L'OEP pour le problème de l'apprentissage de Modèles de Markov Cachés

Adaptation, hybridation et comparaison

Sébastien Aupetit, Nicolas Monmarché et Mohamed Slimane

Equipe Handicap et Nouvelles Technologies (HaNT)
Laboratoire d'Informatique
Université François Rabelais de Tours

Plan

Les Modèles de Markov Cachés (MMC)

Les Modèles de Markov Cachés (MMC)

- Forme la plus simple de MMC : discret et en temps discret
- Outil mathématique permettant de modéliser une série temporelle par deux processus stochastiques :
  - un processus caché :
    - modélise les états (cachés) du système
    - c'est une chaîne de Markov
      L'apparition d'un état ne dépend que de l'état précédant
  - un processus observé :
    - modélise l'émission de symboles observés
    - dépendant du processus caché
      L'émission d'un symbole ne dépend que de l'état courant
- Paramètres :
  - Π : probabilités initiales des états cachés
  - A : probabilités de transition entre états cachés
  - B : probabilités d'émission des symboles par rapport aux états cachés

Les Modèles de Markov Cachés (MMC)

- Forme la plus simple de MMC : discret et en temps discret
- Outil mathématique permettant de modéliser une série temporelle par deux processus stochastiques :
  - un processus caché :
    - modélise les états (cachés) du système
    - c'est une chaîne de Markov
      L'apparition d'un état ne dépend que de l'état précédant
  - un processus observé :
    - modélise l'émission de symboles observés
    - dépendant du processus caché
      L'émission d'un symbole ne dépend que de l'état courant
- Paramètres :
  - Π : probabilités initiales des états cachés
  - A : probabilités de transition entre états cachés
  - B : probabilités d'émission des symboles par rapport aux états cachés

Les Modèles de Markov Cachés (MMC)

- Forme la plus simple de MMC : discret et en temps discret
- Outil mathématique permettant de modéliser une série temporelle par deux processus stochastiques :
  - un processus caché :
    - modélise les états (cachés) du système
    - c'est une chaîne de Markov
      L'apparition d'un état ne dépend que de l'état précédant
  - un processus observé :
    - modélise l'émission de symboles observés
    - dépendant du processus caché
      L'émission d'un symbole ne dépend que de l'état courant
- Paramètres :
  - Π : probabilités initiales des états cachés
  - A : probabilités de transition entre états cachés
  - B : probabilités d'émission des symboles par rapport aux états cachés

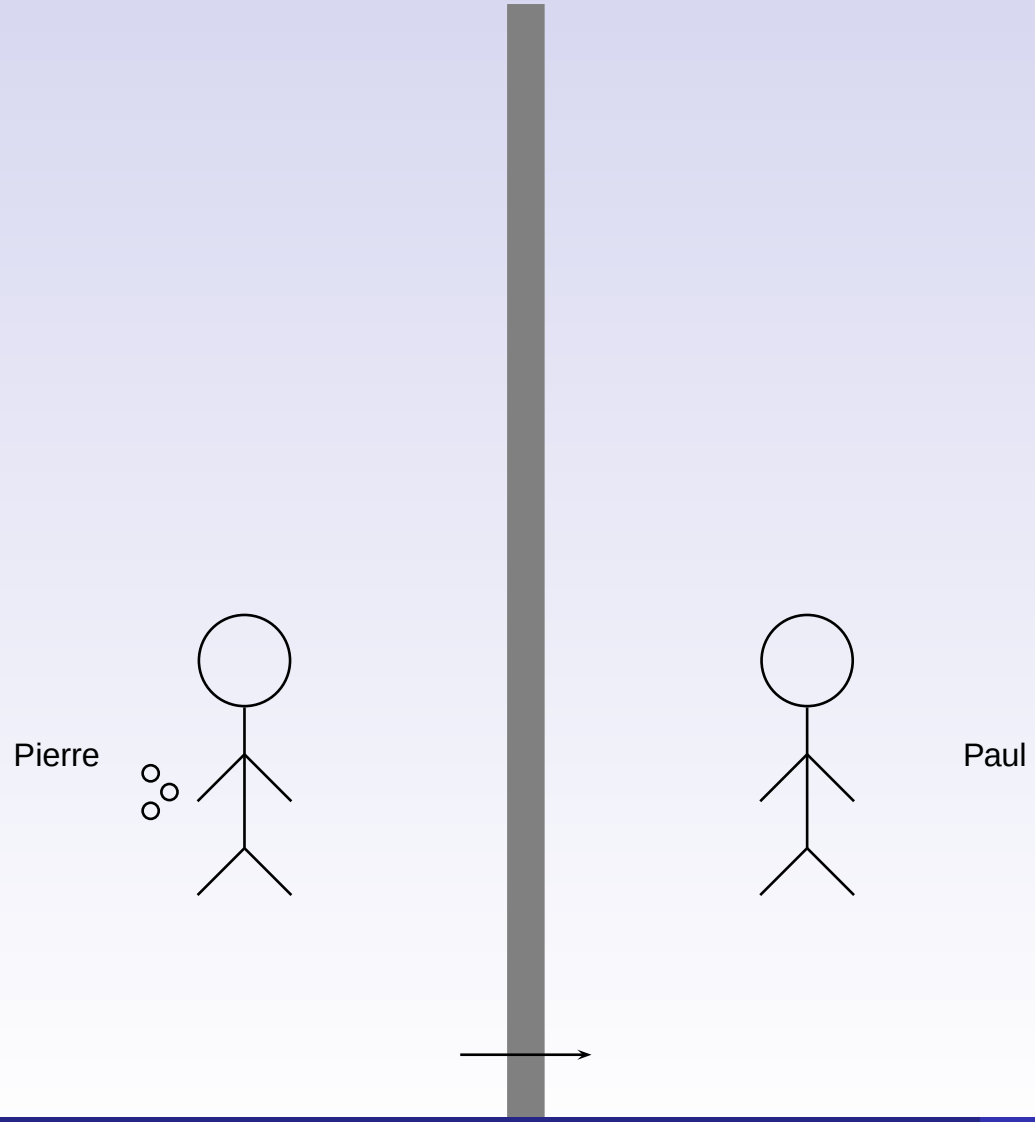
Les Modèles de Markov Cachés (MMC)

- Forme la plus simple de MMC : discret et en temps discret
- Outil mathématique permettant de modéliser une série temporelle par deux processus stochastiques :
  - un processus caché :
    - modélise les états (cachés) du système
    - c'est une chaîne de Markov
      L'apparition d'un état ne dépend que de l'état précédant
  - un processus observé :
    - modélise l'émission de symboles observés
    - dépendant du processus caché
      L'émission d'un symbole ne dépend que de l'état courant
- Paramètres :
  - Π : probabilités initiales des états cachés
  - A : probabilités de transition entre états cachés
  - B : probabilités d'émission des symboles par rapport aux états cachés

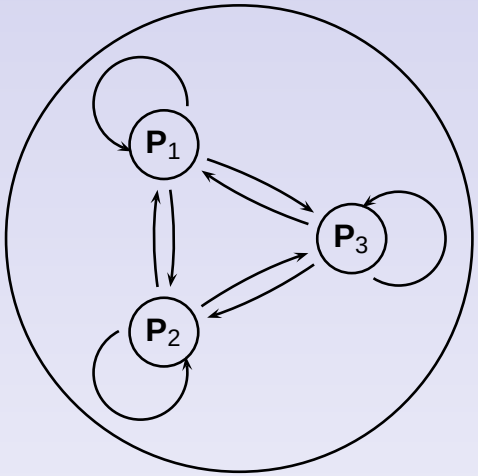
Les Modèles de Markov Cachés (MMC)

- Forme la plus simple de MMC : discret et en temps discret
- Outil mathématique permettant de modéliser une série temporelle par deux processus stochastiques :
  - un processus caché :
    - modélise les états (cachés) du système
    - c'est une chaîne de Markov
      L'apparition d'un état ne dépend que de l'état précédant
  - un processus observé :
    - modélise l'émission de symboles observés
    - dépendant du processus caché
      L'émission d'un symbole ne dépend que de l'état courant
- Paramètres :
  - Π : probabilités initiales des états cachés
  - A : probabilités de transition entre états cachés
  - B : probabilités d'émission des symboles par rapport aux états cachés

Exemple

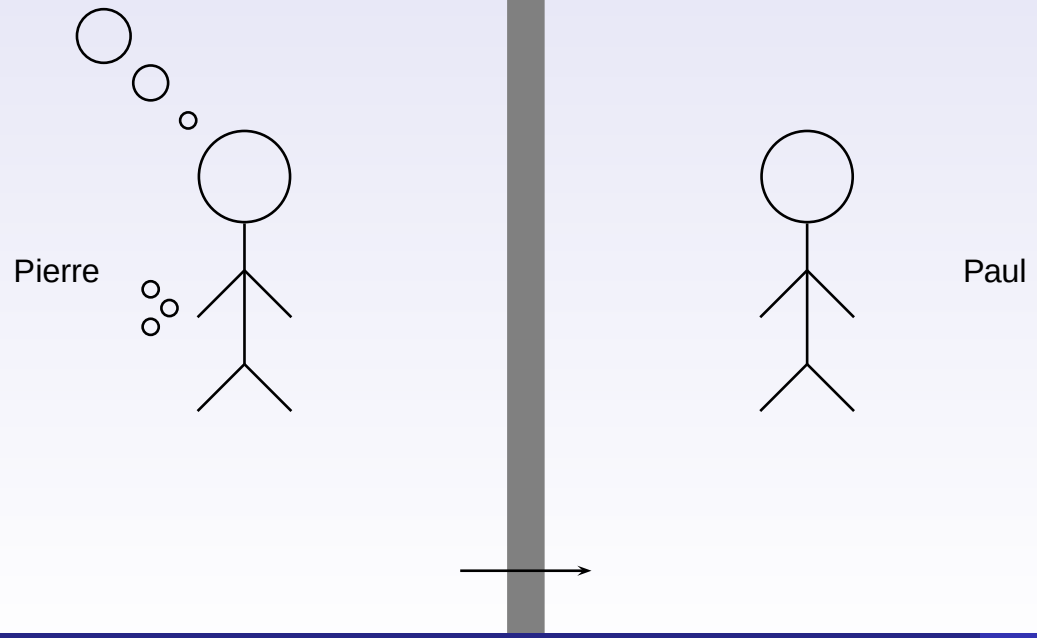


Exemple

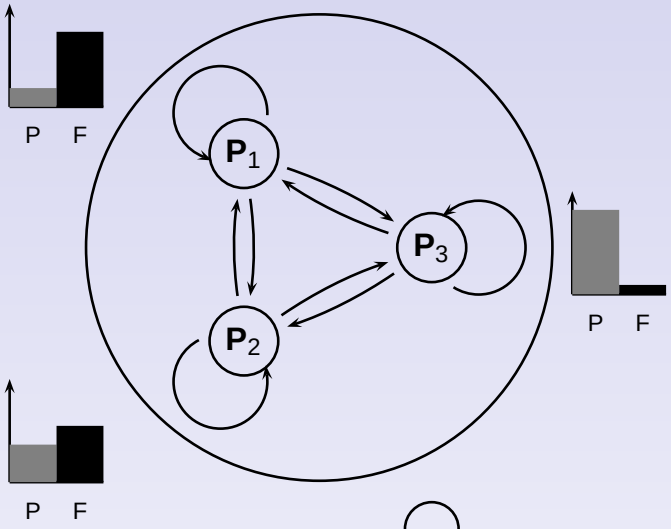


$$\Pi = (0.2\ 0.4\ 0.4)$$

$$A = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$



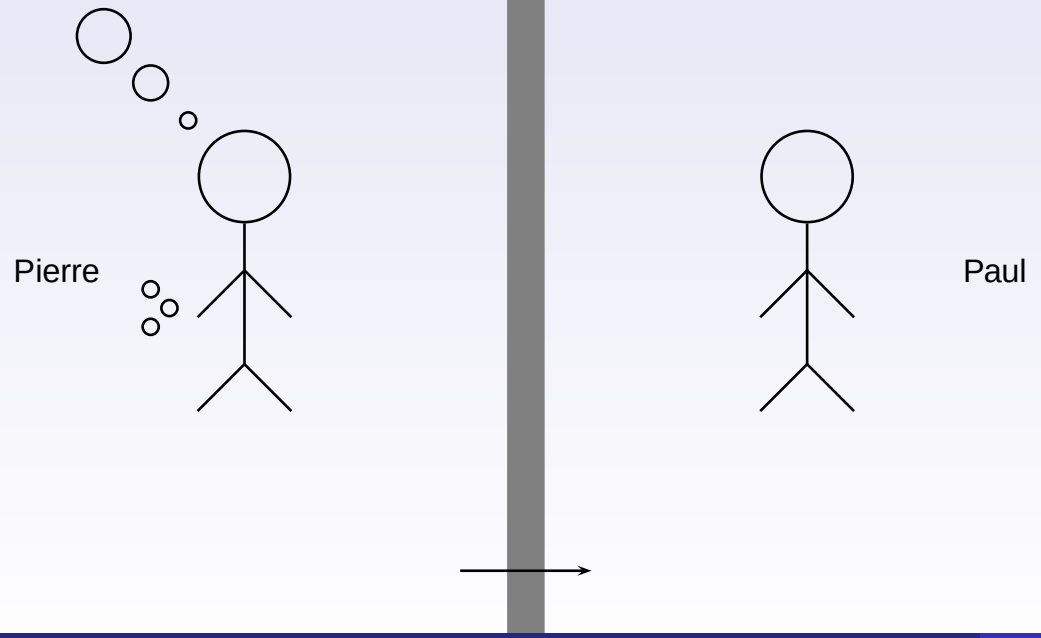
Exemple



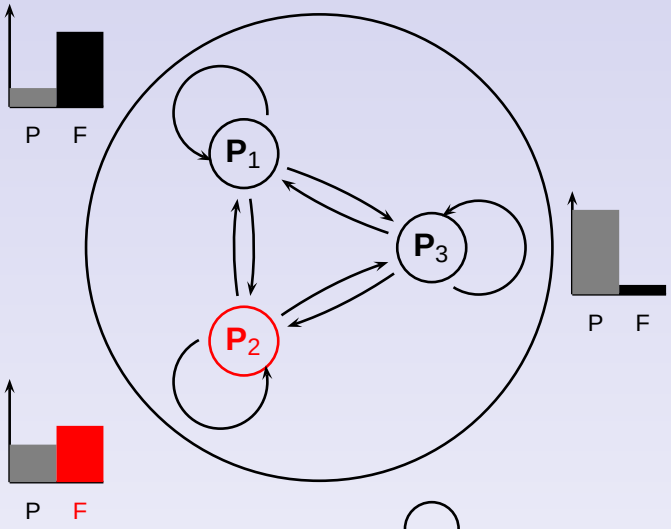
$$\Pi = (0.2\ 0.4\ 0.4)$$

$$A = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix}$$



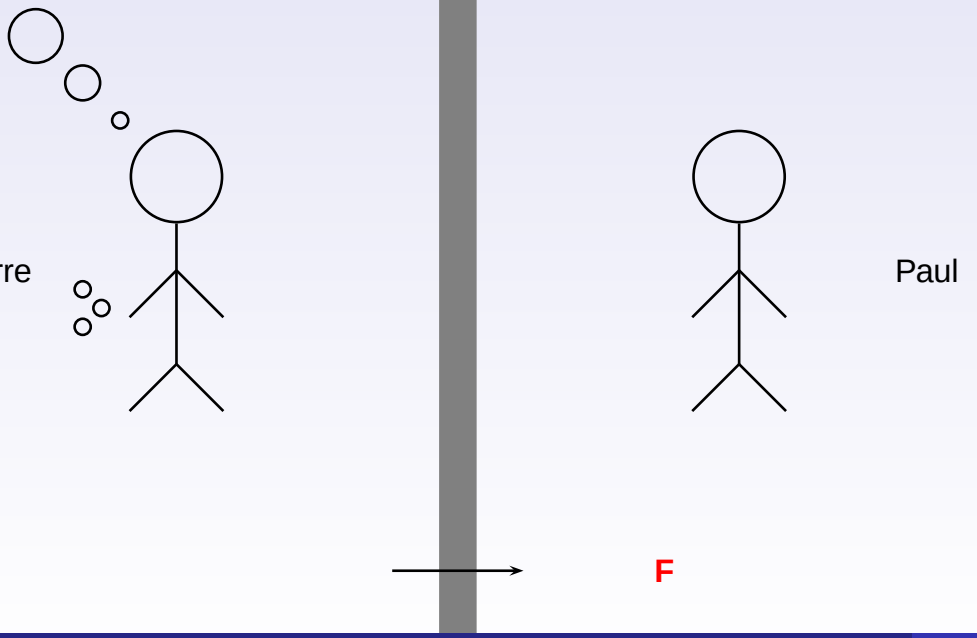
Exemple



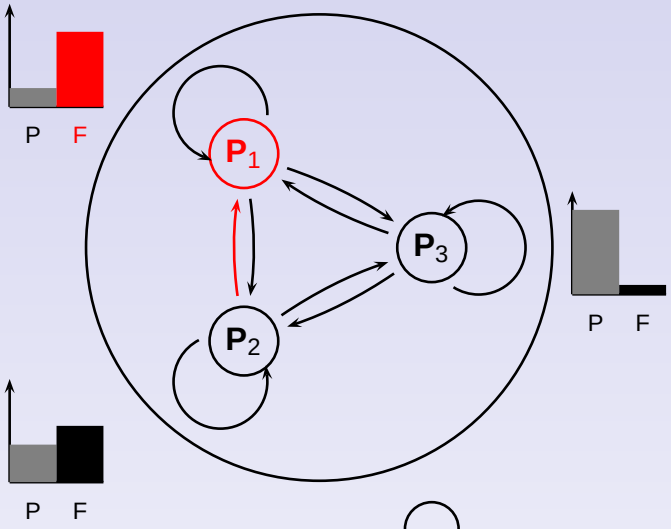
$$\Pi = (0.2\ 0.4\ 0.4)$$

$$A = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix}$$



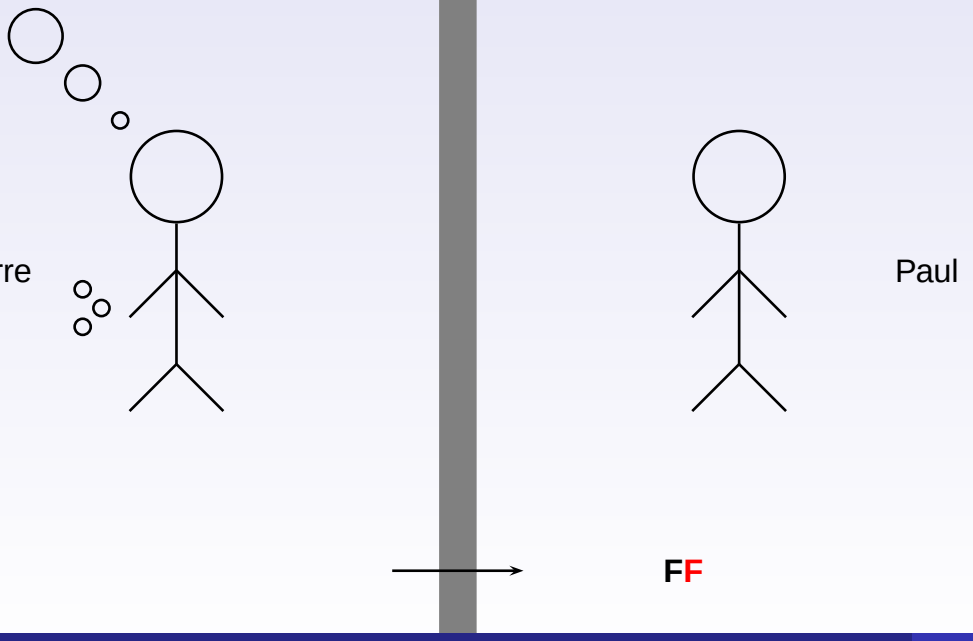
Exemple



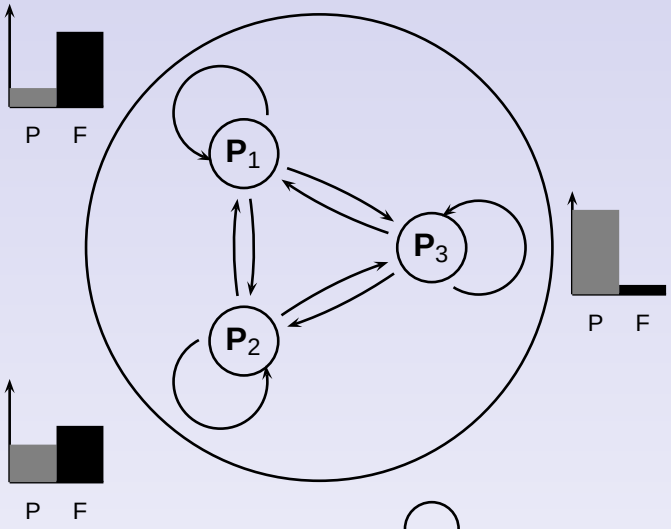
$$\Pi = (0.2\ 0.4\ 0.4)$$

$$A = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix}$$



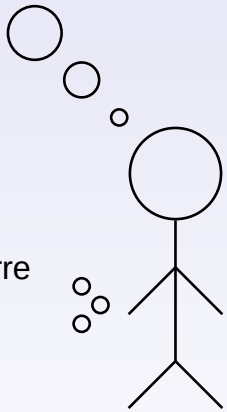
Exemple

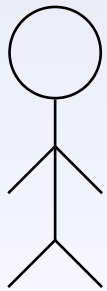


$$\Pi = (0.2\ 0.4\ 0.4)$$

$$A = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.5 & 0.4 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix}$$





$P_2P_1P_2P_2P_3P_1P_2P_1P_1P_1P_3P_1P_2P_3P_1P_2P_3P_1$
F F P P P P P F P F P F F P P P P F

FFPPPPPFPFPFFPPPPF

Les trois problèmes fondamentaux des MMC I

- Un modèle : $\lambda = (A, B, \Pi)$
- Trois problèmes fondamentaux :
  - Calcul de la vraisemblance :
    - $P(O/\lambda) = \sum_{Q \in \mathbb{S}^T} P(O, Q/\lambda) = \sum_{Q \in \mathbb{S}^T} \left(\pi_{q_1} \left(\prod_{t=1}^{T-1} a_{q_t q_{t+1}} \right) \left(\prod_{t=1}^{T} b_{q_t}(o_t) \right) \right)$
    - Algorithmes Forward et Backward
  - Calcul de la séquence d'états cachés la plus probablement suivie :
    - Trouver Q^* telle que $P(O, Q^*/\lambda)$ est maximale
    - Algorithme de Viterbi
  - Apprendre une séquence d'observations :
    - Trouver λ^* qui maximise un critère
    - Nombreux critères : information mutuelle, discrimination, erreur de classification, ...

Les trois problèmes fondamentaux des MMC II

- Critère le plus utilisé :
  - Maximisation de la vraisemblance
  - Pas de solution dans le cas général
  - Mais algorithme de Baum-Welch : «gradient-like»
  - Inévitablement piégé dans des optima locaux $\Longrightarrow$ métaheuristiques
- Difficultés du problème :
  - Optimisation continue
  - Avec contraintes de stochasticité
  - Un polynôme de degré élevé
  - Beaucoup de variables
  - Nombreux optima locaux
  - Nombreuses «symétries» dans l'espace de recherche (renumérotation des états)
  - Vitesse de convergence de Baum-Welch imprévisible

Les trois problèmes fondamentaux des MMC III

- Les heuristiques et méta heuristiques pour l'apprentissage de MMC avec le critère de maximum de vraisemblance :

Algorithme	
Baum-Welch	(Baum et Eagon, 1967)
Recuit simulé	(Paul, 1985), (Haman et Al Ani, 1996)
Recherche tabou	(Chen et al, 2004)
Algorithme génétique	(Slimane et al, 1999), (Brouard, 1999), (Thomsen, 2002)
Apprentissage incrémental à base de population	(Maxwell et Anderson, 1999)
API (fourmis artificielles)	(Monmarché, 2000), (Aupetit, 2005)
Optimisation par essaim particulaire	(Rasmussen, 2003), (Aupetit, 2005)

L’apprentissage de MMC avec l’OEP

Constats

- Problème d'optimisation continue :

  $\Longrightarrow$ l'OEP est une approche pertinente

- L'algorithme de Baum-Welch converge vers des optima locaux à des vitesses imprévisibles

  $\Longrightarrow$ hybrider l'OEP et n'utiliser que quelques itérations de Baum-Welch sur chaque solution

- Optimisation sous contraintes :
  1. Utiliser une fonction de pénalité : possible mais pas idéal
  2. Ignorer les contraintes puis «normaliser» lorsque nécessaire :
     - Principe suivi par (Rasmussen, 2003)
     - Mais ajoute une infinité d'optima locaux et de «symétries»
  3. Incorporer naturellement les contraintes en changeant d'espace de solutions

Constats

- Problème d'optimisation continue :

  $\Longrightarrow$ l'OEP est une approche pertinente

- L'algorithme de Baum-Welch converge vers des optima locaux à des vitesses imprévisibles

  $\Longrightarrow$ hybrider l'OEP et n'utiliser que quelques itérations de Baum-Welch sur chaque solution

- Optimisation sous contraintes :
  1. Utiliser une fonction de pénalité : possible mais pas idéal
  2. Ignorer les contraintes puis «normaliser» lorsque nécessaire :
     - Principe suivi par (Rasmussen, 2003)
     - Mais ajoute une infinité d'optima locaux et de «symétries»
  3. Incorporer naturellement les contraintes en changeant d'espace de solutions

L'espace vectoriel des MMC

- Trois étapes :
  1. L'espace des vecteurs stochastiques comme espace vectoriel :
     Soit $P_K^* = \{(x_1, \ldots, x_K) / \forall\, i = 1..K\,,\ x_i > 0 \text{ et } \sum_{i=1}^K x_i = 1\}$
     Soient $\oplus : P_K^* \to P_K^*$ et $\odot : \mathbb{R} \times P_K^* \to P_K^*$ tels que

     $$(x \oplus y)_i = \frac{x_i y_i}{\sum_{k=1}^K x_k y_k} \text{ et } (c \odot y)_i = \frac{x_i^c}{\sum_{k=1}^K x_k^c}$$

     $(P_K^*, \oplus, \odot)$ est un espace vectoriel
  2. L'espace des matrices stochastiques comme espace vectoriel :

     $$\mathbb{M}_{X \times Y}^* \equiv P_Y^* \times \ldots \times P_Y^*$$

  3. L'espace des MMC comme espace vectoriel :

     $$\Lambda^* = \mathbb{M}_{1 \times N}^* \times \mathbb{M}_{N \times N}^* \times \mathbb{M}_{N \times M}^*$$

     $(\Lambda^*, \oplus, \odot)$ espace vectoriel
- Tout vecteur de Λ^* est un MMC et vice versa (sauf ...)

L'OEP simple sur Λ^*

- La structure vectorielle $(\Lambda^*, \oplus, \odot)$ permet une adaptation facile de l'OEP
- Les équations de mouvement vectorielles :
  1. Déplacement :

$$x_i(t+1) = BW(x_i(t) \oplus v_i(t))$$

  2. Compensation de l'opérateur d'optimisation local :

$$v_i(t) = x_i(t+1) \ominus x_i(t)$$

  3. Mise à jour de la vitesse :

$$\begin{aligned} v_i(t+1) = \; & \omega \odot v_i(t) \\ & \oplus (c_1 \cdot \mathscr{U}_1) \odot (x_i^+(t) \ominus x_i(t)) \\ & \oplus (c_2 \cdot \mathscr{U}_2) \odot (\hat{x}_i(t) \ominus x_i(t)) \end{aligned}$$

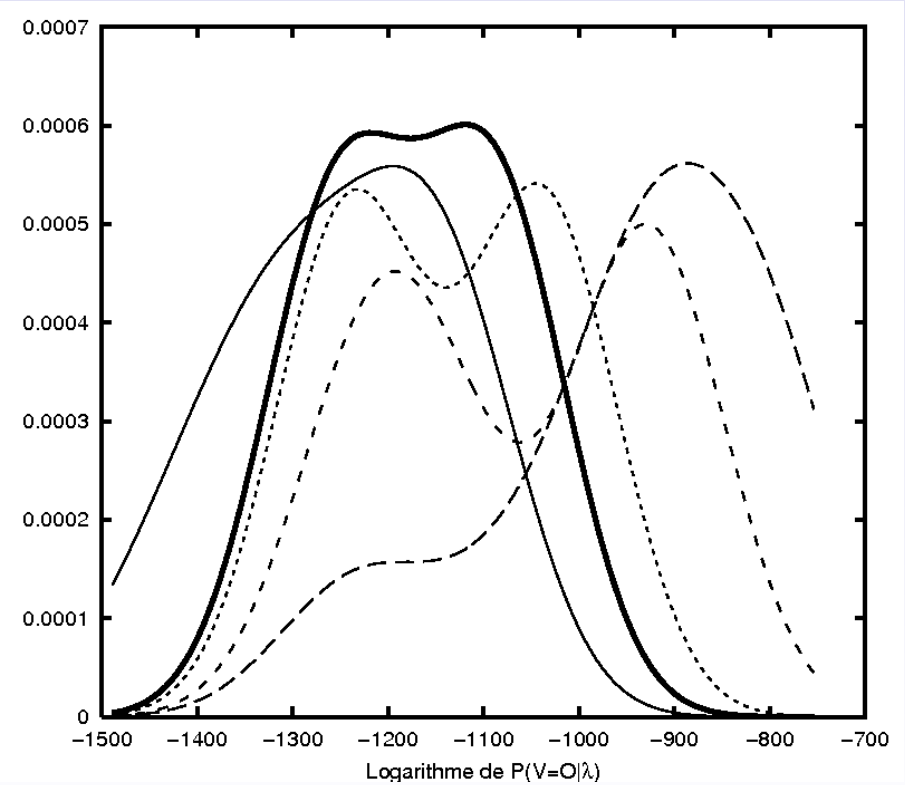
Etudes expérimentales

Recherche de bons paramètres I

- Apprentissage d'un jeu de 8 images :
  - Plus de 48 000 combinaisons de paramètres
  - 15 apprentissages par combinaison
  - 1000 itérations de Baum-Welch ou 30000 évaluations de MMC par apprentissage
  - 10, 20 et 40 états cachés
  - Voisinnage circulaire de taille V
- Bon paramètre :
  - Garantit de bonnes performances quasi-indépendemment des autres paramètres
  - Pas de garantie d'optimalité mais relativement indépendant de l'observation apprise

Recherche de bons paramètres II

- Recherche manuelle des bons paramètres grâce à des courbes moyennes estimant de façon lissée la probabilité d'apparition d'un paramètre en fonction de la performance :



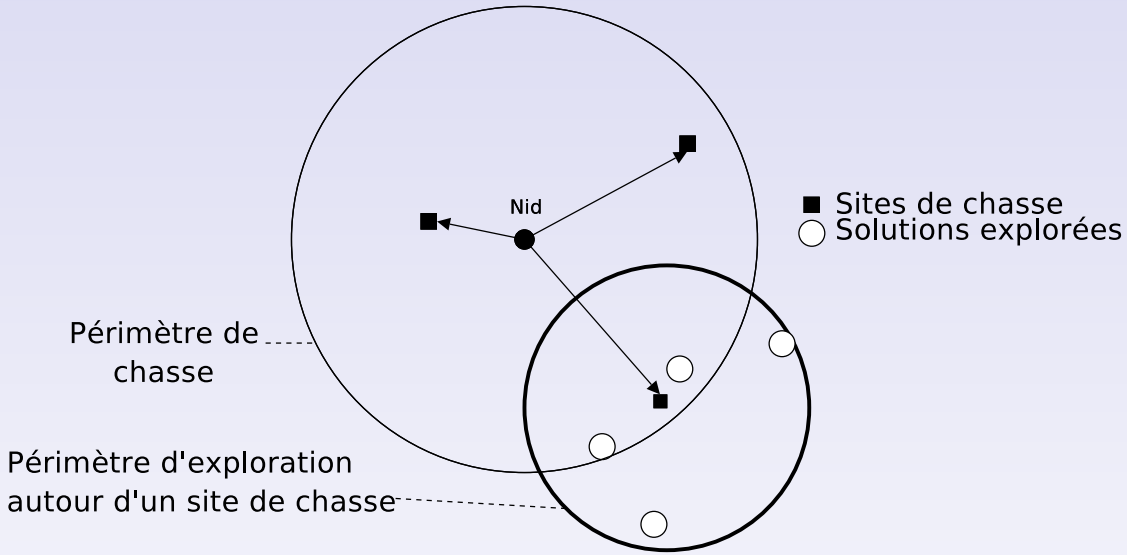
Recherche de bons paramètres III

- Résultats :

	$\mathcal{N}_{BW}$	$\mathcal{N}$	ω	c_1	c_2	V
OEP1	0	20	0.8	1.5	2.5	9
OEP2	2	30	0.8	1.5	0	0
OEP3	5	50	0.4	0	0.5	6

Comparaison des performances

- L'algorithme de colonie de fourmis API :
  - Inspiré du comportements des Pachycondyla Apicalis



- Adapté :
  - sur la structure vectorielle (Aupetit, 2005),
  - sans structure vectorielle (Monmarché, 2000)
- Les mêmes principes de réglage ont été appliqués sur API

Résultats I

$\overline{e}(A)$ performance moyenne normalisée sur [0 ;1] sur plus d'images et pour 5000 évaluations de MMC

Algorithme A	Moyenne $\overline{e}(A)$	$\mathcal{N}_{\text{BW}}$
Random0	14.27%	0
OEP1	59.23%	0
Random2	67.94%	2
Random5	82.32%	5
API vectoriel	98.89%	2
OEP3	99.22%	5
API ordinaire	99.46%	5
OEP2	99.53%	2

- Sans hybridation : performances basses, coût de calcul plus élevé
- Avec hybridation : performances élevées
- L'OEP est compétitive avec le plus performant de nos algorithmes

Résultats II

- Evolution du classement selon le nombre total d'itérations de Baum-Welch

	100 itérations		1000 itérations		5000 itérations	
Algorithme A	$\overline{e}(A)$	Ecart	$\overline{e}(A)$	Ecart	$\overline{e}(A)$	Ecart
Random2	53.57%	53%	0.00%	0%	67.94%	8%
Random5	81.67%	82%	50.26%	8%	82.32%	7%
API vectoriel	3.48%	10%	92.71%	3%	98.89%	1%
OEP3	N/A	N/A	84.18%	6%	99.22%	1%
API ordinaire	98.82%	1%	98.24%	2%	99.46%	1%
OEP2	64.00%	16%	99.34%	1%	99.53%	1%

Conclusion

Conclusion

- L'adaptation de l'OEP au problème de l'apprentissage de MMC se fait simplement :
  - $(\Lambda^*, \oplus, \odot)$
  - Hybridation avec l'algorithme de Baum-Welch
- L'hybridation est rentable
- Compétitif avec nos algorithmes les plus efficaces
- Efficacité inférieure avec peu d'itérations de Baum-Welch mais se rattrape par la suite

Perspectives

- L'OEP que nous avons considérée est très simple
- Les équations de mouvement vectorielles ne sont pas les plus efficaces pour l'OEP :
  - D'autres équations de mouvement permettraient peut être d'améliorer les performances ?
- La structure d'espace $(\Lambda^*, \oplus, \odot)$ pourrait être utilisée :
  - pour mieux comprendre ce que font nos méta heuristiques d'apprentissage ?
  - pour adapter naturellement d'autres méta heuristiques à ce problème ?
- Nos algorithmes se concentrent sur le maximum de vraisemblance :
  - Sont-ils performants pour les autres critères ?