MVA, Projet PGM : Rapport
Factorial HMM

Théis Bazin Valentin De Bortoli

20 janvier 2017

Table des matières

1 Présentation du modèle

1.1 Comparaison avec HMM

Le but de ce rapport est de présenter le modèle *Factorial HMM*, extension du modèle HMM, *Hidden Markov Model*. Ce dernier suppose qu'une variable aléatoire cachée, dont l'évolution temporelle est régie par une chaine de variables cachées, est à l'origine d'une observation. Le modèle que l'on se propose d'étudier ici met en parallèle M chaînes[1] de variables cachées qui sont à l'origine d'une observation. Ce modèle décrit bien des situations d'évolution temporelle dans lesquelles plusieurs facteurs indépendants entrent en jeu. L'espace d'état de chaque variable cachée est supposé être le même et est fini de cardinal K.

On pourrait tenter de regrouper les variables cachées en une seule variable et considérer le modèle HMM connu. Ce modèle n'est pas satisfaisant pour deux raisons :

— l'espace d'état de cette unique variable cachée est alors de cardinal K^M, on arrive rapidement aux limites numériques des ordinateurs.

— l'information sur l'indépendance des variables cachées est perdue.

Ces deux remarques justifient l'intérêt et la complexité de Factorial HMM.

1.2 Graphe associé et inférence

Compte tenu de la proximité entre le modèle Factorial HMM et HMM il est naturel de se poser la question de l'inférence et de la possibilité de construire un algorithme pour apprendre les paramètres du modèle. Dans ce but, on présente le graphe de Factorial HMM. Ce graphe permet de mettre à nouveau en évidence la complexité du nouveau modèle.

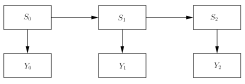

Figure 1 – Modèle HMM

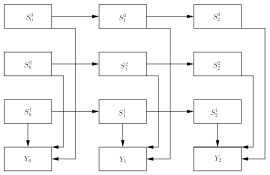

Figure 2 – Modèle Factorial HMM

Dans le cas de Factorial HMM on perd la structure d'arbre. Néanmoins, on peut développer un algorithme semblable à celui développé dans HMM qui permet de calculer exactement les marginales des probabilités se factorisant sur le graphe. Il est alors possible de résoudre exactement E-step dans un algorithme *Expectation-Maximization*, EM. Néanmoins, cet algorithme exact n'est pas satisfaisant dès lors que le nombre de chaînes devient important. On se tourne alors vers trois méthodes d'approximation :

— l'échantillonnage de Gibbs (*Gibbs sampling*),

— *mean-fields*,

— *structured mean-fields*.

Toutes ces méthodes ont été implémentées et testées sur des données de synthèse.

On renvoie à 6.1 où toutes les notations qui vont être utilisées dans la suite du rapport sont exposées et précisées.

2 Le problème d'inférence et l'algorithme EM

On rappelle de manière détaillée le principe de l'algorithme EM en appendice **??**. Ici, on rappelle simplement qu'il s'agit de trouver un moyen d'optimiser la log-vraisemblance $\log(p(Y_{(t)})$ qui peut être très compliquée à calculer. En utilisant l'inégalité de Jensen et les probabilités conditionnelles on peut construire un algorithme itératif qui maximise coordonnée par coordonnée une borne inférieure de la log-vraisemblance (on montre même que la log-vraisemblance augmente à chaque étape).

En appendice, **??**, on présente la mise à jour effectuée à cette étape. Il est à noter que l'on retrouve des formes closes proches de celles obtenues dans HMM.

On remarque que les mises à jour données par M-step dépendent de E-step uniquement pour le calcul de trois quantités :

1. Malgré la similitude avec M chaînes de Markov indépendantes et homogènes, il est incorrect d'utiliser cette terminologie ici puisque les chaînes sont couplées via les variables d'observation

— $\forall (t,m) \in [\![0,T]\!] \times [\![1,M]\!],\ \mathbb{E}_{q_n}(S_t^m)$
— $\forall (t,m_1,m_2) \in [\![0,T]\!] \times [\![1,M]\!]^2,\ \mathbb{E}_{q_n}(S_t^{m_1} S_t^{m_2})$
— $\forall t \in [\![1,T]\!] \times [\![1,M]\!],\ \mathbb{E}_{q_n}(S_{t-1}^m S_t^m)$

On rappelle que les multiplications considérées ici le sont élément par élément. Il s'agit donc de calculer ces trois quantités afin de pouvoir les réinjecter dans M-STEP. Pour cela, on considère tout d'abord une méthode d'inférence *exacte*, deux méthodes d'inférences *variationnelles* et une méthode d'*échantillonnage*.

3 E-step

3.1 Inférence exacte

Pour présenter cet algorithme il est bon d'avoir à l'esprit les calculs effectués lors de la récurrence alpha-beta de HMM. Les calculs sont très semblables, néanmoins le problème est ici plus complexe puisque l'on n'a non pas une chaîne de Markov mais M. On pose :

$$\begin{cases} \alpha_t = p(S_t^1, \dots, S_t^M, Y_{(1,t)}|\theta) \\ \forall m \in [\![1,M]\!],\ \alpha_t^m = p(S_{t-1}^1, \dots, S_{t-1}^m, S_t^{m+1}, \dots, S_t^M, Y_{(1,t-1)}|\theta) \\ \beta_t = p(Y_{(t+1,T)}|S_t^1, \dots, S_t^M, \theta) \\ \forall m \in [\![1,M]\!],\ \beta_t^m = p(Y_{(t,T)}|S_t^1, \dots, S_t^m, S_{t-1}^{m+1}, \dots, S_{t-1}^M, \theta) \end{cases} \tag{1}$$

On remarque que :

$$\begin{cases} \forall t \in [\![1,T]\!],\ \alpha_{t-1} = \alpha_t^M \\ \forall t \in [\![0,T]\!],\ \beta_t = \beta_t^0 \end{cases} \tag{2}$$

Quatre autres équations permettent de compléter la récurrence :

$$\begin{cases} \alpha_t = p(Y_t|S_t^{(m)}, \theta)\alpha_t^0 \\ \forall m \in [\![0,M-1]\!],\ \alpha_t^m = \sum\limits_{S_{t-1}^{m+1}} p(S_t^{m+1}|S_{t-1}^{m+1}, \theta)\alpha_t^{m+1} \\ \beta_{t-1}^M = P(Y_t\ |S_t^{(m)}, \theta)\beta_t \\ \forall m \in [\![0,M-1]\!],\ \beta_t^m = \sum\limits_{S_{t+1}^{m+1}} p(S_{t+1}^{m+1}|S_t^{(m+1)}, \theta)\beta_t^{m+1} \end{cases} \tag{3}$$

On obtient le schéma suivant :

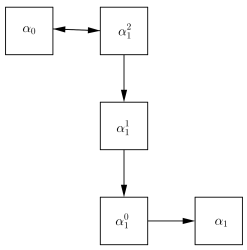

FIGURE 3 – Récurrence sur α

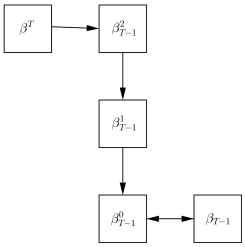

FIGURE 4 – Récurrence sur β

On peut alors calculer γ_t de la manière suivante :

$$\gamma_t = p(S_t|Y_{(t)}, \theta) = \frac{\alpha_t \beta_t}{\sum\limits_{S_t} \alpha_t \beta_t} \tag{4}$$

Il convient alors de calculer les trois quantités issues de M-STEP :
— $\mathbb{E}_{q_n}(S_t^m) = \sum\limits_{S_t^n,\ n \neq m} \gamma_t$
— $\mathbb{E}_{q_n}(S_t^{m_1} S_t^{m_2}) = \sum\limits_{S_t^n,\ n \neq m_1,\ n \neq m_2} \gamma_t$
— $\mathbb{E}_{q_n}(S_{t-1}^m S_t^m) = \frac{\sum\limits_{S_{t-1}^n, S_t^r,\ n \neq m,\ r \neq m} \alpha_{t-1} p(S_t|S_{t-1}) p(Y_t|S_t) \beta_t}{\sum\limits_{S_{t-1}, S_t} \alpha_{t-1} p(S_t|S_{t-1}) p(Y_t|S_t) \beta_t}$

Il est important de préciser que les formules données sont exactes seulement lorsqu'elles sont appliquées en un point. En effet, $\alpha_t, \beta_t, \gamma_t$ sont des éléments de $\mathbb{R}^{[\![1,K]\!]^M}$. Pour plus de détails sur le sens de ces formules on renvoie à l'implémentation en Python de cet algorithme https ://github.com/eliemichel/fHMM.

3.2 Échantillonnage de Gibbs

L'inférence exacte peut être très couteuse en termes d'opérations (avec une complexité temporelle en $\mathcal{O}(TMK^{M+1})$). L'algorithme d'échantillonnage de Gibbs fournit, lui, un moyen d'échantillonner approximativement selon $p(S_{(t)}^{(m)}|Y_{(t)},\theta)$ de manière rapide. En effet, l'échantillonnage de Gibbs peut être vu comme un cas particulier de *Monte Carlo Markov Chain*, pour laquelle la convergence en probabilité est assurée. Néanmoins, on ne possède pas d'information sur la vitesse de convergence de ces algorithmes vers la probabilité voulue.

Le choix des auteurs de [2] a été de ne pas prendre en compte le temps de mise en route de l'échantillonneur de Gibbs (« *burn-in period* », durant laquelle les échantillons ne sont pas supposés suivre la distribution de probabilité souhaitée) et de directement sélectionner les premiers échantillons produits par l'algorithme. Ce choix sera discuté par la suite.

On précise le déroulement de l'échantillonnage :

— les différents états $S_{(t)}^{(m)}$ sont initialisés selon une loi uniforme sur $[\![1,K]\!]$,

— les états $S_0^{(m)}$ sont mis à jour de manière successive (d'abord S_0^1 tiré selon $p(S_0^1|Y_0,S_1^1,S_0^{(-1)}$, puis S_0^2 tiré selon $p(S_0^2|Y_0,S_1^2,S_0^{(-2)})$),

— les états $S_t^{(m)}$ sont mis à jour de manière successive pour $t \in [\![1,T]\!]$,

— on extrait l'échantillon obtenu.

Les opérations sont répétées N_s fois à partir de la seconde étape pour obtenir autant d'échantillons de Gibbs.

On détaille simplement le calcul de la probabilité pour $t \in [\![1,T-1]\!]$ (les autres calculs sont similaires) :

$$\begin{aligned} p(S_t^m|Y_t,S_{(t)}^{(m)}\backslash S_t^m) &= \frac{1}{Z_1}p(S_t^m,S_{(-t)}^m,S_t^{(-m)},Y_t) \\ &= \frac{1}{Z_2}p(Y_t|S_t^m,S_t^{(-m)})p(S_t^m|S_{t-1}^m)p(S_{t+1}^m|S_t^m) \end{aligned} \tag{5}$$

La constante Z_2 est facilement calculable en sommant en notant que $\sum_{S_t^m} p(S_t^m|Y_t,S_{(t)}^{(m)}\backslash S_t^m) = 1$.

3.3 Approche variationnelle complètement factorisée

Un des problèmes de l'échantillonnage de Gibbs est l'absence de résultats concernant la vitesse de convergence des algorithmes MCMC. Celle-ci peut être très lente, comme on le verra dans 4. On présente ici une autre approche. On ne va pas chercher à calculer la probabilité $p(S_t^m|Y_{(t)})$ (résolution exacte de E-STEP) mais plutôt à l'approcher via une autre mesure de probabilité plus simple à calculer. En effet, on rappelle que l'algorithme EM procède à une descente coordonnée par coordonnée. La descente selon la mesure de probabilité est facile à calculer et donne $p(S_{(t)}^{(m)}|Y_{(t)})$. Imposons maintenant la contrainte que tous les états $S_{(t)}^{(m)}$ sont indépendants et reprenons l'étape E-STEP de EM. Ce procédé s'appelle MEAN-FIELD (ou approche variationnelle complètement factorisée). On a :

$$\begin{aligned} \log(p(Y_{(t)})) - \mathcal{L}(Y_{(t)},\theta,q) &= \mathbb{E}_q \log\left(\frac{p(S_{(t)}^{(m)},Y_{(t)})}{q(S_{(t)}^{(m)})}\right) \\ &= \mathbb{E}_q \log(p(Y_{(t)})) - \mathbb{E}_q \log\left(\frac{p(S_{(t)}^{(m)},Y_{(t)})}{q(S_{(t)}^{(m)})}\right) \\ &= \mathbb{E}_q \log\left(\frac{q(S_{(t)}^{(m)})}{p(S_{(t)}^{(m)}|Y_{(t)})}\right) \\ &= \mathrm{KL}(q\|\hat{q}) \end{aligned} \tag{6}$$

Où KL est la divergence de Kullback-Leiber. Cette quantité est positive et vaut zéro seulement si $q = \hat{q}$. On veut donc minimiser cette divergence.

Les détails sont fournis dans 6.4 mais on obtient :

$$\begin{cases} \widehat{\theta_0^m} \propto \exp\left({}^tW^m C^{-1} Y_0 - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_0^n - \frac{1}{2}\Delta^m + {}^t\log(A^m)\theta_1^m + \log(\Pi^m) \right) \\ \forall t \in [\![1, T-1]\!],\ \widehat{\theta_t^m} \propto \exp\left({}^tW^m C^{-1} Y_t - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_t^n - \frac{1}{2}\Delta^m + \log(A^m)\theta_{t-1}^m + {}^t\log(A^m)\theta_{t+1}^m \right) \\ \widehat{\theta_T^m} \propto \exp\left({}^tW^m C^{-1} Y_T - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_T^n - \frac{1}{2}\Delta^m + \log(A^m)\theta_{T-1}^m \right) \end{cases} \tag{7}$$

3.4 Approche variationnelle structurée

On présente enfin une dernière approche, l'approche variationnelle structurée qui permet de prendre en compte la structure de FACTORIAL HMM qui avait été perdue dans l'approche complètement factorisée. Ce nouveau modèle se nomme *structured mean field*. Au lieu de considérer les variables cachées toutes indépendantes on considère M chaînes de Markov indépendantes. Le modèle devient alors :

$$q(S_{(t)}^{(m)}) = \frac{1}{Z_q} \prod_{m=1}^{M} \prod_{k=1}^{K} \left(h_{1,k}^m \Pi_k^m\right)^{S_{0,k}^m} \prod_{t=1}^{T} \prod_{k_1=1}^{K} \left(h_{t,k_1}^m \prod_{k_2=1}^{K} \left(A_{k_2,k_1}^m\right)^{S_{t-1,k_2}^m} \right)^{S_{t,k_1}^m} \tag{8}$$

On remarque que cette probabilité peut très facilement s'écrire sous la forme d'une famille exponentielle de probabilités. On peut reprendre les calculs effectués précédemment et on obtient (beaucoup plus de détails sont fournis dans 6.5) :

$$\forall t \in [\![0, T]\!],\ \hat{h_t^m} = \exp\left({}^tW^m C^{-1} \left(Y_t - \sum_{n=1, n\neq m}^{M} W^n \right) - \frac{1}{2}\Delta^m \langle S_t^n \rangle \right) \tag{9}$$

Il reste à calculer les espérances via un algorithme de type *forward-backward* détaillé en appendice.

4 Résultats

On présente ici les résultats de l'implémentation de ces différents modèles. On va d'abord présenter l'évolution de la log-vraisemblance au cours de l'algorithme EM.

4.1 Évolution de la log-vraisemblance

4.1.1 Inférence exacte

Pour tester nos différentes implémentations on trace l'évolution de la log-vraisemblance au cours des itérations de EM. Celle-ci doit être strictement croissante. Les tests ont été effectués avec une mesure de probabilité initiale choisie aléatoirement, une matrice d'influence des variables cachées sur les variables observées choisie aléatoirement et une matrice de transition choisie aléatoirement. Le nombre d'états est fixé à $K = 2$ pour ce test. Le nombre de chaînes superposées est fixé à $M = 3$. L'espace des variables observées est de dimension $D = 2$ et le nombre d'observations est fixé à $T = 10$. Pour l'inférence exacte on teste deux possibilités. Dans un premier temps on se place dans le cadre HMM connu avec un espace d'états de taille K^M.

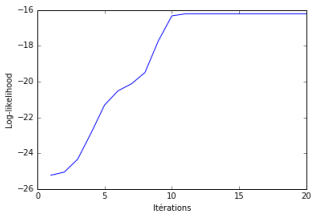

FIGURE 5 – Évolution de la log-vraisemblance pour inférence exacte HMM

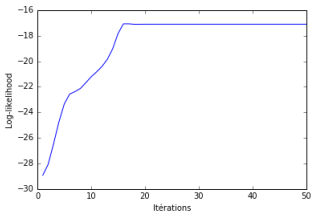

FIGURE 6 – Évolution de la log-vraisemblance pour inférence exacte FACTORIAL HMM

Dans les deux cas on observe une stricte croissance de la log-vraisemblance

4.1.2 Échantillonnage de Gibbs

Dans [2] les auteurs utilisent l'échantillonnage de Gibbs en ne considérant pas de période de *burn-in*, aucun échantillon n'est rejeté. Les auteurs justifient cette approche par leur volonté de comparer les approches variationnelles avec un échantillonnage de Gibbs même loin de la convergence (cet échantillonnage est appelé échantillonnage de Gibbs impatient). Les tests d'apprentissage sont toujours effectués avec les mêmes paramètres aléatoires. Ici on observe l'évolution de la log-vraisemblance avec l'approche décrite dans l'article [2]. L'étape de *burn-in* n'est pas prise en compte et aucun échantillon n'est rejeté. Ici le nombre d'échantillons est posé à $N_s = N_r = 10$ où N_r est le nombre d'échantillons créé et N_s le nombre d'échantillons conservé.

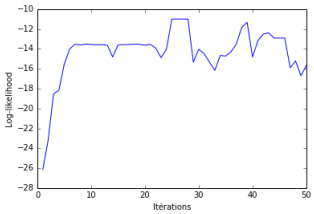

FIGURE 7 – Évolution de la log-vraisemblance pour l'échantillonneur de Gibbs impatient

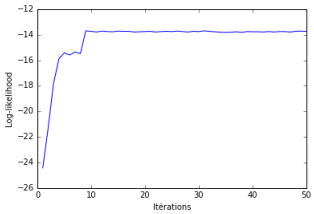

FIGURE 8 – Évolution de la log-vraisemblance pour l'échantillonneur de Gibbs

Bien qu'on observe une tendance croissante et une stabilisation de la log-vraisemblance on conserve une grande variabilité. Une solution pour tenter d'échantillonner de manière correcte et donc de mieux vérifier la propriété de croissance est d'introduire du *burn-in* dans l'échantillonnage de Gibbs. Ici, on a créé $N_r = 30$ échantillons et on en conserve $N_s = 10$.

On constate que les propriétés de croissance sont meilleures (même si des oscillations restent présentes). Néanmoins, on paye cher l'augmentation du nombre d'échantillons créés puisque l'algorithme devient bien plus lent que l'inférence exacte (avec notre nombre d'états et la taille de l'espace d'états, il est évident que si K et M sont plus grands cet échantillonnage restera plus rapide que l'inférence exacte). Or le but de l'échantillonnage de Gibbs est justement d'éviter d'avoir à passer par une inférence exacte assez lente pour des valeurs élevées de M et K.

4.1.3 Approches variationnelles

Comme décrit précédemment une autre approche du problème consiste à ne pas essayer d'échantillonner selon la mesure $p(S_{(t)}^{(m)}|Y_{(t)})$ mais selon une mesure de probabilité qui s'en approche soumise à des contraintes. La contrainte retenue dans le modèle MEAN-FIELD est l'indépendance entre les variables cachées. On rappelle brièvement qu'il convient ensuite de minimiser la divergence de Kullback-Lieber entre la probabilité $p(S_{(t)}^{(m)}|Y_{(t)})$ et les probabilités qui se factorisent dans ce nouveau graphe. La minimisation de cette divergence se fait de manière itérative en opérant une descente selon chaque coordonnée. Les auteurs de [2] remarque que la convergence est très rapide ($N_{it} < 10$ avec N_{it} le nombre d'itérations. On observe l'évolution suivante pour la log-vraisemblance avec $N_{it} = 5$.

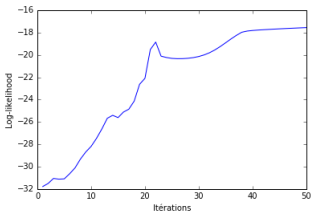

FIGURE 9 – Évolution de la log-vraisemblance pour MEAN-FIELD

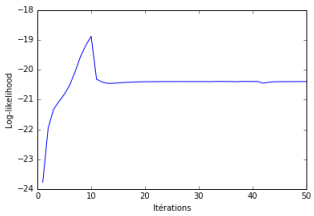

FIGURE 10 – Évolution de la log-vraisemblance pour STRUCTURED MEAN-FIELD

Le fait que cette log-vraisemblance ne soit pas strictement croissante est justifiée par le fait que la log-vraisemblance est ici une log-vraisemblance approchée puisque la mesure de probabilité considérée n'est plus $p(S_{(t)}^{(m)}|Y_{(t)})$.

On présente également les résultats obtenus pour le modèle STRUCTURED MEAN-FIELD. On rappelle que l'hypothèse faite sur les états cachés n'est plus une indépendance mais une structure de M chaînes de Markov superposées.

Il est à préciser qu'en appendice **??**, on étudie le comportement de notre algorithme EM aux temps longs.

4.2 Un critère : complexité et temps d'exécution

On a vu dans la partie précédente plusieurs possibilités pour calculer E-STEP. Il s'agit maintenant d'obtenir des critères pour décider quelle méthode employer. On peut d'abord s'intéresser à la complexité algorithmique de ces méthodes. Ces complexités sont récapitulées dans le tableau suivant.

Méthode	Complexité algorithmique
HMM- inférence exacte	$\mathcal{O}(TK^{2M})$
FACTORIAL HMM- inférence exacte	$\mathcal{O}(TK^{M+1})$
échantillonnage de Gibbs	$\mathcal{O}(TKMN_r)$
MEAN-FIELD	$\mathcal{O}(TK^2MN_{it})$
STRUCTURED MEAN-FIELD	$\mathcal{O}(TK^2MN_{it})$

Dans le tableau suivant on présente les résultats obtenus lors de nos tests sur des modèles génératifs avec paramètres d'entrée aléatoires et $M = 3$ et $K = 2$. On a posé $N_{it} = 5$, $N_r = 30$ et $N_s = 10$. On a toujours $T = 10$.

Méthode	Temps itération (s)
HMM- inférence exacte	0.453
FACTORIAL HMM- inférence exacte	0.429
échantillonnage de Gibbs	1.100
MEAN-FIELD	0.053
STRUCTURED MEAN-FIELD	0.319

Ces résultats correspondent à ceux obtenus dans [2], sauf pour l'inférence exacte où FACTORIAL HMMest plus rapide que HMM. Néanmoins vu les données de complexité énoncées plus haut il semble normal que FACTORIAL HMMsoit plus rapide que HMM. Il convient aussi de remarquer que les méthodes variationnelles possèdent l'avantage de fonctionner même si l'espace d'état est grand. Cela devient rapidement faux pour les méthodes d'inférence exacte et pour l'échantillonnage de Gibbs.

4.3 Estimation des paramètres

Dans cette sous-section on tente d'utiliser notre implémentation pour estimer les paramètres d'un modèle. Ici on se base sur un jeu de données généré avec un modèle de type FACTORIAL HMM avec les paramètres suivants :

— $K = 2$ et $M = 2$

— $W_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ et $W_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$

— $C = 0.025Id$

— mesure de probabilité initiale et matrice de passage aléatoire

Ainsi pour une observation de taille 2000 on obtient les données d'entrainement suivantes. Les différentes couleurs correspondent aux différentes configurations d'états cachés activés pour un état observé (on en a K^M, ici $2^2 = 4$).

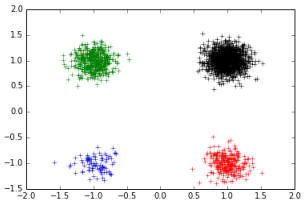

FIGURE 11 – An observation with $T = 2000$

En estimant les paramètres avec l'algorithme FACTORIAL HMM on obtient les résultats suivants concernant les moyennes :

	inférence exacte FACTORIAL HMM	STRUCTURED MEAN-FIELD	moyenne attendue
Moyenne 1	$(-0.996, -0.021)$	$(-0.996, -0.021)$	$(-1, 0)$
Moyenne 2	$(0.954, 0.036)$	$(0.954, 0.036)$	$(1, 0)$
Moyenne 3	$(0.032, 1, 012)$	$(0.032, 1, 012)$	$(0, 1)$
Moyenne 4	$(-0.070, -0.998)$	$(-0.07, -0.998)$	$(0, -1)$

On présente également les résultats obtenus avec HMM.

	inférence exacte HMM	moyenne attendue
Moyenne 1	$(-0.940, -1, 153)$	$(-1, -1)$
Moyenne 2	$(0.980, 0.959)$	$(1, 1)$
Moyenne 3	$(1.000, -1.052)$	$(1, -1)$
Moyenne 4	$(-0.923, 1.038)$	$(-1, 1)$

Les initialisations de ces algorithmes sont aléatoires sauf pour la matrice de covariance initiale fixée à $C = 0.025$. Le nombre d'itérations dans STRUCTURED MEAN-FIELD est fixé à 10. Le nombre d'itérations de EM est fixé à 20. Les résultats obtenus avec les deux algorithmes sont identiques et très proches de la réalité. Il est à préciser que c'est le cas uniquement car les données sont très concentrées (comme on peut l'observer sur le graphe). Lorsque le nombre d'états augmente et que les données sont moins concentrées l'estimation est beaucoup moins bonne.

Remarque : pour confirmer nos résultats on a voulu tester l'implémentation de Ghahramani http ://mlg.eng.cam.ac.uk/zoubin/software.html en Matlab. En le testant avec l'implémentation STRUCTURED MEAN-FIELD l'algorithme renvoyait une erreur (NaN)... On n'a pas eu le temps de se pencher en détail sur le code de Zoubin Ghahramani pour déterminer d'où venait l'erreur.

En appendice, on commente une expérience menée par les auteurs de [2], la réduction de l'*overfitting*.

5 Conclusion

En conclusion, on a dans ce rapport étudié mathématiquement l'algorithme EM ainsi que quatre méthodes pour calculer E-STEP. Il est à préciser que ces trois méthodes se partagent en trois catégories :

— inférence exacte
— méthode de Monte-Carlo
— méthodes variationnelles

On a détaillé les calculs et discuté les méthodes variationnelles et les résultats de convergence. Globalement, il est à noter que malgré des résultats théoriques assez faibles (l'algorithme EM peut converger vers un point stationnaire), celui-ci produit des résultats numériquement satisfaisants et possède l'immense avantage de pouvoir être exécuté en un temps raisonnable.

Dans nos expériences, nous nous sommes concentrés sur la vérification de la bonne convergence de nos algorithmes via l'étude de l'évolution de la log-vraisemblance et les discussions autour de cette valeur. Nous avons ensuite proposé une étude comparative des temps d'exécution et de la complexité de chacune

des implémentations de laquelle il ressort l'importance d'utiliser des méthodes variationnelles pour des espaces d'états de grande taille. Enfin, on a étudié l'estimation des paramètres via notre algorithme.

Nous comptons continuer de travailler sur notre implémentation (une version Python est disponible sur https ://github.com/eliemichel/fHMM) afin de produire une version demo facile à utiliser et réduire le temps de calcul sur certaines fonctions codées naïvement. Le but sera alors de tester notre algorithme sur des données réelles afin de vérifier les propriétés de réduction de l'*overfitting*.

6 Appendices

6.1 Notations

On reporte dans le Tableau 1 les notations qui seront utilisées dans toute la suite de notre rapport.

$T \in \mathbb{N}$	temps de la dernière observation
$M \in \mathbb{N}^*$	nombre de chaînes de Markov
$\Delta_K = \{X \in \{0,1\}^K,\ \sum_{k=1}^{K} X_k = 1\}$	espace d'état des variables cachées
$D \in \mathbb{N}^*$	dimension des variables observées
$(\Pi_k^m)_{(m,k)\in[\![1,M]\!]\times[\![1,K]\!]} \in [0,1]^{MK}$	mesures de probabilité initiales pour chaque chaîne
$(A^m_{k_1,k_2})_{(m,k_1,k_2)\in[\![1,M]\!]\times[\![1,K]\!]^2} \in [0,1]^{MK^2}$	matrices de transition pour chaque chaîne
$C \in \mathcal{S}_D^{++}(\mathbb{R})$	matrice de covariance
$(W^m_{x,k})_{(m,x,k)\in[\![1,M]\!]\times[\![1,D]\!]\times[\![1,K]\!]} \in \mathbb{R}^{MDK}$	matrices d'influence des variables cachées sur les variables observées
$(S_t^m)_{(m,t)\in[\![1,M]\!]\times[\![0,T]\!]} \in \Delta_K^{(T+1)\times M}$	variables cachées
$(Y_t)_{t\in[\![0,T]\!]} \in \mathbb{R}^D$	variables observées

TABLE 1 – Notations

Les différentes contraintes (sommation à 1) sur les mesures de probabilité n'ont pas été rappelées ici. Pour plus de simplicité on notera $Y_{(t)}$ le vecteur $(Y_t)_{t\in[\![0,T]\!]}$, $Y_{(-t)}$ le vecteur $(Y_t)_{t\in[\![0,T]\!]\setminus\{t\}}$, $Y_{(t_1,t_2)}$ le vecteur $(Y_t)_{t\in[\![t_1,t_2]\!]}$. Ces notations s'étendent aux variables cachées. Enfin on notera $\theta = (\Pi, A, C, W)$ le vecteur de paramètres. Si il n'y pas d'ambiguïté la dépendance par rapport à θ est sous-entendue.

6.2 Algorithme EM

Soit p une mesure de probabilité qui se factorise selon FACTORIAL HMM. On a alors :

$$p(S_{(t)}^{(m)}, Y_{(t)}|\theta) = \prod_{m=1}^{M}\prod_{k=1}^{K}(\Pi_k^m)^{(S_0^m)_k}\prod_{t=1}^{T}\prod_{m=1}^{M}\prod_{k_1=1}^{K}\prod_{k_2=1}^{K}(A^m_{k_1,k_2})^{(S_t^m)_{k_2}(S_{t-1}^m)_{k_1}}$$
$$\prod_{t=0}^{T}\frac{1}{(2\pi)^{D/2}}\frac{1}{|C|^{1/2}}\exp\left(-1/2^t\left(Y_t - \sum_{m=1}^{M}W^m S_t^m\right)C^{-1}\left(Y_t - \sum_{m=1}^{M}W^m S_t^m\right)\right) \quad (10)$$

De manière analogue au cas HMM ou GAUSSIAN MIXTURE il n'est pas facile de calculer $p(Y_{(t)})$. L'algorithme EM est une méthode itérative qui permet de considérer seulement la probabilité jointe 10 dont la forme est facile à manipuler. L'idée est la suivante :

$$\begin{aligned} p(Y_{(t)}) &= \int_{S_{(t)}^{(m)}} p(S_{(t)}^{(m)}, Y_{(t)}) \\ &= \int_{S_{(t)}^{(m)}} \frac{p(S_{(t)}^{(m)}, Y_{(t)})}{q(S_{(t)}^{(m)})} q(S_{(t)}^{(m)}) \\ &= \mathbb{E}_q\left(p(S_{(t)}^{(m)}, Y_{(t)})\right) \end{aligned} \quad (11)$$

Si on considère la log-vraisemblance on peut utiliser l'inégalité de Jensen et on obtient :

$$\begin{aligned} l(Y_{(t)},\theta) &= \log(p(Y_{(t)}) \\ &= \log\left(\mathbb{E}_q\left(\frac{p(S_{(t)}^{(m)},Y_{(t)})}{q(S_{(t)}^{(m)})}\right)\right) \quad \geq \mathcal{L}(Y_{(t)},\theta,q) \end{aligned} \tag{12}$$

Avec $\mathcal{L}(Y_{(t)},\theta,q) = \mathbb{E}_q \log\left(p(S_{(t)}^{(m)},Y_{(t)})\right) + H(q)$ où $H(q)$ est l'entropie de q. Il s'agit alors de maximiser cette log-vraisemblance approchée en q et en θ. La maximisation à θ fixé implique

$$\hat{q} = p(S_{(t)}^{(m)}|\theta, Y_{(t)}^{(m)}) \tag{13}$$

Une autre conséquence est l'égalité suivante :

$$\mathcal{L}(Y_{(t)},\theta,\hat{q}) = l(Y_{(t)},\theta) \tag{14}$$

L'algorithme consiste à appliquer un algorithme de Gauss-Seidel avec contraintes sur la log-vraisemblance approchée, les deux variables considérées étant q [2] et θ. On distingue ainsi deux étapes :

— E-STEP : $q_{n+1} = p(S_{(t)}^{(m)}|\theta_n, Y_{(t)}^{(m)})$

— M-STEP : maximisation de $\mathbb{E}_{q_n}\left(\log\left(p(S_{(t)}^{(m)},Y_{(t)}|\theta)\right)\right)$

Les conditions initiales n'ont pas été discutées ici. Néanmoins elles jouent un rôle crucial dans cet algorithme au comportement très local. En effet, très peu de résultats assurent la convergence théorique de cet algorithme vers un maximum local. Dans [4], les auteurs montrent que sous certaines conditions on a l'assurance de la convergence vers un point stationnaire. Les conditions de maximalité sont plus dures à obtenir. Il est cependant important de remarquer que 14 assure une croissance de la log-vraisemblance à chaque étape de l'algorithme.

6.3 M-step

On détaille désormais l'étape M-STEP de notre algorithme. On rappelle qu'il s'agit de maximiser $\mathcal{L}_{approx}(\theta) = \mathbb{E}_{q_n}\left(\log\left(p(S_{(t)}^{(m)},Y_{(t)}|\theta)\right)\right)$. La maximisation selon $(\Pi_k^m)_{(m,k)\in[\![1,M]\!]\times[\![1,K]\!]}$ (respectivement selon $(A_{k_1,k_2}^m)_{(m,k_1,k_2)\in[\![1,M]\!]\times[\![1,K]\!]^2}$) pouvant se faire indépendamment des autres variables, on obtient après utilisation des multiplicateurs de Lagrange [3] :

$$\widehat{\Pi_k^m} = q_n((S_0^m)_k = 1) \tag{15}$$

De même on obtient :

$$\widehat{A_{k_1,k_2}^m} = \frac{\sum_{t=1}^{T} q_n((S_{t-1}^m)_{k_1} = 1, (S_t^m)_{k_2} = 1)}{\sum_{t=1}^{T} q_n((S_{t-1}^m)_{k_1} = 1)} \tag{16}$$

Pour la suite, on note $W \in \mathcal{M}_{DM,K}(\mathbb{R})$ la matrice obtenue en concaténant les matrices $(W_m)_{m\in[\![1,M]\!]}$. De la même manière on note S_t le vecteur obtenu en concaténant tous les vecteurs $(S_t^m)_{m\in[\![1,M]\!]}$. Il s'agit alors de maximiser en (W,C) la fonction suivante

$$\mathcal{L}'(W,C) = \mathbb{E}_{q_n}\left(-\frac{T+1}{2}\log(|C|) - \frac{1}{2}{}^t(Y_t - WS_t)\,C^{-1}\,(Y_t - WS_t)\right)$$

. Cette étude est classique (maximisation des paramètres pour la log-vraisemblance d'une gaussienne) et les calculs ont déjà été effectués à plusieurs reprises. On rappelle simplement que la maximisation se fait d'abord sur W, l'argument maximum ne dépendant pas de C on peut facilement réinjecter $\widehat{W}$ dans $\mathcal{L}'(\widehat{W},C)$ et optimiser selon C [4]. On obtient les expressions suivantes :

2. q peut être vue comme une mesure de probabilité mais d'un point de vue optimisation on considère q comme une variable de $[0,1]^{K^M}$

3. Le fait que l'on travaille sur des ouverts, i.e Π_k^m et A_{k_1,k_2}^m ne s'annulent jamais, n'est pas détaillé ici. Cette propriété est issue de la divergence du logarithme en 0

4. En réalité on optimise selon C^{-1} qui donne une expression plus facile à manipuler

$$\widehat{W} = \left(\sum_{t=0}^{T} Y_t \times \mathbb{E}_{q_n}({}^tS_t)\right)\left(\sum_{t=0}^{T}\mathbb{E}_{q_n}\left(S_t\,{}^tS_t\right)\right)^{+} \tag{17}$$

Où M^+ est la pseudo-inverse de Moore-Penrose. On a ici utilisé le fait que la pseudo-inverse de Moore-Penrose donne une solution au problème des moindres carrés [1]. La maximisation selon C^{-1} donne :

$$\widehat{C} = \frac{1}{T+1}\sum_{t=0}^{T} Y_t\,{}^tY_t - \frac{1}{T+1}\sum_{t=0}^{T}\sum_{m=1}^{M} W^m\mathbb{E}_{q_n}(S_t^m)\,{}^tY_t \tag{18}$$

6.4 Approche variationnelle complètement factorisée

FIGURE 12 – Modèle sous-jacent pour les états cachés

Sans contrainte on retrouve $\hat{q}$ (E-STEP exacte). Maintenant ajoutons la contrainte que tous les états sont indépendants. q est alors de la forme :

$$q(S_{(t)}^{(m)}) = \prod_{t=0}^{T}\prod_{m=1}^{M}\prod_{k=1}^{K}\left(\theta_{t,k}^m\right)^{S_{t,k}^m} \tag{19}$$

Avec :

$$\forall (t,m) \in [\![0,T]\!] \times [\![1,M]\!],\ \sum_{k=1}^{K}\theta_{t,k}^m = 1 \tag{20}$$

On va alors opérer une descente de gradient coordonnée par coordonnée. La justification de ce choix est expliquée dans [3] et détaillée dans 6.6. La divergence de Kullback-Leiber est de la forme suivante (on note $\theta_t^m = (\theta_{t,k}^m)_{k\in[\![1,K]\!]}$ et la fonction logarithme sur un vecteur correspond au logarithme sur chacune des coordonnées) :

$$\begin{aligned}\mathrm{KL}(q\|\hat{q}) = &\sum_{t=0}^{T}\sum_{m=1}^{M} {}^t\theta_t^m\log(\theta_t^m) + \\ \frac{1}{2}\sum_{t=0}^{T}\Bigg(&{}^tY_tC^{-1}Y_t - 2\sum_{m=1}^{M}{}^tY_tC^{-1}W^m\theta_t^m + \sum_{m=1}^{M}\sum_{n=1,n\neq m}^{M}\mathrm{Tr}\left({}^tW^mC^{-1}W^n\theta_t^n{}^t\theta_t^m\right) + \sum_{m=1}^{M}\mathrm{Tr}\left({}^tW^mC^{-1}W^m\mathrm{diag}(\theta_t^m)\right)\Bigg) \\ &- \sum_{m=1}^{M}{}^t\theta_1^m\log(\Pi^m) - \sum_{t=1}^{T}\sum_{m=1}^{M}\mathrm{Tr}\left({}^t\theta_{t-1}^m\theta_t^m\log(A^m)\right) - \log(Z_q) - \log(Z_{\hat{q}})\end{aligned} \tag{21}$$

Où diag est l'opérateur qui a un vecteur associe la matrice dont les coefficients diagonaux sont les composantes du vecteur et Z_q et $Z_{\hat{q}}$ les facteurs de normalisation des probabilités q et $\hat{q}$. On dérive par rapport à θ_t^m et on obtient :

$$\frac{\partial KL}{\partial\theta_t^m} = \log(\theta_t^m) - {}^tW^mC^{-1}Y_t + \sum_{n=1,n\neq m}^{m}{}^tW^mC^{-1}W^n\theta_t^n + \frac{1}{2}\Delta^m - \log(A^m)\theta_{t-1}^m - {}^t\log(A^m)\theta_{t+1}^m + c \tag{22}$$

Où Δ^m est le vecteur des coefficients diagonaux de ${}^TW^mC^{-1}W^m$ et c un vecteur colinéaire à $(1,\ldots,1)$. Il est à noter que cette équation n'est valable que pour $t \in [\![1,T-1]\!]$. En égalant à 0 et en isolant θ_t^m on

obtient :

$$\begin{cases} \widehat{\theta_0^m} \propto \exp\left({}^tW^m C^{-1} Y_0 - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_0^n - \frac{1}{2}\Delta^m + {}^t\log(A^m)\theta_1^m + \log(\Pi^m) \right) \\ \forall t \in [\![1, T-1]\!],\ \widehat{\theta_t^m} \propto \exp\left({}^tW^m C^{-1} Y_t - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_t^n - \frac{1}{2}\Delta^m + \log(A^m)\theta_{t-1}^m + {}^t\log(A^m)\theta_{t+1}^m \right) \\ \widehat{\theta_T^m} \propto \exp\left({}^tW^m C^{-1} Y_T - \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \theta_T^n - \frac{1}{2}\Delta^m + \log(A^m)\theta_{T-1}^m \right) \end{cases} \tag{23}$$

Il convient de remarquer que ces équations sont des relations de proportionnalité et non de véritables égalités. Ici on applique l'algorithme de descente coordonnée par coordonnée sous contrainte. On projette donc à chaque étape de minimisation sur la sous-variété définie par les contraintes. Autrement dit, on fixe la constante de proportionnalité de telle sorte à ce que θ_t^m soit une mesure de probabilité.

Notons en outre que les nombres que l'on manipule peuvent être très petits (proches de la précision machine). Pour éviter ce problème on effectue tous les calculs en considérant le logarithme de θ_t^m. On passe à l'exponentielle pour la dernière étape seulement.

6.5 Approche variationnelle structurée

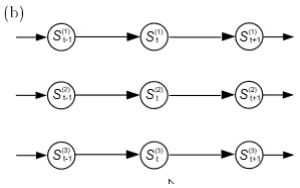

FIGURE 13 – Modèle sous-jacent pour les états cachés

$$\begin{aligned} \mathrm{KL}(q\|\hat{q}) = \sum_{t=0}^{T}\sum_{m=1}^{M} \langle S_t^m \rangle \log(h_t^m) + \frac{1}{2}\sum_{t=0}^{T} \Bigg({}^tY_t C^{-1} Y_t - 2\sum_{m=1}^{M} {}^tY_t C^{-1} W^m \langle S_t^m \rangle + \sum_{m=1}^{M}\sum_{n=1, n\neq m}^{M} \mathrm{Tr}\left({}^tW^m C^{-1} W^n \langle S_t^n \rangle {}^t\langle S_t^m \rangle \right) \\ + \sum_{m=1}^{M} \mathrm{Tr}\left({}^tW^m C^{-1} W^n \mathrm{diag}(\langle S_t^m \rangle) \right) \Bigg) - \log(Z_q) - \log(Z) \end{aligned} \tag{24}$$

Il est important de signaler que les espérances calculées dans les crochets le sont vis à vis de la mesure de probabilité q. Puisqu'on a une famille exponentielle on sait que :

$$\frac{\partial \log(Z_q)}{\partial \log h_t^m} = \langle S_t^m \rangle \tag{25}$$

Donc l'étape de dérivation par rapport à $\log(h_\tau)$ donne :

$$\frac{\partial \mathrm{KL}}{\partial \log(h_\tau^l)} = \langle S_t^l \rangle + \sum_{t=0}^{T}\sum_{m=1}^{M} \left(\log(h_t^m) - {}^tW^m C^{-1} Y_t + \sum_{n=1, n\neq m}^{M} {}^tW^m C^{-1} W^n \langle S_t^n \rangle + \frac{1}{2}\Delta^m \right) \frac{\partial \langle S_t^m \rangle}{\partial \log(h_\tau^m)} - \langle S_\tau^l \rangle \tag{26}$$

Où Δ^m est défini comme dans le modèle complètement factorisé. On obtient donc, en annulant le terme à l'intérieur de la somme :

$$\forall t \in [\![0, T]\!],\ \hat{h_t^m} = \exp\left({}^tW^m C^{-1} \left(Y_t - \sum_{n=1, n\neq m}^{M} W^n \right) - \frac{1}{2}\Delta^m \langle S_t^n \rangle \right) \tag{27}$$

Formulons quelques remarques. Dans l'approche MEAN-FIELD, les différentes quantités étaient approchées coordonnée par coordonnée. Ici, ce couplage est caché dans les espérances $\langle S_t^n \rangle$ qui doivent être calculées

avec un algorithme de type *message-passing*. Dans le modèle précédent ces quantités étaient très simples à calculer et les interdépendances apparaissaient via les matrices de transition. Il est à préciser qu'ici les quantités A^m et Π^m n'apparaissent plus. Cela est dû à la forme du modèle. On donne les formes des messages à faire passer durant l'algorithme en commençant par les messages *forward* :

$$\begin{cases} \mu^m_{0\to 1}(S^m_1) = \sum_{S^m_0} h_{0,S^m_0} \Pi^m_{S^m_0} A^m(S^m_0, S^m_1) \\ \forall t \in [\![1, T-1]\!],\ \mu^m_{t\to t+1}(S^m_{t+1}) = \sum_{S^m_t} h_{t,S^m_t} A^m_{S^m_t, S^m_{t+1}} \mu_{t-1\to t}(S^m_t) \end{cases} \tag{28}$$

On donne ensuite la forme des messages *backward* :

$$\begin{cases} \mu^m_{T\to T-1}(S^m_{T-1}) = \sum_{S^m_T} h_{T,S^m_T} A^m(S^m_{T-1}, S^m_T) \\ \forall t \in [\![1, T-1]\!],\ \mu^m_{t\to t-1}(S^m_{t-1}) = \sum_{S^m_t} h_{t,S^m_t} A^m_{S^m_{t-1}, S^m_t} \mu_{t+1\to t}(S^m_t) \end{cases} \tag{29}$$

6.6 Quelques remarques sur les modèles variationnels

Les deux derniers modèles introduisent une étape itérative pour trouver une mesure de probabilité proche la postérieure sous contrainte. L'algorithme utilisé est une mise à jour coordonnée par coordonnée. Or, l'algorithme de descente coordonnée par coordonnée n'est assuré de fonctionner que dans le cas d'une fonction objectif convexe. Ainsi, on peut se retrouver bloquer dans un maximum local. Ce problème de non- convexité s'observe dans le cadre de FACTORIAL HMMpour les deux approches variationnelles. On détaille le cas MEAN-FIELDoù il est plus facile à identifier.

On remarque tout d'abord qu'en considérant 21 on constate que la fonction obtenue en fixant $(\theta_{(-t)})^{(-m)}$ et en ne considérant que les variations en θ^m_t est concave. Ainsi, pour maximiser, il est bien justifié d'annuler la dérivée. Cependant 21 n'est pas concave en $(\theta_{(t)})^{(m)}$. En effet, il comporte le terme suivant :

$$\sum_{t=0}^{T} \sum_{m=1}^{M} \sum_{n\neq m} \mathrm{Tr}\left({}^t W^m C^{-1} W^n \theta^{nt}_t \theta^m_t\right) \tag{30}$$

Celui-ci n'est pas concave et donc on n'a pas assurance de la convergence de l'algorithme coordonnée par coordonnée vers le maximum global. L'interprétation est plus compliquée pour le modèle STRUCTURED MEAN-FIELD. Néanmoins, malgré ces problèmes de convergence vers un point stationnaire, les modèles variationnels sont très utilisés car ils fournissent de bonnes bornes inférieures.

On peut formuler une remarque complémentaire pour le modèle STRUCTURED MEAN-FIELD. En effet, on a annulé une somme en considérant que chacun des termes était nul. Malheureusement cela peut ne pas être le cas. En effet, le terme :

$$\left(\log(h^m_t) - {}^t W^m C^{-1} Y_t + \sum_{n=1, n\neq m}^{M} {}^t W^m C^{-1} W^n \langle S^n_t \rangle + \frac{1}{2}\Delta^m\right) \frac{\partial \langle S^m_t \rangle}{\partial \log(h^m_\tau)} \tag{31}$$

n'est pas nécessairement toujours positif ou toujours négatif et des compensations peuvent se produire. Ainsi, dans le modèle STRUCTURED MEAN-FIELD on n'est pas assuré de la bonne maximisation selon chaque coordonnée. Néanmoins, cette formulation de l'algorithme est soutenue par l'intuition qui nous permet d'aboutir à une forme close proche de celle obtenue dans le cas MEAN-FIELD.

6.7 Comportement aux temps longs

Dans les études précédentes ainsi que dans [2] les études sont menées sur de courtes chaines ($T = 20$). On illustre ici pour le cas $T = 100$ les résultats obtenus via l'inférence exacte.

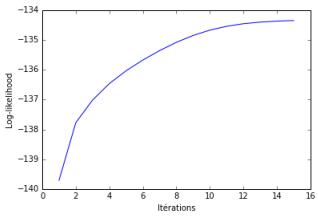

FIGURE 14 – $M = 2$, $K = 2$, $T = 100$, inférence exacte

Il est à préciser que puisque le nombre de paramètres des modèles variationnels augmente linéairement avec T, il est naturel de devoir augmenter les paramètres itératifs pour s'approcher de la convergence. Les résultats empiriques obtenus suggèrent que, même pour $T = 100$, on observe une stabilisation de la log-vraisemblance pour $N_{it} < 100$.

6.8 Overfitting

On commente rapidement une propriété de FACTORIAL HMM : la réduction de l'*overfitting*. Il est à préciser que nous n'avons pas pu reproduire les résultats énoncés dans l'article [2] faute de temps et puissance de calcul. Néanmoins, on évoque cette propriété en s'appuyant sur le graphe suivant que l'on peut retrouver dans [2].

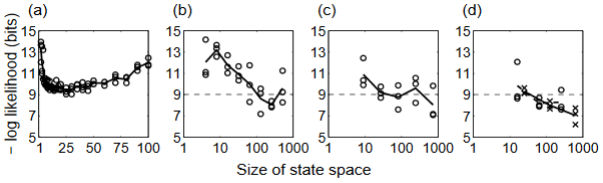

FIGURE 15 – Réduction de l'*overfitting*

La première courbe représente l'évolution de la log-vraisemblance sur les données de test après entraînement sur des données moyenne pour HMM en fonction du nombre d'états. Le même graphique est reproduit pour $K = 2$ (deuxième graphe), $K = 3$ (troisième graphe) et $K = 4$ (quatrième graphe). La propriété à observer ici est l'*overfitting*. En effet, lorsque le nombre d'états augmentent l'estimation se fait plus précise sur l'ensemble d'entraînement, mais il arrive un moment où cette estimation "colle" trop bien aux données d'entraînement et oublie la liberté inhérente au modèle. Dans ce cas, la log-vraisemblance augmente de nouveau. On obtient donc une courbe en cloche (ici retournée car on considère la log-vraisemblance multipliée par -1). Il est à noter que ce phénomène est très visible sur HMM et disparaît progressivement sur FACTORIAL HMM lorsque K augmente.

Une interprétation possible est qu'il y a beaucoup plus de paramètres déterminés par HMM (K^M moyennes) que par FACTORIAL HMM (KM moyennes). Ainsi le modèle FACTORIAL HMM conserve plus de liberté que le modèle HMM.

Références

[1] Adi Ben-Israel and Thomas NE Greville. *Generalized inverses : theory and applications*, volume 15. Springer Science & Business Media, 2003.

[2] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3) :245–273, 1997.

[3] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2) :1–305, 2008.

[4] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.