelle méthode. Celles-ci permettent d'accélérer la convergence de l'algorithme, tout en obtenant des résultats similaires à celui-ci. Dans ce travail, nous nous concentrons sur l'aspect classification. Nous réalisons une étude comparative entre les différentes variantes sur des données simulées et réelles et proposons une stratégie d'utilisation de notre méthode qui s'avère très efficace.

1 Introduction

L'utilisation des modèles de mélange dans la classification est devenue une approche classique et très puissante (voir par exemple [Banfield et Raftery, 1993], et [Celeux et Govaert, 1995]). En traitant la classification sous cette approche, l'algorithme EM [Dempster *et al.*, 1977] composé de deux étapes : Estimation et Maximisation, est devenu quasiment incontournable. Celui-ci est très populaire pour l'estimation de paramètres. Ainsi, de nombreux logiciels sont basés sur cette approche, comme Mclust-EMclust [Fraley et Raftery, 1999], EMmix [McLachlan et Peel, 1998] ou MIXMOD [Biernacki *et al.*, 2001]. Ce succès tient à sa simplicité, à ses propriétés théoriques et à son bon comportement pratique. De plus, un intérêt grandissant se fait ressentir actuellement pour les données qualitatives. On peut citer le logiciel AutoClass [Cheeseman et Stutz, 1996], très utilisé dans la communauté Fouille des Données.

Malheureusement, son principal inconvénient réside dans sa lenteur due au nombre élevé d'itérations parfois nécessaire pour la convergence, ce qui rend son utilisation inappropriée pour les données de grande taille. Plusieurs versions ont été faites pour accélérer cet algorithme et beaucoup d'entres elles agissent sur l'étape maximisation. Ici, comme nous nous intéressons au modèle des classes latentes, l'étape de maximisation ne présente aucune difficulté pour le calcul des paramètres. Nous avons donc choisi d'étudier des versions particulièrement adaptée aux données de grande taille et qui utilisent une étape partielle d'estimation au lieu d'une étape Estimation complète. Cette version semble très efficace pour des mélanges Gaussiens, nous proposons ici de l'appliquer sur le modèle de mélange des classes latentes et de discuter son comportement.

2 Modèle de mélange et algorithme EM

Dans l'approche modèle de mélange, les individus $\mathbf{x}_i,\ldots,\mathbf{x}_n$ à classifier sont supposés provenir d'un mélange de s densités dans des proportions inconnus $p_1,\ldots,p_s$. Ainsi, chaque objet $\mathbf{x}_i$ est une réalisation d'une densité de probabilité (p.d.f.), décrite par :

$$\varphi(\mathbf{x}_i;\theta) = \sum_{k=1}^{s} p_k \varphi_k(\mathbf{x}_i;\alpha_k)$$

où $\varphi_k(\mathbf{x}_i;\alpha_k)$ représente la densité de $\mathbf{x}_i$ de paramètre α_k. Le vecteur des paramètres à estimer θ est composé de $\mathbf{p} = (p_1,\ldots,p_k)$ et $\alpha = (\alpha_1,\ldots,\alpha_k)$. De ceci, nous en déduisons la log-vraisemblance du vecteur $\mathbf{x} = (\mathbf{x}_i,\ldots,\mathbf{x}_n)$, donnée par :

$$L(\mathbf{x},\theta) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{s} p_k \varphi_k(\mathbf{x}_i;\alpha_k)\right). \quad (1)$$

Dans la suite, nous allons aborder le problème de la classification sous l'approche estimation : les paramètres sont d'abord estimés, puis la partition en est déduite par la méthode du maximum a posteriori (MAP). L'estimation des paramètres du modèle passe par la maximisation de $L(\mathbf{x},\theta)$. Une solution itérative pour la résolution de ce problème est l'algorithme EM [Dempster *et al.*, 1977]. Le principe de cet algorithme est de maximiser de manière itérative l'espérance de la log-vraisemblance complétée conditionnellement aux données $\mathbf{x}$ et la valeur du paramètre courant $\theta^{(q)}$:

$$Q(\theta|\theta^{(q)}) = \sum_{i=1}^{n}\sum_{k=1}^{s} t_{ik}^{(q)} \left(\log(p_k) + \log\varphi_k(\mathbf{x}_i;\alpha_k)\right)$$

où $t_{ik}^{(q)} \propto p_k^{(q)}\varphi_k(\mathbf{x}_i;\alpha_k^{(q)})$ est la probabilité conditionnelle a posteriori. Chaque itération de EM a deux étapes:

- **Estimation** : On calcule $Q(\theta|\theta^{(q)})$, notons que dans le contexte modèle de mélange, cette étape se réduit aux calculs des $t_{ik}^{(q)}$.
- **Maximisation** : On cherche la paramètre $\theta^{(q+1)}$ qui maximise $Q(\theta|\theta^{(q)})$.

3 Données qualitatives et modèle des classes latentes

Le principe de base du modèle des classes latentes, qui a été proposé par Lazarfeld et Henry [1968] est la supposition d'une variable qualitative latente à s modalités dans les données. Dans ce modèle, les associations entre chaque paire de variables disparaissent, si la variable latente est constante. C'est le modèle basique de l'analyse des classes latentes, avec l'hypothèse fondamentale d'indépendance locale. Cette hypothèse est courament choisie quand les données sont de type qualitatif ou binaire [Celeux et Govaert, 1992, Cheeseman et Stutz, 1996]. Ainsi, la densité d'une observation $\mathbf{x}_i$ peut se décrire comme suit :

$$\varphi_k(\mathbf{x}_i;\alpha_k) = \prod_{j=1}^{d}\prod_{e=1}^{c^j}(a_k^{je})^{x_i^{je}}, \text{ avec } \sum_{e=1}^{c^j} a_k^{je} = 1$$

où chaque attribut $j = 1,\ldots,d$ est de type qualitatif, a un nombre fini de modalités c^j et $x_i^{je} = 1$ si la modalité e de la variable j est observée, et 0 sinon. L'hypothèse d'indépendance locale permet d'estimer les paramètres séparement. Cette hypothèse simplifie grandement les calculs, principalement quand le nombre de variables est grand. Bien que cette affirmation est toujours fausse dans les données réelles, l'indépendance locale est généralement très performante pour la classification. Ce paradoxe est expliqué par [Domingos et Pazzani, 1997].

4 Accélération de EM

4.1 Incremental EM (IEM)

L'algorithme Incremental EM [Neal et Hinton, 1998], ou IEM, qui est une variante de EM, est destiné à réduire le temps de calcul en réalisant des étapes Estimation partielles. Soit $y = y_1,\ldots,y_B$ une partition des données en B blocs disjoints (B est proposé par l'utilisateur, ou bien est déduit par rapport à la taille des blocs demandés par l'utilisateur, en pourcentage de la taille initiale). L'algorithme IEM parcourt tous les blocs de façon cyclique. A chaque itération, on met à jour les probabilités a posteriori d'un bloc dans l'étape Estimation. Ci-dessous, on décrit plus en détail la $n$ième itération :

- **Estimation** : Dans cette étape, on retient un bloc y_b et on met à jour les probabilités a posteriori $t_{ik}^{(q)}$ pour toutes les observations appartenant aux bloc y_b, quant aux autres observations (appartenant aux autres blocs) nous avons $t_{ik}^{(q)} = t_{ik}^{(q-1)}$. L'espérance conditionnelle associée au bloc b notée Q_b est mise à jour, quant à celles associées aux autres blocs elles restent inchangées. Autrement dit la quantité globale qu'on cherchera à maximiser dans l'étape maximisation peut s'écrire :

$$Q(\theta|\theta^{(q)}) = Q(\theta|\theta^{(q-1)}) - Q_b(\theta|\theta^{(q-1)}) + Q_b(\theta|\theta^{(q)}) \quad (2)$$

- **Maximisation** : On cherche comme dans l'algorithme EM classique, le paramètre $\theta^{(q+1)}$ qui maximise $Q(\theta|\theta^{(q)})$.

De cette façon, chaque observation $\mathbf{x}_i$ est visitée après les B étapes d'estimation partielles. Une approximation de la log-vraisemblance ne décroît pas à chaque itération. La justification théorique de cet algorithme a été faite par Neal et Hinton [Neal et Hinton, 1998]. Récemment, nous avons étudié son comportement avec le modèle de mélange de Bernoulli [Jollois et Nadif, 2003]. Enfin, notons que lorsque $B = 1$, IEM se réduit à EM.

4.2 Sparse EM

L'algorithme Sparse EM, introduit par [Neal et Hinton, 1998], ou SpEM, minimise le coût de l'étape Estimation en choisissant les calculs à effectuer à partir d'un seuil, contrairement à IEM. SpEM cherche donc les probabilités a posteriori très petites (inférieur à un certain seuil), et ne les recalcule plus pendant un certain nombre d'itérations. L'idée est qu'un individu qui a une faible probabilité d'appartenance à une classe a peu de chance de la voir devenir grande en une seule itération. Et donc, ces probabilités sont bloqués pendant un certain temps, puis recalculer au cours d'une itération standard de EM. L'algorithme peut être défini comme suit.

- **Estimation**
  - Étape standard : Calculer les probabilités a posteriori. Identifier y^i_{sparse} comme l'ensemble des classes à ignorer durant les étapes *sparse*, pour chaque individu i.
  - Étape *sparse* : Dans cette étape, on calcule les probabilités a posteriori $t^{(q)}_{ik}$ pour toutes les classes appartenant aux blocs $y^i_{\overline{sparse}}$. Seule l'espérance conditionnelle associée au bloc $y_{\overline{sparse}}$ notée $Q_{\overline{sparse}}$ est mise à jour. Autrement dit la quantité globale qu'on cherchera à maximiser dans l'étape maximisation est celle de l'équation 2 (en remplacant Q_b par $Q_{\overline{sparse}}$).
- **Maximisation** : On cherche comme dans l'algorithme EM classique, le paramètre $\theta^{(q+1)}$ qui maximise $Q(\theta|\theta^{(q)})$.

L'algorithme débute sur une itération standard de EM, puis effectue un nombre n_{sparse} d'itération avec une étape Estimation *sparse*, pour revenir ensuite à une itération standard, et ainsi de suite jusqu'à la convergence.

4.3 Lazy EM (LEM)

Ayant le même objectif que les deux précédentes méthodes IEM et SpEM, l'algorithme Lazy EM [Thiesson *et al.*, 2001], ou LEM, cherche donc à réduire le temps de l'étape Estimation. Pour ceci, il cherche à identifier régulièrement les individus importants, et à porter son attention sur ceux-ci pour plusieurs itérations. Un individu est considéré comme important si le changement de la probabilité t_{ik} entre deux itérations successives est grande. Notons $y_{\overline{lazy}}$ cet ensemble de cas significatif, et y_{lazy} l'ensemble de cas restants. Suivant un déroulement similaire à SpEM, chaque itération requiert soit une étape Estimation standard, soit une étape Estimation *lazy*, suivi ensuite par une étape Maximisation standard. L'étape complète calcule pour tous les individus les probabilités a posteriori. De plus, elle établit la liste des individus importants. Une étape *lazy* ne met à jour qu'une partie des probabilités a posteriori.

- **Estimation**
  - Étape standard : Calculer les probabilités a posteriori. Identifier y_{lazy} comme l'ensemble d'individus à ignorer durant les étapes *lazy*.
  - Étape *lazy* : Dans cette étape, on calcule les probabilités a posteriori $t^{(q)}_{ik}$ pour toutes les observations appartenant au bloc $y_{\overline{lazy}}$, quant aux autres observations (appartenant à y_{lazy}), nous avons $t^{(q)}_{ik} = t^{(q-1)}_{ik}$. Seule l'espérance

conditionnelle associée au bloc $y_{\overline{lazy}}$ notée $Q_{\overline{lazy}}$ est mise à jour. Autrement dit la quantité globale qu'on cherchera à maximiser dans l'étape maximisation est celle de l'équation 2 (en remplacant Q_b par $Q_{\overline{lazy}}$).

– **Maximisation** : On cherche comme dans l'algorithme EM classique, le paramètre $\theta^{(q+1)}$ qui maximise $Q(\theta|\theta^{(q)})$.

Le déroulement de LEM est le même que SpeM, avec une itération standard suivie de n_{lazy} itérations dites *lazy*. Ce schéma est répété jusqu'à la convergence de l'algorithme.

La viabilité de l'algorithme LEM réside partiellement dans l'idée que toutes les données ne sont pas d'importance égale. Elle dépend aussi du coût de calcul pour déterminer l'importance de chaque individu et du coût de stockage pour garder cette information. Pour les modèles de mélange, ces deux coûts peuvent être grandement réduits, voire simplement supprimés pour le coût de stockage, grâce à un critère d'importance. L'idée derrière ce critère est la suivante : si un individu a une forte probabilité d'appartenir à une classe, il n'est pas approprié de l'assigner à une autre. Et s'il le fallait, cela ne serait pas soudainement mais plutôt progressivement. Ainsi, nous supposons que les observations qui ne sont pas fortement liées à une classe contribuent le plus à l'évolution des paramètres. Et donc, les individus sont considérés comme importants s'ils ont toutes leur probabilités d'appartenance t_{ik} inférieures à un certain seuil.

Due à la démonstration de [Neal et Hinton, 1998], la convergence de l'algorithme est justifiée théoriquement et est applicable pour chaque découpage arbitraire des individus, du moment qu'on visite régulièrement tous les cas.

4.4 Lazy EM basé sur les différences (LEM-D)

L'idée de base de [Thiesson *et al.*, 2001] est d'écarter un certain nombre d'individus que l'on peut considérer comme peu important dans les calculs. Cette notion d'importance peut être rapportée à l'évolution des probabilités a posteriori. En effet, si un individu ne montre pas d'évolution importante entre deux étapes, c'est qu'il est a priori stable et donc, il a de fortes chances de le rester un long moment. Dans ce cas, on prend la décision de l'écarter des calculs et de ne plus le prendre en compte pendant un certain nombre d'itérations. Au contraire, si son évolution est significative, il est intéressant de le garder dans les calculs.

A partir de là, un nouveau problème se pose à nous. Comment déterminer si un individu a évolué significativement ou non ? Pour ceci, nous avons choisi de mesurer les différences entre les probabilités a posteriori avant et après l'étape Estimation standard, où on remet à jour tous les t_{ik} de tous les individus. Plus précisement, nous comparons la moyenne des valeurs absolues de ces différences pour chaque classe avec un seuil α :

$$\frac{\sum_{k=1}^{s} |t_{ik}^{(q)} - t_{ik}^{(q-1)}|}{s} < \alpha$$

Nous avons testé plusieurs valeurs de ce seuil. Il est apparu que si celui-ci est grand (généralement au-dessus de 0.0250), l'algorithme met de côté tous les individus, et s'arrête donc automatiquement. Pour cette raison, dans nos expériences, les tests seront réalisés avec un seuil inférieur à 0.0250.

5 Résultats

5.1 Données simulées

Afin de comparer les 4 différentes méthodes d'accélération de EM, nous avons lancé tous ces algorithmes sur des données simulées. Pour ceci, nous avons choisi de créer 10 tableaux avec les mêmes paramètres : $n = 5000$, $d = 10$, $p = \{0.5,0.2,0.3\}$, $c^j = 3,\forall j = 1,\ldots,d$. Nous avons simulé des données moyennement mélangées (environ 11 % d'observations mal-classées). Dans le tableau suivant, nous récapitulons les paramètres choisis pour chacune des méthodes utilisées.

Méthode	Paramètres	
	Taille des blocs	
IEM	0.5, 1, 2.5, 5, 10, 25 et 50 %	
	Itération	Seuils
SpEM	1, 2, 3 et 4	0.001, 0.005, 0.010 et 0.050
LEM	1, 2, 3 et 4	0.50, 0.60, 0.70, 0.80, 0.90, 0.95 et 0.99
LEM-D	1, 2, 3 et 4	0.001, 0.005, 0.010, 0.015 et 0.020

Dans le tableau 1, nous décrivons les meilleurs résultats, ainsi que les moins bons pour chaque valeur résultat intéressante (L, temps, Coefficient d'accélération et % de mal-classés) pour chaque méthode, sur la moyenne des 10 tableaux. Dans les figures 1, 2, 3, et 4, nous présentons les accélérations moyennes et les pourcentages moyens de mal-classés pour chaque méthode (respectivement SpEM, LEM, LEM-D et IEM), en fonction des paramètres choisis.

Il est très clair que LEM-D se montre le plus rapide de tous. Au minimum, il va 3,54 fois plus vite que l'algorithme EM, et est toujours plus rapide que toutes les autres méthodes. De plus, LEM, SpEM et IEM vont parfois moins vite que EM.

En terme de pourcentage de mal-classés, et donc d'adéquation entre la partition simulée et celle obtenue, on remarque que notre méthode LEM-D s'avère encore une fois la plus stable et la plus proche de EM. Il n'y a au pire que 8 % de la population en plus qui est mal classés. Par contre, SpEM se montre moins performant, en ne trouvant jamais une partition proche de celles de EM.

Si nous regardons les résultats en fonction de paramètres choisis, pour chaque méthode, il en ressort les éléments suivants :

- SpEM va vite pour un petit seuil, quelque soit le nombre important d'itérations *sparse*. C'est d'ailleurs là qu'il est le plus performant pour la partition (surtout pour 3 ou 4 itérations).
- Pour LEM, les plus grosses accélérations se font avec un seuil important (proche de 1), et ce, pour tous les nombres d'itérations *lazy* choisis. Malheureusement, cette accélération s'effectue au détriment de la qualité de la partition obtenue. Il semble préférable de choisir un seuil plus faible, au risque de ne pas aller trop vite.
- LEM-D obtient les meilleures accélérations pour un seuil entre 0.010 et 0.020 (donc assez grand pour cette méthode), principalement pour une itération *lazy*. Et c'est avec ces paramètres qu'il obtient des partitions assez proches de celles simulées.

TAB. 1 – *Le meilleur (+) et le moins bon (-) résultat pour chaque méthode d'accélération de l'algorithme EM, sur la moyennes des 10 tableaux simulés.*

	Méthode	+		-	
		Résultat	Paramètres	Résultat	Paramètres
L	EM	-83720.91	-	-83720.91	-
	LEM	-83616.04	3, 0.90	-83721.24	4, 0.50
	LEM-D	-83688.15	1, 0.010	-83731.53	4, 0.020
	SpEM	-83610.54	1, 0.010	-83703.77	1, 0.001
	IEM	-83620.86	25.0 %	-83915.80	0.5 %
Coefficient	EM	1.00	-	1.00	-
d'accélération	LEM	1.46	3, 0.99	0.72	4, 0.60
	LEM-D	4.01	1, 0.015	3.54	4, 0.005
	SpEM	1.29	4, 0.001	0.98	4, 0.050
	IEM	1.98	50.0 %	0.28	0.5 %
Pourcentage	EM	11.74	-	11.74	-
de mal-classés	LEM	11.72	4, 0.50	28.49	3, 0.90
	LEM-D	11.72	1, 0.020	19.78	4, 0.005
	SpEM	14.54	4, 0.005	28.78	2, 0.001
	IEM	11.69	1.0 %	25.74	25.0 %

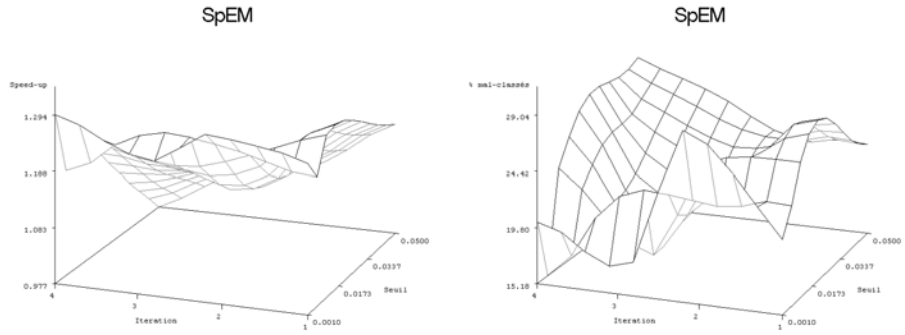

FIG. 1 – *Coefficient d'accélération moyen (à gauche) et pourcentage de données mal-classés moyen (à droite) pour l'algorithme SpEM sur les 10 tableaux simulés, en fonction du seuil et du nombre d'itération sparse*

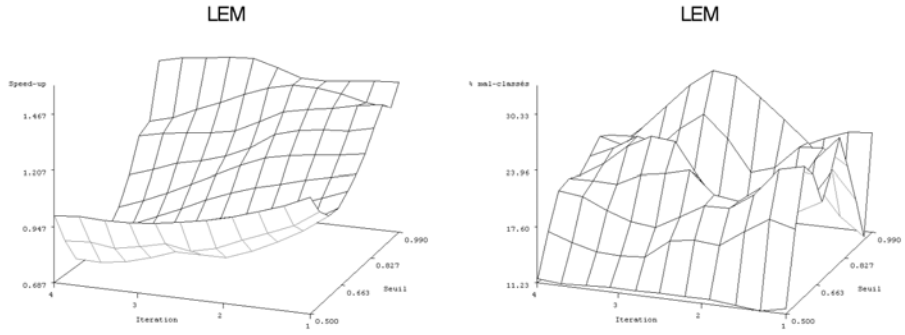

FIG. 2 – *Coefficient d'accélération moyen (à gauche) et pourcentage de données mal-classés moyen (à droite) pour l'algorithme LEM sur les 10 tableaux simulés en fonction du seuil et du nombre d'itération lazy*

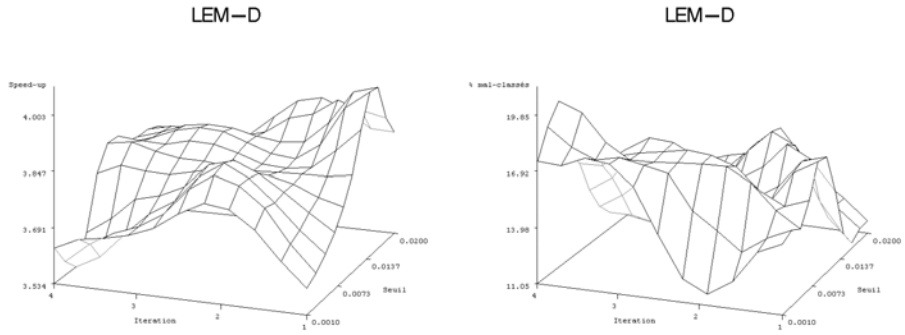

FIG. 3 – *Coefficient d'accélération moyen (à gauche) et pourcentage de données mal-classés moyen (à droite) pour l'algorithme LEM-D sur les 10 tableaux simulés en fonction du seuil et du nombre d'itération lazy*

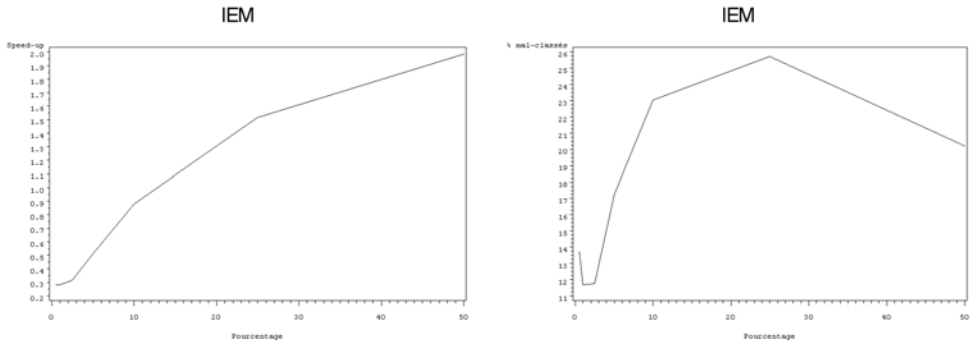

FIG. 4 – *Coefficient d'accélération moyen (à gauche) et pourcentage de données mal-classés moyen (à droite) pour l'algorithme IEM sur les 10 tableaux simulés en fonction de la taille des blocs choisie (avec un pourcentage de la population générale)*

– IEM est rapide lorsque la taille des blocs est égale à 50 % (donc, avec 2 blocs). Avec une taille de blocs inférieure à 10 %, il ralentit l'algorithme. Mais en ralentissant, il obtient de bien meilleurs résultats.

A la vue de ces premiers résultats, il nous semble donc que la méthode la plus appropriée et la plus performante, tant en terme d'accélération que de partition obtenue, est notre méthode, LEM-D. Nous allons maintenant voir sur des données réelles si celà se confirme.

5.2 Données réelles

Nous avons testé ces 4 méthodes d'accélération de l'algorithme EM sur 4 tableaux de données réelles (Congressional Votes, Titanic, ADN et Mushroom) que nous présentons par la suite. Sachant que pour les algorithmes de type EM, la solution dépend fortement de l'initialisation, nous lancons chaque algorithme 20 fois. Pour le paramètrage des différentes méthodes, nous avons utilisé tous ceux testés dans le paragraphe précédent. Nous avons choisi de retenir l'essai où la méthode a été la plus rapide, tout en étant performante en terme de partition. Puis, nous avons aussi testé un mode d'utilisation de notre méthode LEM-D, qui consiste à prendre comme paramètre 1 itération *lazy* et 4 seuils : 0.001, 0.005 et 0.010. Ici, puisque nous utilisons 3 seuils, nous lancons 7 fois l'algorithme pour chacun des seuils testés. Nous ne retenons que la solution fournissant L maximum.

Congressional Votes Ce tableau comprend les votes pour chacun des 435 représentants du congrès américain [1], pour 16 votes clés, sur différents sujets (handicap, religion, immigration, armée, éducation, ...). Pour chaque vote, trois réponses ont été prises en compte : pour, contre, et abstention. Les individus sont séparées en deux classes distinctes :

– Démocrates (267)
– Républicains (168)

Titanic Ce tableau décrit l'âge (adulte ou enfant), le sexe et la classe (première, deuxième, troisième ou équipage) des 2201 personnes présentes sur le Titanic lors de son naufrage en pleine mer, et s'ils ont été naufragés (1490) ou rescapés (711) de cet accident [2].

ADN Ce tableau de données a été obtenu à partir d'exemples tirés de Genbank 64.1 [3], et a été plusieurs fois utilisé dans des articles d'apprentissage automatique. Il contient 3186 observations, chacune décrites par 60 variables, représentant des nucléotides, toutes avec 4 modalités possibles (A, C, G ou T). Ces observations sont réparties en 3 classes :

– 'intron → exon' ou *ie* (parfois nommé donneurs, 767 objets),

1. Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL : Congressional Quarterly Inc. Washington, D.C., 1985
2. http://www2.ncsu.edu/ncsu/pams/stat/info/jse/homepage.html
3. ftp://genbank.bio.net

– 'exon → intron' ou *ei* (parfois nommé receveurs, 765 objets),
– ni l'un, ni l'autre, noté n (1654 objets).

A partir de ce tableau, et selon les indications apportées pas les créateurs des données, nous avons choisi de ne retenir que les variables 21 à 40, qui représentent les nucléotides les plus proches de la jonction du gène.

Mushroom Ce tableau de données a été obtenu sur le site de l'UCI Machine Learning Repository [4]. Il contient les descriptions de 8124 champignons, grâce à 22 variables nominales (couleur, forme, taille, habitat, ...). Ils sont répartis en deux classes :

– Comestible (4208 champignons),
– Vénéneux ou inconnu et donc considéré comme potentiellement dangereux (3916).

Données	Méthode	Paramètres	L	Temps (en s)	Coefficien d'accélération	% de mal classés
Votes	EM		-4464.82	25.27	1.00	13.1
	IEM	25 %	-4464.82	16.27	1.55	13.1
	SpEM	1,0.005	-4464.82	16.28	1.55	13.1
	LEM	1,0.60	-4464.82	6.78	3.72	13.1
	LEM-D	3,0.001	-4464.83	3.78	6.69	12.9
	LEM-D	(*)	-4464.82	4.13	6.12	13.1
Titanic	EM		-4132.01	21.27	1.00	22.4
	IEM	25 %	-4132.48	22.67	0.94	22.4
	SpEM	3,0.050	-4131.80	13.07	1.63	22.7
	LEM	3,0.50	-4383.57	1.65	12.89	22.7
	LEM-D	1,0.010	-4135.71	9.65	2.20	22.7
	LEM-D	(*)	-4135.83	10.55	2.02	22.7
ADN	EM		-82507.50	485.68	1.00	4.8
	IEM	50 %	-82507.52	230.85	2.10	4.8
	SpEM	1,0.050	-82507.47	292.20	1.66	4.8
	LEM	1,0.99	-82507.52	321.80	1.51	4.8
	LEM-D	1,0.005	-82513.28	141.39	3.44	4.9
	LEM-D	(*)	-82512.43	155.30	3.13	4.8
Mushroom	EM		-150988.36	19028.05	1.00	10.0
	IEM	50 %	-150995.17	529.04	36.31	10.1
	SpEM	1,0.001	-150986.33	475.65	40.00	10.6
	LEM	3,0.99	-151138.49	557.04	34.16	10.6
	LEM-D	1,0.005	-151189.60	226.53	84.00	10.6
	LEM-D	(*)	-151089.06	265.20	71.75	11.0

TAB. 2 – *Comparaison de différentes méthodes d'accélération et d'initialisation de EM, sur des données réelles et un tableau simulé. (*) signifie que l'on a utilisé 3 seuils (0.001, 0.005 et 0.010) pour LEM, avec une itération lazy.*

Les résultats sont présentés dans le tableau 2. Mis à part pour Titanic, notre méthode LEM-D est la plus rapide, tout en fournissant de très bons résultats. No-

4. http://www.ics.uci.edu/~mlearn/MLRepository.html

tons toutefois que pour le tableau Titanic, la vraisemblance obtenue par LEM-D est très proche de celle obtenue avec EM, contrairement à LEM. Les bons résultats de notre de méthode s'obtiennent avec des seuils différents, selon le tableau, ainsi qu'un nombre d'itérations *lazy* variant. Par contre, en utilisant notre stratégie d'utilisation de LEM-D, (*), on remarque que l'accélération proposée est plutôt intéressante et même très proche de celle obtenue avec un seul seuil, et les résultats sont toujours bons. Elle semble donc très intéressante à utiliser, afin d'éviter à l'utilisateur de choisir un seuil et un nombre d'itération *lazy*.

6 Conclusion

Nous avons présenté ici plusieurs variantes de l'algorithme EM, permettant d'accélérer sa convergence. Se basant sur l'une d'entre elles, nous avons proposé une nouvelle méthode, LEM-D, reposant sur l'évolution des probabilités a posteriori de chaque individu entre deux étapes pour limiter le nombre de calculs.

Malheureusement, chaque algorithme présenté ici nécessite soit le choix d'un nombre de blocs (IEM), soit le choix d'un seuil et d'un nombre d'itérations (LEM, LEM-D et SpEM). Ce choix semble difficile à effectuer a priori. C'est pourquoi nous avons proposé une stratégie intéressante d'utilisation de LEM-D. Celle-ci utilise plusieurs seuils, avec un nombre d'itérations toujours égal à 1. Nous l'avons validée sur des données réelles, où elle s'avère très performante.

Références

[Banfield et Raftery, 1993] J. D. Banfield et A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[Biernacki *et al.*, 2001] C. Biernacki, G. Celeux, G. Govaert, F. Langrognet, et Y. Vernaz. Mixmod : High performance model-based cluster and discriminant analysis. http://www-math.univ-fcomte.fr/MIXMOD/index.php, 2001.

[Celeux et Govaert, 1992] G. Celeux et G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14:315–332, 1992.

[Celeux et Govaert, 1995] G. Celeux et G. Govaert. Gaussian parcimonious clustering methods. *Pattern Recognition*, 28:781–793, 1995.

[Cheeseman et Stutz, 1996] P. Cheeseman et J. Stutz. Bayesian classification (autoclass): Theory and results. In U. Fayyad, G. Piatetsky-Shapiro, et R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 61–83. AAAI Press, 1996.

[Dempster *et al.*, 1977] A. Dempster, N. Laird, et D. Rubin. Mixture densities, maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statitical Society*, 39(1):1–38, 1977.

[Domingos et Pazzani, 1997] P. Domingos et M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, 29:103–130, 1997.

[Fraley et Raftery, 1999] C. Fraley et A. E. Raftery. Mclust: Software for model-based cluster and discriminant analysis. Technical Report 342, University of Washington, 1999.

[Jollois et Nadif, 2003] F.-X. Jollois et M. Nadif. L'algorithme iem pour données binaires. In *SFC 2003, Méthodes et Perspectives en Classification*, pages 141–144, Neuchâtel, Suisse, 10-12 septembre 2003.

[Lazarfeld et Henry, 1968] P.F. Lazarfeld et N.W. Henry. *Latent Structure Analysis*. Houghton Mifflin, Boston, 1968.

[McLachlan et Peel, 1998] G. J. McLachlan et D. Peel. User's guide to emmix-version 1.0. Technical report, University of Queensland, 1998.

[Neal et Hinton, 1998] R. Neal et G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–371, 1998.

[Thiesson *et al.*, 2001] B. Thiesson, C. Meek, et D. Heckerman. Accelerating em for large databases. Technical Report MSR-TR-99-31, Microsoft Research, 2001.

Summary

The EM algorithm is a popular and efficient method for the estimation of mixture model parameters. The major inconvenient is its slow convergence. Its application on large databases can spend a lot of time, and then is unsuitable. In order to solve this problem, we study here some known variants, and we propose a novel one. It allows one to speed up the convergence of the algorithm EM, and to obtain similar results ones. In this work, we focus on clustering aspect, and we perform a comparative study between all variants on simulated and real data. Finally, we propose a interesting strategy of our method, called LEM-D, which seems to be very efficient.