C A C H A N

Méthodes MCMC et applications
Online EM Algorithm for Hidden Markov Model*
Article d'Olivier Cappé

De Myttenaere Arnaud et Dadi Charles

6 janvier 2013

Introduction

L'algorithme EM a pour but initial de permettre d'estimer le maximum de vraisemblance dans un modèle dépendant de données latentes non observables. Ses applications sont très nombreuses dans le cadre notamment de l'imagerie médicale ou du traitemet de la parole. La version classique de l'algorithme peut être qualifiée "d'hors ligne" dans la mesure où elle est exécutée à partir d'un échantillon d'observations statiques (dans le sens où cet échantillon n'évolue pas au cours de la procédure). Toutefois, l'apparition de problématiques liées au stockage des données de grande dimension, ou encore de l'adaptation de l'algorithme lorsque l'acquisition des données se fait de façon continue durant la procédure, conduit à modifier l'algorithme EM classique afin de prendre en compte les nouvelles observations au fur et à mesure de leur arrivée.

Dans l'article présenté ici [4], il s'agit de traiter des HMM (Hiden Markov Model), qui constituent une forme de modèle à données cachées, à savoir une chaine de Markov de paramètres inconnus. On se restreint dans cet article à des HMM à espace d'états fini, avec une vraisemblance complète qui suit une distribution exponentielle.

*http://arxiv.org/pdf/0908.2359.pdf

Table des matières

1 De l'algorithme EM "hors" ligne à une version online

1.1 Rappel sur l'algorithme EM Classique :

L'algorithme Maximisation-Expectation permet de maximiser une log-vraisemblance lorsque les observations dépendent d'une variable cachée. Ce modèle répond aux problématiques de données manquantes, d'observations bruitées.. L'algorithme se compose de deux étapes :

- l'étape (E) consiste à calculer l'espérance de la vraisemblance des données complètes.
- l'étape (M) permet ensuite de maximiser cette quantité afin d'actualiser les paramètres

L'idée générale pour justifier la convergence est la croissance de la log-vraisemblance à chaque itération. Cet algorithme se décline en de nombreuses versions adaptées à des modèles particulier tel que l'algorithme Forward-Backward pour estimer les paramètres d'une chaîne de Markov cachée (HMM - Hidden Markov Chain).

1.2 Introduction de l'algorithme EM Online

Dans le cadre de l'algorithme EM en ligne, les données sont considérées au fur et à mesure qu'elles sont disponibles. L'algorithme EM ne peut donc pas être lancé une fois toutes les données stockées. L'idée de base est de remplacer dans l'étape (E) le calcul de l'espérance de la vraisemblance complète par une étape d'approximation stochastique. La méthode, développée dans [4], propose une étape E différente. L'idée principale repose sur l'utilisation de la formule des probabilités totale pour approximer l'espérance, se ramenant ainsi au calcul de plusieurs espérances conditionnelles, en conditionnant par rapport au données observées. Comme dans de nombreux algorithme de MCMC, l'itération nécessite maintenant l'introduction d'une suite $(\gamma_n)_{n\leq 0}$, telle que $\sum \gamma_n = +\infty$ et $\sum \gamma_n^2 < +\infty$. Après l'étape d'initialisation, les itérations s'effectuent le calcul au préalable des quantités suivantes :

$$\hat{\phi}_{n+1}(x) = \frac{\sum_{x'\in\mathcal{X}} \hat{\phi}(x') q_{\hat{\theta}_n}(x', x) g_{\hat{\theta}_n}(x, Y_{n+1})}{\sum_{x',x''\in\mathcal{X}^2} \hat{\phi}(x') q_{\hat{\theta}_n}(x', x'') g_{\hat{\theta}_n}(x'', Y_{n+1})}$$

$$\hat{\rho}_{n+1}(x) = \sum_{x'\in\mathcal{X}} \left(\gamma_{n+1} s(x', x, Y_{n+1}) + (1-\gamma_{n+1})\hat{\rho}_n(x')\right) \frac{\hat{\phi}_n(x') q_{\hat{\theta}_n}(x', x)}{\sum_{x''\in\mathcal{X}} \hat{\phi}_n(x'') q_{\hat{\theta}_n}(x'', x)}$$

L'espérance est alors estimée par :

$$\frac{1}{n}\widehat{\mathbb{E}}\big[\sum_{t=1}^{n} s(X_{t-1}, X_t, Y_t)\big|Y0:n\big] = \sum_{xin\chi} \hat{\phi}_{n+1}(x)\hat{\rho}_{n+1}(x)$$

Enfin, comme les données arrivent au fur et à mesure, il faut laisser passer un certain "temps de chauffe" avant d'actualiser l'estimation de θ pour la première fois. Dans la pratique, on fixe donc un valeur n_{min}, et pour $n > n_{min}$ pour effectue l'étape M : $\hat{\theta}_{n+1} = \bar{\theta}\left(\sum_{x\in\mathcal{X}} \hat{\phi}_{n+1}(x)\hat{\rho}_{n+1}(x)\right)$

1.3 Extension de l'algorithme EM Online à des données non indépendantes, HMM

Dans le cadre de l'algorithme EM classique pour un modèle HMM, on suppose que (X_t, Y_t) sont générés suivant une chaine de markov cachée de paramètre θ inconnu, avec l'hypothèse que X_t prend ses valeurs dans un espace fini $\mathcal{X}$ et que seul les Y_i sont observés.

Une hypothèse importante de l'algorithme EM est le fait que le modèle est régit suivant une famille exponentielle, c'est-à-dire que la densité jointe des états et des observations s'écrit sous la forme :

$$p_\theta(x_t, y_t|x_{t-1}) = h(x_t, y_t) \cdot exp\big(\langle \psi(\theta), s(x_{t-1}, x_t, y_t)\rangle - A(\theta)\big)$$

L'étape M de l'algorithme EM consiste alors à déterminer le maximum de la vraisemblance pour un certain θ (évoluant à chaque étape et convergeant vers l'identification de la chaine de Markov cachée), en résolvant pour $S \in \mathcal{S}$:

$$\nabla_\theta \psi(\theta) S - \nabla_\theta A(\theta) = 0$$

La solution est notée $\bar{\theta}(S)$.

En pratique, la k^{ieme} itération de l'algorithme EM, appliquée sur l'ensemble des observations, noté $Y_{0:n} = (Y_0, \ldots, Y_n)$, est :

E - Step :

$$S_{k+1} = \frac{1}{n}\mathbb{E}_{\nu,\theta_k}\left[\sum_{t=1}^{n} s(X_{t-1}, X_t, Y_t)|Y_{0:n}\right]$$

M - Step : Mise à jour de l'estimation du paramètre θ : $\theta_{k+1} = \bar{\theta}(S_{k+1})$

L'étape E nécessite le calcul d'une espérance, ce qui n'est pas toujours possible en pratique. Il est donc envisageable d'utiliser des méthodes de MCMC, comme nous le verrons dans la partie. En 1988, Zeitouni et Dembo [9] proposent une méthode de calcul de cette espérance (en utilisant la formule des probabilités totales) à partir des quantités ϕ et ρ définies par :

$$\phi_{n,\nu,\theta}(x) = \mathbb{P}_{\nu,\theta}(X_n = x|Y_{0:n})$$

$$\rho_{n,\nu,\theta}(x) = \frac{1}{n}\mathbb{E}\left[\sum_{t=1}^{n} s(X_{t-1}, X_t, Y_t)|Y_{0:n}, X_n = x\right]$$

L'étape E peut alors être remplacée par le calcul de $\sum_{x\in\mathcal{X}} \phi_{n,\nu,\theta}(x)\rho_{n,\nu,\theta}(x)$. Cette méthode est aussi décrite dans [4]

1.4 Condition de convergence

Dans le cadre des HMM, la convergence est liée au théorème suivant, présenté dans [4] :
Si :

i) la vraisemblance des données complète est de la forme exponentielle, et les paramètres sont des fonctions continument différentiable à l'intérieur de Θ,

ii) $\mathcal{X}$ est un ensemble de cardinal fini,

iii) Θ est un ensemble compact et $\theta^\star$ est dans l'intérieur de Θ,

iv) la matrice de transition vérifie $q_\theta(x, x') \geq \varepsilon > 0$ pour tout $\theta \in \Theta$,

v) $\sup_\theta \sup_y \bar{g}_\theta(y) < \infty$

vi) $\mathbb{E}_{\theta^\star}[|\log \inf_\theta \bar{g}_\theta(Y_0)] < \infty$, où $\bar{g}_\theta(y) = \sum_x g_\theta(x, y)$

Alors :

i)

$$\frac{1}{n}\mathbb{E}_{\nu,\theta}\left[\sum_{t=1}^{n} s(X_{t-1}, X_t, Y_t)|Y_{0:n}\right] \underset{n\to\infty}{\to} \mathbb{E}_{\theta^\star}\left[\mathbb{E}_{\theta}[s(X_{t-1}, X_t, Y_t)|Y_{-\infty;+\infty}]\right]$$

ii) Les points fixes de l'algorithme sont les points stationnaires de la vraisemblance.

La vérification des l'ensemble des hypothèses dans le cadre du modèle HMM présenté dans l'article fera l'objet de la partie 3 de ce rapport.

1.5 Comparaison des performances

Les expériences développées dans [4] montrent que sur les données utilisées l'algorithme EM en ligne converge, et que le choix de la suite γ_n influe sur la vitesse de convergence. Par ailleurs, la comparaison des résultats montre qu'à nombre d'observations identique l'algorithme classique propose des intervalles de confiance plus précis que l'algorithme en ligne, et est donc plus satisfant. Ce résultat se comprend bien dans la mesure où pour chaque itération l'algorithme EM classique utilise l'ensemble des observations, tandis que l'algorithme en ligne n'utilise que la nouvelle observation. Toutefois, les performances de l'algorithme EM online apparaissent très satisfaisantes.

Dans la partie suivante, nous reproduirons les résultats présentés dans [4] pour les HMM, et nous présenterons les résultats de l'estimation des paramètres dans le cas d'un mélange de gaussiennes pour les algorithmes EM classique et en ligne.

2 Implémentation et comparaison des algorithmes

2.1 EM classique pour un mélange de gaussiennes

Pour pouvoir souligner les avantages et les limites potentielles de l'algorithme EM Online, nous avons choisi de comparer les résultats obtenus avec un algorithme EM "hors" ligne. Dans un premier temps nous comparerons les deux algorithmes sur des données jouets échantillonnées selon un modèle de mélange de deux gaussiennes de même variance. Le modèle du mélange de deux gaussiennes répond à l'hypothèse d'indépendance des données (donc $L(y_{1:n}, x_{1:n}) = \prod_{t=1}^{n} L(y_t, x_t)$) et la vraisemblance complète se factorise sous la forme d'un modèle exponentiel de la façon suivante :

$$\begin{aligned}
L(y_t, x_t) &= \mathbb{P}(y_t, x_t) = \mathbb{P}(y_t|x_t)\mathbb{P}(x_t) \\
&= \prod_{i=1}^{2} [\mathbb{P}(y_t|x_t = i)\mathbb{P}(x_t = i)]^{\mathbb{I}_{(x_t=i)}} \\
&= \prod_{i=1}^{2} \frac{1}{\sqrt{2\pi\sigma}} \exp[(-\frac{1}{2\sigma}(y_t - \mu_i)^2 + \log(pi))\mathbb{I}_{(x_t=i)}] \\
&= \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\log(\sigma) - \frac{1}{2\sigma}\sum_{i=1}^{2} y_t^2 \mathbb{I}_{x_t=i} + \frac{1}{\sigma}\sum_{i=1}^{2} y_t\mu_i - \frac{2}{\sigma}\mathbb{I}_{(x_t=i)} \sum_{i=1}^{2} \mu_i \mathbb{I}_{(x_t=i)} + \sum_{i=1}^{2} \log(p_i)\mathbb{I}_{(x_t=i)} \right]
\end{aligned}$$

D'où,

$$s(y_{1:n}, x_{1:n}) = \left(\sum_{t=1}^{n} y_t^2 \mathbb{1}_{(x_t=1)}, \sum_{t=1}^{n} y_t^2 \mathbb{1}_{(x_t=2)}, \sum_{t=1}^{n} y_t \mathbb{1}_{(x_t=1)}, \sum_{t=1}^{n} y_t \mathbb{1}_{(x_t=2)}, \sum_{t=1}^{n} \mathbb{1}_{(x_t=1)}, \sum_{t=1}^{n} \mathbb{1}_{(x_t=2)} \right)$$

Ce modèle et l'algorithme ont été implémentés sous R (le code est mis en annexe). Les figures 1, 2 et 3 représentent respectivement les données et la convergence des variances et enfin la convergence des moyennes dans le cadre de l'algorithme EM "hors ligne" sur gaussiennes. Pour chaque graphique présenté tout au long de ce rapport, nous avons effectué 100 simulations, et nous avons représenté en rouge la valeur moyenne, et en vert premier et troisième quartiles, qui permettent de proposer des intervalles de confiance.

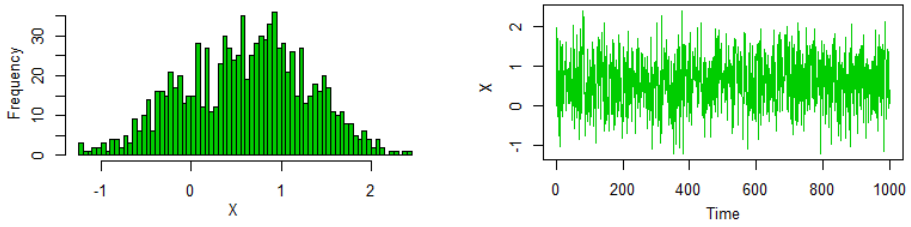
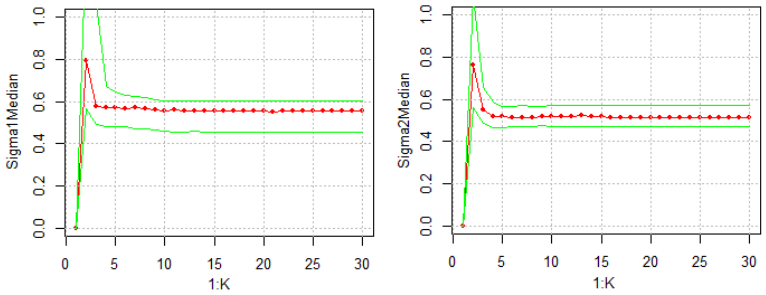
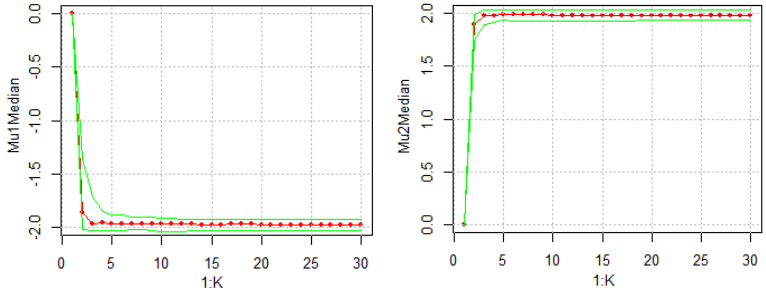


FIGURE 1 – Histogramme et trajectoire des observations générées selon un mélange de gaussiennes



(a) Sigma1 (b) Sigma2

FIGURE 2 – Estimations de la variance des gaussiennes



(a) Mu1 (b) Mu2

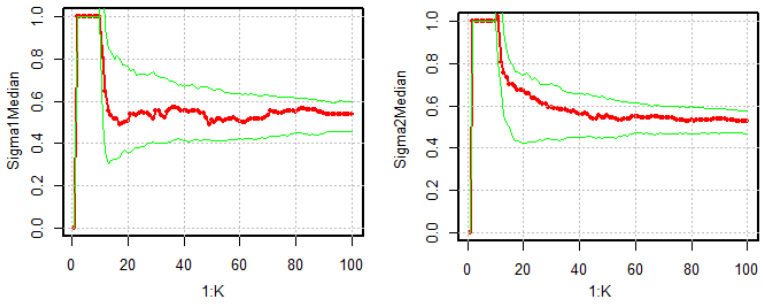
FIGURE 3 – Estimations de la moyenne des gaussiennes

On remarque une convergence très rapide (5 itérations) de l'algorithme EM classique. Pour confronter ces résultats à ceux de l'algorithme en ligne, intéressons-nous maintenant à l'implémentation online.

2.2 EM online pour un mélange de gaussiennes

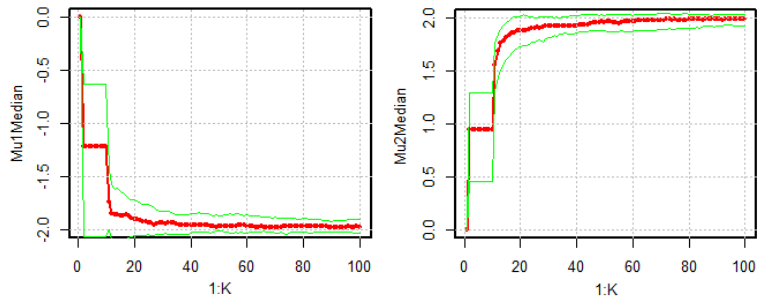
Les figures 4 et 5 représentent la convergence de l'algorithme EM online pour le même mélange de gaussiennes que précédemment. Nous avons considéré une période de chauffe de 10 itérations, durant lesquelles les paramètres ne sont pas ajustés.

La convergence s'établit ici beaucoup plus lentement que dans le cas classique, puisque elle nécessite 30 à 40 itérations pour le cas en ligne. La période constant pour les premières itérations correspond à la période de temps de chauffe. Mais cette différence de vitesse de convergence s'explique par le fait que le nombre de données utilisées à chaque itérations est considérablement moindre dans le cas en ligne. En effet, l'estimation des paramètres ($\hat{\mu}$ et $\hat{\sigma}$) repose sur 20 observations à la 20^{eme} itération en ligne, alors qu'elle repose sur 100 observations à chaque itération de l'algorithme EM classique.



(a) Sigma1 (b) Sigma2

FIGURE 4 – Estimations de la variance des gaussiennes



(a) Mu1 (b) Mu2

FIGURE 5 – Estimations de la moyenne des gaussiennes

Nous venons de voir que dans le cas des mélanges de gaussiennes les deux types d'algorithme EM convergent très rapidement (de 5 à 40 itérations). Cette vitesse de convergence, associée à la vitesse d'exécution des algorithmes, nous ont permi de réaliser plusieurs simulations pour obtenir des intervalles de confiance, et ainsi améliorer la précision de nos résultats. Etudions maintenant la convergence des algorithmes EM dans le cas d'un modèle à chaine de Markov cachée.

2.3 EM classique pour un HMM

Le modèle EM et les algorithmes EM associés sont décrits dans [4]. Comme dans le modèle de mélange, le modèle HMM fait intervenir une structure sous jacente sous la forme d'une variable X_t. Une subtilité de l'article [4] est que la variable X_t est implicitement introduite, et qu'il s'agit d'un étiquette pointant vers une moyenne μ_i, ce qui permet en particulier d'alléger les notations.

On note ν la densité de X_0 et $\phi_k|n$ la densité de $X_k|Y_{0:n}$. On a alors

$$f_n(x, y, \theta) = \nu(x_0, \theta)g(x_0, x_1, \theta)q(x_0, x_1, \theta)g(x_1, y_1, \theta)...q(x_{n-1}, x_n, \theta)g(x_n, y_n, \theta)$$

L'expression de la log-vraisemblance est donc :

$$\log f_n(x_{0:n}, y_{0:n}, \theta) = \log \nu(x_0, \theta) + \sum_{t=0}^{n} \log(q(x_t, x_{t+1}, \theta) + \sum_{t=0}^{n} \log g(x_t, y_{t+1}, \theta)$$

Nous nous intéressons à l'inférence dans ce modèle, et en particulier nous souhaiton calculer $\gamma_t(i) = \mathbb{P}(X_t = i|Y_{0:T})$, qui est la probabilité a posteriori de X_t sachant les données observées. Pour cela, nous avons utilisé la méthode introduite par Baum-Welch, appelée méthode *Forward Backward* [8], [6].

$$\begin{aligned}
\gamma_t(i) &= \mathbb{P}(X_t = i|Y_{0:T}) \\
&= \frac{\mathbb{P}(Y_{0:T}|X_t = i)\mathbb{P}(X_t = i)}{\mathbb{P}(Y_{0:T})} \\
&= \frac{\mathbb{P}(Y_{0:t}|X_t = i)\mathbb{P}(Y_{t+1:T}|X_t = i)\mathbb{P}(X_t = i)}{\mathbb{P}(Y_{0:T})} \\
&= \frac{\alpha_t(i)\beta_i(i)}{\mathbb{P}(Y_{0:T})}
\end{aligned}$$

On peut réécrire $\alpha_t(i)$, $\beta_t(i)$ et on obtient :

$$\begin{aligned}
\gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(Y_{0:T})} \\
\alpha_{t+1}(i) &= \sum_{j=1}^{2} f_i(y_{t+1})\alpha_t(j)q_{i,j} \\
\beta_{t+1}(i) &= \sum_{j=1}^{2} q_{i,j}f_j(y_{t+1})\beta_{t+1}(j)
\end{aligned}$$

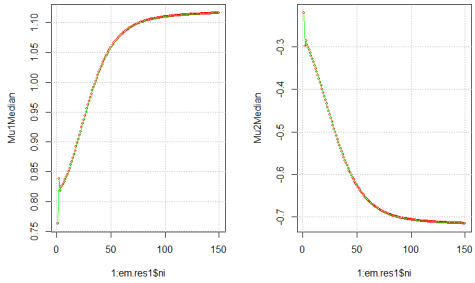
Ainsi nous pouvons calculer à présent les probabilités jointes pour la chaine de Markov, $\mathbb{P}(X_t, X_{t+1}|Y)$ que l'on note dorénavant $\xi(x_t, x_{t+1})$. Grâce à la formule de Bayes et à la formule des probabilités totales on peut écrire :

$$\xi(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j)q_{i,j}f_j(y_{t+1})}{\sum_{i=1}^{2}\sum_{j=1}^{2}\alpha_t(i)\beta_{t+1}(j)q_{i,j}f_j(y_{t+1})}$$

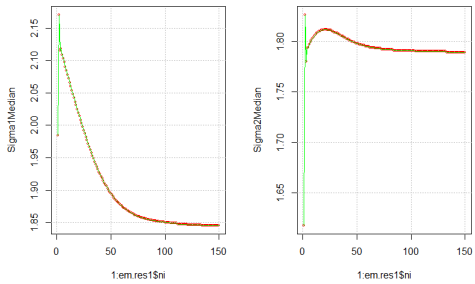
Remarque : en pratique, lors de l'implémentation nous avons été amenés à renormaliser les vecteurs α et β pour des problèmes d'*overflow*.

L'algorithme EM est alors le suivant :

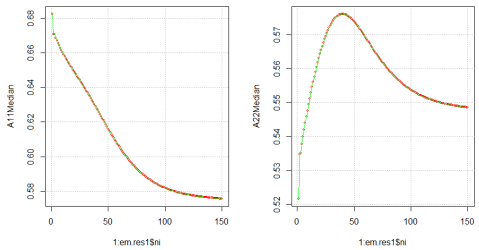
L'implémentation de l'algorithme EM "hors ligne" pour HMM se trouve en annexe. Les figures 6 représentent la convergence pour les termes de moyennes de variance et de la matrice de transition.



(a) Estimation des moyennes



(b) Estimation des variances



(c) Estimation de la diagonale de la matrice de transition

FIGURE 6 – Algorithme EM (version classique) appliqué à un modèle HMM

Un premier constat immédiat est la vitesse de convergence, qui est de l'ordre de 10 fois plus élevées que dans le cas du mélange de gaussienne. On peut donc s'attendre aussi à une vitesse de convergence beaucoup plus lente dans le cas de l'algorithme online.

2.4 EM online pour un HMM

Pour l'algorithme EM online dans le cas des HMM, nous avons d'abord généré une chaîne de Markov cachée à états, que nous avons ensuite bruitée pour obtenir des observations de la forme $Y_t = X_t + V_t$, où $V_t \rightsquigarrow \mathcal{N}(0, v^2)$. Puis nous avons estimé les valeurs de la matrice de transition q de la chaîne X_t, ainsi que ses états μ_1 et μ_2. Enfin, comme précédemment, nous avons répété 100 fois cette expérience (de la génération des données à l'estimation des paramètres), afin de pouvoir tracer l'évolution des estimations moyennes et les premier et troisième quartiles, qui constituent en quelque sorte des intervalles de confiance.

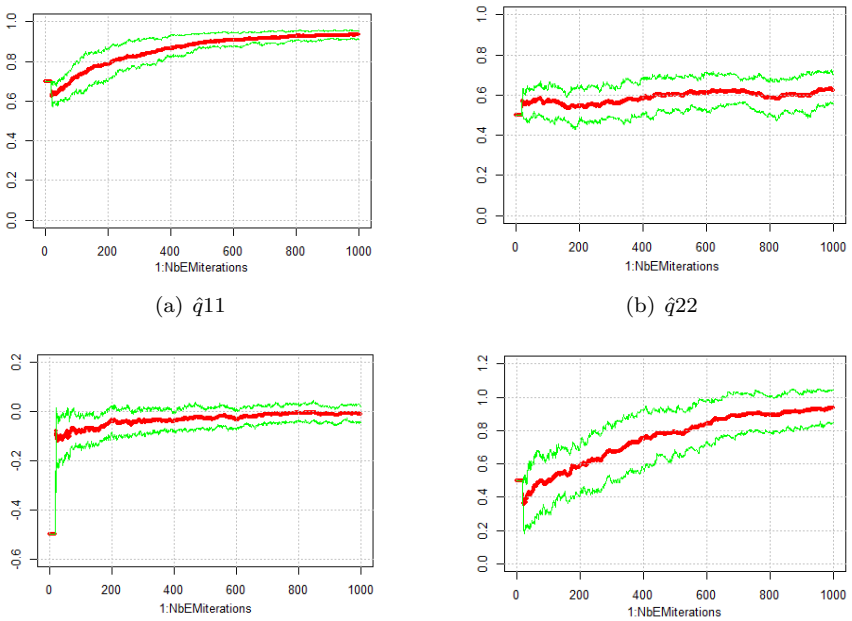
Afin de comparer nos résultats avec ceux avancés dans l'article, nous avons utilisé les mêmes paramètres pour définir les observations, et les mêmes initialisations pour l'algorithme. Les observations sont donc définies par :

$$q_{1,1} = 0.95 \ , \ q_{2,2} = 0.5 \ , \ \mu_1 = 0 \ , \ \mu_2 = 1 \text{ et } v = 0.5$$

Et l'algorihme est initialisé à :

$$q_{1,1}^{(0)} = 0.7 \ , \ q_{2,2}^{(0)} = 0.5 \ , \ \mu_1 = -0.5 \ , \ \mu_2 = 0.5 \text{ et } v = 2$$

La résultats pour la convergence de l'estimation de la matrice de transition et pour les états de la chaîne X_t sont présentés respectivement à la figure 7.



(a) $\hat{q}11$ (b) $\hat{q}22$

(c) $\hat{\mu}(1)$ (d) $\hat{\mu}(2)$

FIGURE 7 – Estimations des paramètres d'un HMM avec l'algorithme EM online

La vitesse de convergence dépend des états : pour q_{11} et μ_2 il faut attendre un grand nombre d'itérations (de l'ordre de 500, ce qui était prévisible d'après les résultats de l'algorithme EM classique), tandis que la convergence est presque acquise dès la sortie de la période de chauffe pour q_{22} et μ_1.

3 Partie Théorique :

3.1 Vérification des hypothèses du théorème

Dans cette section nous vérifierons si le modèle *Markov Chain Observed in Gaussian noise* présenté en section 5.2 respecte les hypothèses du théorème 1, en page 8, énoncé dans [4]. Pour rappel, les théorèmes est énoncé dans la partie 1.4 de ce rapport.

Le modèle est le suivant : $Y_t = X_t + V_t$ où $V_t \rightsquigarrow N(0, v)$ et X_t est une chaine de markov à deux états possibles $\mu(1), \mu(2)$. Pour l'expérimentation numérique nous avons posé les valeurs de θ initiales suivantes :

$$q_0 = \begin{pmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{pmatrix} \ , \ v_0 = 2 \ , \ \mu_0(1) = -0.5 \text{ et } \mu_0(2) = 0.5$$

Hypothèse (i) : Vérifiions que le modèle est exponentiel et la différentiabilité.

Calculons $\mathbb{P}[y_t, x_t | x_{t-1}]$:

L'attention du lecteur est attirée sur la variable x_t est la variable d'état, ou de régime. Elle prend ses valeurs dans $\{1, 2\}$ mais pointe sur les moyennes μ_1, μ_2.

$$\begin{aligned}
\mathbb{P}[y_t, x_t | x_{t-1}] &= \mathbb{P}[y_t | x_t, x_{t-1}] \mathbb{P}[x_t | x_{t-1}] \\
&= \prod_{i=1}^{2} \prod_{j=1}^{2} \left[\mathbb{P}(y_t | x_t, x_{t-1}) q_{i,j} \right]^{\mathbb{1}_{(x_t = i, x_{t-1} = j)}} \\
&= \frac{1}{\sqrt{2\pi}} \exp \left[\sum_{i=1}^{2} \sum_{j=1}^{2} \left(-\frac{1}{2} \log(\nu) - \frac{1}{2\nu} (y - \mu_i)^2 + \log(q_{i,j}) \right) \cdot \mathbb{1}_{(x_t = i, x_{t-1} = j)} \right] \\
&= \frac{1}{\sqrt{2\pi}} \exp \left[-2 \log(\nu) - \frac{1}{2\nu} \sum_{i=1}^{2} 2(y_t^2 - 2\mu_i y_t + \mu_i^2) \mathbb{1}_{(x_t = i)} + \sum_{i,j=1}^{2} \log(q_{i,j}) \mathbb{1}_{(x_t = i, x_{t-1} = j)} \right]
\end{aligned}$$

En notant $\Lambda = \nu^{-1}$, on a

$$\mathbb{P}[y_t, x_t | x_{t-1}] = \frac{1}{\sqrt{2\pi}} \exp \left[\log(\Lambda) - \Lambda \sum_{i=1}^{2} y_t^2 \mathbb{1}_{(x_t = i)} + 2\Lambda \sum_{i=1}^{2} \mu_i y_t \mathbb{1}_{(x_t = i)} - \Lambda \sum_{i=1}^{2} \mu_i^2 \mathbb{1}_{(x_t = i)} + \sum_{i,j=1}^{2} \log(q_{i,j}) \mathbb{1}_{(x_t = i, x_{t-1} = j)} \right]$$

Par identification, il s'agit d'un modèle exponentiel :

$$\mathbb{P}[y_t, x_t | x_{t-1}] = h(x_t, y_t) \exp \left[\psi(\theta), s(x_{t-1}, x_t, y_t) - A(\theta) \right]$$

Où l'on a : $\psi(\theta) = \begin{pmatrix} \Lambda \\ \Lambda \\ \Lambda\mu(1) \\ \Lambda\mu(2) \\ \Lambda\mu(1)^2 \\ \Lambda\mu(2)^2 \\ \log(q_{1,1}) \\ \log(q_{1,2}) \\ \log(q_{2,1}) \\ \log(q_{2,2}) \end{pmatrix}$, $s(x_{t-1}, x_t, y_t) = \begin{pmatrix} -y^2 \mathbb{1}_{(x_t = 1)} \\ -y^2 \mathbb{1}_{(x_t = 2)} \\ 2y_t \mathbb{1}_{(x_t = 1)} \\ 2y_t \mathbb{1}_{(x_t = 2)} \\ \mathbb{1}_{(x_t = 1)} \\ \mathbb{1}_{(x_t = 2)} \\ \mathbb{1}_{(x_t = 1, x_{t-1} = 1)} \\ \mathbb{1}_{(x_t = 1,, x_{t-1} = 2)} \\ \mathbb{1}_{(x_t = 2,, x_{t-1} = 1)} \\ \mathbb{1}_{(x_t = 2,, x_{t-1} = 2)} \end{pmatrix}$,

et $A(\theta) = 2\log(\Lambda)$. Nous retrouvons ainsi les équations que respectent la statistique suffisante énoncées page 11 de [4].

Ainsi si on considère l'intérieur de Θ nous pouvons affirmer que ν^{-1}, $\mu_k\nu^{-1}$, $\log(q_{i,j})$ sont continument différentiables .

hypothèse (ii) : $\chi = \{0, 1\}$, donc χ est bien un ensemble de cardinal fini ($card(\chi) = 2$).

hypothèse (iii) :

$$\theta = \{q, \mu(1), \mu(2), \nu\} \in \mathcal{S}_{++}^2 \times \mathbb{R}^2 \times \mathbb{R}_+^\star$$

Cet ensemble n'est pas compact. En effet, il suffit de considérer la suite $u_n = (q, \mu_1, \mu_2, \frac{1}{n})$.

$$\forall n \in \mathbb{N}^\star, u_n \in \Theta, u_n \longrightarrow (q, \mu_1, \mu_2, 0) \notin \Theta$$

On peut cependant remarquer que cet ensemble est convexe. La notion de compacité nous assure une convergence de l'algorithme EM vers un point stationnaire de la log-vraisemblance. Nous savons que la vraisemblance est croissante avec le nombre d'itération de l'algorithme. Ainsi, supposons dans un premier temps l'existence de $\theta_\star$ nous savons alors que $\forall i \in \{1, .., K\}$, $l(\theta_\star) \geq l(\theta^i)$ où K est le nombre d'itération de l'algorithme EM.

Supposons à présent que $\theta^i \to \theta_\star$ alors le fait que Θ soit compact nous assure que $\theta_\star \in \Theta$. Donc pour toute fonction continue s, $\{s(\theta^i)\}_{i>0}$ converge vers $s(\theta_\star)$. Mais ceci ne nous assure pas l'existence de $\theta_\star$ ni la convergence de θ^i vers cette valeur. Toutefois la compacité de Θ nous assure par le théoreme de Bolzano-Weierstrass qu'il existe au moins une sous suite de θ^i convergente.

Dorénavant on considérera un ensemble $\tilde{\Theta}$ compact inclut dans Θ. Si Θ est compact alors $\left\{\theta \in \Theta : l(\theta) \geq l(\tilde{\theta})\right\}, \forall \theta \in \Theta$.

hypothèse (iv) : A-t-on $q_\theta(x, x') \geq \varepsilon > 0$ $\forall \theta \in \Theta$? La réponse n'est pas évidente puisqu'en faisant tendre ν vers 0 on a un problème de définition. Il est cependant possible de montrer que la limite de $q(x, x')$ est strictement positive lorsque $\nu \to 0$, en remarquant que

$$\phi_{n+1}(k) = \frac{\sum_{k'} \phi_n(k') q_n(k', k) \psi_\theta(k, k^\star, y, v)}{\sum_{k', k''} \phi_n(k') q_n(k', k'') \psi_\theta(k'', k^\star, y, v)}$$

en notant $k^\star$ le k tel que $(y - \mu(k^\star))^2 - (y - \mu(k))^2 \leq 0$ (possible car il y a seulement deux k), et

$$\psi(k, k', y, v) = \frac{g_\theta(k, y)}{g_\theta(k', y)} = exp\left(\frac{(y - \mu(k'))^2 - (y - \mu(k))^2}{2v}\right)$$

Alors $\psi(k, k^\star, y, 0) = 0$ pour $k \neq k^\star$ et 1 sinon.

Donc

$$\lim_{\nu \to 0} \phi_{n+1}(k, \nu) = \begin{cases} 0 \text{ si } k \neq k^\star \\ 1 \text{ sinon} \end{cases}$$

Et on peut montrer qu'à partir du rang 2 tous les ρ^q sont strictement positifs. Alors $S_{n+1}^q(i, j) = \sum_{k'} \rho_{n+1}^q(i, j, k') \phi_{n+1}(k') > 0$ pour tout (i, j). Et donc $q_{n+1}(i, j) > 0$ pour tout (i,j).

Si ν ne tend pas vers 0, remarquons que

$$\sum_{k=0}^{m} \phi(k) = 1 \text{ et } \forall k \in \{1, \ldots, m\}, 0 \leq \phi(k) \leq 1$$

Donc il existe k tel que $\phi_n(k) \geq 1/m > 0$, et on en conclut comme précédemment que $q_{n+1}(i,j) > 0$ pour tout (i,j).

En conclusion, nous n'avons pas $q_\theta(x,x') \geq \varepsilon > 0 \ \forall \theta \in \Theta$, mais nous pouvons affirmer que, sur tout compact inclut dans Θ, $\forall (x,x') \in \chi^2, q_\theta(x,x') \geq \varepsilon > 0$

hypothèse (v) : Montrons que $sup_\theta sup_y \bar{g}_\theta(y) < \infty$

On a :

$$\begin{aligned}
\sup_\theta \sup_y \bar{g}_\theta(y) &= \sup_y \sup_\theta \bar{g}_\theta(y) \\
&= \sup_y \sup_\theta \sum_{x \in \{\mu(1),\mu(2)\}} g_\theta(x,y) \\
&\leq \sup_y \sum_{x \in \{\mu(1),\mu(2)\}} \sup_\theta g_\theta(x,y)
\end{aligned}$$

Or ces densités étant gaussiennes nous avons

$$\forall \theta \in \Theta, g_\theta(\mu(1),y) + g_\theta(\mu(2),y) = \frac{1}{\sqrt{2\pi\nu}} e^{-\frac{1}{2\nu}(y-\mu(1))} + \frac{1}{\sqrt{2\pi\nu}} e^{-\frac{1}{2\nu}(y-\mu(2))^2}$$

Nous pouvons maximiser ces deux fonctions objectives par rapport à ν. On obtient une valeur maximale pour $\nu = (y-\mu(k))^2$.

Donc

$$\sup_y \sup_\theta \bar{g}_\theta(y) \leq \sup_y \left(\frac{1}{\sqrt{(2\pi(y-\mu(1))}} + \frac{1}{\sqrt{(2\pi(y-\mu(2))}}\right) \underset{p.s.}{<} \infty$$

hypothèse (vi) : A-t-on $\mathbb{E}_{\theta^\star}[|\log \inf_\theta \bar{g}_\theta(Y_0)] < \infty$?

Cette hypothèse n'est pas vérifiable car $\inf_\theta \bar{g}_\theta(Y_0) = 0$ (il suffit de faire tendre ν vers 0). En revanche, cette hypothèse est vraie pour tout compact inclut dans Θ. En effet :

Soit $\mathcal{K}$ un compact inclut dans Θ.

La fonction $\bar{g}_\theta(Y_0)$ étant continue sur $\theta \in \mathcal{K}$ elle atteint ses bornes. Par ailleurs elle est strictement potivite sur $\mathcal{K}$. Donc il existe c tel que $\bar{g}_\theta(Y_0) > c$, pour tout $\theta \in \mathcal{K}$. Alors

$$\inf_{\theta \in \mathcal{K}} \bar{g}_\theta(Y_0) > c$$

Par ailleurs, soit $\theta_0 \in \mathcal{K}$. On a :

$$\inf_{\theta \in \mathcal{K}} \bar{g}_\theta(Y_0) < \bar{g}_{\theta_0}(Y_0)$$

Ainsi

$$c < \inf_{\theta \in \mathcal{K}} \bar{g}_\theta(Y_0) < \bar{g}_{\theta_0}(Y_0)$$

Et $c > 0$, donc par croissance de la fonction log :

$$log(c) < log\big(\inf_{\theta \in \mathcal{K}} \bar{g}_\theta(Y_0)\big) < log\big(\bar{g}_{\theta_0}(Y_0)\big)$$

Et donc

$$\mathbb{E}\Big[\big|log\big(\inf_{\theta\in\mathcal{K}}\bar{g}_\theta(Y_0)\big)\big|\Big] \leq max\Big(\big|log(c)\big|, \big|log\big(\bar{g}_{\theta_0}(Y_0)\big)\big|\Big)$$

D'où

$$\mathbb{E}\Big[\big|log\big(\inf_{\theta\in\mathcal{K}}\bar{g}_\theta(Y_0)\big)\big|\Big] < \infty$$

Bilan : pour tout compact $\mathcal{K}$ dans Θ, l'hypothèse est vraie.

3.2 Autour des preuves de fin d'article :

En annexe, [4] propose un preuve du théorème, par l'introduction de théorèmes énoncés dans [1], d'un lemme et d'un corrolaire. Afin de comprendre la nécessité des hypothèses, nous nous sommes intéressés à la preuve du théorème 1. Plusieurs hypothèses apparaissent comme nécéssaires dans les preuves :

- famille exponentielle et différentiabilité (théorème 3 de [1])
- le fait que χ soit de cardinal fini (corrolaire 2),
- moments d'ordre 1 et 2 finis (théorème 1)
- $\theta^\star$ est dans l'intérieur de Θ (théorème 1)

En revanche, la condition de compacité, qui est une hypothèse très forte et qui nous a posé plusieurs difficultés dans la partie vérification des hypothèses, n'apparait pas clairement. Il est peut être possible d'alléger cette condition en supposant l'ensemble Θ convexe. Par ailleurs, les preuves nous apprennent que, dans un cadre plus général, une autre hypothèse peut être vérifiée par q et g simultanément : la condition présentée dans [1] et appelée *Strong Mixing Condition* :

Il existe un noyau de transition K et des fonctions mesurables ξ^- et ξ^+ telle que pour tout $(A, y) \in \xi \times \mathcal{Y}$,

$$\xi^-(y)K(y, A) \leq \int_A q(x, dx')g(x', y) \leq \xi^-(y)K(y, A)$$

3.3 Utilisation des méthodes MCMC :

Les méthodes de simulations MCMC permettent de générer des échantillons de la distribution a posteriori des paramètres du modèle. Dans l'étape E de l'algorithme EM Online il s'agit de calculer $\frac{1}{n}\mathbb{E}_{\nu,\theta_k}[\sum_{t=1}^n s(X_{t-1}, X_t, Y_t)|Y_{0:n}]$. On peut penser à utiliser une seule chaine MCMC pour un nombre d'itérations pour laquelle la convergence est supposée atteinte et ainsi utiliser la dynamique de la chaine pour échantillonner selon la loi a posteriori.

Cette méthode peut être une alternative intéressante à la simulation de plusieurs chaines en parallèle, et améliorer sensiblement le temps de calcul.

Toutefois, il est nécessaire de poser des hypothèses sur le noyau de la chaine. En effet, en simulant une seule chaine on prend le risque de ne pas explorer tous les états . On peut par exemple supposer qu' une condition d'irréductibilité est nécessaire.

Conclusion

A la lumière de [4] et d'expérience portant sur les modèles de mélange de gaussiennes et de chaines de Markov cachées, nous avons pu tester la pertinence de l'algorithme EM en ligne, comparée un l'algorithme EM classique. Il apparait alors que la version en ligne est très satisfaisante et qu'elle permet une application de l'algorithme EM dans un contexte plus souple, puisqu'il n'est plus nécessaire de disposer de l'ensemble des données pour estimer un modèle. Il est cependant important de noter que la convergence est plus lente dans le cas de l'algorithme en ligne. La version on line nécessite donc moins d'observations mais un nombre d'itérations plus importants.

Toutefois, nous avons vu que la convergence de l'algorithme en ligne s'effectue par le théorème 1 sous de nombreuses hypothèses, parfois très restrictives (par exemple la compacité de Θ). L'analyse de ces hypothèses et les expériences montrent que ces conditions ne sont pas forcément nécessaires. Une réflexion nous a alors amené à penser que l'hypothèse portant sur la compacité peut peut-être être allégée, en supposant par exemple que l'ensemble des paramètres et convexe.

Au terme de cette étude il apparait que l'algorithme EM Online permet une plus grande souplesse que l'algorithme EM classique, et son utilisation pourrait être préférée dans de nombreux domaines applicatifs pour faisant intervenir l'estimation de paramètres en temps réels, tel que le traitement automatique de la parole, ...

Références

[1] Moulines et Ryden Cappé. Inference in hidden markov models. 1995.

[2] Olivier Cappé. Version récursive de l'algorithme em pour l'estimation en ligne des paramètres de modèles de markov cachés.

[3] Olivier Cappé. Modéles de mélange et modèles de markov cachés pour le traitement automatique de la parole. 2000.

[4] Olivier Cappé. Online em algorithm for hidden markov chains. 2011.

[5] Leggetter and Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. 1995.

[6] MacKey. Ensemble learning for hidden markov models.

[7] Eric Moulines Olivier Cappé and Tobias Ryden. Inference in hidden markov models. 2009.

[8] Ganapathiraju Wu and Picone. Baum-welch re-estimation of hidden markov model. 1999.

[9] Zeitouni and Dembo. Exact filters for the estimation of the number of transitions of finite-state continuous-time markov processes. 1988.

4 Annexe

4.1 Code R pour l'algorithme EM Classique d'un mélange de gaussiennes

```
#########################################################################
# Donnees issues dun melange gaussien
#########################################################################
GaussianSmoothing=function(N,p,mu1,mu2,sigma1,sigma2){
  X=NULL
  X<-matrix(data=0,nrow=N,ncol=1)
### Boucle permettant de tirer N valeurs issues
### dun  melange gaussien :
for (i in 2:N) {
  if(runif(1)<p){X[i]=rnorm(1,mu1,sigma1)}
  else{X[i]=rnorm(1,mu2,sigma2)}
}
  X
}

X=GaussianSmoothing(N=1000,p=0.4,mu1=0,mu2=1,sigma1=0.5,sigma2=0.5)
par(mfrow=c(2,2))
hist(X,col="3",breaks=100)
ts.plot(X,col="3")

##############
# Algo EM : #
##############
#Definition et initialisation des parametres :
t1=proc.time()#horloge au debut de lalgo.
K = 20
#on fera K iterations de lalgorithme

hatP<-matrix(data=0,nrow=K,ncol=2) #matrice K,2 ..1ere colonne : p, 2nd:1-p
mu<-matrix(data=0,nrow=K,ncol=2)
sigma<-matrix(data=0,nrow=K,ncol=2)

hatP[1,1]=runif(1) #Initialisation de p uniforme
hatP[1,2]=runif(1)
mu[1,1] = runif(1,min=-1,max=1)
mu[1,2]= runif(1,min=1,max=2)
sigma[1,1] =  runif(1,min=0,max=1)
sigma[1,2] =  runif(1,min=0,max=1)

for (i in 2:K) {

  vrais1 = hatP[i-1,1]*dnorm(X,mean=mu[i-1,1],sd=sigma[i-1,1])
  vrais2 = (1-hatP[i-1,1])*dnorm(X,mean=mu[i-1,2],sd=sigma[i-1,2])

  vrais12 = vrais1 / (vrais1 + vrais2) # probas a posteriori p_{i,1}
  vrais22 = vrais2 / (vrais1 + vrais2) # probas a posteriori p_{i,2}
  ## Mise a jour de lambda1 = P(Z=1 | X,Thteta) :
  hatP[i,1]=mean(vrais12)   #p estime
  hatP[i,2]= 1-hatP[i,1]

  ## Mise  a jour de mu1 et mu2 :
  mu[i,1] = sum(vrais12*X)/sum(vrais12)
  mu[i,2] = sum(vrais22*X)/sum(vrais22)
  ## Mise a jour de sigma1 et sigma2 :
  sigma[i,1] = sqrt(sum(vrais12*(X-mu[i,1])^2)/(sum(vrais12)))
```

```
  sigma[i,2] = sqrt(sum(vrais22*(X-mu[i,2])^2)/(sum(vrais22)))
}

Mu1Median = rep(0,K)
Mu1Sup = rep(0,K)
Mu1Inf = rep(0,K)
for(i in 1:K){
  s = summary(mu[,i])
  Mu1Inf[i] = s[2]
  Mu1Median[i] = s[4]
  Mu1Sup[i] = s[5]
}


plot(1:K, sigma[,1],ylim=c(-2,2), pch = 16, type='o', col = "red", cex = 0.6)
plot(1:K, mu[,1], ylim=c(-2,2),pch = 16, type='o', col = "red",cex = 0.6)

t2=proc.time()-t1 # calcul le temps pour effecuter algo EM
t2

#boxplot(sigma[,1], main="Simple Box plot", col="lightblue")
```

4.2 Algorithme EM online pour les HMM

```
source("fonctions.R")

Nsimulations = 100;NbEMiterations = 1000

### Pour stocker les valeurs
Q11Array = array(0,dim = c(Nsimulations, NbEMiterations))

########################################
### Simulations
########################################

for(simulation in 1:Nsimulations){
  ########################################
  ### Generation des observations
  ########################################

  NbObs = 1000
  nmin = 20 ## Iteration a partir de laquelle on modifie les parametres

  Q<-matrix(data=c(0.95,0.3,0.05,0.7),nrow=2,ncol=2)# to define matrix Q
  mu=matrix(data=c(0,1),nrow=1,ncol=2) # to define vector mu

  Y=MarkovChain(N=NbObs,q=Q,mu1=mu[1],mu2=mu[2],sigma=.5)

  ########################################
  ### Algorithme
  ########################################
  Mu=c(-0.5,0.5)
  q = matrix(c(0.7,0.5,0.3,0.5), nrow = 2)
  v = 0.5
  nu = c(-0.5, 0.5)  ### nu est la densite de X0 donc soit +- 0.5
  gamma = 1

  ### Initialisation des parametres
  phi = c( nu[1]*g(y,v)[1] , nu[2]*g(y,v)[2] ) / (g(y,v)[1] + g(y,v)[2])
  Rho = array(0, dim = c(2,2,2))
  RhoD = array(0,dim = c(2,2,3))
  for(i in 1:2){
    for(k in 1:2){
      for(d in 1:3){
        RhoD[i,k,d] = delta(i-k)*(y^(d-1))
      }
    }
  }

  Q11Array[simulation, 1] = q[1,1]

  ### C'est parti pour la boucle !
  for(n in 2:NbEMiterations){
    y=Y[n]
    gamma = n^(-0.6)

    ### Approx. Filter Update
    newPhi = actualiserPhi()

    ### E-Step
    newR = actualiserR()
    newRho = actualiserRho()
    newRhoD = actualiserRhoD()
```

```
60
      ### M-Step
62    if(n>=nmin){
        newS = actualiserS()
64      q = actualiserQ()
        newSD = actualiserSD()
66      Mu = actualiserMu()
        v = actualiserV()
68    }

70    phi = newPhi
      Rho = newRho
72    RhoD = newRhoD

74    Q11Array[simulation, n] = q[1,1]
    }
76  }

78  Q11Median = rep(0,NbEMiterations)
    Q11Sup = rep(0,NbEMiterations)
80  Q11Inf = rep(0,NbEMiterations)
    for(i in 1:NbEMiterations){
82    s = summary(Q11Array[,i])
      Q11Inf[i] = s[2]
84    Q11Median[i] = s[4]
      Q11Sup[i] = s[5]
86  }

88  plot(1:NbEMiterations, Q11Median, main = "Q_11", ylim = c(0,1), pch = 16, type='o'
        , col = "red", cex = 0.6)
    grid(col = "grey", lty = "dotted", lwd = 1)
90  points(1:NbEMiterations, Q11Inf, pch = 20, col = "green", type='l', lwd = 1, cex =
        1)
    points(1:NbEMiterations, Q11Sup, pch = 20, col = "green", type='l', lwd = 1, cex =
        1)
```

Les fonctions :

```
1

    ##########################################
3   ### Generation des donnees
    ##########################################
5   MarkovChain=function(N,q,mu1,mu2,sigma){
      X=NULL
7     X<-matrix(nrow=N, ncol=1)
      X[1,1]=mu2
9     V=rnorm(N,0,sigma)

11    for(i in 2:N){
        u=runif(1)
13      if(X[i-1,1]==mu1){
          if(u<q[1,1]){X[i,1]=mu1}else X[i,1]=mu2}
15      if(X[i-1,1]==mu2){
          if(u<q[2,2]){X[i,1]=mu2}else X[i,1]=mu1}
17    }
      Y=X+rnorm(N,0,sigma)
19    Y
    }
21

    g = function(y,v){
```

```
  c(exp(-((y - Mu[1])^2) / (2*v)), exp(-((y - Mu[2])^2) / (2*v)))
}

delta = function(x){
  1*(x==0)
}

##########################################
### Approx. Filter Update
##########################################
actualiserPhi = function(){
  newPhi = c(0,0)
  for(i in 1:2){
    newPhi[i] = (phi[1]*q[1,i]*g(y,v)[i]+phi[2]*q[2,i]*g(y,v)[i])
  }
  newPhi/(sum(newPhi))
}


##########################################
### E-Step
##########################################
actualiserR = function(){
  newR = array(0, dim = c(2,2))
  for(i in 1:2){
    for(j in 1:2){
      newR[i,j] = phi[i]*q[i,j]/(phi[1]*q[1,j] + phi[2]*q[2,j])
    }
  }
  newR
}

actualiserRho = function(){
  newRho = array(0, dim = c(2,2,2))
  for(i in 1:2){
    for(j in 1:2){
      for(k in 1:2){
        newRho[i,j,k] = gamma*delta(j-k)*newR[i,j] + (1-gamma)*(Rho[i,j,1]*newR[1,
            k] + Rho[i,j,2]*newR[2,k])
      }
    }
  }
  newRho
}

actualiserRhoD = function(){
  newRhoD = array(0, dim = c(2,2,3))
  for(i in 1:2){
    for(k in 1:2){
      for(d in 1:3){
        newRhoD[i,k,d] = gamma*delta(i-k)*(y^(d-1)) + (1-gamma)*(RhoD[i,1,d]*newR
            [1,k] + RhoD[i,2,d]*newR[2,k])
      }
    }
  }
  newRhoD
}

##########################################
### M-Step
##########################################
actualiserS = function(){
```

```
  newS = array(0, dim = c(2,2))
  for(i in 1:2){
    for(j in 1:2){
      newS[i,j] = newRho[i,j,1]*newPhi[1] + newRho[i,j,2]*newPhi[2]
    }
  }
  newS
}

actualiserQ = function(){
  q = array(0,dim = c(2,2))
  for(i in 1:2){
    for(j in 1:2){
      q[i,j] = newS[i,j] / (newS[i,1] + newS[i,2])
    }
  }
  q
}

actualiserSD = function(){
  newSD = array(0, dim = c(2,3))
  for(i in 1:2){
    for(d in 1:3){
      newSD[i,d] = newRhoD[i,1,d]*newPhi[1] + newRhoD[i,2,d]*newPhi[2]
    }
  }
  newSD
}

actualiserMu = function(){
  c(newSD[1,2]/ newSD[1,1], newSD[2,2] / newSD[2,1])
}

actualiserV = function(){
  numerateur = newSD[1,3] - (Mu[1]^2)*newSD[1,1] + newSD[2,3] - (Mu[2]^2)*newSD
      [2,1]
  numerateur /(newSD[1,1] + newSD[2,1])
}
```

4.3 Code R pour l'algorithme EM Classique de HMM

```
1
# Ce programme fonctionne a partir de 3 fonctions differentes
3   # On genere les donnees HMM a partir de 'rdata.hmm'
    #On realise la procedure foward backward grace a la fonction 'bwfw.hmm
5   #Algorithme Em dans la fonction 'em.hmm'

7  # le nombre d observation
NUM1<-8000
9


11 ##########################################
#Parametres pour generer la chaine de Markov cachees
13 ##########################################

15  # distribution initiale
 pii<-c(1, 0)
17  # matrice de transition
A<-matrix(c(0.95, 0.05, 0.3, 0.70), 2, 2, byrow=T)
19  # 1ere distribution de y|x
 f0<-c(0, 2)
21  #  2nd distribution de y|x
 f1<-c(1, 2)
23  # simulation de HMM

25 rdata1<-rdata.hmm(NUM1, pii, A, f0, 1, f1)
 # donnees observees y
27 x1<-rdata1$o

29 par(mfrow=c(1,2))
 hist(rdata1$o,col="blue",breaks=100)
31 ts.plot(rdata1$o,col="blue")

33  # donnees latentes
 theta1<-rdata1$s
35
##############################################
37 # Procedure forward/backward et scaling (etape E)
##############################################
39


41 x1<-rdata1$o
 fb.res1<-bwfw1.hmm(x1, pii, A, f0, f1)
43
# variable backward (beta)
45 backward.var<-fb.res1$bw

47 # variable forward (alpha)
 forward.var<-fb.res1$fw
49
##########################################################
51 # Reestimation des paramatres (etape M)
##########################################################
53 L<-1

55 em.res1<-em1.hmm(x1, maxiter=150)

57 em.res1$A
 em.res1$f1
59 em.res1$pii
```

```
em.res1$muiter
em.res1$sditer
A11<-rep(0,em.res1$ni)
for(i in 1:em.res1$ni){A11[i]=em.res1$aiter[[i]][1,1]}
A22<-rep(0,em.res1$ni)
for(i in 1:em.res1$ni){A22[i]=em.res1$aiter[[i]][2,2]}

#On creer des tableau pour les graphiques de convergences.
Mu1Median = rep(0,em.res1$ni)
Mu1Sup = rep(0,em.res1$ni)
Mu1Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
  s = summary(em.res1$muiter[i,1])
  Mu1Inf[i] = s[2]
  Mu1Median [i] = s[3]
  Mu1Sup [i] = s[5]
}
Mu2Median = rep(0,em.res1$ni)
Mu2Sup = rep(0,em.res1$ni)
Mu2Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
  s = summary(em.res1$muiter[i,2])
  Mu2Inf[i] = s[2]
  Mu2Median [i] = s[3]
  Mu2Sup [i] = s[5]
}

Sigma1Median = rep(0,em.res1$ni)
Sigma1Sup = rep(0,em.res1$ni)
Sigma1Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
  s = summary(em.res1$sditer[i,1])
  Sigma1Inf[i] = s[2]
  Sigma1Median [i] = s[3]
  Sigma1Sup [i] = s[5]
}


Sigma2Median = rep(0,em.res1$ni)
Sigma2Sup = rep(0,em.res1$ni)
Sigma2Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
  s = summary(em.res1$sditer[i,2])
  Sigma2Inf[i] = s[2]
  Sigma2Median [i] = s[3]
  Sigma2Sup [i] = s[5]
}

A11Median = rep(0,em.res1$ni)
A11Sup = rep(0,em.res1$ni)
A11Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
  s = summary(A11[i])
  A11Inf[i] = s[2]
  A11Median[i] = s[3]
  A11Sup [i] = s[5]
}

A22Median = rep(0,em.res1$ni)
A22Sup = rep(0,em.res1$ni)
A22Inf = rep(0,em.res1$ni)
for(i in 1:em.res1$ni){
```

```
  s = summary(A22[i])
  A22Inf[i] = s[2]
  A22Median[i] = s[3]
  A22Sup [i] = s[5]
}


##############################################
### Graphiques
##############################################
par(mfrow=c(1,2))
#mu1
plot(1:em.res1$ni, Mu1Median, type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, Mu1Inf, pch = 20, col = "green", type='l', lwd = 1, cex = 1)
points(1:em.res1$ni, Mu1Sup, pch = 20, col = "green", type='l', lwd = 1, cex = 1)

#mu2
plot(1:em.res1$ni, Mu2Median, type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, Mu2Inf, pch = 20, col = "green", type='l', lwd = 1, cex = 1)
points(1:em.res1$ni, Mu2Sup, pch = 20, col = "green", type='l', lwd = 1, cex = 1)

#Sigma1
plot(1:em.res1$ni, Sigma1Median, type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, Sigma1Inf, pch = 20, col = "green", type='l', lwd = 1, cex =
    1)
points(1:em.res1$ni, Sigma1Sup, pch = 20, col = "green", type='l', lwd = 1, cex =
    1)

#Sigma2
plot(1:em.res1$ni, Sigma2Median, type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, Sigma2Inf, pch = 20, col = "green", type='l', lwd = 1, cex =
    1)
points(1:em.res1$ni, Sigma2Sup, pch = 20, col = "green", type='l', lwd = 1, cex =
    1)

#A11
plot(1:em.res1$ni, A11Median,,type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, A11Inf, pch = 20, col = "green", type='l', lwd = 1, cex = 1)
points(1:em.res1$ni, A11Sup, pch = 20, col = "green", type='l', lwd = 1, cex = 1)

#A22
plot(1:em.res1$ni, A22Median ,type='o', col = "red", cex = 0.6)
grid(col = "grey", lty = "dotted", lwd = 1)
points(1:em.res1$ni, A22Inf, pch = 20, col = "green", type='l', lwd = 1, cex = 1)
points(1:em.res1$ni, A22Sup, pch = 20, col = "green", type='l', lwd = 1, cex = 1)


#####################################################################
#Fonction Backward/Forward et scaling
#####################################################################
bwfw1.hmm<-function(x, pii, A, f0, f1)
{

#####################################################################

## USAGE
```

```
 # bwfw.hmm(x, pii, A, f0, f1)

## ARGUMENTS
 # y=(y[1], ..., y[m]): donnee observees
 # pii=(pii[0], pii[1]):distribution initiale
 # A=(A[0,0], A[0,1]\\ A[1,0], A[1,1]): matrice de transition
 # f0=(mu, sigma): parametre pour y|x=1
 # f1=(mu, sigma):  parametre pour y|x=2

## DETAILS
 # bwfw.hmm calcule les valeurs forward et backward en tenant compte des pb d
     overflow
 # --Procedure (Baum et al.)

#########################################################################

## Initialisation

NUM<-length(y)

## Densities

f0y<-dnorm(y, f0[1], f0[2])
f1y<-dnorm(y, f1[1], f1[2])

## procedure Forward backward
alpha<-matrix(rep(0, NUM*2), NUM, 2, byrow=T)
# scaling variable c_0
c0<-rep(0, NUM)

alpha[1, 1]<-pii[1]*f0y[1]
alpha[1, 2]<-pii[2]*f1y[1]
#scalling
c0[1]<-1/sum(alpha[1, ])
alpha[1, ]<-c0[1]*alpha[1, ]

for (k in 1:(NUM-1))
{
  alpha[k+1, 1]<-(alpha[k, 1]*A[1, 1]+alpha[k, 2]*A[2, 1])*f0y[k+1]
  alpha[k+1, 2]<-(alpha[k, 1]*A[1, 2]+alpha[k, 2]*A[2, 2])*f1y[k+1]
  # scalling
  c0[k+1]<-1/sum(alpha[k+1, ])
  alpha[k+1, ]<-c0[k+1]*alpha[k+1, ]
}



beta<-matrix(rep(0, NUM*2), NUM, 2, byrow=T)

beta[NUM, 1]<-c0[NUM]
beta[NUM, 2]<-c0[NUM]

for (k in (NUM-1):1)
{
  beta[k, 1]<-A[1, 1]*f0y[k+1]*beta[k+1, 1]+A[1, 2]*f1y[k+1]*beta[k+1, 2]
  beta[k, 2]<-A[2, 1]*f0y[k+1]*beta[k+1, 1]+A[2, 2]*f1y[k+1]*beta[k+1, 2]
  # rescaling beta
  # using the same scaling factors as alpha
  beta[k, ]<-c0[k]*beta[k, ]
}

lfdr<-rep(0, NUM)
```

```
241
for (k in 1:NUM)
243 {
  q1<-alpha[k, 1]*beta[k, 1]
245   q2<-alpha[k, 2]*beta[k, 2]
  lfdr[k]<-q1/(q1+q2)
247 }

249 # probabilite des etats caches


251
gamma<-matrix(1:(NUM*2), NUM, 2, byrow=T)
253
gamma[NUM, ]<-c(lfdr[NUM], 1-lfdr[NUM])
255 dgamma<-array(rep(0, (NUM-1)*4), c(2, 2, (NUM-1)))

257 for (k in 1:(NUM-1))
{
259   denom<-0
  for (i in 0:1)
261   {
    for (j in 0:1)
263     {
      fx<-(1-j)*f0x[k+1]+j*f1x[k+1]
265       denom<-denom+alpha[k, i+1]*A[i+1, j+1]*fx*beta[k+1, j+1]
    }
267   }
  for (i in 0:1)
269   {
    gamma[k, i+1]<-0
271     for (j in 0:1)
    {
273       fx<-(1-j)*f0x[k+1]+j*f1x[k+1]
      dgamma[i+1, j+1, k]<-alpha[k, i+1]*A[i+1, j+1]*fx*beta[k+1, j+1]/denom
275       gamma[k, i+1]<-gamma[k, i+1]+dgamma[i+1, j+1, k]
    }
277   }
}
279
# on retourne les resultats
281
bwfw.var<-list(bw=alpha, fw=beta, lsi=lfdr, pr=gamma, ts=dgamma)
283

285 return(bwfw.var)

287 }

289 ###################################################################
# ALGORITHME EM
291 ###################################################################
em1.hmm<-function(y, maxiter=50)
293 {

295 #
    ##############################################################################################


297 ## ARGUMENTS
 # y: donnes observee
299  # maxiter: Nombre d iteration maximum
```

```
#
     ##############################################################################


#Tableau des stockages des valeurs des parametres au cours de EM
A_ITER<-vector("list",maxiter)
for(niter in 1:maxiter){A_ITER[[niter]] <- matrix(data = 0,nrow=2,ncol=2)}
MU_ITER<-matrix(data=0,nrow=maxiter,ncol=2)
SIGMA_ITER<-matrix(data=0,nrow=maxiter,ncol=2)
#################################################################
NUM<-length(x)

niter<-0

###initialisation des parametres
pii.new<-c(1, 0)
A.new<-matrix(c(0.7, 0.3, 0.5, 0.5), 2, 2, byrow=T)
f0.new<-c(-0.5, 2)
f1.new<-c(0.5, 2)



### The E-M Algorithm

for(niter in 1:maxiter)
{


pii.old<-pii.new
A.old<-A.new
f1.old<-f1.new
f0.old<-f0.new

bwfw.res<-bwfw1.hmm(y, pii.old, A.old, f0.old, f1.old)

#probabilite des donnes cachee
gamma<-bwfw.res$pr
# the transition variables
dgamma<-bwfw.res$ts

##   (M STEP)

for (i in 0:1)
{
   pii.new[i+1]<-gamma[1, i+1]
}



for (i in 0:1)
{
   for (j in 0:1)
   {
      q1<-sum(dgamma[i+1, j+1, ])
      q2<-sum(gamma[1:(NUM-1), i+1])
      A.new[i+1, j+1]<-q1/q2

   }
}

A_ITER[[niter]]= A.new
```

```
361
    q11<-sum(gamma[, 2])
363 q22<-sum(gamma[, 2]*y)
    mu1<-q22/q11
365 q12<-sum(gamma[, 1])
    q22<-sum(gamma[, 1]*y)
367 mu2<-q22/q12
    MU_ITER[niter ,]=c(mu1,mu2)
369 q31<-sum(gamma[, 2]*(y-mu1)*(y-mu1))
    q32<-sum(gamma[, 2]*(y-mu1)*(y-mu1))
371
    sd1<-sqrt(q31/q11)
373 sd2<-sqrt(q32/q12)
    SIGMA_ITER[niter ,]=c(sd1,sd2)
375 f1.new<-c(mu1, sd1)
    f0.new<-c(mu2,sd2)
377
    }
379
    lfdr<-gamma[, 1]
381
    em.var<-list(pii=pii.new, A=A.new, f1=f1.new, lf=lfdr, ni=niter, muiter=MU_ITER,
        sditer=SIGMA_ITER, aiter=A_ITER)
383 return (em.var)

385 }
```