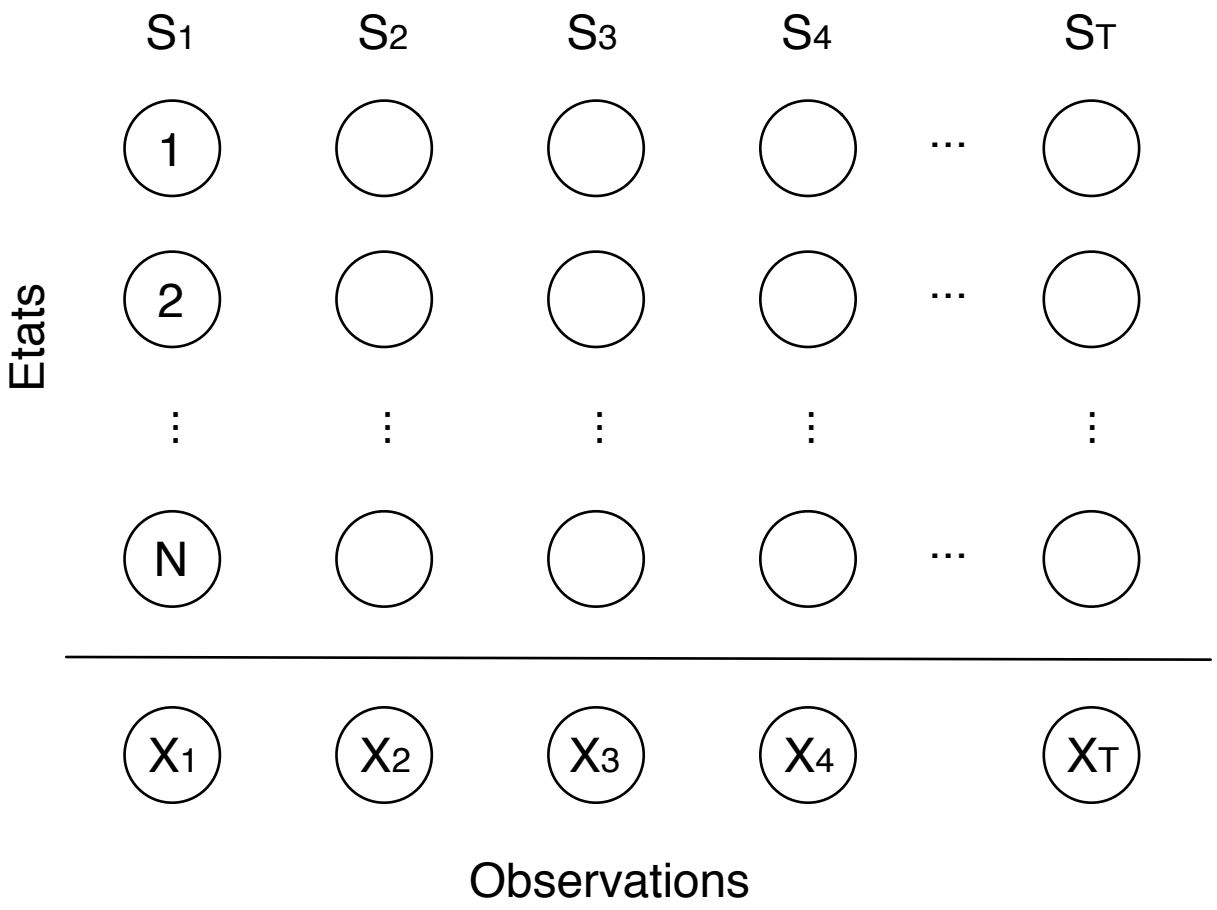
MAPSI — cours 8 : Apprentissage des chaîne de Markov cachée

Vincent Guigue, Thierry Artières
vincent.guigue@lip6.fr

LIP6 – Université Paris 6, France

Rappels sur la structure d'un MMC

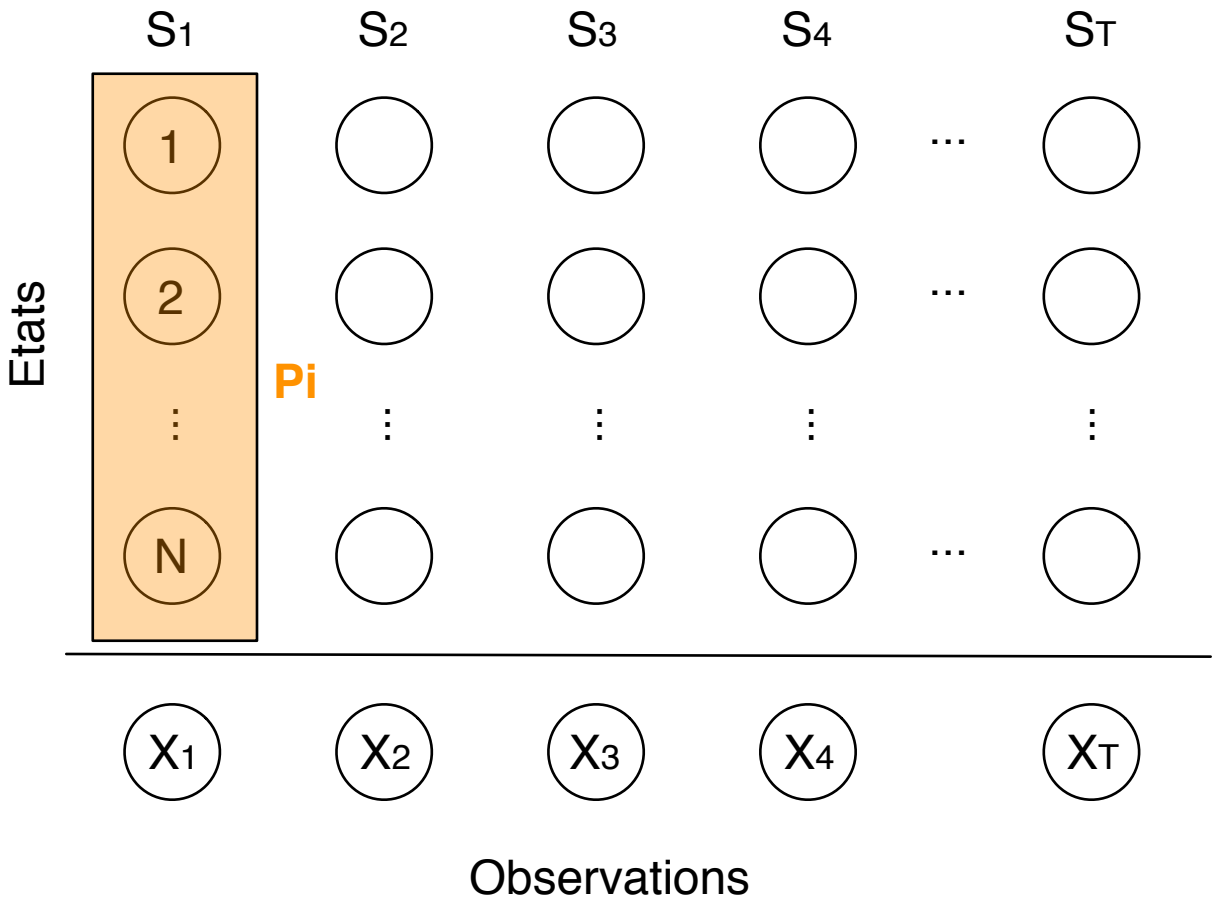
Constitution d'un MMC :



- Les états sont inconnus...

Rappels sur la structure d'un MMC

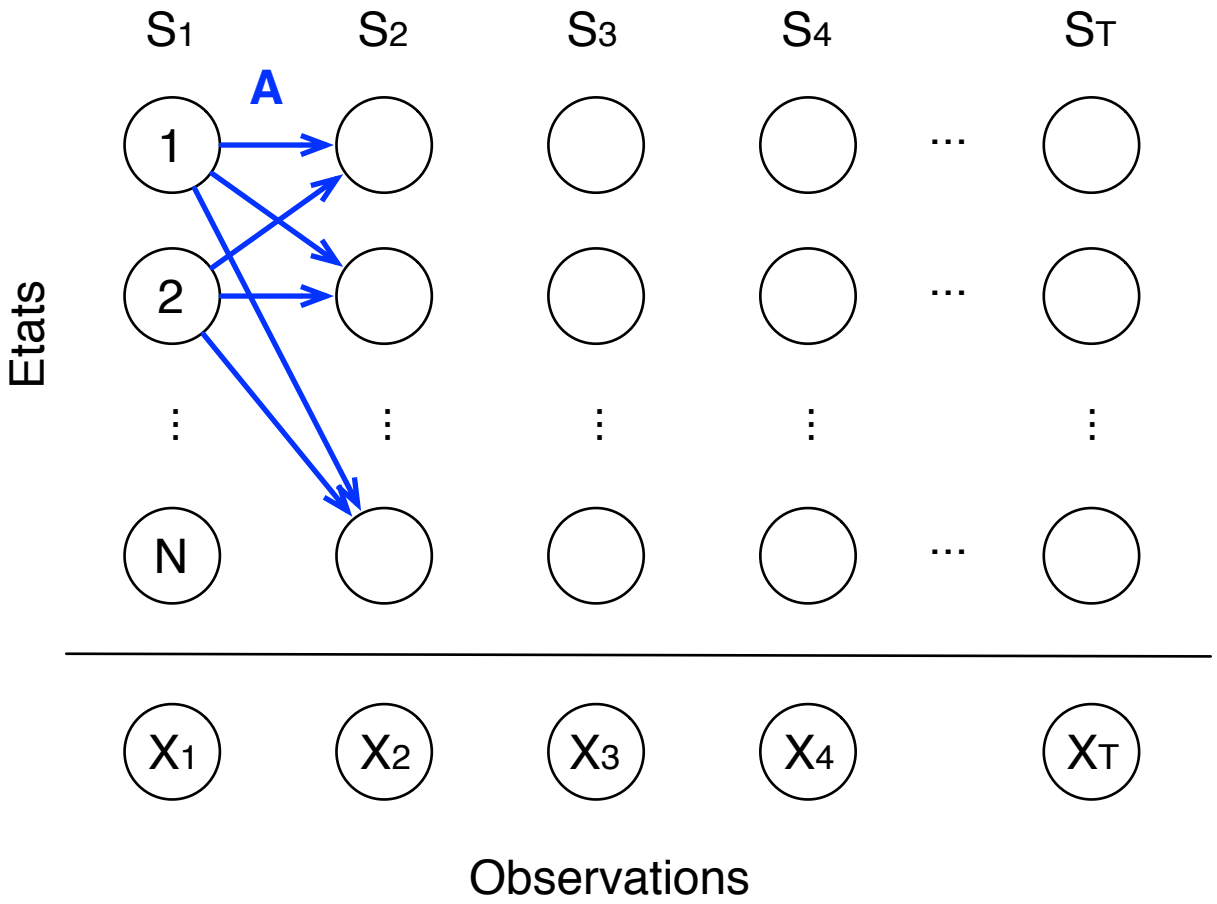
Constitution d'un MMC :



- Les états sont inconnus...

Rappels sur la structure d'un MMC

Constitution d'un MMC :

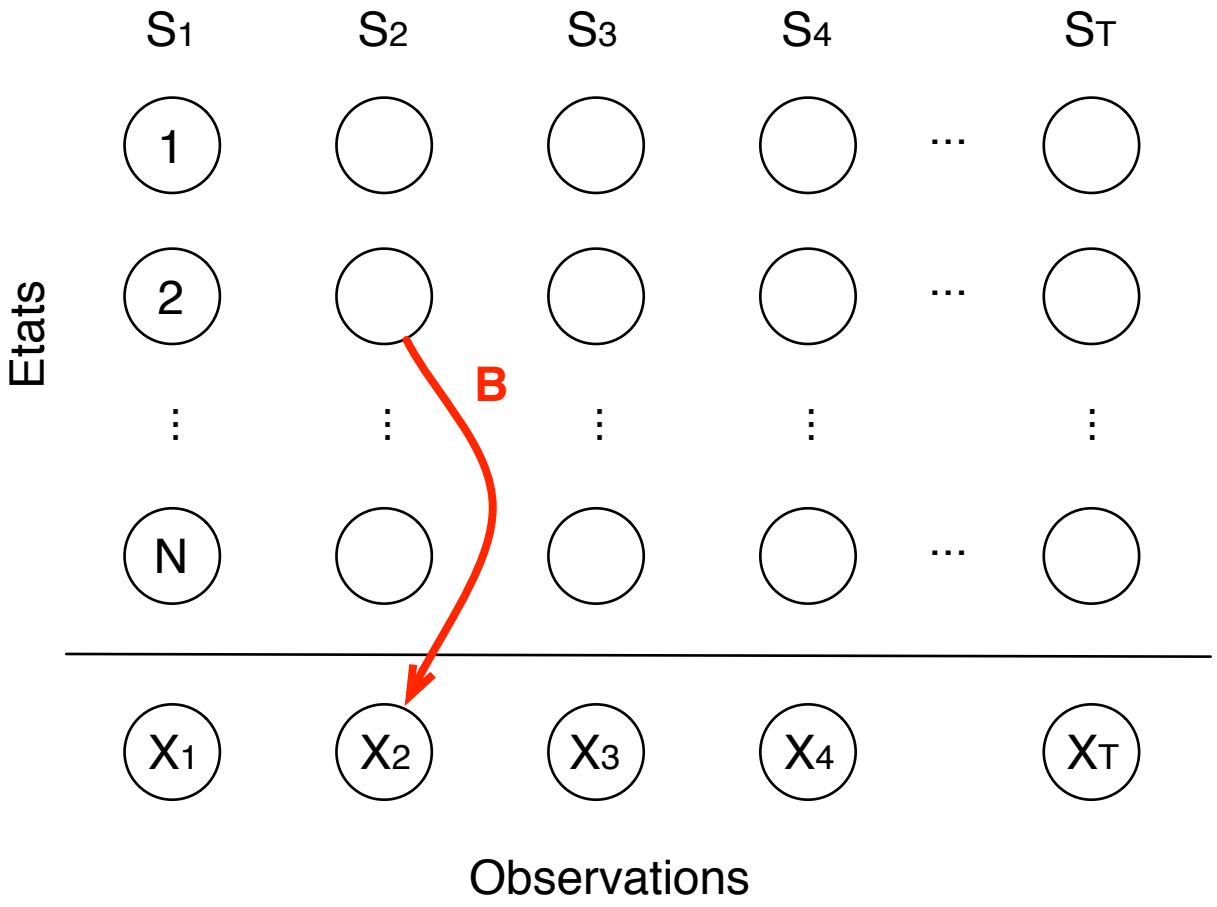


Hyp. Ordre 1 : chaque état ne dépend que du précédent

- Les états sont inconnus...
  La combinatoire à envisager est problématique !

Rappels sur la structure d'un MMC

Constitution d'un MMC :



Hyp. Ordre 1 : chaque état ne dépend que du précédent

Chaque obs. ne dépend que de l'état courant

- Les états sont inconnus...

Les trois problèmes des MMC (Fergusson - Rabiner)

- **Evaluation** : λ donné, calcul de $p(x_1^T|\lambda)$
- **Décodage** : λ donné, quelle séquence d'états a généré les observations ?

$$s_1^{T\star} = \arg\max_{s_1^T} p(s_1^T|x_1^T,\lambda) = \arg\max_{s_1^T} p(x_1^T, s_1^T|\lambda)$$

- **Apprentissage** : à partir d'une série d'observations, trouver $\lambda^\star$

$$\lambda^\star = \{\Pi^\star, A^\star, B^\star\} = \arg\max_{\lambda} p(x_1^T|\lambda)$$

PB1 : Algorithme *forward*

$$\alpha_t(i) = p(x_1^t, s_t = i|\lambda)$$

- Initialisation :

$$\alpha_{t=1}(i) = p(x_1^1, s_1 = i|\lambda) = \pi_i b_i(x_1)$$

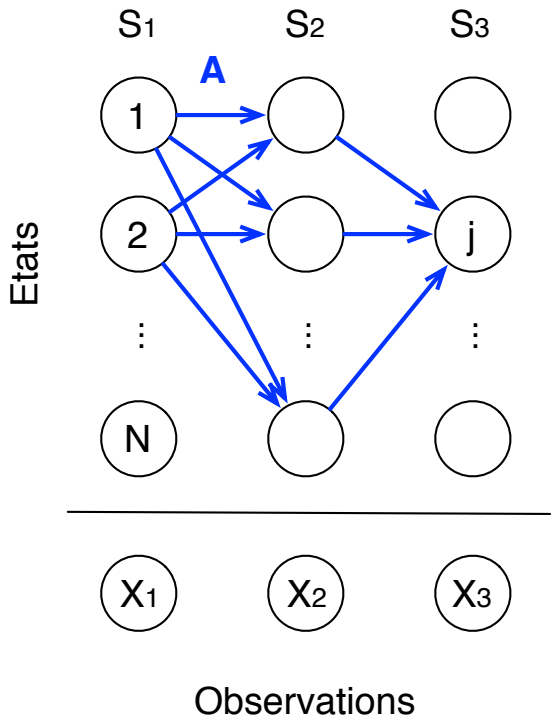
- Itération :

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(x_t)$$

- Terminaison :

$$p(x_1^T|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

- Complexité linéaire en T
  - Usuellement : $T >> N$



PB2 : Viterbi (récapitulatif)

$$\delta_t(i) = \max_{s_1^{t-1}} p(s_1^{t-1}, s_t = i, x_1^t | \lambda)$$

1. Initialisation

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(x_1) \\ \Psi_1(i) &= 0 \end{aligned}$$

2. Récursion

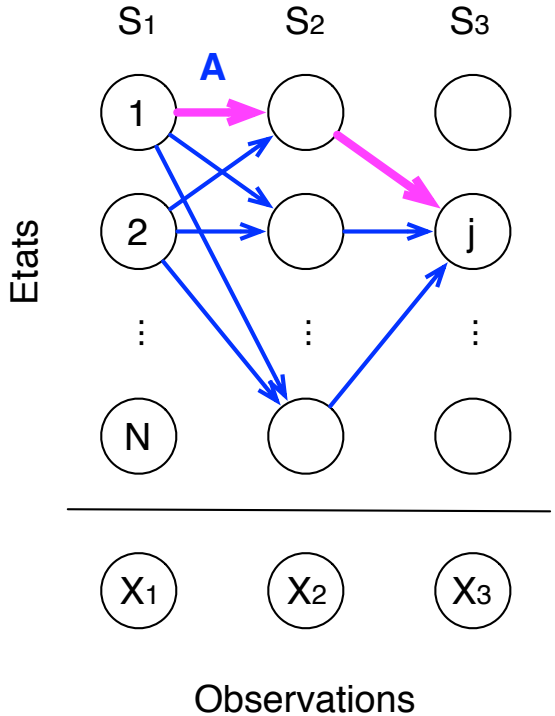
$$\begin{aligned} \delta_t(j) &= \left[\max_i \delta_{t-1}(i) a_{ij}\right] b_j(x_t) \\ \Psi_t(j) &= \arg \max_{i \in [1,\, N]} \delta_{t-1}(i) a_{ij} \end{aligned}$$

3. Terminaison

$$S^\star = max_i \delta_T(i)$$

4. Chemin

$$\begin{aligned} q_T^\star &= \arg \max_i \delta_T(i) \\ q_t^\star &= \Psi_{t+1}(q_{t+1}^\star) \end{aligned}$$



PB3 : Apprentissage des MMC

- **Version simplifiée** (*hard assignment*) : type k-means
- Nous disposons de :
  - **Evaluation** : $p(x_1^T|\lambda)$
  - **Décodage** : $s_1^{T\star} = \arg\max_{s_1^T} p(x_1^T|\lambda)$
- Proposition :

Algorithm 1: Baum-Welch simplifié pour l'apprentissage d'un MMC

Data: Observations : X, Structure= N, K
Result: $\tilde{\Pi}^\star, \tilde{A}^\star, \tilde{B}^\star$
Initialiser $\lambda_0 = \Pi^0, A^0, B^0$;
  $\rightarrow$ finement si possible;
$t = 0$;
while *convergence non atteinte* **do**
  $S_{t+1} = decodage(X, \lambda_t)$;
  $\lambda^\star_{t+1} = \Pi^{t+1}, A^{t+1}, B^{t+1}$ obtenus par comptage des transitions ;
  $t = t + 1$;

Vous avez déjà tous les éléments pour faire ça !

Apprentissage complet

Algorithm 2: Baum-Welch simplifié pour l'apprentissage d'un MMC

Data: Observations : X
Result: $\tilde{\Pi}^\star, \tilde{A}^\star, \tilde{B}^\star$
Initialiser $\lambda_0 = \Pi^0, A^0, B^0$;
$\rightarrow$ finement si possible;
$t = 0$;
while *convergence non atteinte* **do**
    Trouver les distributions de probabilité des variables manquantes (états);
    Ré-estimer les paramètres $\lambda^\star_{t+1}$;
    $t = t + 1$;

Affectation dure $\Rightarrow$ estimation des distributions d'appartenance
Algorithme type EM

App. complet : avant propos (et révision)

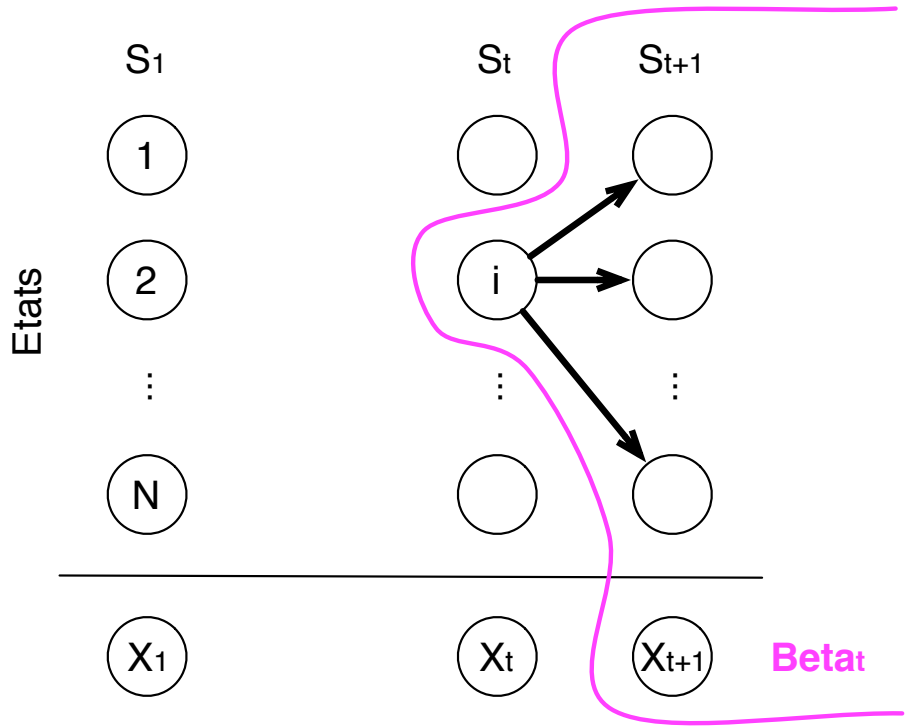
On a vu le calcul des α... Définissons maintenant les β :

$$\beta_t(i) = p(x_{t+1}^T | s_t = i, \lambda), \qquad \text{(sym. des } \alpha\text{)}$$

- Initialisation (arbitraire) :

$$\forall i, \qquad \beta_{t=T}(i) = 1$$

- Récursion :



App. complet : avant propos (et révision)

On a vu le calcul des α... Définissons maintenant les β :

$$\beta_t(i) = p(x_{t+1}^T | s_t = i, \lambda), \qquad \text{(sym. des } \alpha\text{)}$$

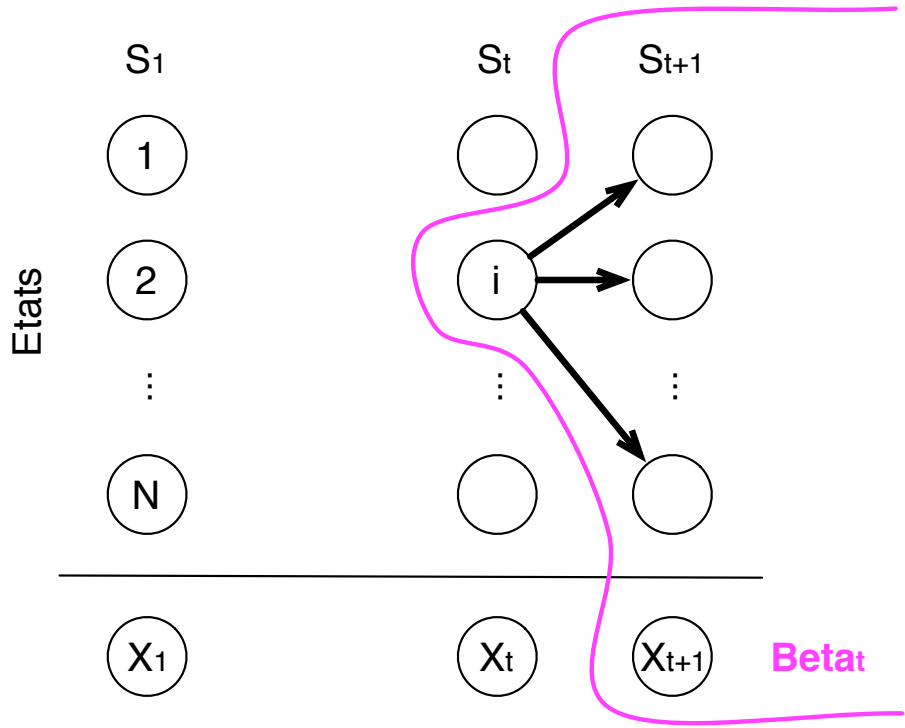
- Initialisation (arbitraire) :

$$\forall i, \qquad \beta_{t=T}(i) = 1$$

- Récursion :

Comment puis partir de l'état i à t et observer la suite des x_{t+1}^T ?

1. Transition de i vers n'importe quel j
2. Observation de x_{t+1} (à partir de j)
3. ... Puis on continue sur $\beta_{t+1}(j)$



App. complet : avant propos (et révision)

On a vu le calcul des α... Définissons maintenant les β :

$$\beta_t(i) = p(x_{t+1}^T | s_t = i, \lambda), \qquad \text{(sym. des } \alpha\text{)}$$

- Initialisation (arbitraire) :

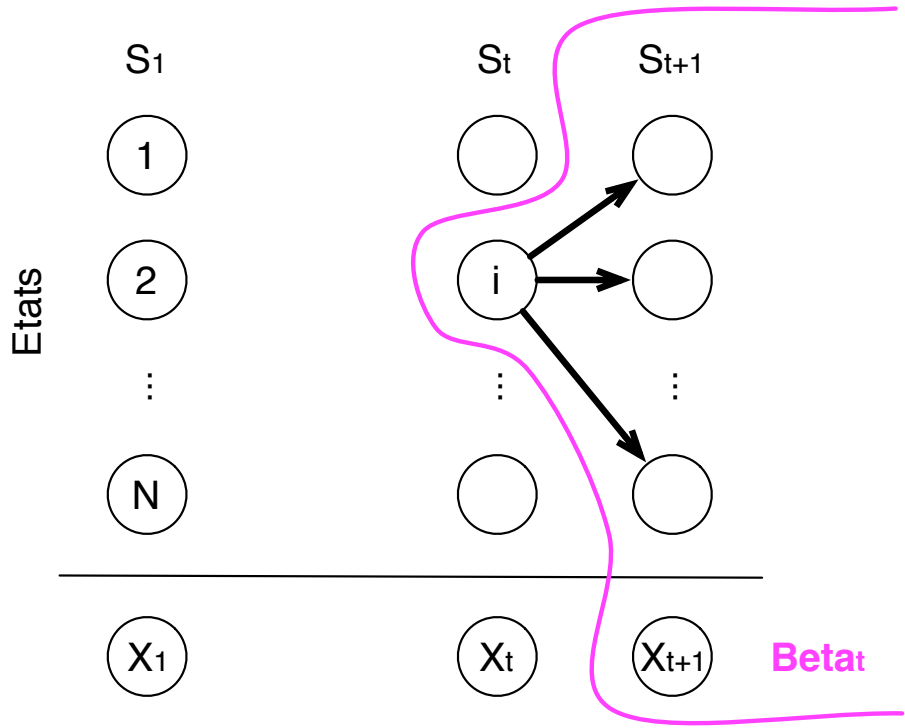
$$\forall i, \qquad \beta_{t=T}(i) = 1$$

- Récursion :

Comment puis partir de l'état i à t et observer la suite des x_{t+1}^T ?

1. Transition de i vers n'importe quel j
2. Observation de x_{t+1} (à partir de j)
3. ... Puis on continue sur $\beta_{t+1}(j)$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$



App. complet : modélisation

Critère : max de vraisemblance sur les observations

- Baum-Welsch **simplifié** : affectation en dur des états :
  + comptage

$$s_t^\star = \arg\max_i p(s_t = i | x_1^T)$$

- Pour la version **complète** :
  $\Rightarrow$ distributions sur les variables manquantes
  + comptage pondéré par les probabilités d'appartenance

$$\gamma_t(i) = p(s_t = i | x_1^T, \lambda)$$

$$\gamma_t(i,j) = p(s_t = i, s_{t+1} = j | x_1^T, \lambda)$$

App. complet : modélisation (2)

$$\gamma_t(i) = p(s_t = i|x_1^T, \lambda), \quad \gamma_t(i,j) = p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

- Les γ se calculent à partir des :

$$\alpha_t(i) = p(x_1^t, s_t = i|\lambda)$$

$$\beta_t(i) = p(x_{t+1}^T|s_t = i, \lambda)$$

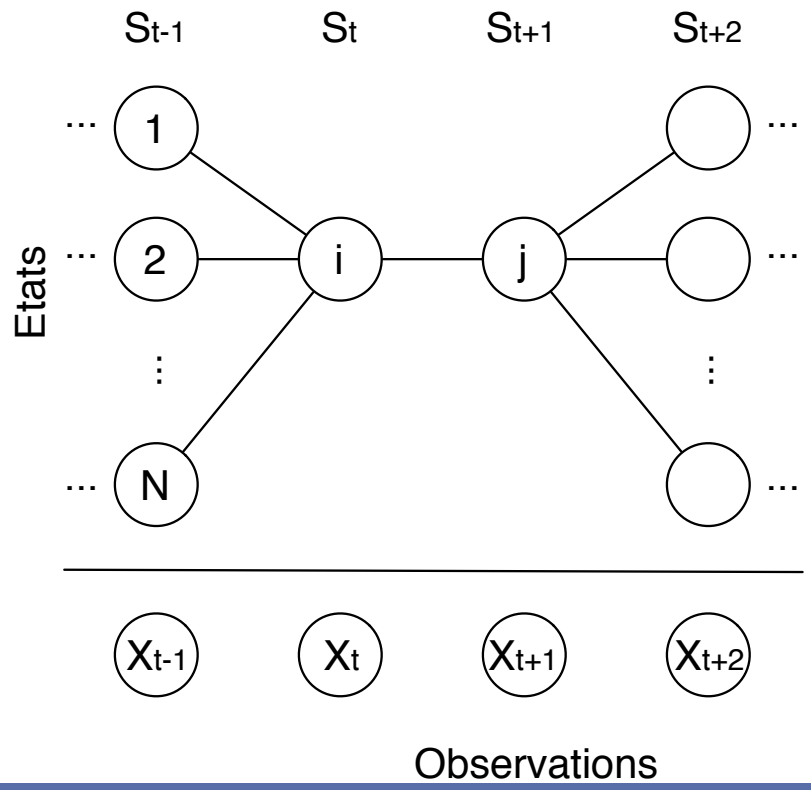
- Exprimer les γ en fonction des α et β ? ?

App. complet : Démo $\gamma(i,j)$

$$\gamma_t(i,j) = p(s_t = i, s_{t+1} = j | x_1^T, \lambda)$$

En fonction de :

$$\alpha_t(i) = p(x_1^t, s_t = i | \lambda), \quad \beta_t(i) = p(x_{t+1}^T | s_t = i, \lambda)$$



App. complet : Démo $\gamma(i,j)$

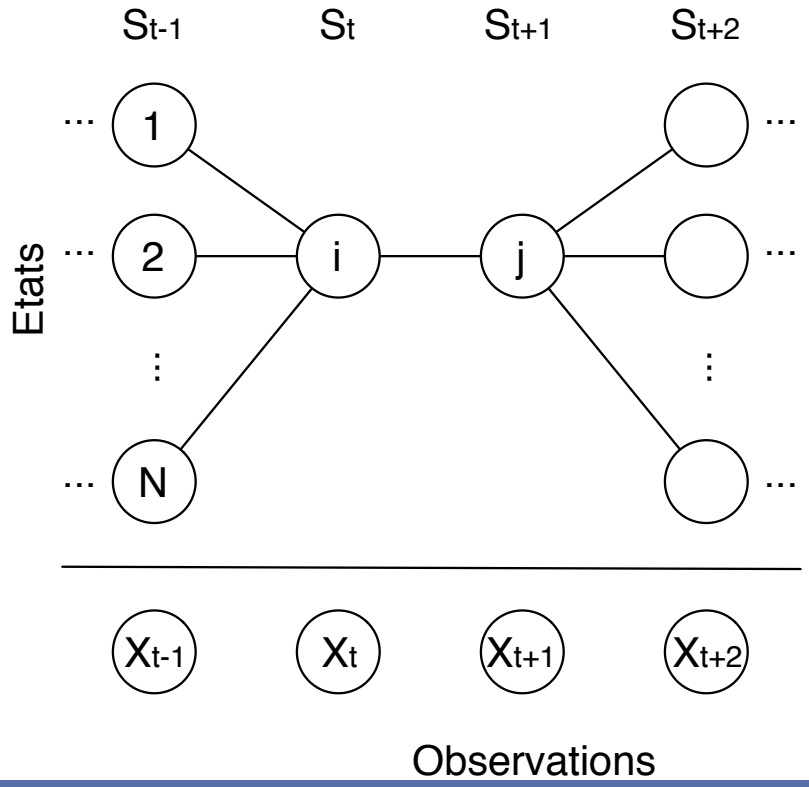
$$\gamma_t(i,j) = p(s_t = i, s_{t+1} = j | x_1^T, \lambda)$$

En fonction de :

$$\alpha_t(i) = p(x_1^t, s_t = i|\lambda), \quad \beta_t(i) = p(x_{t+1}^T | s_t = i, \lambda)$$

Début ($p(A|B) = \frac{p(A,B)}{p(B)}$) :

$$\begin{aligned}
\gamma_t(i,j) &= p(s_t = i, s_{t+1} = j | x_1^T, \lambda) \\
&= \frac{p(s_t = i, s_{t+1} = j, x_1^T | \lambda)}{p(x_1^T|\lambda)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{p(x_1^T|\lambda)}
\end{aligned}$$



App. complet : Démo $\gamma(i)$

$$\gamma_t(i) = p(s_t = i|x_1^T, \lambda), \quad \gamma_t(i,j) = p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

- $\gamma_t(i)$ est une marginale par rapport à $\gamma_t(i,j)$!
- D'où :

$$\gamma_t(i) = \sum_{j=1}^{N} \gamma_t(i,j)$$

De γ aux paramètres du MMC

$$\gamma_t(i) = p(s_t = i|x_1^T, \lambda), \quad \gamma_t(i,j) = p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

- Quel est le lien entre γ et les paramètres du modèle ?
- Interprétation :

$$\sum_t \gamma_t(i,j) = \sum_t p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

$$\sum_t \gamma_t(i) = \sum_t p(s_t = i|x_1^T, \lambda)$$

De γ aux paramètres du MMC

$$\gamma_t(i) = p(s_t = i|x_1^T, \lambda), \quad \gamma_t(i,j) = p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

- Quel est le lien entre γ et les paramètres du modèle ?
- Interprétation :

$$\sum_t \gamma_t(i,j) = \sum_t p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

Espérance du nombre de transitions de *i* à *j*

$$\sum_t \gamma_t(i) = \sum_t p(s_t = i|x_1^T, \lambda)$$

De γ aux paramètres du MMC

$$\gamma_t(i) = p(s_t = i|x_1^T, \lambda), \quad \gamma_t(i,j) = p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

- Quel est le lien entre γ et les paramètres du modèle ?
- Interprétation :

$$\sum_t \gamma_t(i,j) = \sum_t p(s_t = i, s_{t+1} = j|x_1^T, \lambda)$$

Espérance du nombre de transitions de *i* à *j*

$$\sum_t \gamma_t(i) = \sum_t p(s_t = i|x_1^T, \lambda)$$

Espérance du nombre de transitions issues de *i*

Mise à jour des paramètres de A

- Etant donné l'interprétation des γ :

$$\tilde{a}_{ij} = \frac{\sum_t \gamma_t(i,j)}{\sum_t \gamma_t(i)}$$

Il s'agit bien d'une sorte de comptage probabiliste des transitions

- Assez simple pour les π_i

$$\tilde{\pi}_i = \gamma_1(i), \qquad \text{(naturellement normalisé)}$$

- Un peu plus compliqué pour les probabilités d'émission :

$$\tilde{b_j(k)} = \frac{\sum_{t \text{ t.q. } x_t=k} \gamma_t(j)}{\sum_t \gamma_t(j)}$$

Comptage probabiliste quand on est dans l'état j de générer l'observation k.

Algorithme de Baum-Welch

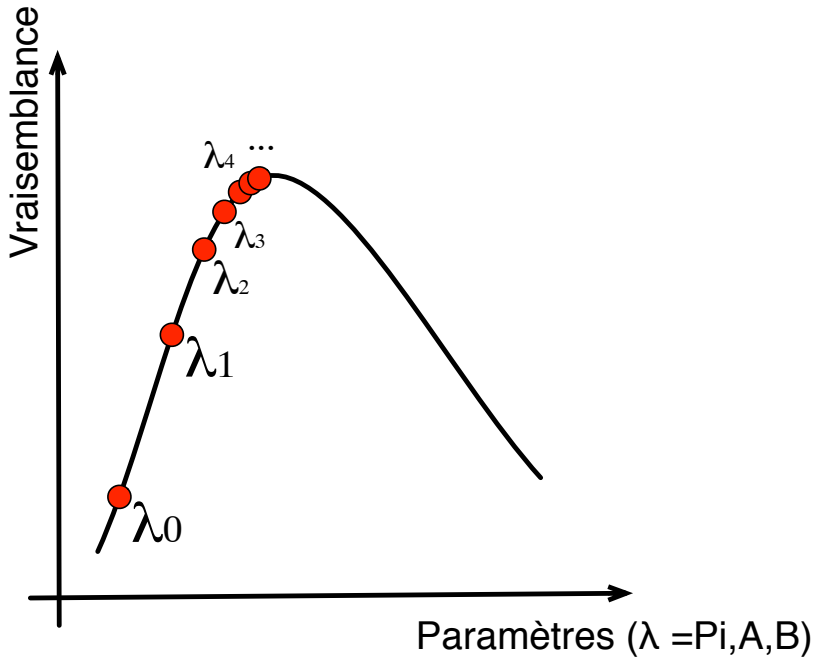
Algorithm 3: Baum-Welch pour l'apprentissage d'un MMC

Data: Observations : X, Structure= N, K
Result: $\tilde{\Pi}^\star, \tilde{A}^\star, \tilde{B}^\star$
Initialiser $\lambda_0 = \Pi^0, A^0, B^0$;
$\rightarrow$ finement si possible;
$t = 0$;
while *convergence non atteinte* **do**
    Etape E
    Forward/Backward : calcul des α, β;
    [OPT] Calcul des γ;
    Etape M
    Mise à jour de Π^t, A^t, B^t;
    $t = t + 1$;

Convergence de la vraisemblance

Objectif de EM

Maximiser la vraisemblance
ie : faire coller un modèle à des observations



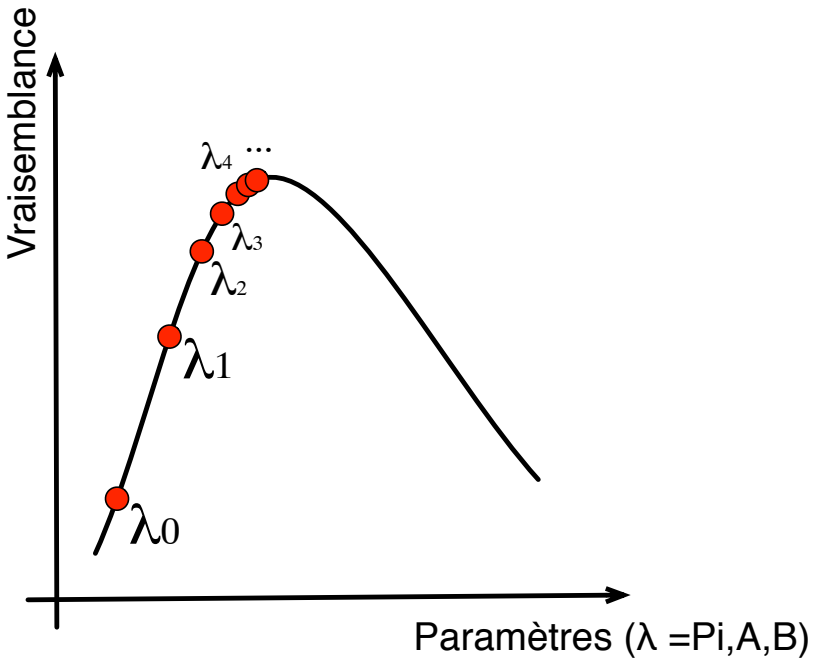
Critère de convergence : maximisation de la vraisemblance
Soit ensemble de séquence d'observations : $X = \{\mathbf{x_i}\}_{i=1,\ldots,n}$

$$\log \mathcal{L}(X, \lambda) = \sum_{i=1}^{n} \log(p(\mathbf{x_i}|\lambda))$$

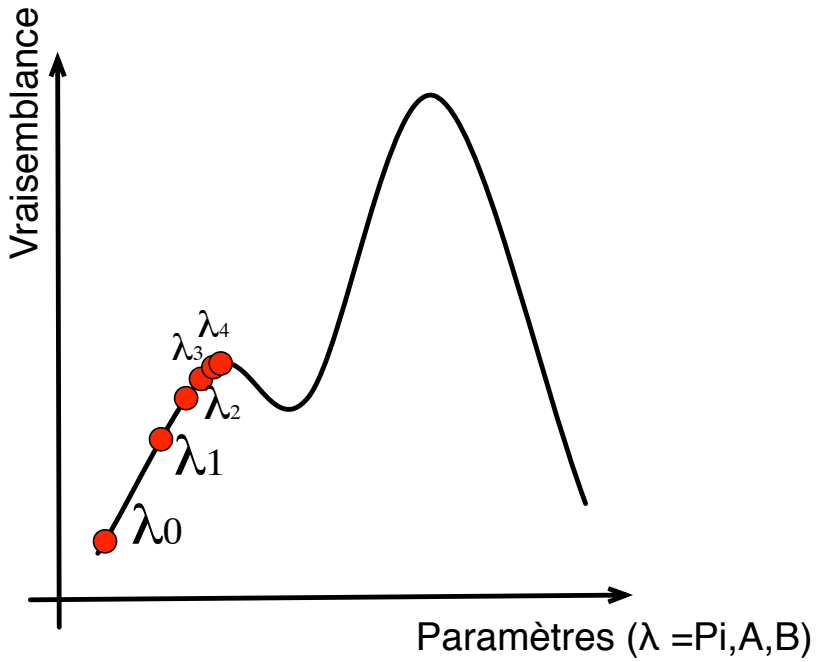
Note : les $\log(p(\mathbf{x_i}|\lambda))$ sont calculés par Viterbi ou la méthode des α, selon la stratégie d'optimisation choisie.

Convexité... Ou pas

Optimisation convexe :



Optimisation non-convexe :

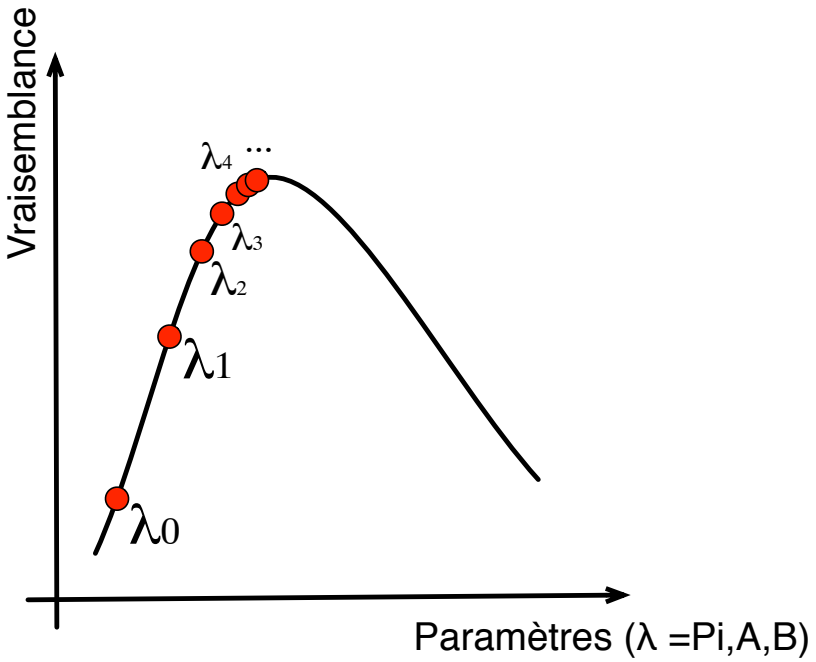


EM

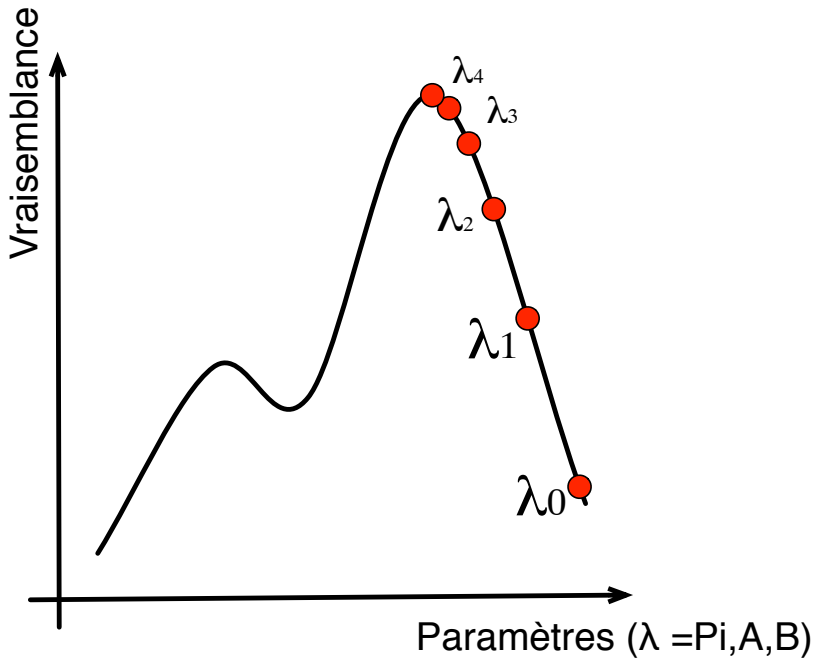
Pas de garantie sur un optimum global dans le cas non-convexe...
MMC = problème non convexe en général

Convexité... Ou pas

Optimisation convexe :



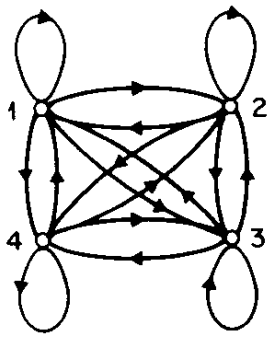
Optimisation non-convexe :



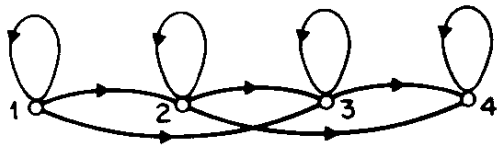
EM

Pas de garantie sur un optimum global dans le cas non-convexe...
MMC = problème non convexe en général $\Rightarrow$**Bien choisir** λ_0 !

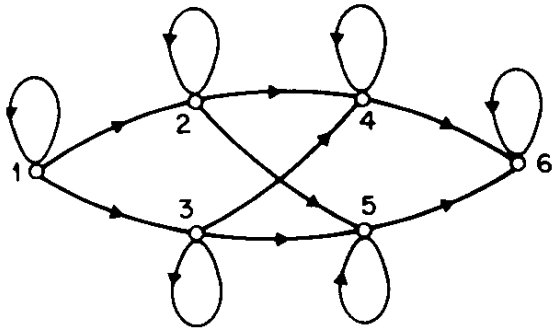
Types de MMC (et importance de l'init.)



(a)



(b)



(c)

- (a) Modèle ergodique (= complètement connecté)
- Modèle gauche-droite
  - exhaustif (cf TME)
  - (b) avec saut possible

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

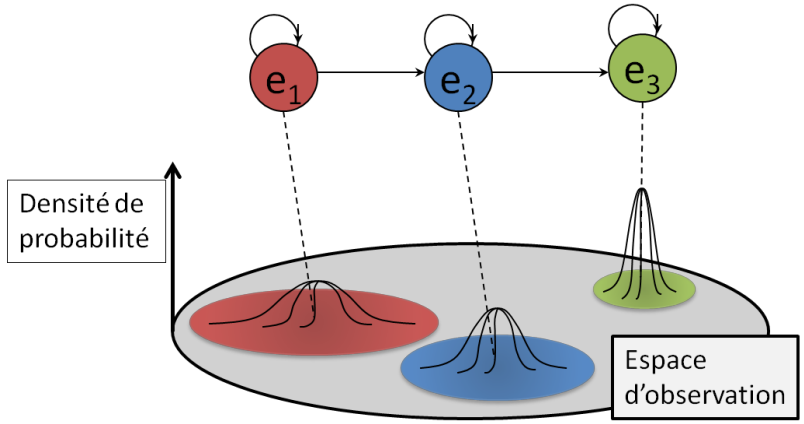
  - (c) chemins parallèles
- Modèle cyclique

Initialisation aléatoire + connaissance *a priori* sur le problème

Variante : MMC à observations continues

Cas le plus simple :

- 1 état $j = 1$ gaussienne $\mathcal{N}(\mu_j, \sigma_j)$
- Pas de changement pour Π, A



Une observation x_t appartient à toute les gaussiennes (avec une pondération) :

$$\tilde{\mu}_j = \frac{\sum_t \gamma_t(j) x_t}{\sum_t \gamma_t(j)} \qquad \sigma_j^2 = \frac{\sum_t \gamma_t(j)(x_t - \mu_j)^2}{\sum_t \gamma_t(j)}$$

Facilement extensible à une mixture de gaussiennes par état (cf Rabiner)

Variante : MMC sur distributions multi-variées

- Distribution des observations multi-variée = plusieurs observations à chaque pas de temps.

$$x_1^T \Rightarrow \mathbf{x}_1^T, \quad \mathbf{x}_t \in \mathbb{R}^d$$

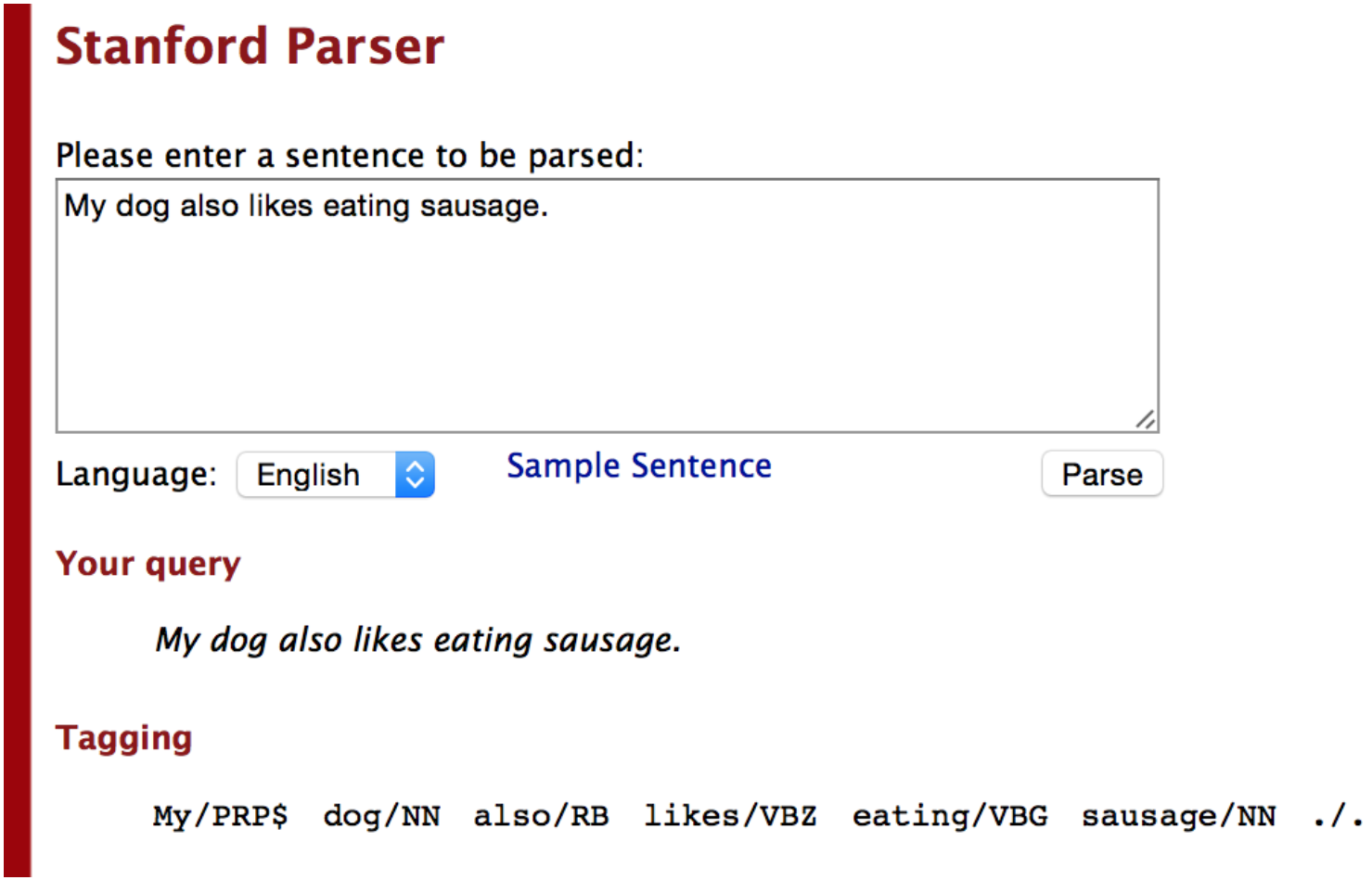
$\Rightarrow$ Il suffit de prendre une distribution multi-variée pour les émissions (eg : une gaussienne à D dimensions)

- Ca ne change rien au reste de la résolution

Exemple : étiquetage morpho-syntaxique par des modèles markoviens

- Etape de traitement intermédiaire pour des applications en **langage naturel** et en **recherche d'information**.
- But :
  - Associer à chaque terme d'une phrase une **étiquette morpho-syntaxique**
  - **Exemple :** *article, préposition, adjectif, nom commun singulier, nom propre, nom commun pluriel, pronom personnel, adverbe, verbe infinitif, verbe au présent ou au passé,* etc. (entre 30 et 150 étiquettes)
  - L'étiquetage doit déterminer les catégories syntaxiques des mots dans la phrase et doit en ce sens résoudre des problèmes d'ambiguïté.
  - **Performances :** autour de 95 % en anglais

Exemple : étiquetage morpho-syntaxique par des modèles markoviens



Démo online : `http://nlp.stanford.edu:8080/parser/`

Historique : étiquetage morpho-syntaxique

- Information syntagmatique : utiliser les séquences d'étiquettes non ambiguës pour désambiguer : 1ers étiqueteurs (1971) 77% étiquettes correctes

Exemple wikipedia :
"*Papa aime Maman*" et "*Le boulanger fait son pain*" ont le même axe syntagmatique. "*Papa*" et "*le boulanger*" sont des paradigmes du sujet, "*aime*" et "*fait*" sont des paradigmes du verbe, "*Maman*" et "son *pain*" sont des paradigmes du complément.

- Information lexicale statistique : attribuer à un mot son étiquette la plus fréquente (1987) : 90 %
- Premiers étiqueteurs performants sont basés sur des modèles markoviens (1990)
- Actuellement bon étiqueteurs statistiques ou/et à base de règles : utiliser à la fois l'information lexicale et l'information sur les séquences d'étiquettes.

Modélisation (Markovienne)

w_i	$i$ème mot de la séquence
t_i	étiquette associée à w_i
$t(i)$	$i$ème étiquette parmi l'ensemble d'étiquettes
$w(i)$	$i$ème mot du corpus
$C(w(i))$	$\sharp$ occurrences de $w(i)$ dans le corpus
$C(t(i))$	$\sharp$ occurrences de $t(i)$ dans le corpus
$C(t(i), t(k))$	$\sharp$ occurrences du couple $t(i) - t(k)$ dans le corpus
$C(w(i), t(k))$	$\sharp$ occurrences de $w(i)$ étiquetées $t(k)$ dans le corpus
w_i^j, t_i^j	séquences de mots (étiquettes) $w_i \dots w_j (t_i \dots t_j)$

- On retrouve les notations markoviennes classiques.
- Subtilité : projection dans la séquence d'une part et sur un dictionnaire d'autre part.
- Questions :
  - Quels sont les états, les observations ?
  - Quelle hypothèse faire sur l'automate ?

Modélisation (2)

- On peut partir d'un modèle ergodique (complètement connecté)
- Modèle MMC (d'ordre 1)
  - Etats : étiquettes

$$p(t_{i+1}|t_1^i) = p(t_{i+1}|t_i), \quad p(t_i|t_j) = \frac{C(t_i, t_j)}{C(t_j)}$$

  - Observation : mots (par rapport aux états=étiquettes)

$$p(w(i)|t(j)) = \frac{C(w(i), t(j))}{C(t(j))}$$

  - Etiquetage optimal (= étiquettes les plus vraisemblables)

$$\begin{aligned} \arg\max_{t_1^n} p(t_1^n|w_1^n) &= \arg\max_{t_1^n} p(w_1^n|t_1^n)p(t_1^n) \\ &= \arg\max_{t_1^n} \prod_{i=1}^{n} p(w_i|t_i)p(t_i|t_{i-1}) \end{aligned}$$

Différences avec les MMC classiques

- Différences :
  - ici, on utilisera des corpus étiquetés au niveau mot
  - ce sont donc des modèles de markov « visibles »
  - on peut également mixer des données non étiquetées et des données étiquetées : apprentissage semi-supervisé
- Algorithme d'apprentissage
  - pour tous les tags $t(i), t(j)$, pour tous les mots $w(i)$:

$$p(t_i|t_j) = \frac{C(t_i, t_j)}{C(t_j)}, \quad p(w(i)|t(j)) = \frac{C(w(i), t(j))}{C(t(j))}$$

- Intérêt principal : le décodage
  - Utilisation de Viterbi, estimation des :

$$\delta_i(j) = \max_{t_1^{i-1}} p(t_1^{i-1}, t_i = j, w_1^i|\lambda), \quad \text{Init} : \delta_i(.) = 1,\ \delta_i(x \neq .) = 0$$

Quelle est la probabilité de l'étiquette j et de l'ensemble du passé (mots + étiquetage)

Reconnaissance de paroles / écrits

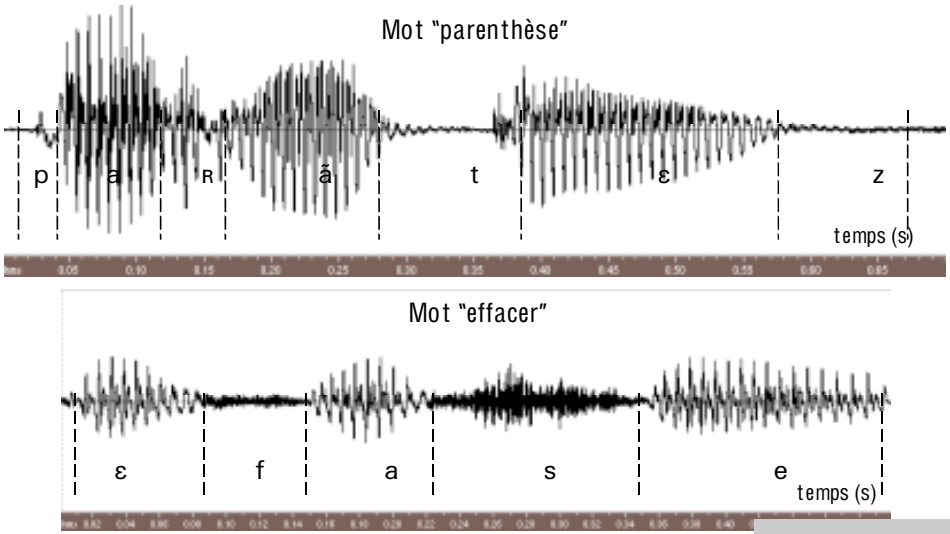
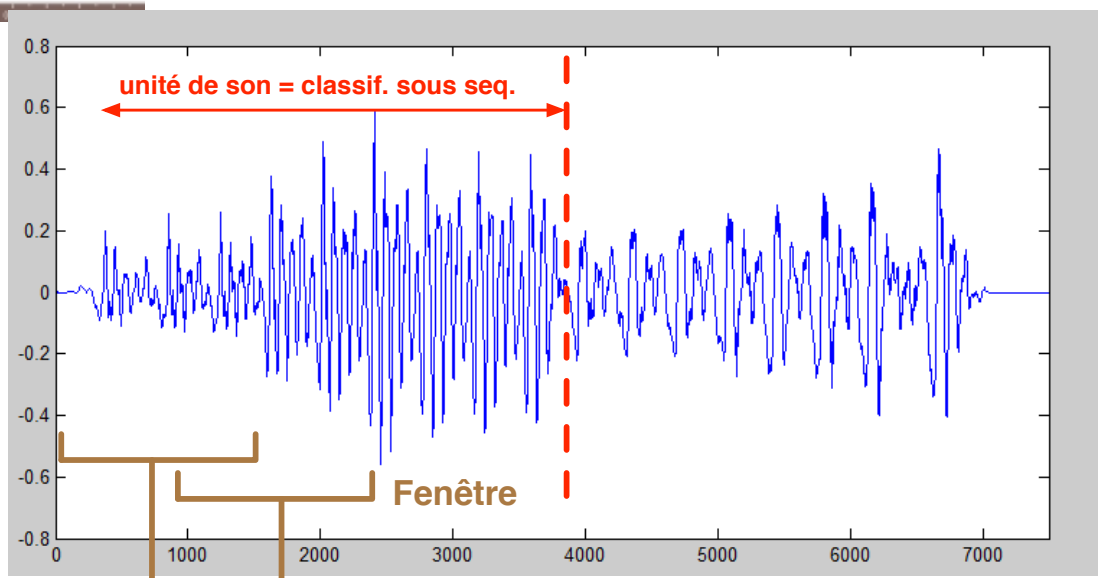


Fig. 1.2 Audiogramme de signaux de parole.

Crédit : Thierry Dutoit (Univ. Mons)

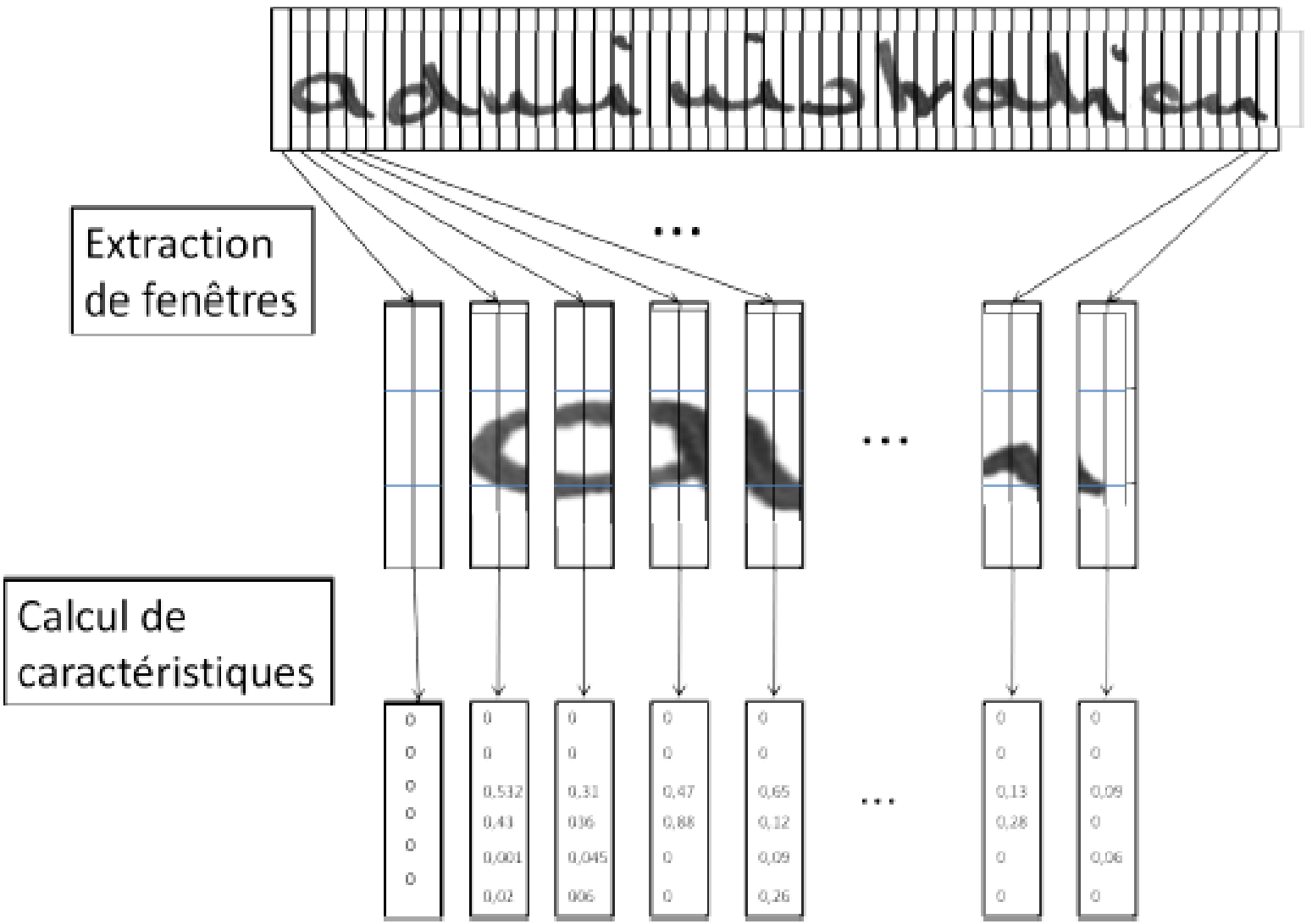
Tâche de segmentation + reconnaissance



Pré-traitements et mise en forme

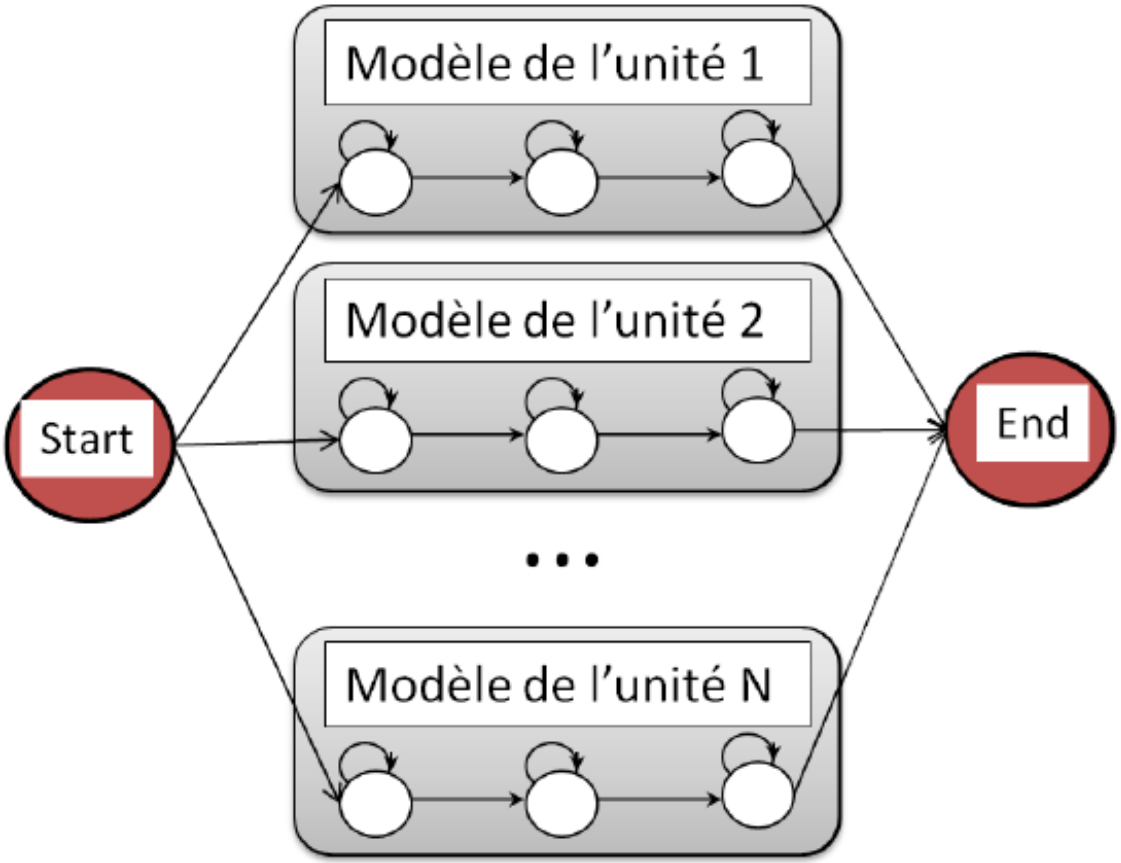
- Caractéristiques fréquentielles
- Puissance
- ...

Reconnaissance de paroles / écrits



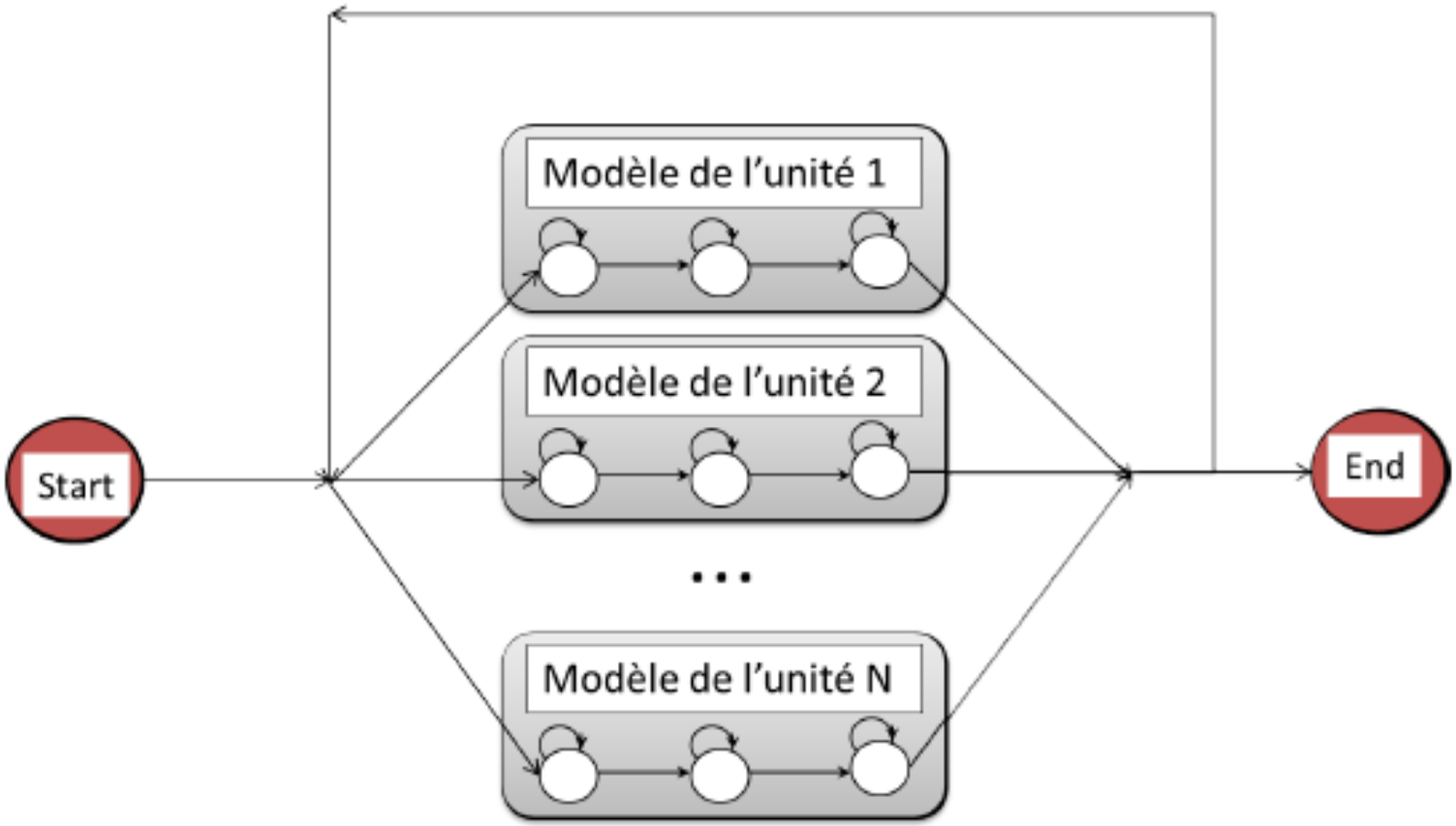
Architecture

- 1 modèle par unité de son (e.g. dictionnaire phonétique)

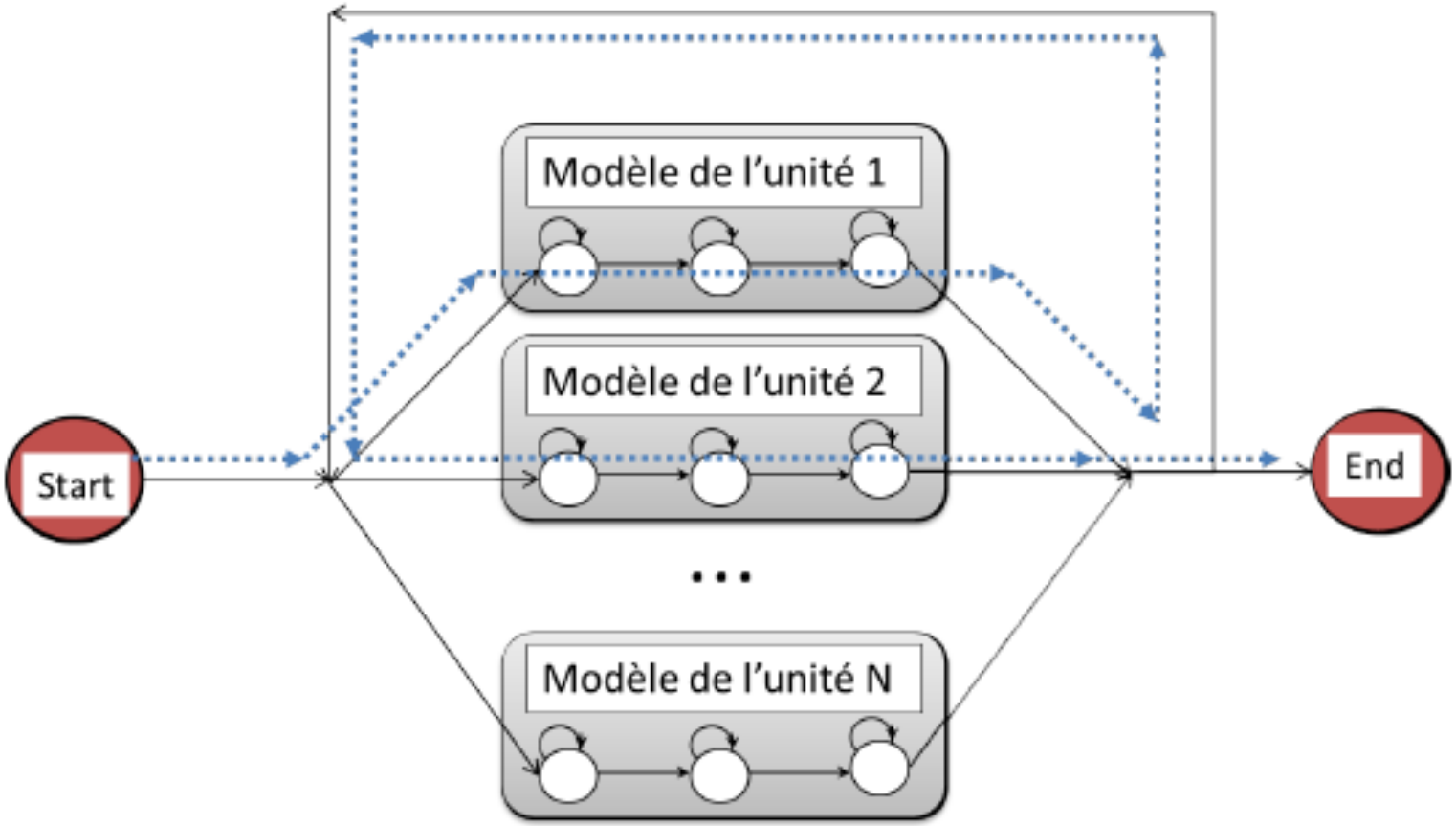


Architecture

- ... Et re-bouclage après détection

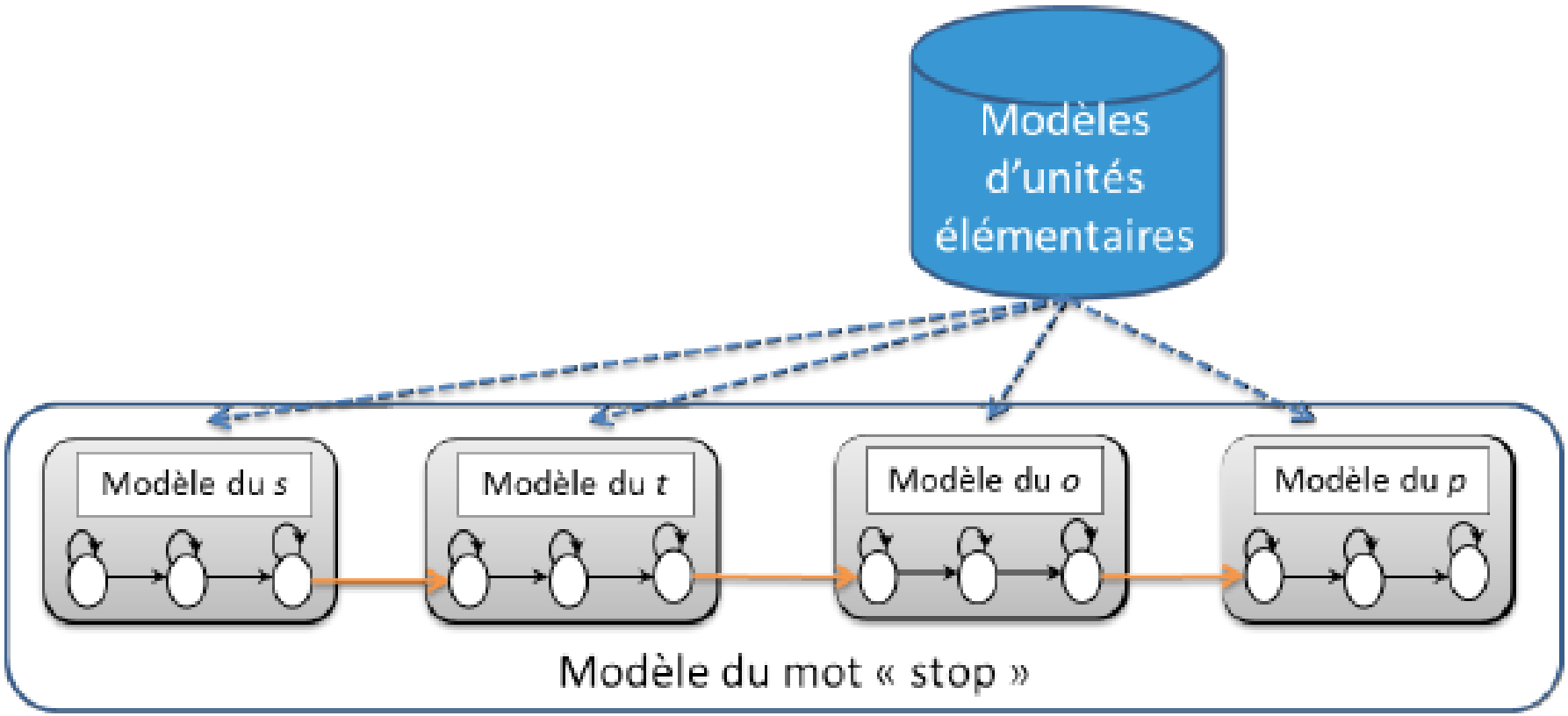


- ... Modèle cyclique



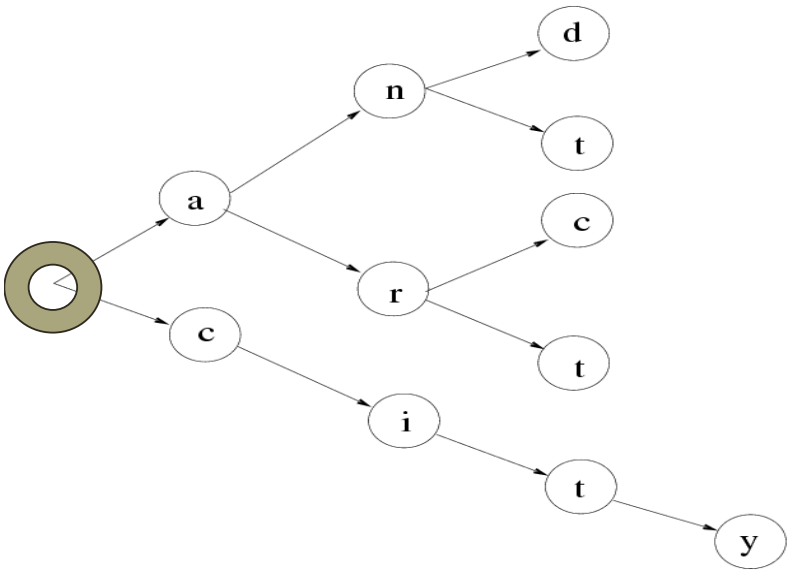
Architecture

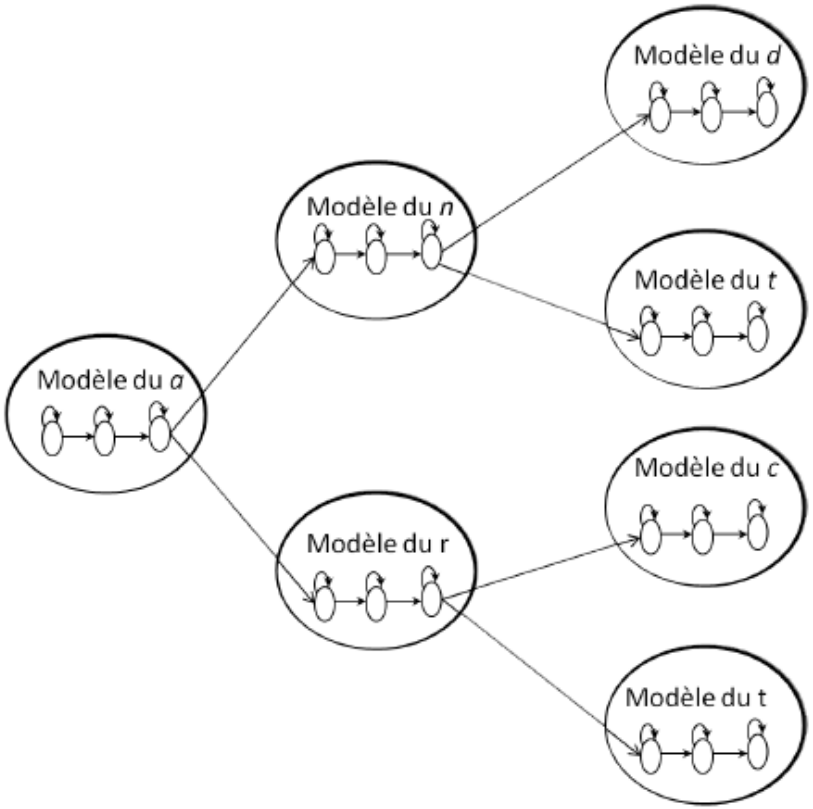
- Apprentissage classique :



Post-processing (affinage)

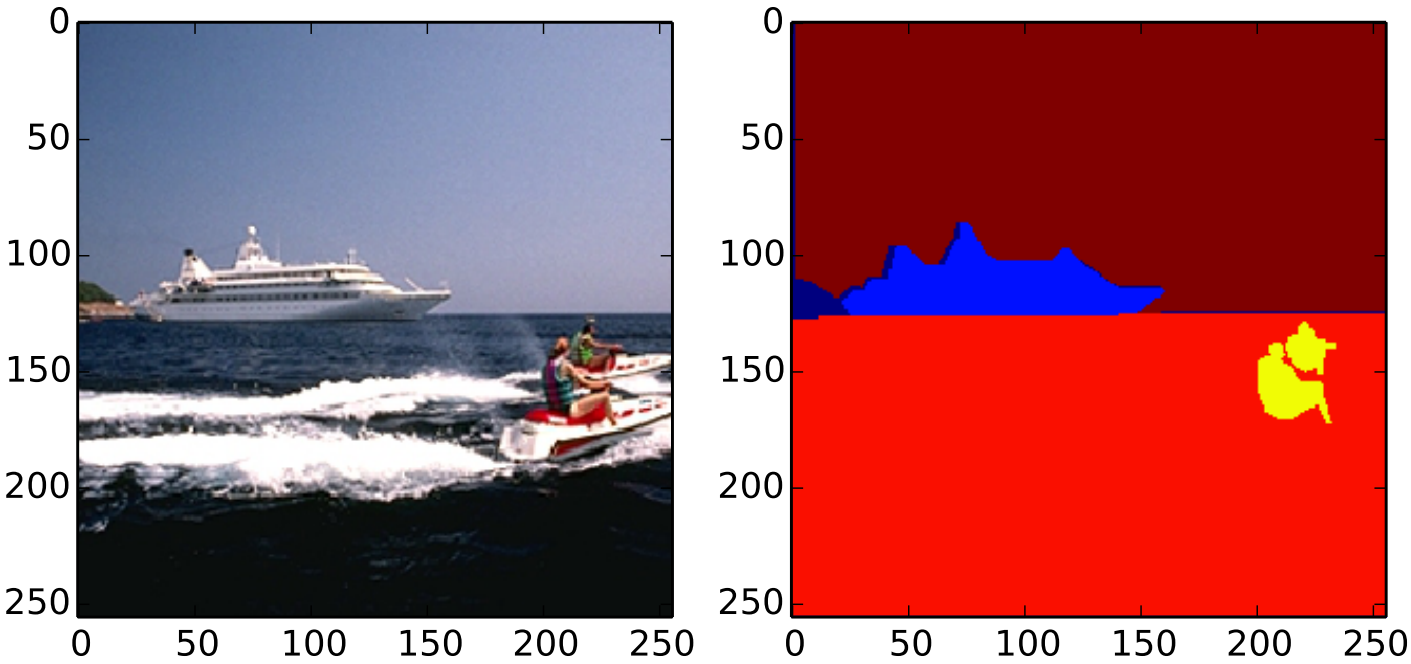
- Soft
  - Stats d'enchainements de caractères (bi-trigrams)
- Hard
  - Utilisation d'une structure d'arbre préfixé
  - + Stratégie d'élagage Beam-search (autour du meilleur)
    - la valeur du beam (nb de faisceaux),





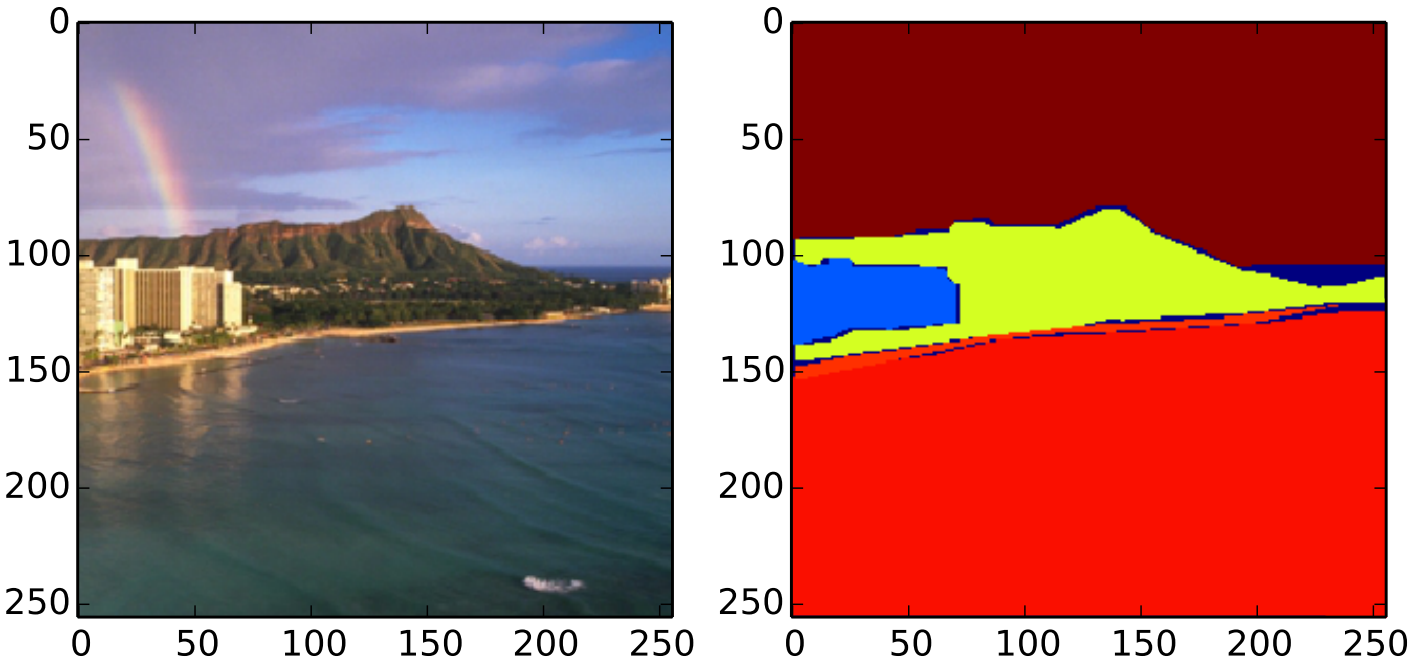
Segmentation image

- Bases étiquetées : exemple Sift flow

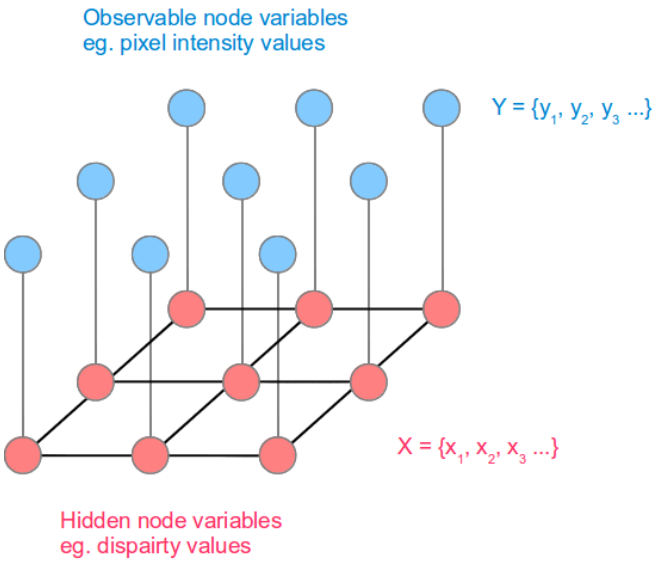


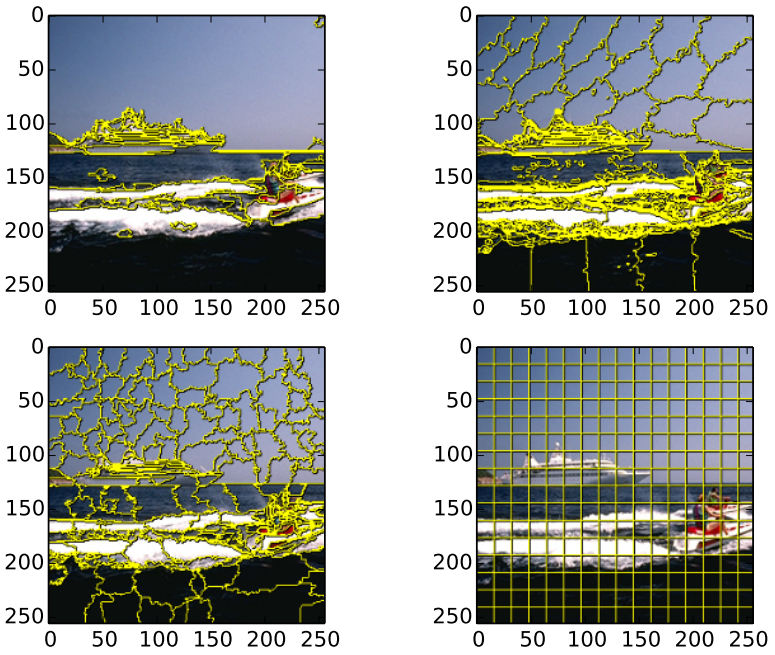
Segmentation image

- Bases étiquetées : exemple Sift flow



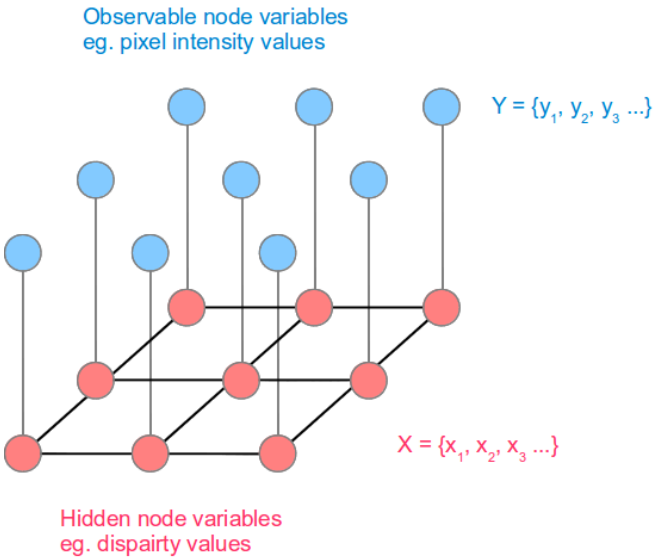
Réseau de Markov (chaine 2D)

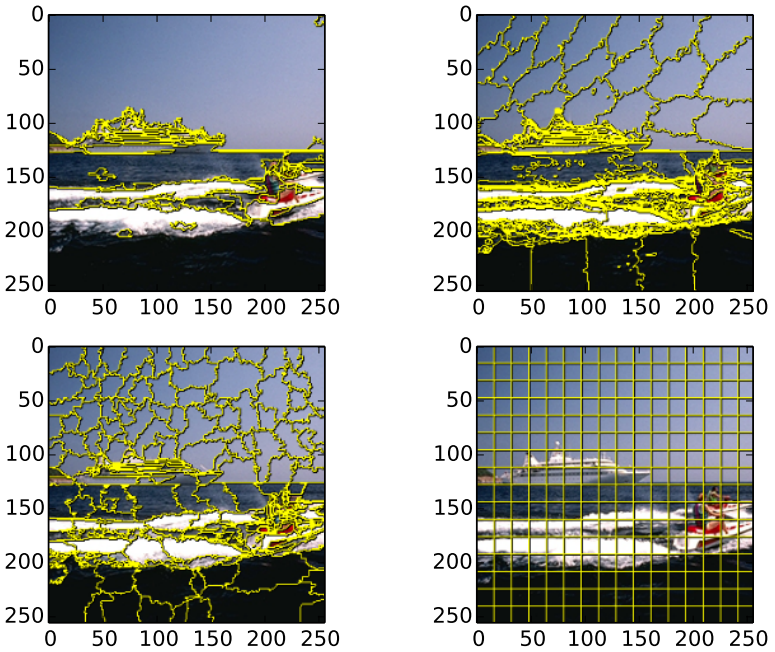




- Est-il possible de définir un modèle de markov 2D ?

Réseau de Markov (chaine 2D)





- Est-il possible de définir un modèle de markov 2D ?
  - Matrice de transition très complexe (même en simplifiant)
  - Viterbi ingérable

Proposition 1

- Utilisation de l'échantillonnage de Gibbs (cas particulier de MCMC, cf semaine prochaine)

1. Initialise $x_{0,1:n}$.
2. For $i = 0$ to $N-1$
   - Sample $x_1^{(i+1)} \sim p(x_1|x_2^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
   - Sample $x_2^{(i+1)} \sim p(x_2|x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.

     $\vdots$
   - Sample $x_j^{(i+1)} \sim p(x_j|x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.

     $\vdots$
   - Sample $x_n^{(i+1)} \sim p(x_n|x_1^{(i+1)}, x_2^{(i+1)}, \ldots x_{n-1}^{(i+1)})$.

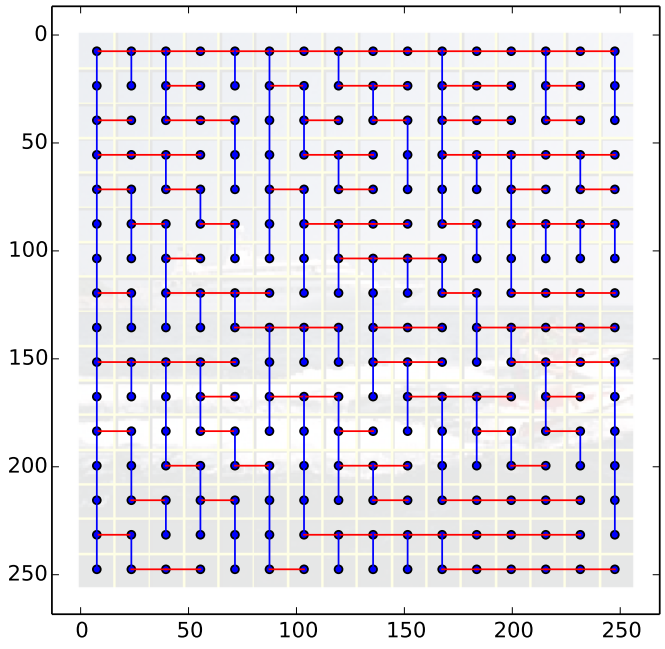
- Générer des points selon une distribution jointe complexe

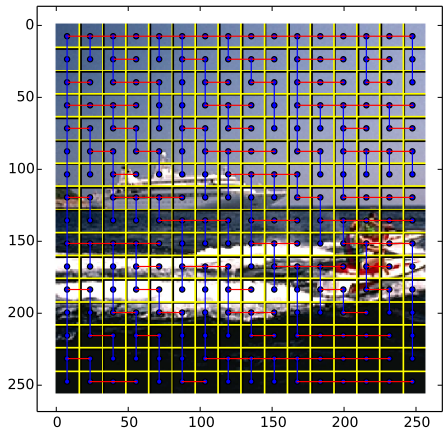
- Génération d'une population
- Statistique sur les états (avec une hypothèse d'indépendance pour relacher les contraintes)

An Introduction to MCMC for Machine Learning
Andrieu, De Freitas, Doucet, Jordan

Proposition 2

- Définir un arbre aléatoire dans l'image





- Utiliser une variante de viterbi dans les arbres...
- Et itérer le processus !

Proposition 3 : Conditional Random Fields

- Séquence/champs de mots/pixels $\mathbf{x} = \{x_1, \ldots, x_T\}$
- Sequence/champ d'étiquettes $\mathbf{s}$

Introduction de fonctions de scoring basées sur des descripteurs structurés :

$$score(\mathbf{s}, \mathbf{x}) = \sum_{j \in \mathcal{C}} \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}, t, s_t, s_{t-1})$$

Si on préfère des probabilités (comme pour la régression logistique) :

$$p(\mathbf{s}, \mathbf{x}) = \frac{1}{Z} \exp(score(\mathbf{s}, \mathbf{x})) \text{ et : } p(\mathbf{s}|\mathbf{x}) = \frac{\exp(score(\mathbf{s}, \mathbf{x}))}{\sum_{\mathbf{s}'} \exp(score(\mathbf{s}', \mathbf{x}))}$$

Mc callum `http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf`
`http://pages.cs.wisc.edu/~jerryzhu/cs838/CRF.pdf`

Apprentissage CRF

Critère à optimiser : vraisemblance conditionnelle (cf regression logistique)

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}, t, s_t, s_{t-1}) \right\}$$

$$\mathcal{L}_{cond} = \sum_n \log p(\mathbf{s}^n|\mathbf{x}^n) = \sum_n \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}^n, t, s_t^n, s_{t-1}^n) - \sum_n \log(Z(\mathbf{x}_n))$$

Comment optimiser ?

Apprentissage CRF

Critère à optimiser : vraisemblance conditionnelle (cf regression logistique)

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}, t, s_t, s_{t-1}) \right\}$$

$$\mathcal{L}_{cond} = \sum_n \log p(\mathbf{s}^n|\mathbf{x}^n) = \sum_n \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}^n, t, s_t^n, s_{t-1}^n) - \sum_n \log(Z(\mathbf{x}_n))$$

Comment optimiser ?

Montée de gradient $\frac{\partial \mathcal{L}}{\partial \lambda_j}$

$$\lambda_j \leftarrow \lambda_j + \sum_{n,t} f_j(\mathbf{x}^n, t, s_t^n, s_{t-1}^n) - \sum_{n,t} \sum_{s'_t, s'_{t-1}} f_j(\mathbf{x}^n, t, s'_t, s'_{t-1}) p(s'_t, s'_{t-1}|\mathbf{x}^n)$$

Apprentissage CRF

Critère à optimiser : vraisemblance conditionnelle (cf regression logistique)

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}, t, s_t, s_{t-1}) \right\}$$

$$\mathcal{L}_{cond} = \sum_n \log p(\mathbf{s}^n|\mathbf{x}^n) = \sum_n \sum_j \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}^n, t, s_t^n, s_{t-1}^n) - \sum_n \log(Z(\mathbf{x}_n))$$

Comment optimiser ?

- La difficulté réside essentiellement dans le facteur de normalisation

CRF : bilan

- Un formalisme discriminant très performant...
- Mais un cout d'inférence élevé

$$score(\mathbf{s}, \mathbf{x}) = \sum_{j \in \mathcal{C}} \sum_{t=1}^{T} \lambda_j f_j(\mathbf{x}, t, s_t, s_{t-1})$$

La fonction de scoring n'est pas triviale