Reconnaissance des formes

Cours 8

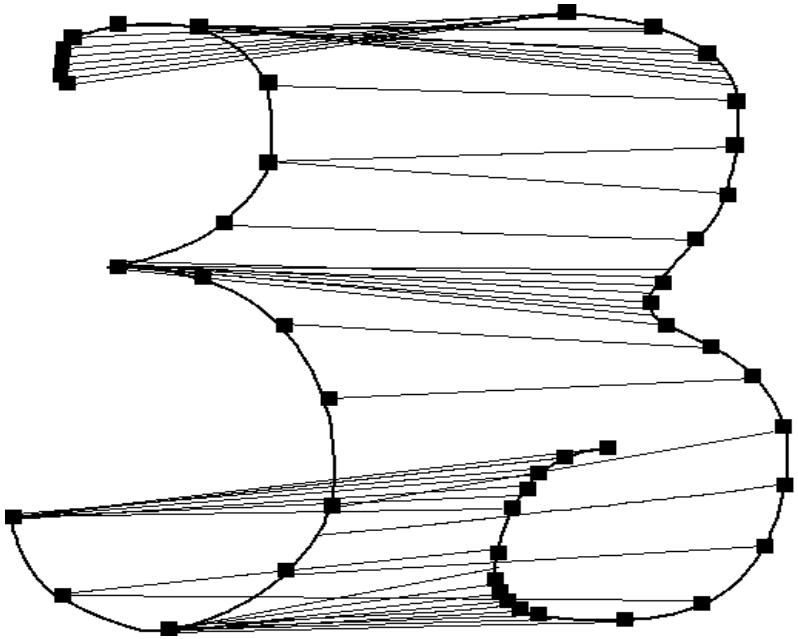
Modèles de Markov Cachés
(HMM)

Plan

- Comparaison élastique
- Programmation dynamique
- Application à la reconnaissance de mots
- Modélisation stochastique
- Modèle de Markov caché

Méthodes de comparaison
Comparaison élastique

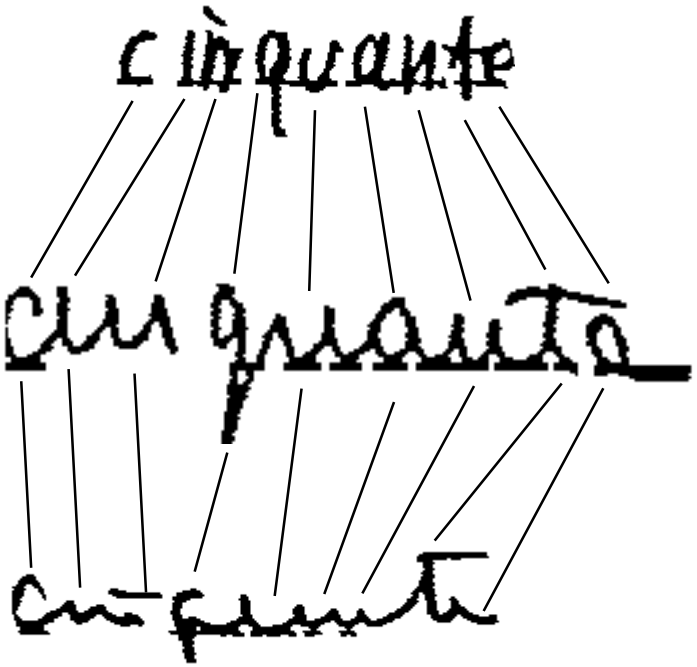
- Rigidité de la méthode directe
  - Problèmes d'optimisation non linéaires
- Recherche de la meilleure correspondance
  - Optimiser un critère global de ressemblance
- Appariement souple ou élastique



Méthodes de comparaison
Comparaison élastique

- Nécessité absolue pour les mots

Modèle

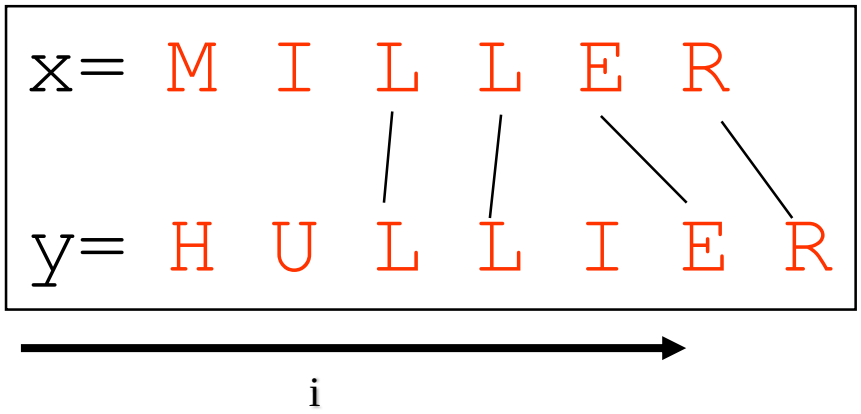


Programmation dynamique

- *Principe*
  - La programmation dynamique est utilisée en comparaison d'objets quand les objets sont décomposables élément par élément
  - A chaque étape, elle compare deux éléments (un dans le modèle et un dans l'échantillon) et essaie d'optimiser le coût à chaque progression
  - Cette optimisation n'est possible que si le problème est décomposable en étapes et si le coût est progressif

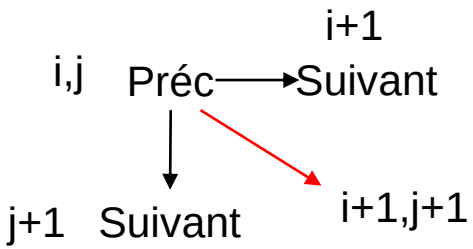
Programmation dynamique

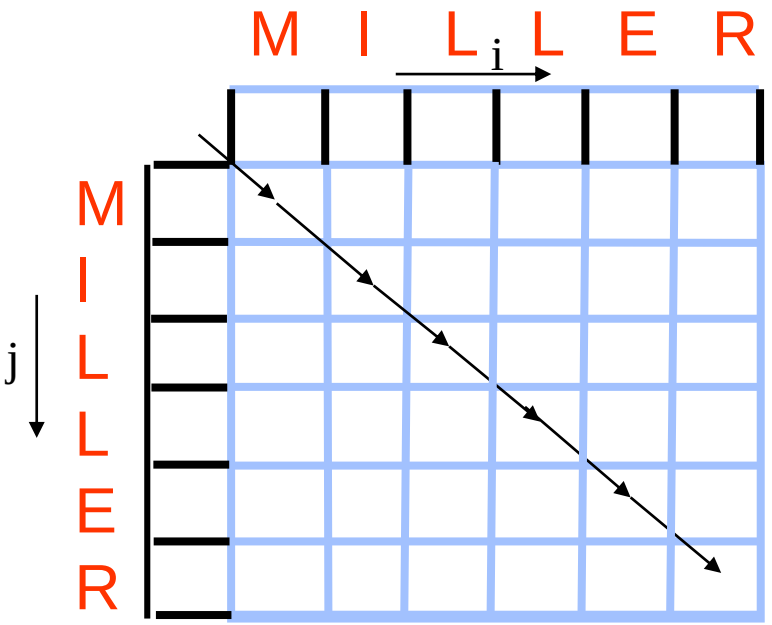
- En chaque étape : Levinston propose :
  - 3 opérations locales (ou symbole) pour réaliser la comparaison :
    - Remplacement ou substitution
    - Ajout
    - Suppression
  - La distance (d'édition) ou le coût C(i) = C(i-1) + min (C(R), C(A), C(S))



Programmation dynamique

- Wagner et Fisher proposent un graphe de comparaison
  - Construction par substitution
    - On avance dans i et dans j





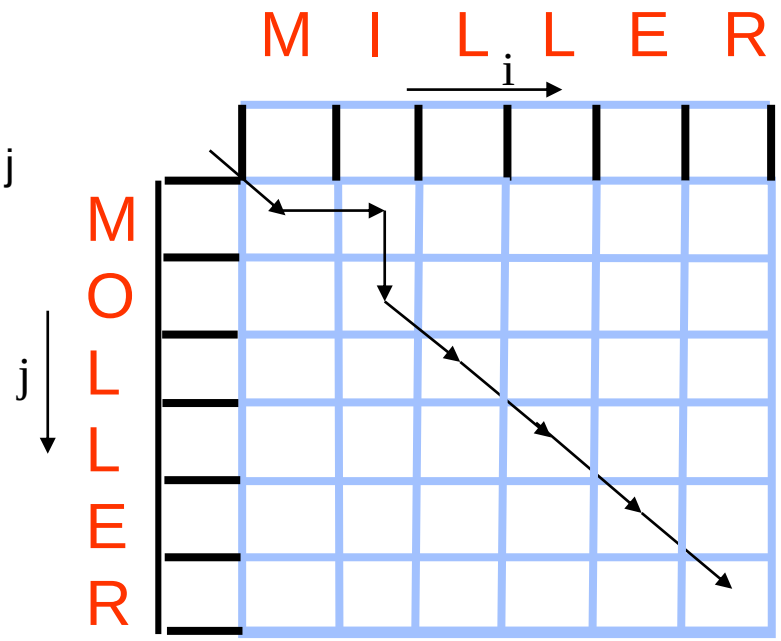
Programmation dynamique

- Construction par ajout :
  - On avance dans i (on accepte i)

$$i \longrightarrow i+1$$
$$j \longrightarrow j$$

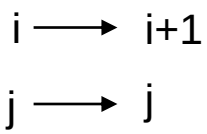
  - puis on avance ensuite dans j (on supprime le o)

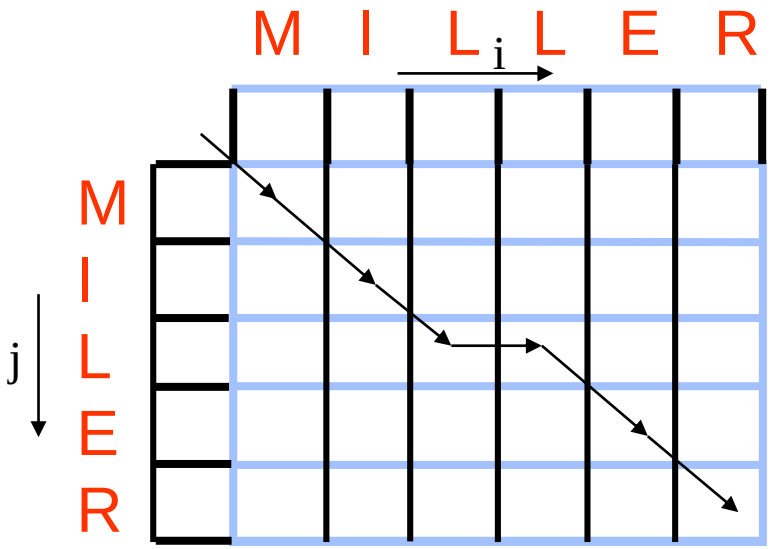
$$i \longrightarrow i$$
$$j \longrightarrow j+1$$



Programmation dynamique

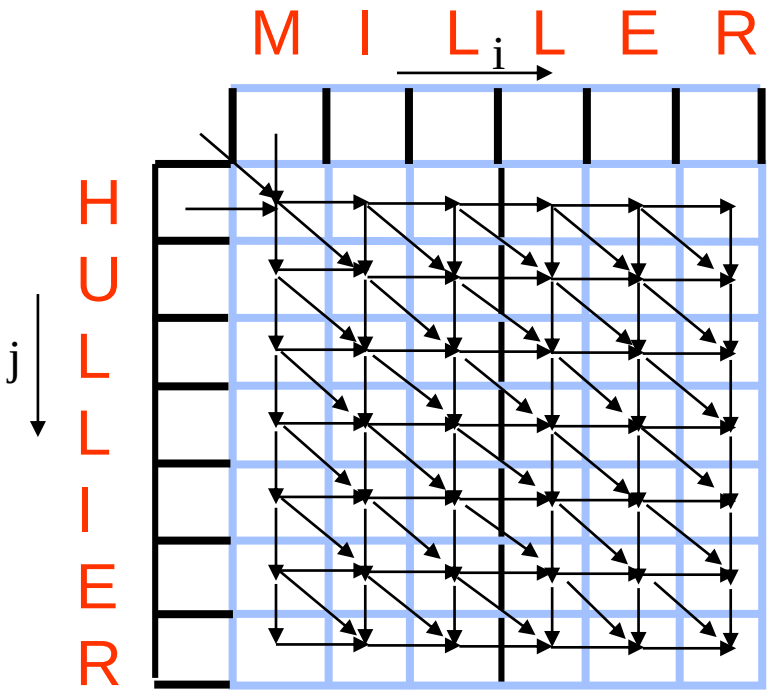
- Construction par suppression :
  - On peut aussi supprimer dans i





Programmation dynamique

- Construction dans le cas général
  - On prévoit toutes les possibilités :
    - Substitution
    - Ajout
    - Suppression
  - Puis on prend le chemin minimum



Programmation dynamique

- Algorithme de Wagner et Fisher en O(m.n) :

On note m et n les longueurs respectives des mots

d(0,0)=0;

Pour i variant de 1 à n effectuer :

$d(i,0)= d(i-1,0) + c(a_i,\lambda)$

Pour j variant de 1 à m effectuer :

$d(0,j)= d(0,j-1) + c(\lambda, b_j)$

} Initialisation de la distance

Pour i variant de 1 à n effectuer :

Pour j variant de 1 à m effectuer :

$d_1 = d(i-1,j-1) + c(a_i,b_j)$

$d_2 = d(i-1,j) + c(a_i,\lambda)$

$d_3 = d(i,j-1) + c(\lambda,b_j)$

$d(i,j)=\min(d_1, d_2, d_3)$

} Développement de la distance

La distance d'édition est le résultat final de l'algorithme c(m,n)

Programmation dynamique

- Exercice :
  - Soient deux listes X={a a b a c} et Y={a b d}, et les coûts :
  - $c(\lambda,y_k) = 0.5\ \forall k$, $c(x_r,\lambda) = 0.5\ \forall r$, $c(x_r,y_k) = 0\ \forall r,k$ si $x_r = y_k$ et $c(x_r,y_k) = 1\ \forall r,k$ si $x_r \neq y_k$
  - Construire le tableau des distances, puis calculer la distance totale entre les deux chaînes

		0	1	2	3	4	5
	X / Y	λ	a	a	b	a	c
0	λ	?	?	?	?	?	?
1	a	?	?	?	?	?	?
2	b	?	?	?	?	?	?
3	d	?	?	?	?	?	?

Programmation dynamique

- Solution :
  - $c(\lambda,y_k) = 0.5\ \forall k$
  - $c(x_r,\lambda) = 0.5\ \forall r$
  - $c(x_r,y_k) = 0\ \forall r,k$ si $x_r = y_k$
  - $c(x_r,y_k) = 1\ \forall r,k$ si $x_r \neq y_k$

$d_1 = d(i-1,j-1) + c(a_i,b_j)$

$d_2 = d(i-1,j) + c(a_i,\lambda)$

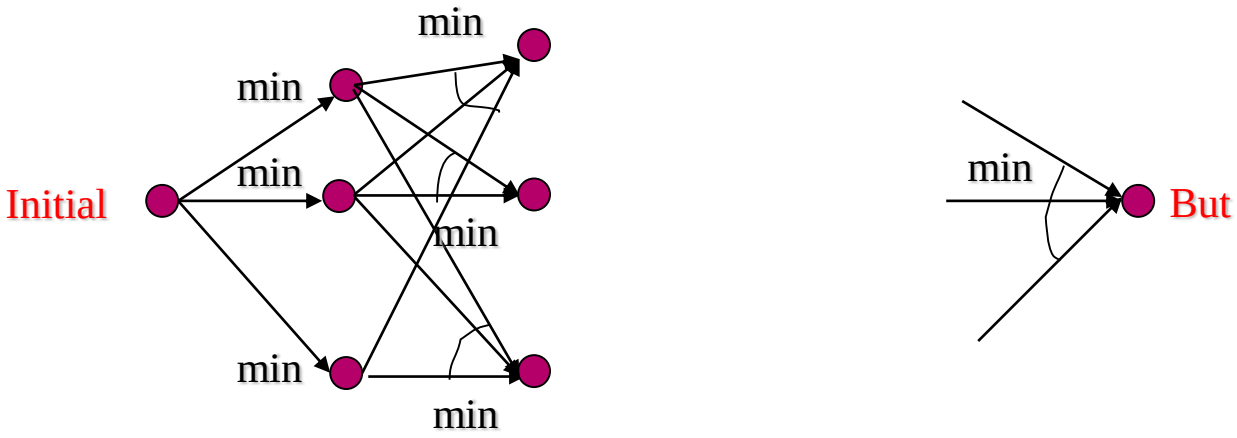
$d_3 = d(i,j-1) + c(\lambda,b_j)$

$d(i,j)=\min(d_1, d_2, d_3)$

X Y	j	a	a	b	a	c
i	0	0.5	1	1.5	2	2.5
a	0.5	0	0	1	1	2
b	1	1	1	0	1	2
d	1.5	2	2	1	1	2

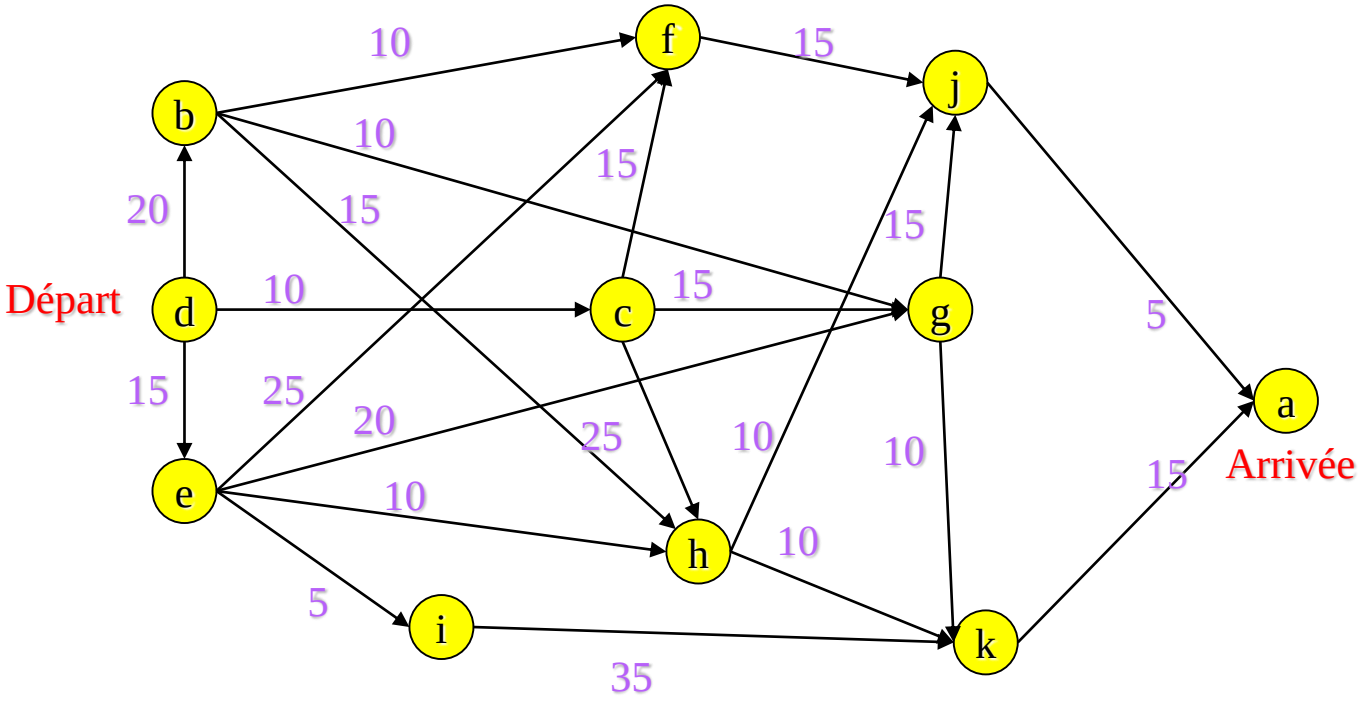
Programmation dynamique

- *Autre problème : recherche de chemin minimum entre 2 villes*
  - *Comment le faire par programmation dynamique ?*
    - *En cherchant le minimum à chaque progression jusqu'au but*



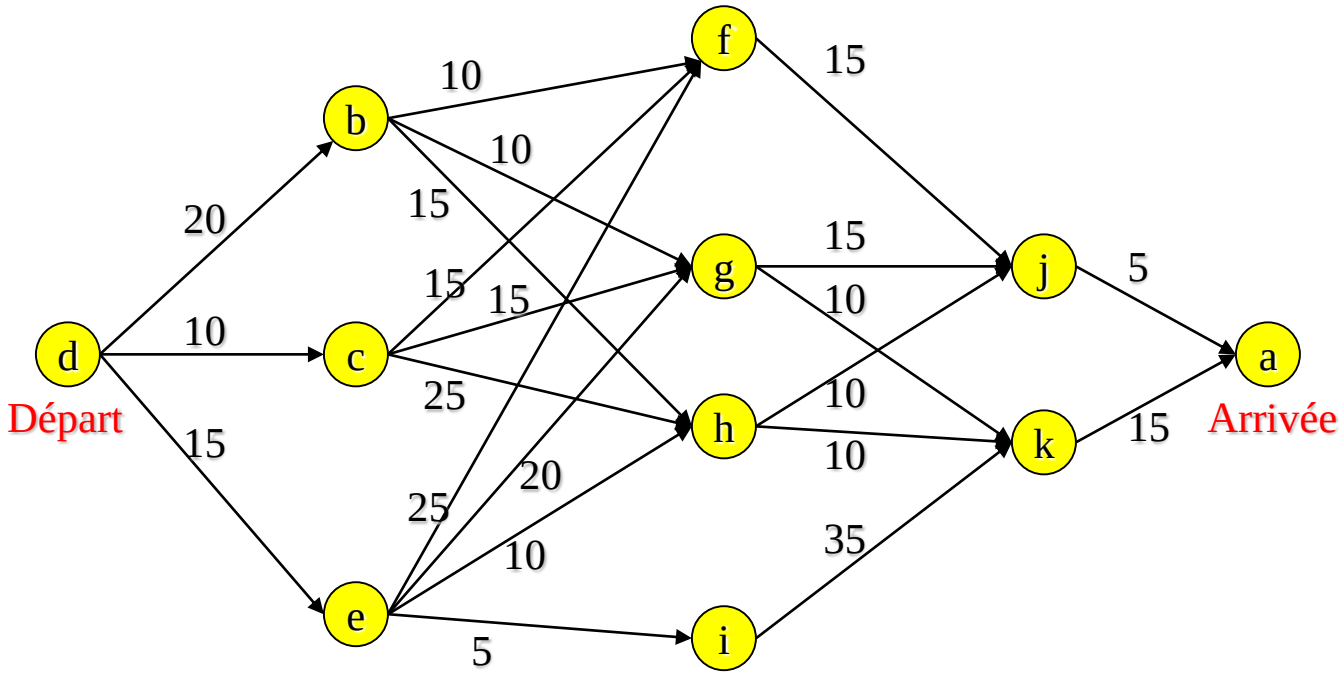
Recherche heuristique
Comparaison dynamique

- *Exercice*
  - Utiliser l'algorithme de programmation dynamique pour trouver le chemin le plus court entre deux villes



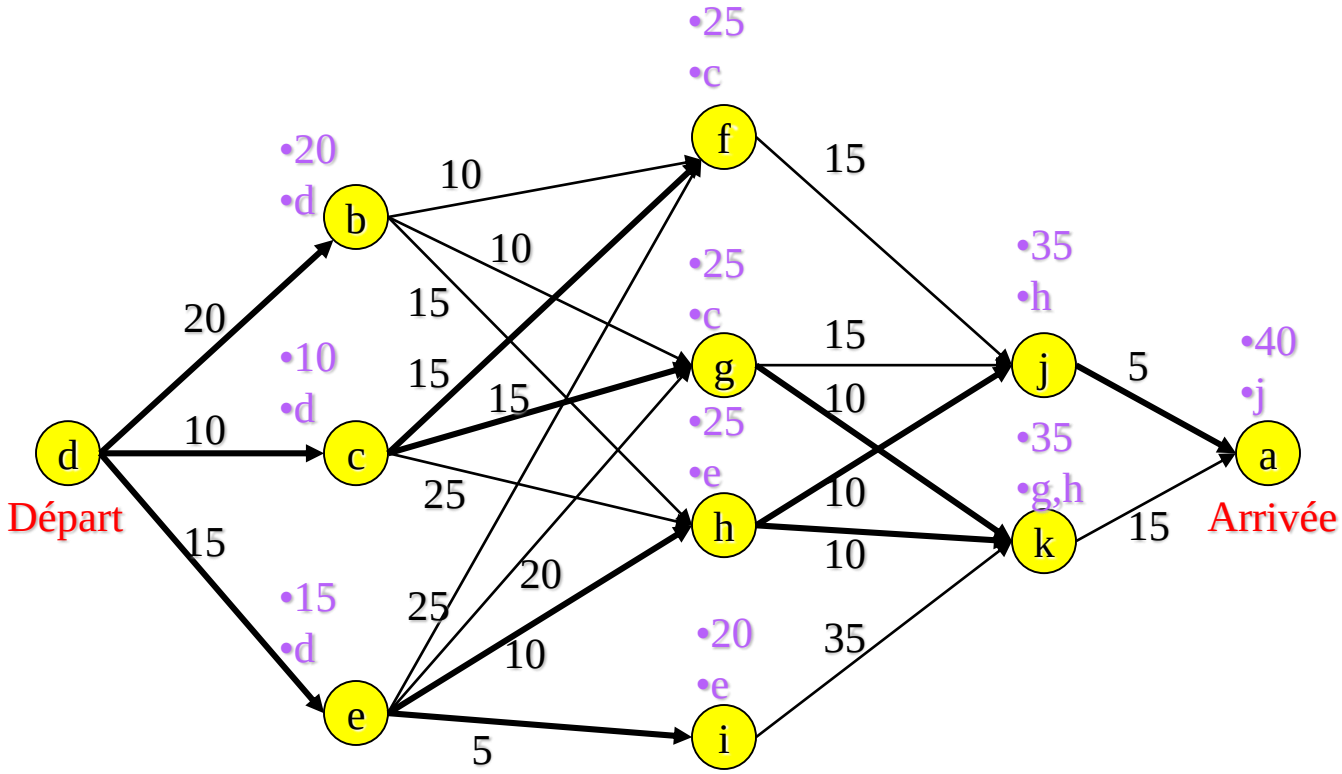
Recherche heuristique
Comparaison dynamique

- *Préparation de la solution*
  - *On construit l'arbre des connexions, si possible par niveau de progression*



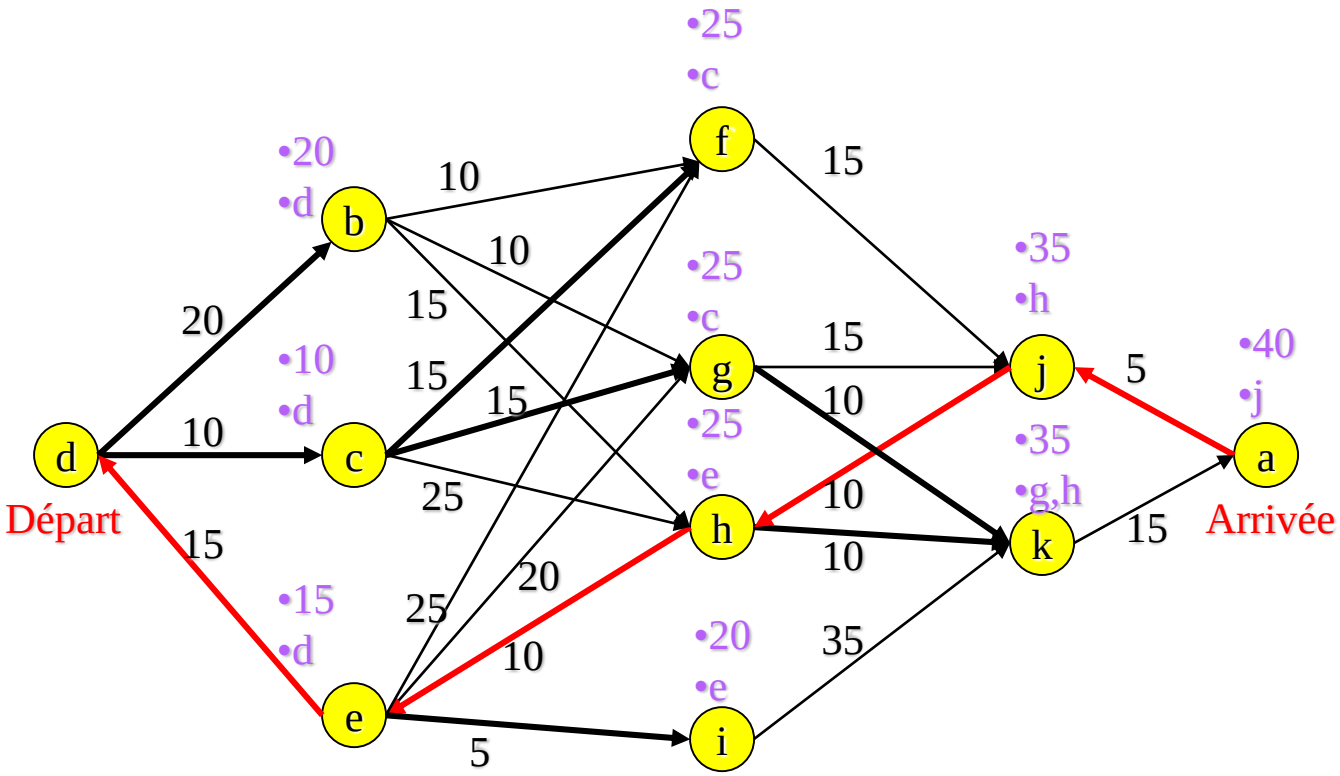
Recherche heuristique
Comparaison dynamique

- *Solution : Étape 1 :*
  - *On construit le minimum par ville en indiquant le prédécesseur amenant le minimum (flèche noire)*



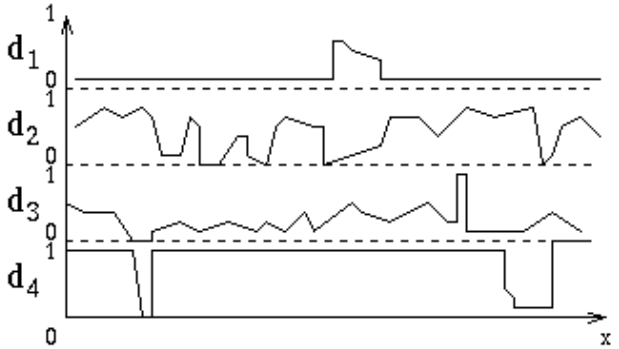
Recherche heuristique
Comparaison dynamique

- *Solution : Étape 2 :*
  - *On remonte le chemin de coût minimum en utilisant les états stockés par ville*



Application à la reconnaissance de mots manuscrits

- Plusieurs problèmes
  - Problème 1 : décomposition du signal d'écriture
    - Comment faire de cette image une liste de « caractéristiques » sur lesquelles je peux opérer la comparaison dynamique ?
    - L'idéal est de la segmenter en une suite de lettres ! puis de la comparer au modèle (comme vu)



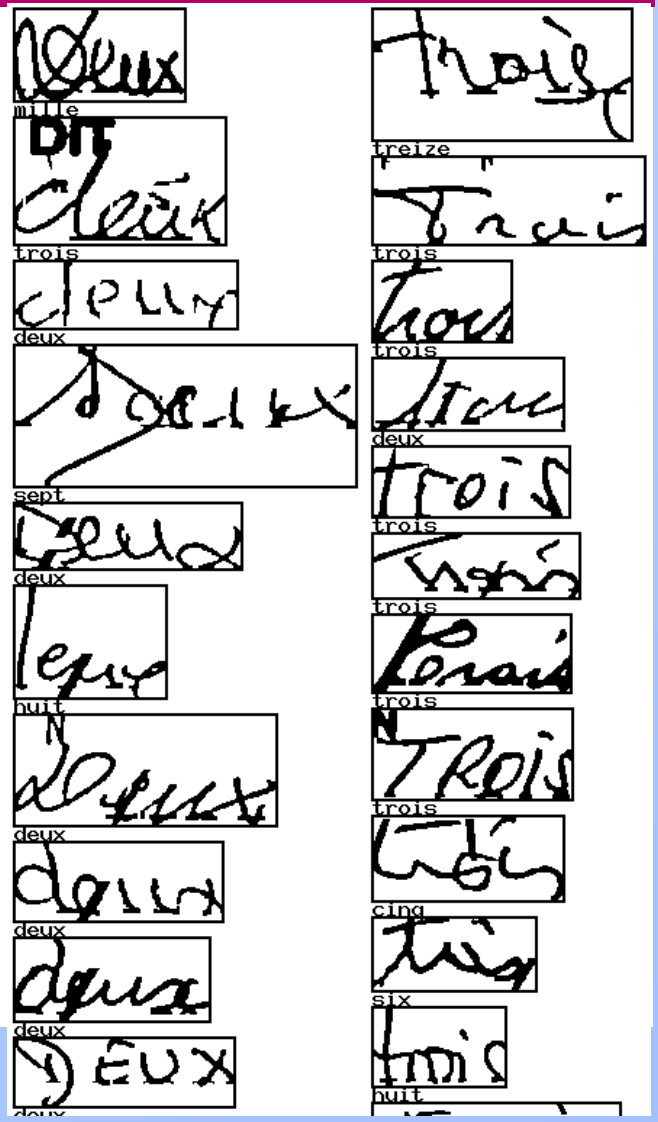
Pas facile ➔ Dilemme de Sayre

Application à la reconnaissance de mots manuscrits

- Exemples de segmentations plus ou moins réussies

Application à la reconnaissance de mots manuscrits

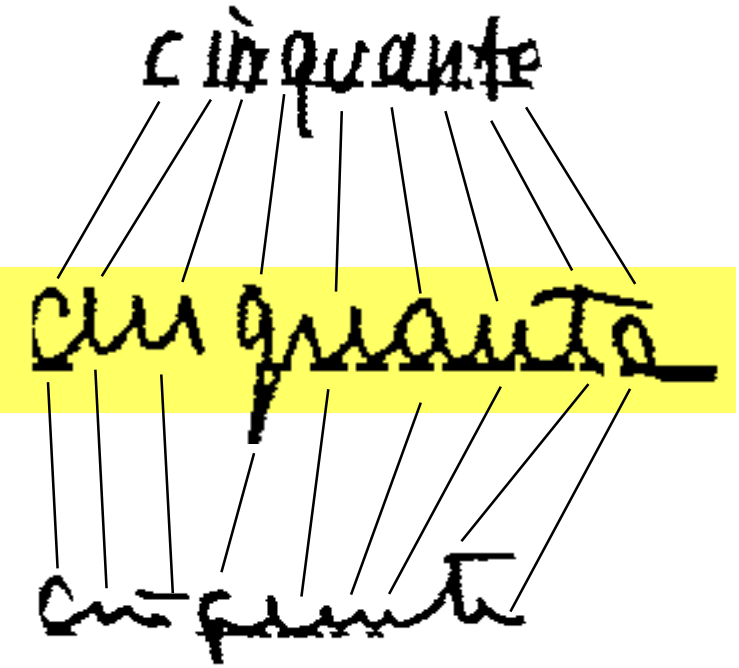
- Difficultés de l'application
  - Mots de montants



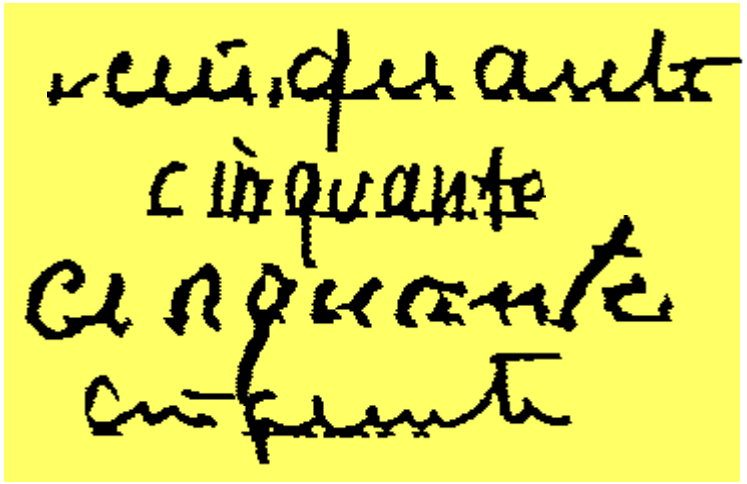
Application à la reconnaissance de mots manuscrits

Problème 2 :

- Comment faire pour tenir compte de toutes les écritures ?
- Comment créer un modèle pour représenter toutes les variations ?

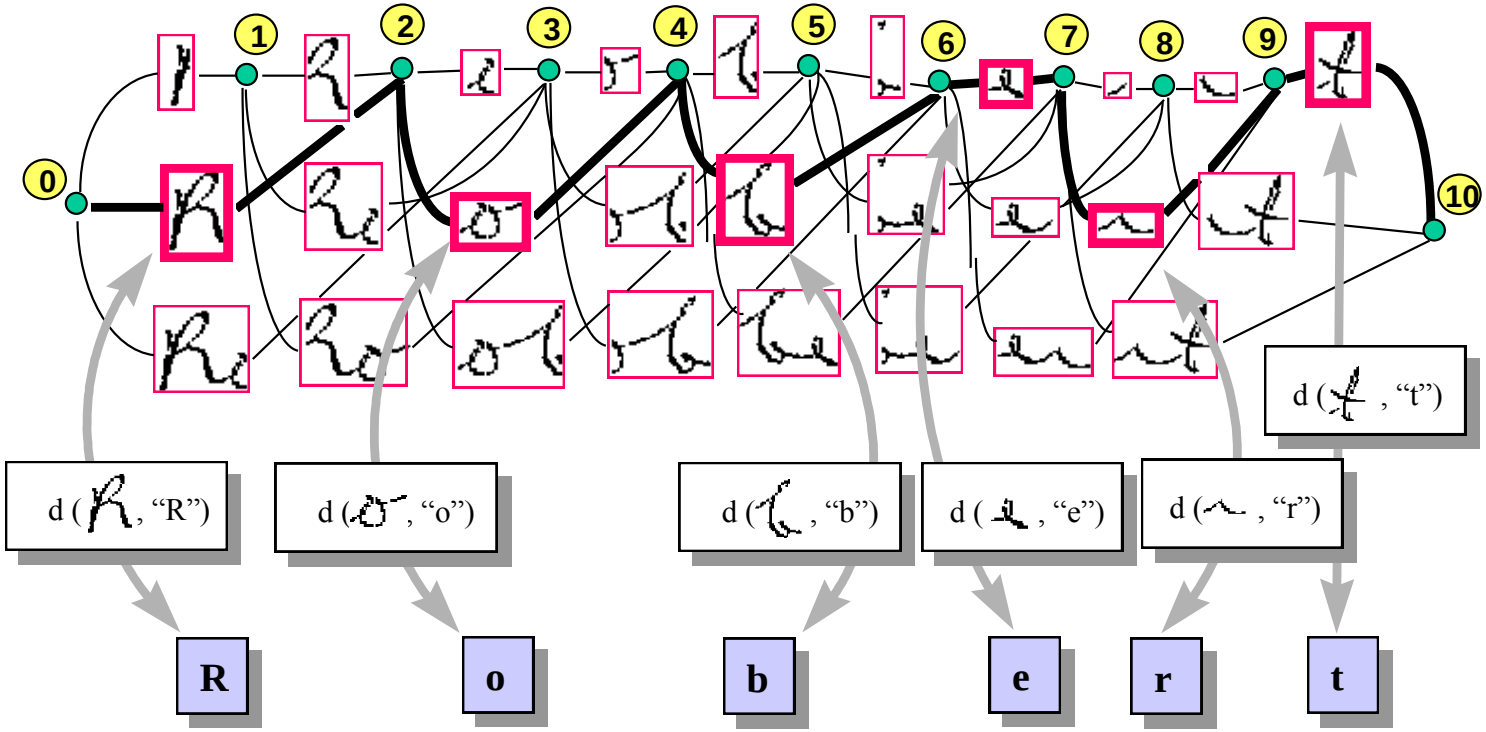


Modèle



Modèle ?

Solution 1 : comparaison à un dictionnaire



- On prend toutes les décompositions possibles du mot, puis chaque regroupement est comparé à chaque entrée du dictionnaire
- On retient la séquence de regroupement la plus pertinente

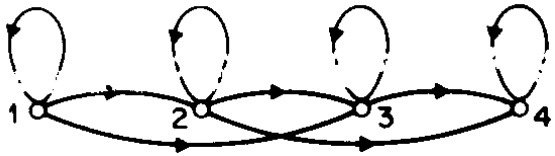
Solution 2 : on crée un modèle regroupant toutes les variations

- Comment faire ?
  - Pour que les primitives les plus fréquentes soient signalées
  - Pour que la liste des primitives la plus représentative de tous ces échantillons soit marquée…
  - Pour que si je présente un mot semblable, le modèle me dise j'ai appris quelque chose de semblable : j'ai un chemin identique (ou presque) ou que ce chemin est probablement identique à ceux que j'ai appris

Création d'un tel modèle de chemins par apprentissage ?

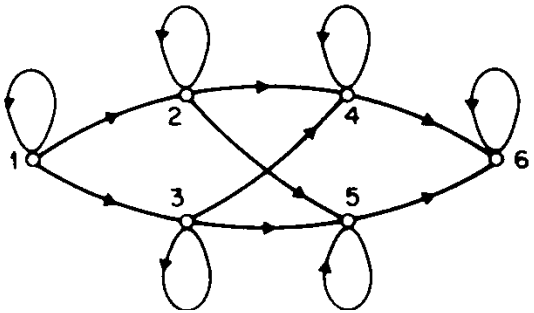
- Solution
  - On crée un graphe stochastique où les chemins représentent des voies probables de segmentation
  - Plusieurs architectures

Bakis

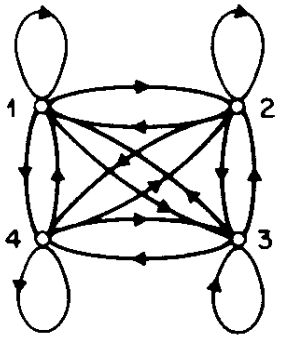


Variations uniquement dans la longueur

parallèle



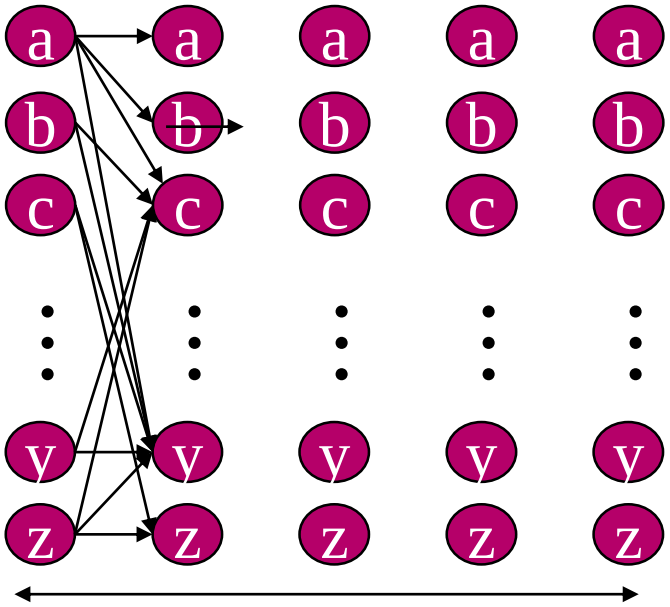
Ergodique



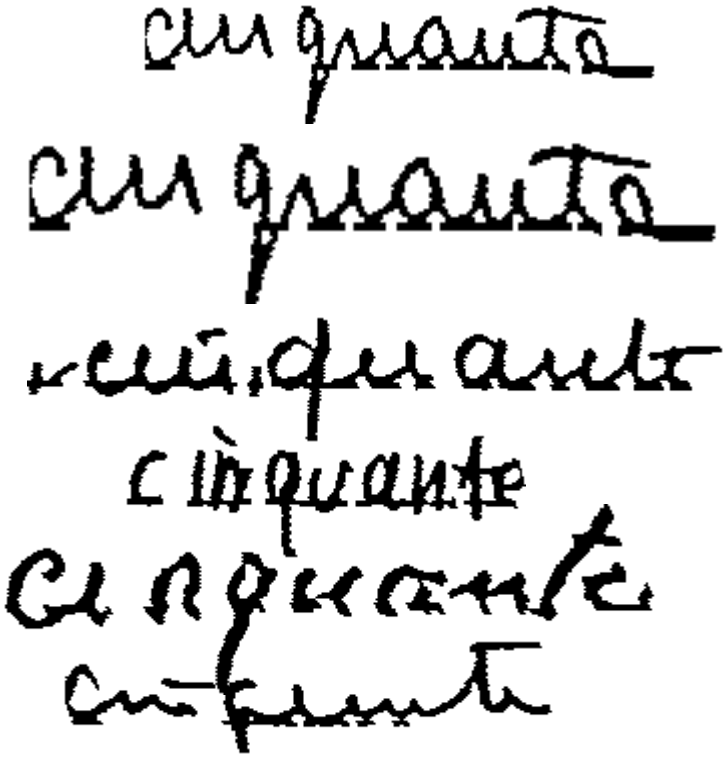
Variations dans la longueur et dans la composition

Création d'un tel modèle de chemins par apprentissage ?

- Idée 1 : observable
  - Créer un graphe complet de lettres avec des probabilités d'apparition d'une lettre à un rang donné

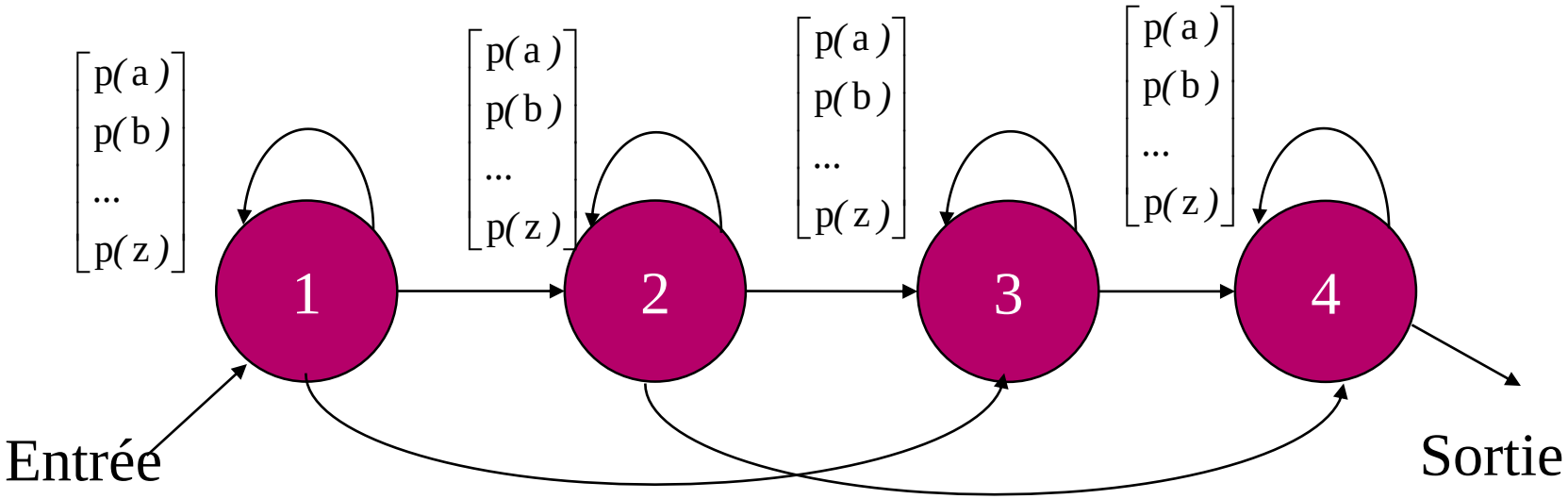


Longueur maximale des mots d'une langue



Création d'un tel modèle de chemins par apprentissage ?

- Idée 2 :
  - Créer un graphe plus réduit sans marquer nécessairement l'endroit d'observation d'une lettre
  - Les lettres peuvent être observées à n'importe quel état avec une probabilité
  - Les probabilités de transitions vont permettre de faire avancer la lecture et de trouver le bon état pour lire la lettre



Un modèle stochastique

- Le modèle est stochastique discret
  - Il associe à chaque état des probabilités d'observation
  - Il associe à chaque arc une probabilité de transition
- Le modèle est évolutif
  - Il évolue d'état en état
    - En évoluant dans ses observations, le système peut rester dans un même état ou passer dans un autre état en fonction de probabilités associées à l'état
  - Soient t= 1,2, …, T ces instants et s_t l'état à l'instant t
    - La probabilité d'observation dépend de tous les états précédents

$$P(s_1, s_2, \ldots, s_T) = P(s_1, s_2, \ldots, s_{T-1}) \times P(s_T / s_1, s_2, \ldots, s_{T-1}) \ldots$$

Un modèle stochastique

- On s'intéresse au modèle de Markov caché
  - défini par 2 suites
    - Suite cachée d'états : q_1, q_2, …, q_T, notée Q(1:T) où les q_i prennent leur valeur parmi l'ensemble des n états du modèle $\{s_1, s_2, \ldots, s_n\}$
    - Suite observable correspondant à la séquence d'observations : o_1, o_2, …, o_T, notée O(1:T) où les o_i sont aussi fonctions du temps et se réalisent parmi un ensemble des M symboles observables $\{v_1, v_2, \ldots, v_M\}$
  - a de bonnes propriétés
    - stationnaire

$$\forall t,k:\ P(q_t = s_i / q_{t-1} = s_j) = P(q_{t+k} = s_i / q_{t+k-1} = s_j)$$

    - a une mémoire limitée

$$P(q_t = s_i / q_{t-1} = s_j, q_{t-2} = s_k, \cdots) = P(q_t = s_i / q_{t-1} = s_j)$$

Modèle de Markov Caché

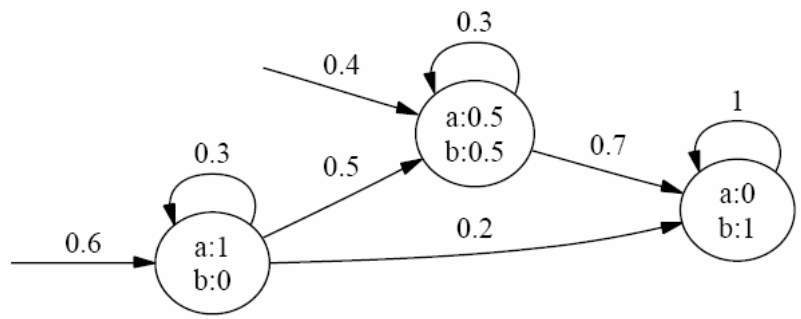
- Définition et exemple

$$\lambda = (\pi, A, B)$$

$$\pi_i = P[q_1 = s_i] \quad 1 \le i \le N$$

$$A = \left[a_{ij} = P(q_t = s_j \,/\, q_{t-1} = s_i) \right] \quad 1 \le i, j \le N$$

$$B = \left[b_j(k) = P(v_k \text{ à } t / q_t = s_j \right] \quad 1 \le j \le N, 1 \le k \le M$$



$$A = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0 & 0.3 & 0.7 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.6 \\ 0.4 \\ 0 \end{pmatrix}$$

Modèle de Markov Caché

- Que peut on faire avec un tel modèle ?
  - Calcul de la vraisemblance d'une observation :
    - $P(O|\lambda)$
      - ➔ calcul nécessaire lors de l'apprentissage
  - Calcul de la séquence d'états ayant généré l'observation présentée :
    - $P(Q/O,\lambda)$
      - ➔ calcul nécessaire pour la reconnaissance
  - Apprentissage :
    - Ajuster les paramètres du HMM $\lambda = (\pi, A, B)$ pour maximiser la vraisemblance de l'ensemble des échantillons
      - ➔ Ajustement nécessaire à l'apprentissage de $P(O|\lambda)$

Problème 1 :
Vraisemblance d'une observation P(O| λ)

- Calcul direct :
  - La probabilité de la suite d'observations O, étant donné le modèle λ, est égale à la somme sur tous les chemins d'états possibles Q des probabilités conjointes de O et de Q

$$P(O \mid \lambda) = \sum_{Q} P(O, Q \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

or, on a les relations :

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$$

$$P(O \mid Q, \lambda) = b_{q_1}(O_1) b_{q_2}(O_2) \dots b_{q_T}(O_T)$$

on déduit donc :

$$P(O \mid \lambda) = \sum_{q_1, q_2, \dots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

Cette formule directe nécessite beaucoup de calcul : $n^T - 1$ additions et $(2^T - 1)n^T$ multiplications. (n^T étant le nombre de chemins possibles de longueur T), soit $2Tn^T$ opérations ($\simeq 10^{72}$ pour n = 5 et T = 100)

Problème 1 :
Simplification par Forward-Backward

- En effet :
  - de nombreuses multiplications sont répétées (portions de sous-séquences communes)
- Idées :
  1. Calculer $P(O|\lambda)$ de manière incrémentale ou inductive
     - c.à.d de manière progressive, permettant à chaque nouvelle étape de ne calculer que ce qui concerne le nouvel état
  2. Calculer $P(O|\lambda)$ de deux manières duales en allant de l'origine et de l'extrémité (pour éviter de se perdre à la recherche de la sortie)
- Calcul inductif vers l'avant :
  - Définissons

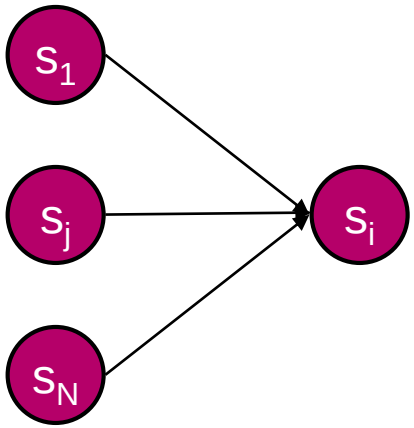
    $\alpha_t(i) = P(o_1,o_2,\ldots,o_t,q_t=s_i|\lambda)$
  - la probabilité de la sous-séquence $o_1,\ldots,o_t$ **et** d'être dans l'état s_i en t

Problème 1 :
Simplification par Forward-Backward

- Calcul en avant : Forward

$$\alpha_1(i) = \pi_i * b_i(o_1)$$

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

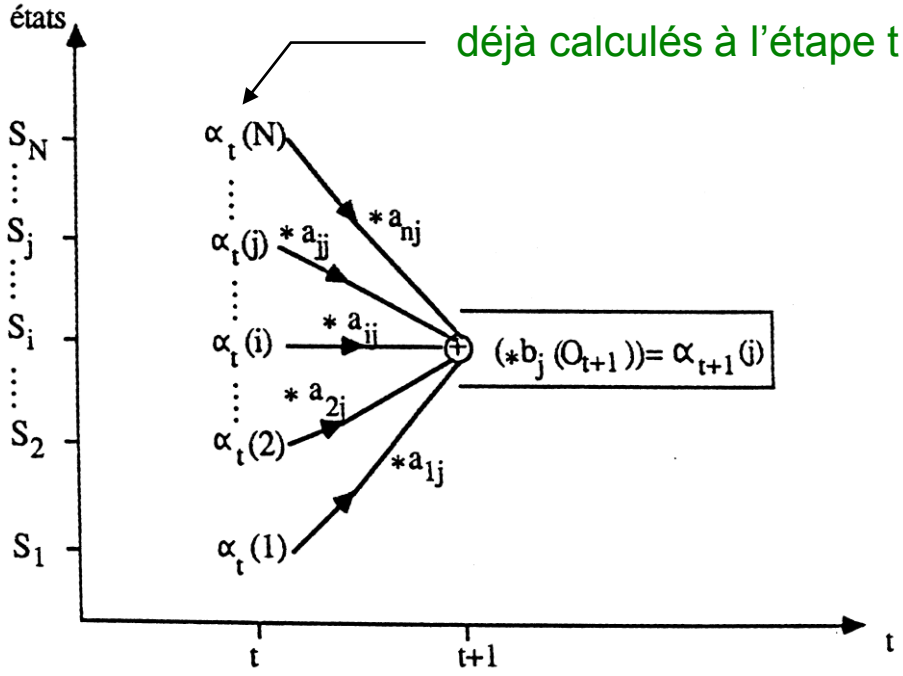


$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i)$$

Forward

Problème 1 :
Simplification par Forward-Backward

- Algorithme en N^2T
  - c'est une accumulation de tout ce qui peut être observé en un état, provenant de tous les chemins précédents (preuve encore de l'optimisation, car évite la répétition du calcul en un état)



Problème 1 :
Simplification par Forward-Backward

- Calcul inductif de l'arrière :
  - Définissons

    $\beta_t(i) = P(o_{t+1},\ldots,o_T,q_t=s_i|\lambda)$

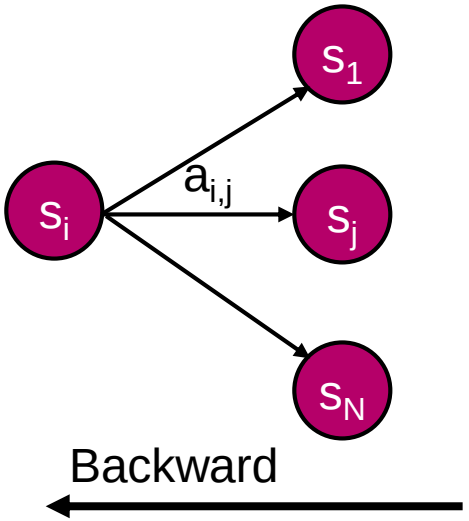
$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i)$$

Initialisation :

$$\beta_T(i) = 1,\ 1 \le i \le N$$

Induction :

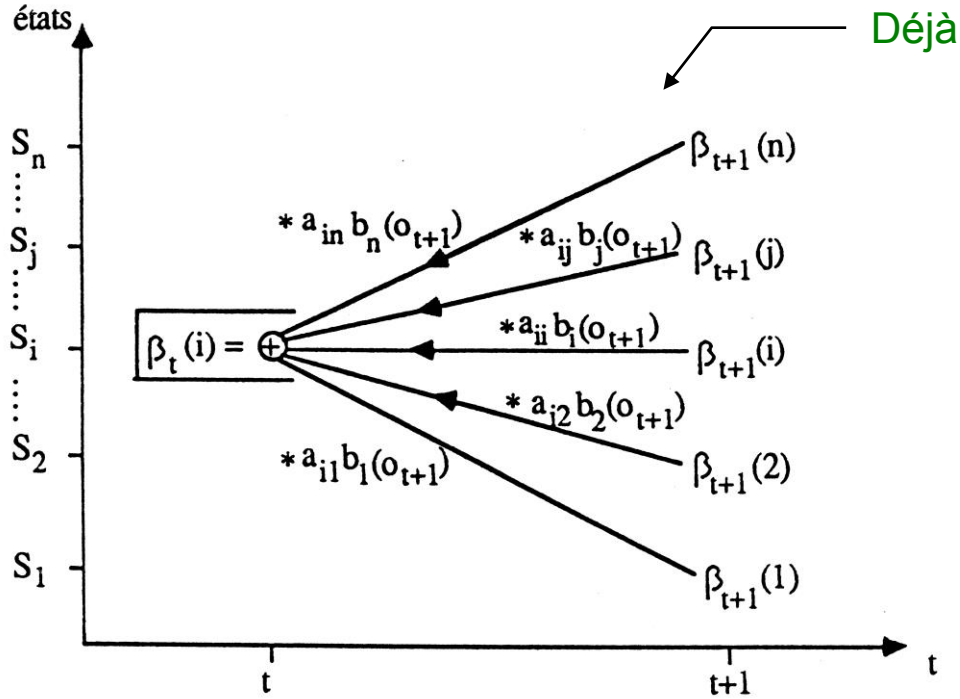
$$\beta_T(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j) \quad T-1 \ge t \ge 1 \quad 1 \le i \le N$$



A chaque étape, $2m^2$ multiplications $\Rightarrow$ complexité en $\Theta(2m^2n)$ en temps, $\Theta(m)$ en espace *(idem pour le forward)*

Problème 1 :
Simplification par Forward-Backward

- le calcul de β synthétise tous les chemins partant de cet état sans le considérer lui même dans le calcul, car il sera calculé dans le α correspondant



Problème 1 :
Simplification par Forward-Backward

- Calcul de la probabilité d'observation
  - Elle est obtenue en prenant les valeurs de α et de β à un instant t quelconque

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$$

  - Cependant, on utilise le plus souvent les valeurs obtenues pour deux cas particuliers (t=0) ou (t=T), ce qui donne :

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \pi_i \beta_0(i)$$

Problème 2
Reconnaissance

- Il s'agit de déterminer la composante cachée du processus, étant donnée la composante observable O et le modèle λ
- La difficulté est la définition du critère d'optimisation
  - Le critère le plus naturel et le plus utilisé est celui de trouver la meilleure suite d'états (chemin) qui maximise :

$$P(q_t = s_i / O, \lambda)$$

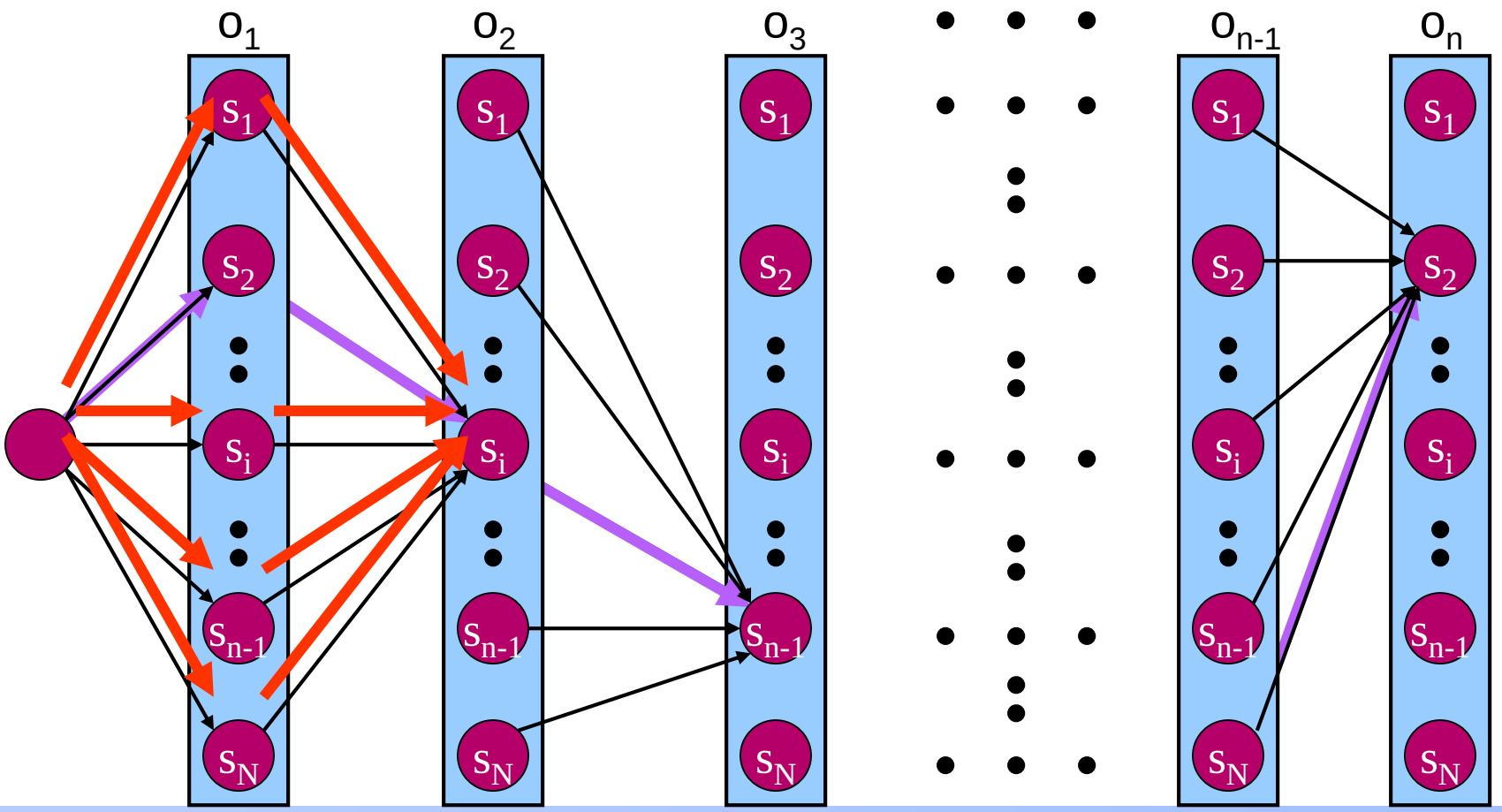
  - La solution pour ce dernier critère est souvent appelée la suite d'états de Viterbi parce qu'elle est trouvée en utilisant l'algorithme de Viterbi [Forney 73] qui est basée sur la technique de programmation dynamique

$$P(Q/O,\lambda) \Rightarrow \text{max}imiser\ P(Q,O/\lambda)$$

Reconnaissance

Algorithme de Viterbi : programmation dynamique

de gauche à droite avec $d(s_{i,t},s_{j,t+1})=a_{i,j}*b_j(o_{t+1})$



Problème 2
Chemin optimal

- Pour trouver le meilleur chemin $Q=(q_1,q_2,\ldots,q_T)$ pour une suite d'observations donnée $O=(O_1,O_2,\ldots,O_T)$, on définit la quantité $\delta_t(i)$ comme la probabilité du meilleur chemin partiel amenant à l'état s_i à l'instant t guidé par les t premières observations

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1 q_2 \ldots q_t = s_i, o_1 o_2 \ldots o_t \mid \lambda)$$

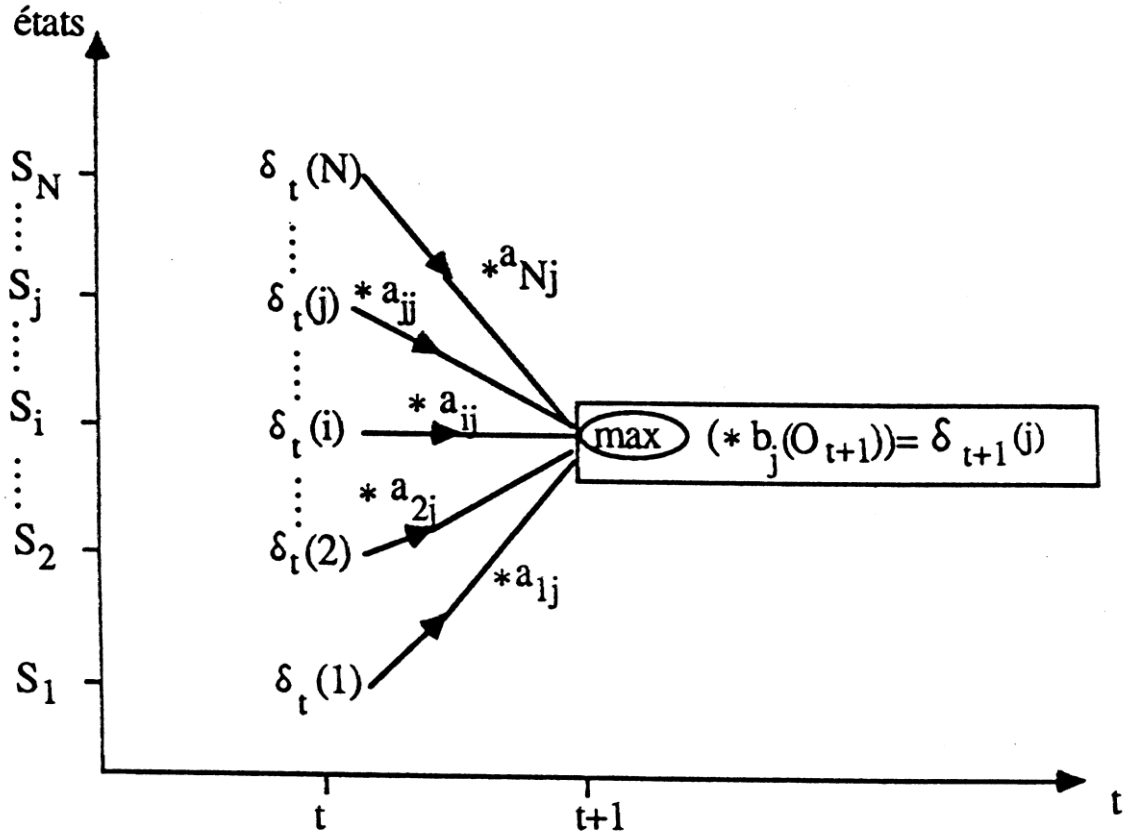
- Par induction, on peut calculer $\delta_{t+1}(i)$ à partir de $\delta_t(i)$

$$\delta_{t+1}(i) = [\max_i \delta_t(i) a_{ij}] . b_j(O(t+1))$$

- et on doit garder trace de la suite d'états qui donne le meilleur chemin amenant à l'état s_i à l'instant t dans un tableau Ψ

Problème 2

Chemin optimal



Problème 2

Chemin optimal

1. Initialisation

$$\delta_1(i) = \Pi_i b_i(O_1) \quad pour \quad 1 \leq i \leq N$$

$$\psi_1(i) = 0$$

2. Induction

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] . b_j(O_t) \quad pour \quad 1 \leq j \leq N$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad pour \quad 1 \leq j \leq N$$

3. Terminaison

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

arg permet de mémoriser l'indice i entre 1 et N, avec lequel on atteint le maximum des quantités $\delta_{t-1}(i)\, a_{ij}$

4. Chemin d'états retenu (backtracking)

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad pour \quad T - 1 \geq t \geq 1$$

Problème 2
Chemin optimal

- Obtention du chemin : liste des états cachés
  - On mémorise en chaque état, l'état précédent qui a permis le calcul du maximum local
  - Ce calcul est répété sur tous les états à l'instant t,
  - Donc, on ne peut déterminer le bon maximum tout de suite
  - Pour cela :
    - On mémorise les maxima à chaque instant t
    - On attend d'avoir fini pour dérouler la liste à l'envers et déterminer enfin, parmi les états des maxima, ceux qui ont conduit au résultat final
  - Ceci peut être considéré comme une généralisation de la programmation dynamique où il s'agit de comparer une liste d'observation à plusieurs (différence avec ce qui a été vu) chemins possibles (voir exemple plus loin)

Problème 3

Apprentissage ou estimation du modèle

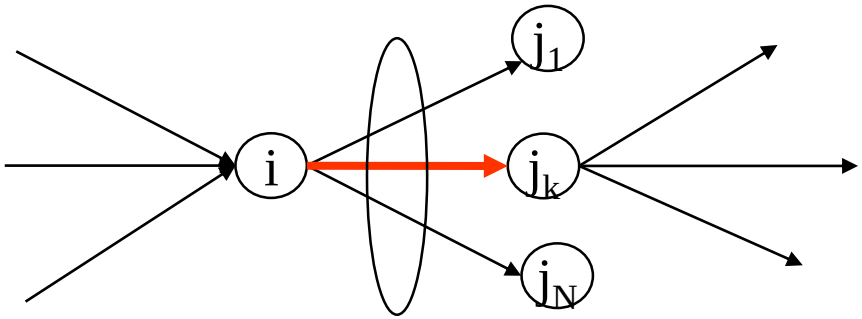
- Principe :
  - Introduire dans le modèle plusieurs échantillons représentatifs de la forme, de manière à disposer de paramètres (π,A,B) maximaux pour la forme
  - Les difficultés de calcul empêchent d'avoir une solution directe d'optimisation globale
  - Les solutions utilisées consistent plutôt à affiner un modèle de base en ré-estimant ses paramètres à chaque itération
    - Il s'agit d'aller progressivement en se laissant guider par les données jusqu'à atteindre un maximum local satisfaisant (point de convergence)

Problème 3

Apprentissage : Ré-estimation des paramètres du modèle

- Principe :
  - Pour chaque nouvelle observation, on calcule son influence sur le modèle (gain qu'elle apporte), précisément sur les états et sur les transitions qu'elle traverse
  - La ré-estimation d'une probabilité qu'elle soit d'observation ou de transition consiste à mesurer son efficacité relativement à toutes les autres possibles et cela sur toute la séquence, et à corriger en conséquence (ré-estimer) cette probabilité

Mesure d'efficacité



Problème 3

Apprentissage : Explications intuitives

- L'efficacité d'une transition se calcule à partir de α, β :
  - Pour sommer les probabilités de tous les chemins passant par cette transition entre t et t+1 :
    - On s'appuie sur α et β :
      - α donne la somme de tous les chemins arrivant sur cette transition et β donne la somme de tous les chemins repartant de cette transition
      - Ceci nous donne l'efficacité entre t et t+1
  - Pour avoir l'efficacité totale de la transition :
    - On effectue cette somme pour tous les t. Ceci se traduit par :

$$\sum_{t=1}^{T-1} \alpha_t(i) \cdot a_{ij} \cdot b_j(t+1) \cdot \beta_{t+1}(j)$$

  - La ré-estimation de la transition consiste à diviser cette efficacité par la somme des efficacités de toutes les transitions sortant de l'état

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

Problème 3

Apprentissage : justifications formelles

- Calcul de l'influence au niveau de la transition :
  - Pour chaque séquence traitée (observée), on définit la probabilité de transition d'un état i à un état j à l'instant t :

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j / O, \lambda)$$

  - On fait passer la séquence d'observation dans la considération du calcul de la proba de transition, car il est tout à fait normal de tenir compte du passé observé. Bayes permet de tenir compte de cette séquence, avec l'hypothèse d'indépendance entre O et λ

$$\xi_t(i,j) = \frac{P(q_t = s_i, q_{t+1} = s_j, O/\lambda)}{P(O/\lambda)}$$

Problème 3

Apprentissage par Baum-Welch (MLE)

- Pour calculer la probabilité de transition de i à j à l'instant t, il faut considérer tous les chemins d'états dans lesquels il y a cette transition i-j à l'instant t
- Grâce à α et β, on peut factoriser tous ces chemins. ξ_t est la probabilité cumulée sur tous les chemins passant de i à j à l'instant t

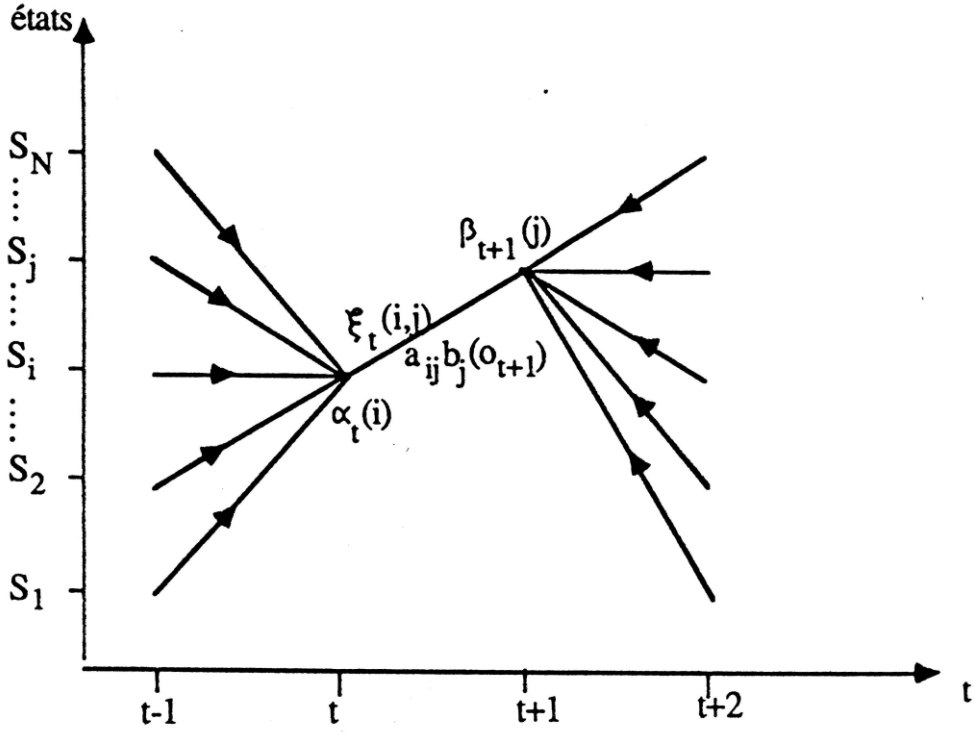
$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O/\lambda)}$$

- α nous dit ce qui peut arriver du début,
- β nous dit tout ce qui peut repartir vers la fin
- et a_{ij} x b_j calcule la proba de transition effective à un instant t

Problème 3

Apprentissage par Baum-Welch (MLE)

- Illustration du calcul de ξ_t : On voit comment α synthétise les chemins arrivant à l'état i et comment β synthétise les chemins partant de l'état j



Problème 3

Calcul de l'influence au niveau de l'état

- De même, on définit la proba d'être dans un état i à un instant t, i.e. on synthétise tous les chemins passant par cet état à l'instant t à l'aide de γ_t :

$$\gamma_t(i) = P(q_t = s_i / O, \lambda)$$

- Comme pour les proba de transitions, ceci est égal à la somme sur tous les chemins passant par cet état à l'instant t, i.e. la somme sur tous les passages de i à tous les états de sortie j

$$\gamma_t(i) = \sum_{j=1}^{N} P(q_t = s_i, q_{t+1} = s_j / O, \lambda) = \alpha_t(i) \cdot \beta_t(i)$$

IL FAUT TOUJOURS SE RAMENER A UN FLUX ENTRANT ET A UN FLUX SORTANT : C'EST CAPITAL POUR COMPRENDRE TOUT LE RESTE

Problème 3

Apprentissage par Baum-Welch (MLE)

- On déduit donc la relation entre $\xi_t(i,j)$ et $\gamma_t(i)$:

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

- Une sommation sur le temps t de γ_t peut être interprétée comme une information sur l'intérêt (le gain) d'être dans l'état s_i ou comme une information sur le nombre de transitions possibles à partir de l'état s_i : γ_t rassemble en l'état tout ce qui passe par cet état, venant des chemins précédant ayant choisi de passer par cet état
- De manière similaire, une sommation sur le temps t de la quantité $\xi_t(i,j)$ peut être interprétée comme une information sur le nombre de transitions possibles de s_i à s_j

Problème 3

Apprentissage par Baum-Welch (MLE)

- Ainsi, on déduit les formules de ré-estimation suivantes :
  - Ré-estimation de la transition : calcul du gain

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{a}_{ij} = \frac{Le\ nombre\ de\ transitions\ possibles\ à\ partir\ de\ l'état\ s_i\ à\ l'état\ s_j}{Le\ nombre\ de\ transitions\ possibles\ à\ partir\ de\ l'état\ s_i}$$

  - Ré-estimation de l'observation : calcul du gain

$$\bar{b}_j(k) = \frac{\sum_{t=1, O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\bar{b}_j(k) = \frac{Le\ nombre\ de\ fois\ possibles\ d'être\ dans\ l'état\ s_j\ en\ observant\ v_k}{Le\ nombre\ de\ fois\ possibles\ d'être\ dans\ l'état\ s_j}$$

Problème 3

Apprentissage par Baum-Welch (MLE)

- Ré-estimation en pratique
  - Tous les calculs précédents ne tiennent compte que d'une seule séquence d'observation
  - Pour ré-estimer le modèle à partir de plusieurs échantillons (tout le corpus), il faut faire le calcul d'efficacité sur l'ensemble des échantillons d'apprentissage. Cela donne la formule suivante :

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, a_{ij}\, b_j(O_{t+1}^k)\, \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(j) + \alpha_{T_k}^k(i)\, a_{iF} \right]}$$

Problème 3
Apprentissage par Baum-Welch (MLE)

- Attention :
  - L'efficacité sur l'ensemble n'est pas égale à la somme des efficacités de chaque séquence. C'est pour cela que l'on fait une somme en haut (proba de transition ou d'observation sur tous les échantillons pour tous les instants t) et une somme en bas qui est égale à la somme de dessus mais généralisée à tous les cas possibles
- De cette manière :
  - On fait l'apprentissage d'un seul coup sur tout le corpus en faisant ces sommations
  - Comme l'algorithme BW ne permet pas d'atteindre l'optimum immédiatement, nous sommes obligés de réitérer l'apprentissage un certain nombre de fois sur tout le corpus

Problème 3

Apprentissage : Théorème de Baum

- Les formules de ré-estimations sont déduites du théorème de Baum, appliqué à la fonction P(O/ λ)
- Le théorème indique que si : $\bar{\lambda}=(\bar{A}, \bar{B}, \bar{\pi})$ est le modèle ré-estimé:
  - Le modèle initial, λ, définit un point critique de la fonction de vraisemblance P(O/ λ). Dans ce cas, nous avons :

$$\bar{\lambda}=\lambda$$

  - Le modèle $\bar{\lambda}$ est plus probable que le modèle λ dans le sens de l'inégalité du théorème :

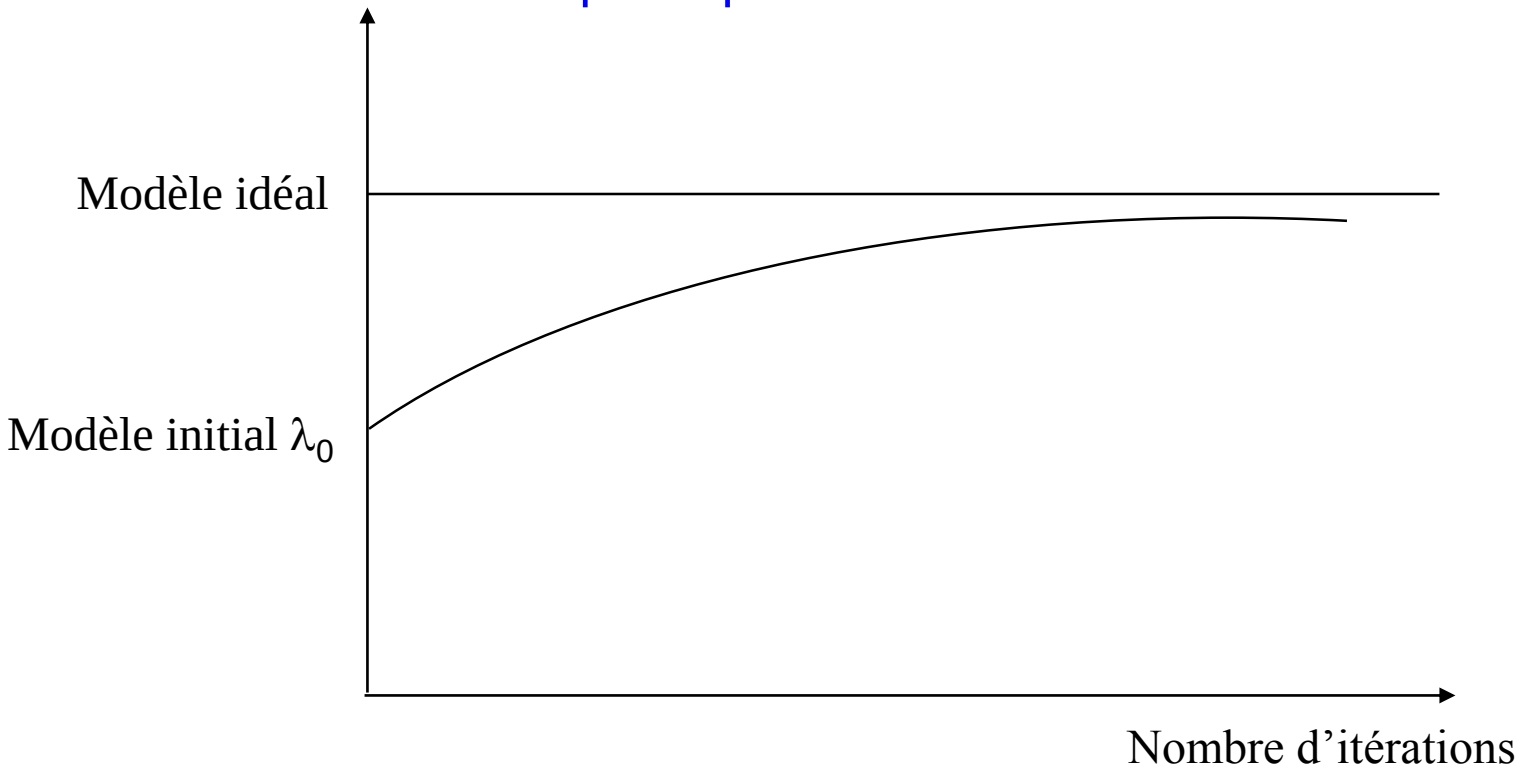
$$P(O/\bar{\lambda})>P(O/\lambda)$$

  - Nous avons donc trouvé un modèle $\bar{\lambda}$ à partir duquel la suite d'observation a plus de chance d'être produite
  - Nous utilisons $\bar{\lambda}$ à la place de λ et nous répétons la ré-estimation jusqu'à un point limite. Le modèle résultant est un HMM obtenu par le Maximum de Vraisemblance : MLE

Problème 3

Apprentissage : Théorème de Baum

- Application en pratique
  - La procédure FB garantit qu'à chaque itération le nouveau modèle est mieux que le précédent



Problème 3

Apprentissage : Théorème de Baum

- Algorithme de Baum Welch
  1. Fixer des valeurs initiales

     a_{ij}^0, $b_j^0(k)$, π_i^0 pour $1 \le i,j \le N$, $1 \le i \le N$, $1 \le k \le M$

  2. Calculer : $\xi_t(i,j)$, $\gamma_t(i)$ pour $1 \le i,j \le N$, $1 \le i \le N$, $1 \le t \le T-1$

     en utilisant les fonctions FB

  3. Nouvelles estimations

     $\bar{a}_{ij}$, $\bar{b}_j(k)$, $\bar{\pi}_i$ pour $1 \le i,j \le N$, $1 \le i \le N$, $1 \le k \le M$ d'où $\bar{\lambda}$

  4. Recommencer en 2 jusqu'à un certain point limite

Problème 3

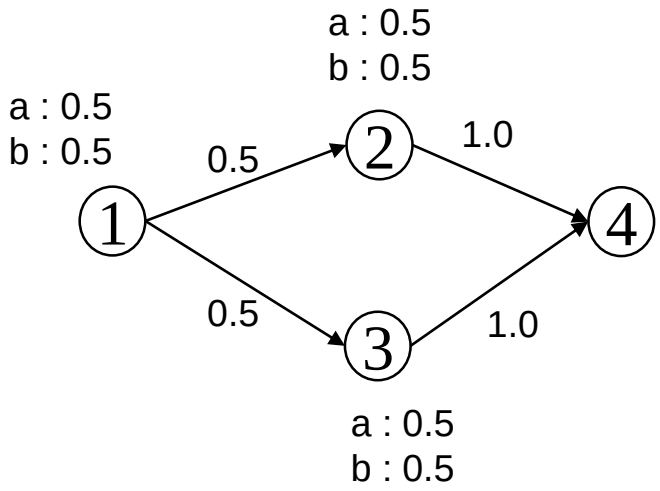
Apprentissage : Explications intuitives

- Effets de la topologie et de l'initialisation du modèle
  1. Un modèle initialisé de manière équiprobable avec une topologie simple (tous les chemins identiques) n'apprendra quasiment rien :
     - Tous les chemins étant identiques, les transitions n'auront pas plus d'importance dans un chemin que dans un autre
     - On tombera sur les mêmes sommes, i.e. toutes les transitions auront la même efficacité pour un état donné
     - Donc, le rapport sera le même et donc la ré-estimation équiprobable (pas de changement par rapport au départ)

Problème 3

Apprentissage : Explications intuitives

- Effets de la topologie et de l'initialisation du modèle
  2. Même une topologie non ergodique peut poser le problème avec une initialisation équiprobable :
     - Exemple 1 : tout est équiprobable, mais départ en l'état 1

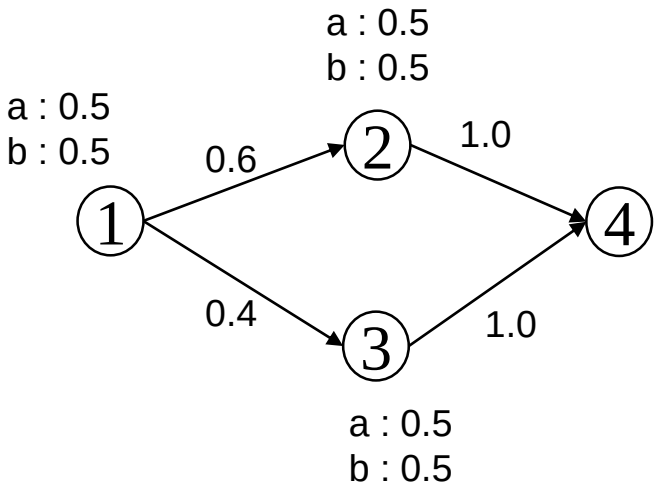


     - Dans ce cas, l'apprentissage ne permettra pas de privilégier un chemin plutôt qu'un autre
     - Les deux chemins garderont la même probabilité de départ

Problème 3

Apprentissage : Explications intuitives

- Effets de la topologie et de l'initialisation du modèle
  - Exemple 2 : les transitions de 1 à 2 et de 1 à 3 ne sont pas équiprobables



  - Dans ce cas, l'apprentissage va favoriser la transition de 1 à 2 et à terme, la transition de 1 à 3 va disparaître
  - Dans ce cas, il aurait été plus intéressant d'avoir un modèle plat dès le début

Problème 3

Apprentissage : Explications intuitives

- Qu'est ce qui fait qu'une transition va se trouver plus efficace qu'une autre ?
  1. Réponse 1 :
     - sa proba est déjà élevée, donc la probabilité d'un chemin passant par cette transition sera renforcée. Comme on s'appuie sur la somme de tous les chemins passant par la transition pour estimer son efficacité, cette somme sera renforcée
  2. Réponse 2 :
     - la transition est souvent empruntée : ex classique : la probabilité de boucler sur un état. Quand on range une donnée longue dans un modèle court, nécessairement, on va beaucoup boucler sur des états, donc ces transitions seront empruntées beaucoup pus que les autres, et donc seront renforcées
  3. Réponse 3 :
     - La transition fait partie de chemins qui sont déjà très probables, dans ce cas elle sera renforcée comme faisant partie de chemins intéressants

Problème 3

Apprentissage : Explications intuitives

- En résumé :
  - Il est intéressant de trouver une topologie adaptée à la structure de donner afin de guider la répartition des observations dans le modèle
  - Avoir des valeurs initiales qui soient représentatives des données de manière à préparer la répartition, ce qui n'est pas simple surtout pour des structures complexes !!!

Problème 3
Apprentissage : par Viterbi

- Cet algorithme permet d'assurer le calcul du maximum de vraisemblance en utilisant uniquement les chemins (maximaux) de Viterbi :
  - C'est une approche certes moins rigoureuse (hypothèse de Viterbi : «tous les autres chemins ont une probabilité nulle ou négligeable»)

$$P(O|\lambda) \approx P(O|\lambda, V)$$

  - …mais c'est un algorithme optimal

Problème 3

Apprentissage : par Viterbi

Rappel : $P(O|\lambda,Q) = \pi_{i(1)} * a_{i(1),i(2)} * b_{i(1)}(o_1) * \ldots * a_{i(n-1),i(n)} * b_{i(n)}(o_n)$

N^i_{jk} : nombre de passages s_j puis en s_k pour la séquence i

M^i_{jl} : nombre d'émissions du symbole o_l par s_j pour la séquence i

$$P(O^i|V^i,\lambda) = \pi_{i(1)} * \prod_{s_j \in S}\left(\prod_{o_l \in \Sigma} b_j(o_l)^{M^i_{jl}} \prod_{s_k \in S} a_{j,k}^{N^i_{jk}}\right)$$

➔ La vraisemblance d'une série d'observation le long du chemin de Viterbi est égale au produit d'autant fois de passages par les transitions du chemin et d'observations aux états du chemin

Problème 3

Apprentissage : par Viterbi

$$P(O|\lambda,V) = \prod_{s_j \in S} \left(\pi_j^{P_j} \prod_{o_l \in \Sigma} b_j(o_l)^{M_{jl}} \prod_{s_k \in S} a_{j,k}^{N_{jk}} \right)$$

N_{jk} : nombre de passages s_j en s_k pour l'ensemble des séquences

M_{jl} : nombre d'émissions du symbole o_l par s_j pour l'ensemble des séquences

P_j : nombre de fois où s_j est premier dans le chemin de Viterbi

Maximiser cette formule ⇔ Maximiser les 3 sous-produits

$$\hat{\pi}_j = \frac{P_j}{T}, \quad \hat{a}_{j,k} = \frac{N_{jk}}{\sum_{i=1}^{m} N_{ji}}, \quad \hat{b}_j(o_l) = \frac{M_{jl}}{\sum_{i=1}^{m} M_{il}}$$

➔ L'apprentissage consiste à estimer ces facteurs en calculant le passage par les états et les transitions des chemins de Viterbi

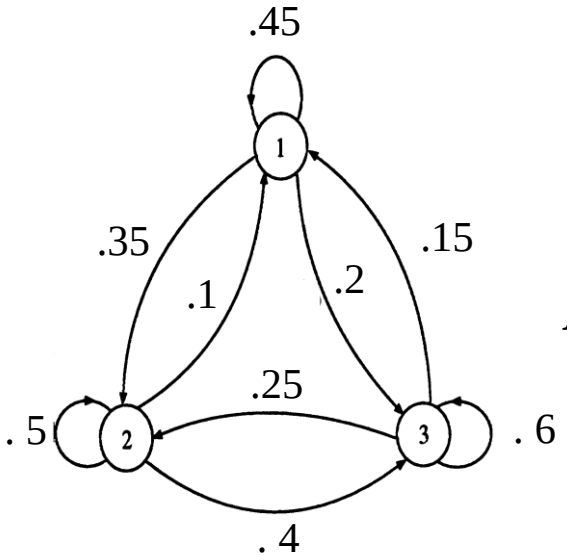
Problème 3

Apprentissage : par Viterbi

- Choix du paramétrage initial du HMM
- Répéter
  - Initialiser les compteurs N_{jk}, M_{jl}, P_j à 0
  - Pour chaque séquence O^i
    - Calculer le chemin de Viterbi pour le HMM courant
    - Maj des compteurs N_{jk}, M_{jl}, P_j
  - Fin pour
  - Ré-estimer les paramètres du HMM avec les formules précédentes
- Jusqu'à stabilité des paramètres

Exemple1

- Soit le modèle : learn_our_method_2.m



$$A=\begin{bmatrix} 0.45 & 0.35 & 0.2 \\ 0.10 & 0.50 & 0.4 \\ 0.15 & 0.25 & 0.6 \end{bmatrix} \quad B=\begin{bmatrix} \text{pile} & \text{face} \\ 1.0 & 0.0 \\ 0.5 & 0.5 \\ 0.0 & 1.0 \end{bmatrix} \quad \pi=\begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix}$$

Exemple1

- Exemple (suite)
  - Considérons la suite d'observations : pffpp
  - Pour calculer P(pffpp/λ) :
    - on n'utilise que la première partie de Baum-Welch (forward)
    - α est déterminé comme suit :

$$\alpha_1(1) = \pi_1 b_1(p) = 0.5 \times 1.0 = 0.5,$$
$$\alpha_1(2) = \pi_2 b_2(p) = 0.3 \times 0.5 = 0.15,$$
$$\alpha_1(3) = \pi_3 b_3(p) = 0.2 \times 0.0 = 0.0,$$

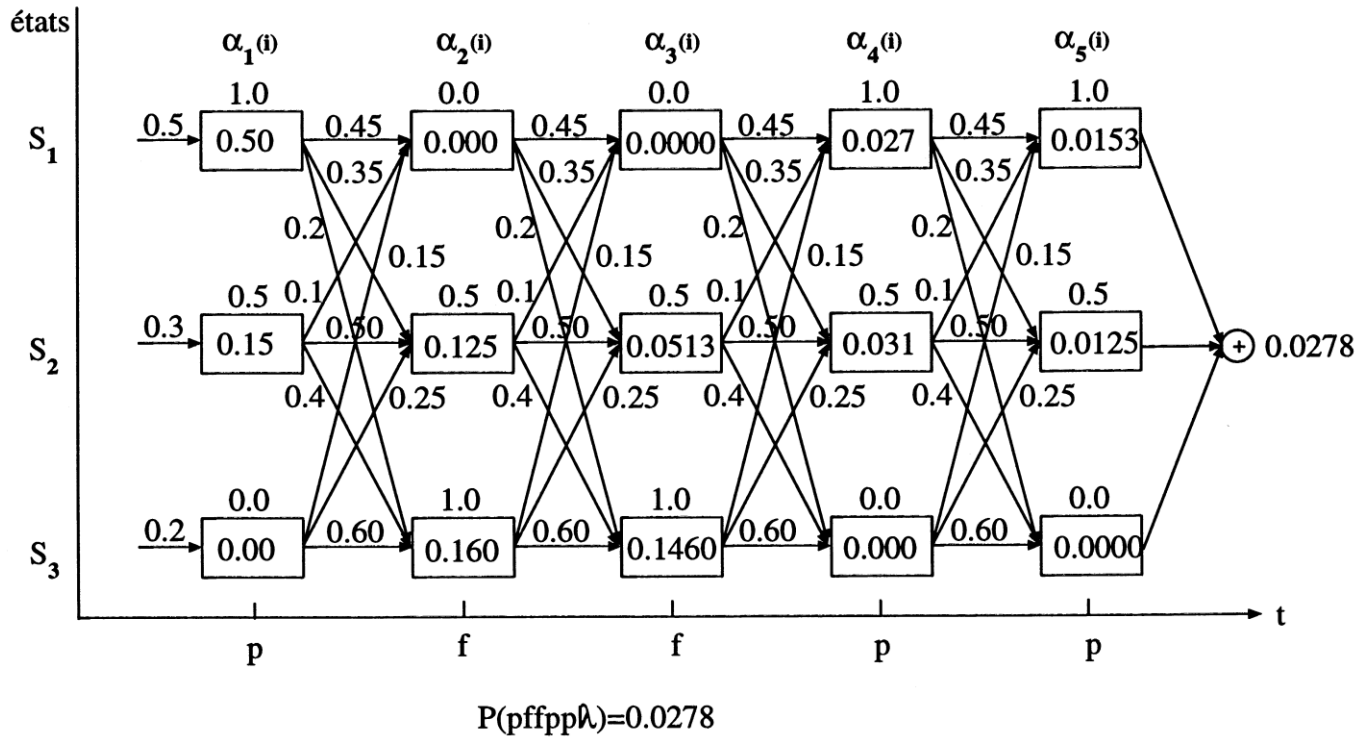
$$\alpha_2(1) = (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_1(f)$$
$$= (0.5 \times 0.45 + 0.15 \times 0.1 + 0.0 \times 0.15) \times 0.0$$
$$= (0.24) \times 0.0 = 0.0$$

$$\alpha_2(2) = (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_2(f)$$
$$= (0.5 \times 0.35 + 0.15 \times 0.5 + 0.0 \times 0.25) \times 0.5$$
$$= (0.25) \times 0.5 = 0.125$$

$$\alpha_2(3) = \ldots$$

Exemple1

- P(pffpp/λ) = 0.0278, comme le montre la figure ci-dessous



Exemple1 : apprentissage

- Pour déterminer le modèle final, c.à.d. calculer les vraies valeurs des paramètres A, B et π, d'après l'échantillon unique, on calcule β :

$$\beta_5(1) = 1.0 \qquad \beta_5(2) = 1.0 \qquad \beta_5(3) = 1.0$$
$$\beta_4(1) = 0.625 \qquad \beta_4(2) = 0.350 \qquad \beta_4(3) = 0.275$$
$$\beta_3(1) = 0.3425 \qquad \beta_3(2) = 0.150 \qquad \beta_3(3) = 0.1375$$
$$\beta_2(1) = 0.0538 \qquad \beta_2(2) = 0.0925 \qquad \beta_2(3) = 0.1013$$
$$\beta_1(1) = 0.0364 \qquad \beta_1(2) = 0.0636 \qquad \beta_1(3) = 0.0723$$

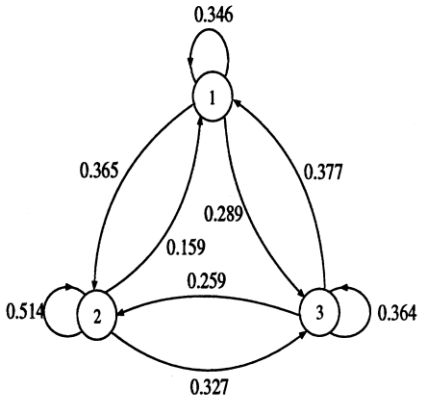
Exemple1 : apprentissage

- Avec ces valeurs de α et β on déduit le modèle ci-dessous. Le nouveau modèle est :

  - Chemins « interdits » (P=0) sont supprimés
  - Paramètres A, B et π ont évolué
  - Pas de départ en 3, contrairement à l'hypothèse de départ (où tout état peut être le départ)
  - Probabilité du modèle très faible

$$A = \begin{pmatrix} 0.346 & 0.365 & 0.289 \\ 0.159 & 0.514 & 0.327 \\ 0.377 & 0.259 & 0.364 \end{pmatrix} \qquad \Pi = \begin{pmatrix} 0.656 \\ 0.344 \\ 0.0 \end{pmatrix}$$

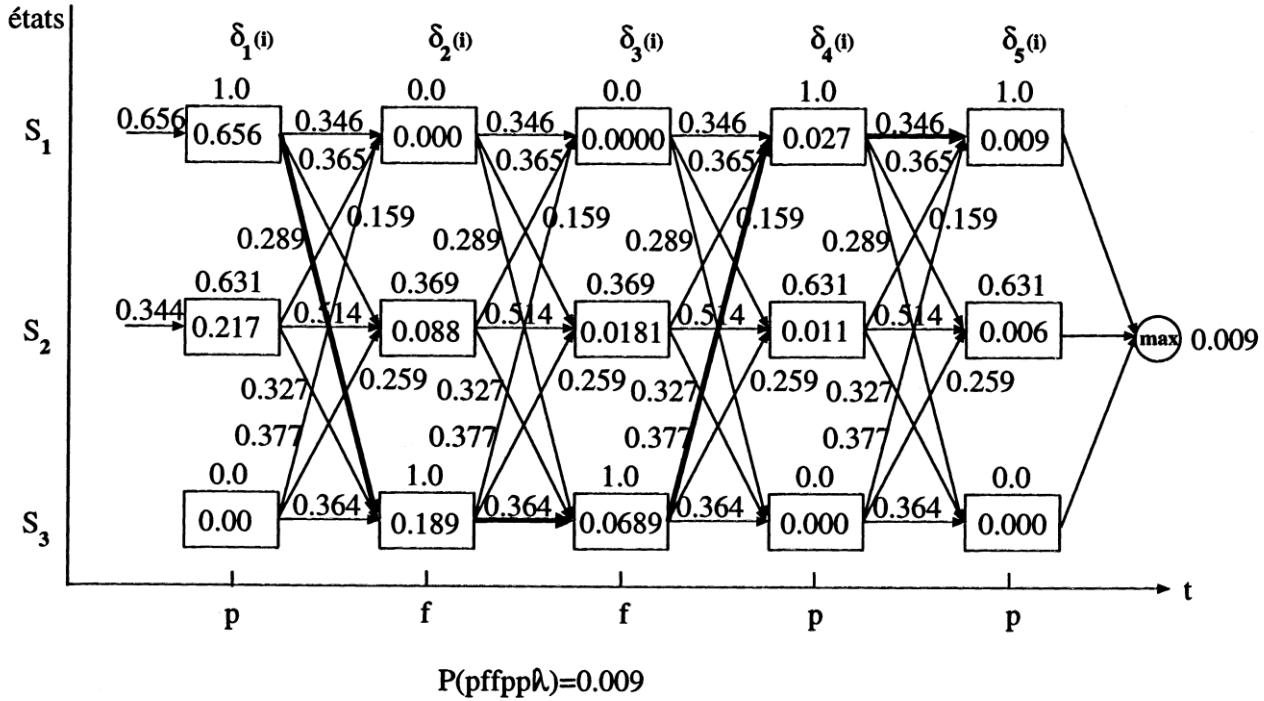
$$B = \begin{pmatrix} 1.0 & 0.0 \\ 0.631 & 0.369 \\ 0.0 & 1.0 \end{pmatrix}$$



Exemple1 : apprentissage

- Calcul de P(pffpp/λ) par l'algorithme de Viterbi
  - Chemin maximal en gras
  - Bien que la chaîne pffpp ait servi à l'apprentissage

  ⇒ P(Viterbi = 0.009)<P(Forward=0.0278)



Exemple1 : reconnaissance

$$\delta_1(1) = \pi_1 b_1(p) = 0.656 \times 1.0 = 0.656,$$

$$\delta_1(2) = \pi_2 b_2(p) = 0.344 \times 0.631 = 0.217,$$

$$\delta_1(3) = \pi_3 b_3(p) = 0.0 \times 0.0 = 0.0,$$

$$\delta_2(1) = \max_{1 \le i \le N} (\delta_1(i) a_{i1}) b_1(f)$$

$$= max \left\{ \begin{array}{l} \delta_1(1) a_{11} \\ \delta_1(2) a_{21} \\ \delta_1(3) a_{31} \end{array} \right\} b_1(f)$$

$$= max \left\{ \begin{array}{l} 0.656 \times 0.346 \\ 0.217 \times 0.159 \\ 0.0 \times 0.377 \end{array} \right\} \times 0.0 = 0.0 \qquad \psi_2(1) = 1$$

$$\delta_2(2) = \max_{1 \le i \le N} (\delta_1(i) a_{i2}) b_2(f)$$

$$= max \left\{ \begin{array}{l} \delta_1(1) a_{12} \\ \delta_1(2) a_{22} \\ \delta_1(3) a_{32} \end{array} \right\} b_2(f)$$

$$= max \left\{ \begin{array}{l} 0.656 \times 0.365 \\ 0.217 \times 0.514 \\ 0.0 \times 0.259 \end{array} \right\} \times 0.369 = 0.088 \qquad \psi_2(2) = 1$$

...

Exemple1 : reconnaissance

$$\underline{\delta_1(1) = 0.656} \quad \delta_1(2) = 0.217 \quad \delta_1(3) = 0.0$$
$$\delta_2(1) = 0.000 \quad \delta_2(2) = 0.088 \quad \underline{\delta_2(3) = 0.189}$$
$$\delta_3(1) = 0.000 \quad \delta_3(2) = 0.018 \quad \underline{\delta_3(3) = 0.069}$$
$$\underline{\delta_4(1) = 0.026} \quad \delta_4(2) = 0.011 \quad \delta_4(3) = 0.0$$
$$\underline{\delta_5(1) = 0.009} \quad \delta_5(2) = 0.006 \quad \delta_5(3) = 0.0$$

$$\psi_2(1) = 1 \quad \psi_2(2) = 1 \quad \underline{\psi_2(3) = 1}$$
$$\psi_3(1) = 3 \quad \psi_3(2) = 3 \quad \underline{\psi_3(3) = 3}$$
$$\underline{\psi_4(1) = 3} \quad \psi_4(2) = 3 \quad \psi_4(3) = 3$$
$$\underline{\psi_5(1) = 1} \quad \psi_5(2) = 1 \quad \psi_5(3) = 1$$

Exemple1 : reconnaissance

$$q_5^* = max \left\{ \begin{array}{l} \delta_5(1) \\ \delta_5(2) \\ \delta_5(3) \end{array} \right\} = 1$$

$$q_4^* = \psi_5(1) = 1$$

$$q_3^* = \psi_4(1) = 3$$

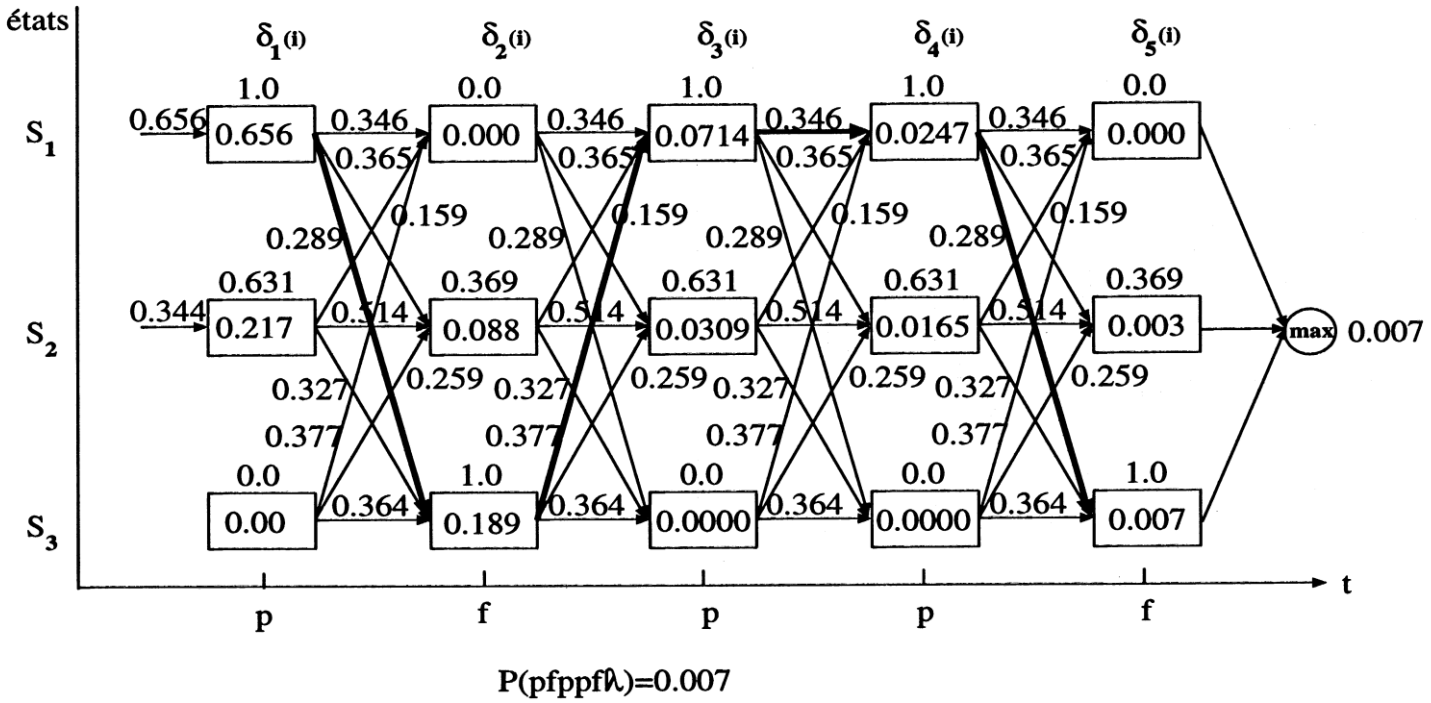
$$q_2^* = \psi_3(3) = 3$$

$$q_1^* = \psi_2(3) = 1$$

On déduit de ce tableau que le chemin conduisant à P^* est : 1 3 3 1 1

Exemple1 : reconnaissance

- Utilisation de Viterbi sur la chaîne pfppf
  - On remarque que la probabilité P(pffpp), tout en étant faible, est plus importante que P(pfppf)



Exemple1

- Remarque
  - Les exemples ont montré :
    - Si la probabilité d'un modèle est faible, son pouvoir de discrimination est également faible
    - Ceci est reflété dans les scores de reconnaissance
    - Donc, pour augmenter la fiabilité de ce modèle :
      - $\Rightarrow$ l'améliorer en itérant avec des valeurs successives de :
      - $\overline{A}, \overline{B}, \overline{\pi}$ tel que :

$$P(O/\overline{\lambda}) > P(O/\lambda)$$

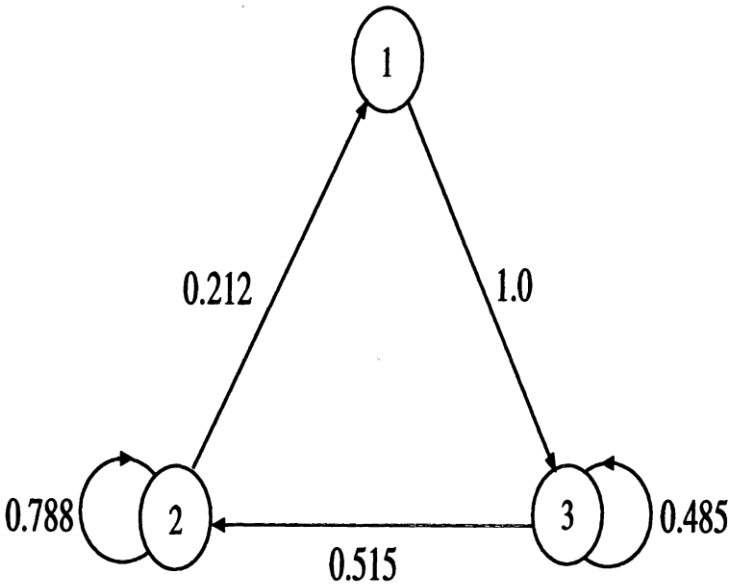
      - Idée : améliorer la probabilité de la séquence à apprendre sachant le modèle
      - Pour ce faire, on ré-estime le modèle afin qu'il soit plus proche des données

Exemple1 : apprentissage

- Résultat
  - Au bout de 15 itérations, on obtient le modèle suivant :

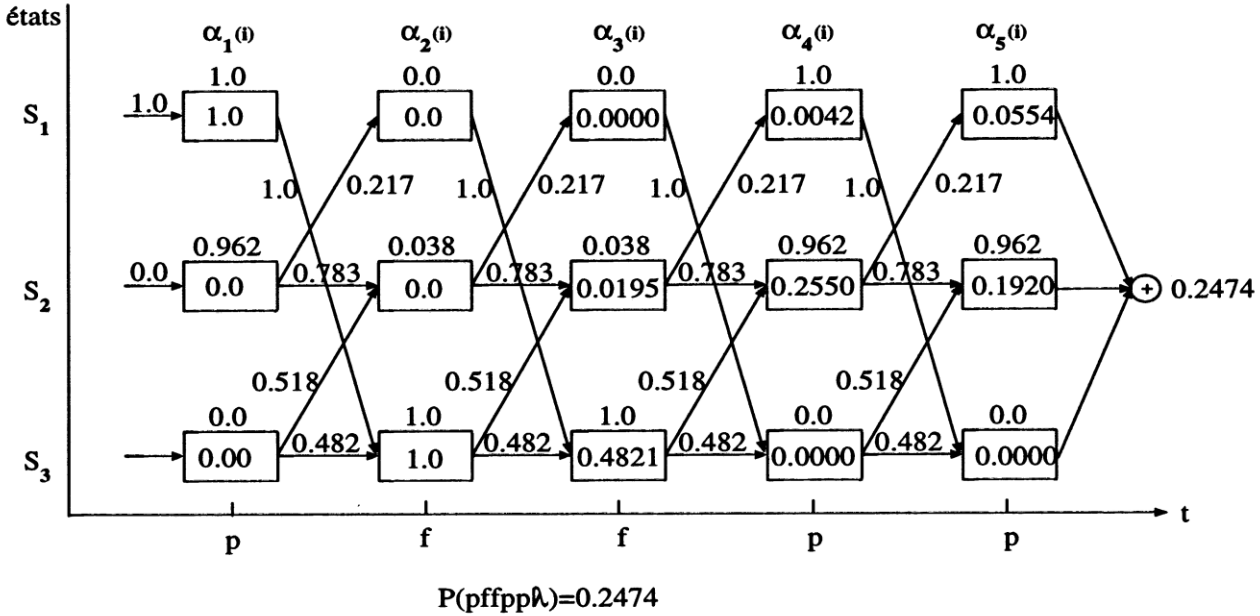
$$A = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.212 & 0.788 & 0.0 \\ 0.0 & 0.515 & 0.485 \end{pmatrix}$$

$$B = \begin{pmatrix} 1.0 & 0.0 \\ 0.969 & 0.031 \\ 0.0 & 1.0 \end{pmatrix} \qquad \Pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$



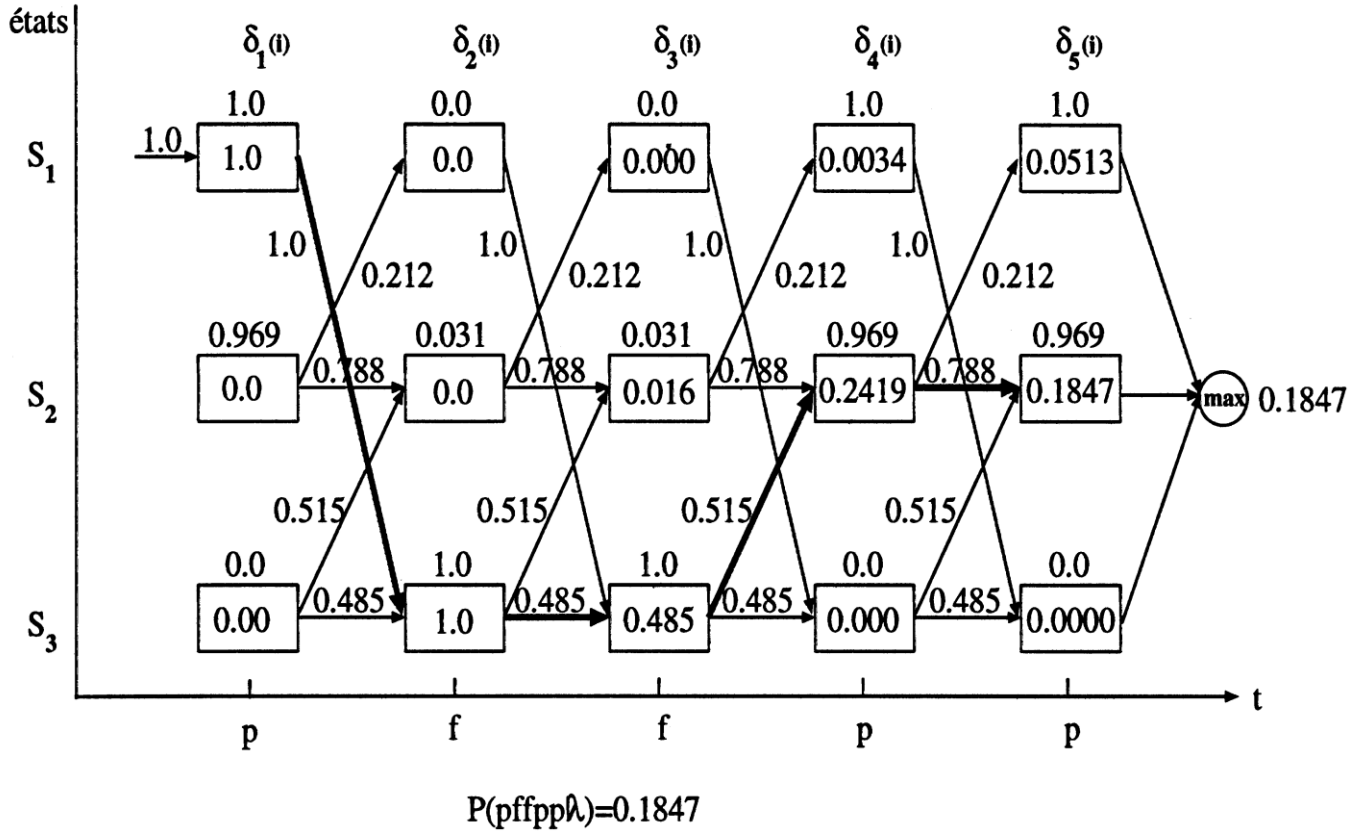
Exemple1 : apprentissage

- Changement de la forme du modèle : un seul candidat pour l'état de départ du modèle, l'état 1
- Modèle plus restrictif avec beaucoup d'interdictions et donc moins de chance de commettre des erreurs



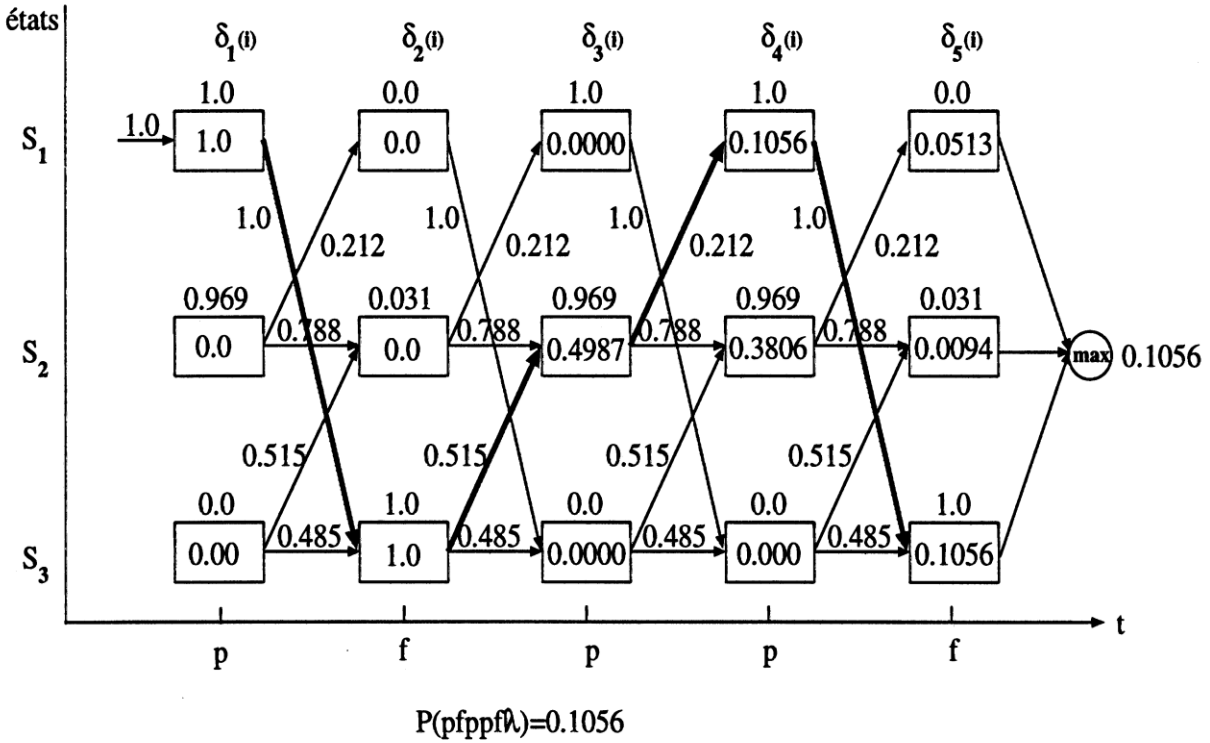
- Nette amélioration de la probabilité : 0.0278 → 0.2474

Exemple1 : apprentissage



Exemple1 : apprentissage

- Remarque :
  - Fort renforcement des scores de Viterbi, avec une nette séparation entre la probabilité de la chaîne la plus proche du modèle et l'autre

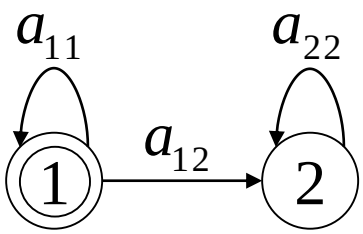


Exemple2

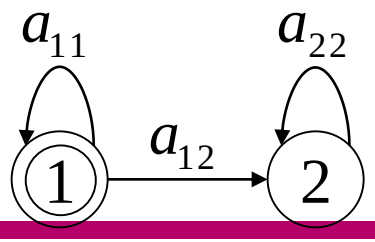
- Modélisation du temps matin/après-midi le week-end
  - On considère 2 observations : soleil et pluie
  - On considère 2 week-ends successifs : 2 séquences
  - Deux jours avec matin et après-midi : séquences de 4 observations
    - MAIS : comme j'ai fait ça dimanche matin, la 2ème séquence ne contient que 3 observations
- Les séquences
  - Premier WE : samedi : soleil, pluie; dimanche : pluie, soleil
  - Second WE : samedi : pluie, pluie; dimanche : soleil et modélisation ☺

Exemple2

- Le modèle
  - HMM à 2 états pour les 2 jours du WE
  - Modèle de type gauche-droite, entrée dans le 1er état
  - Initialisation : on initialise « aléatoirement » les probabilités d'observation
    - P1(soleil)=0.3, P1(pluie)=0.7
    - P2(soleil)=0.8, P2(pluie)=0.2
  - Transitions équiprobables : $a_{11}=a_{12}=0.5$, $a_{22}=1$



Exemple2



- Calcul des $\alpha_t(i)$ et $\beta_t(i)$ pour la première séquence

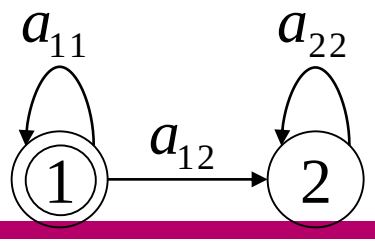
$$\alpha_1(1)=0.3 \quad \alpha_2(1)=0.105 \quad \alpha_3(1)=0.03675 \quad \alpha_4(1)=0.0055125$$
$$\alpha_1(2)=0 \quad \alpha_2(2)=0.03 \quad \alpha_3(2)=0.0165 \quad \alpha_4(2)=0.0279$$

$$\beta_4(1)=1 \quad \beta_3(1)=0.55 \quad \beta_2(1)=0.2725 \quad \beta_1(1)=0.111375$$
$$\beta_4(2)=1 \quad \beta_3(2)=0.8 \quad \beta_2(2)=0.16 \quad \beta_1(2)=0.032$$

Probabilité=0.0334125

$$=\alpha_1(1)*\beta_1(1)+\alpha_1(2)*\beta_1(2)=\alpha_2(1)*\beta_2(1)+\alpha_2(2)*\beta_2(2)$$
$$=\alpha_3(1)*\beta_3(1)+\alpha_3(2)*\beta_3(2)=\alpha_4(1)*\beta_4(1)+\alpha_4(2)*\beta_4(2)$$

Exemple2



- Calcul des $\alpha_t(i)$ et $\beta_t(i)$ pour la deuxième séquence

$$\alpha_1(1)=0.7 \quad \alpha_2(1)=0.245 \quad \alpha_3(1)=0.03675$$
$$\alpha_1(2)=0 \quad \alpha_2(2)=0.07 \quad \alpha_3(2)=0.154$$

$$\beta_3(1)=1 \quad \beta_2(1)=0.55 \quad \beta_1(1)=0.2725$$
$$\beta_3(2)=1 \quad \beta_2(2)=0.8 \quad \beta_1(2)=0.16$$

Probabilité = 0.19075

$$=\alpha_1(1)*\beta_1(1)+\alpha_1(2)*\beta_1(2)=\alpha_2(1)*\beta_2(1)+\alpha_2(2)*\beta_2(2)$$
$$=\alpha_3(1)*\beta_3(1)+\alpha_3(2)*\beta_3(2)$$

Apprentissage Baum-Welch
Exemple

- Ré-estimation de la transition a_{12}
  - Contribution de la première séquence : tous les chemins passant par a_{12}
    - Tous les chemins allant de l'état 1 vers l'état 2
    - $\alpha_t(i)$: tous les chemins arrivant à l'état i à l'instant t
    - $\beta_t(j)$: tous les chemins partant de l'état j à l'instant t
    - tous les chemins passant par a_{12} entre t et t+1 : $\alpha_t(1) * a_{12} * b_{t+1}(2) * \beta_{t+1}(2)$
    - tous les chemins passant par a_{12} : $\sum_{t=1}^{t=3} \alpha_t(1) * a_{12} * b_{t+1}(2) * \beta_{t+1}(3)$ =0.0279
    - On fait pareil pour a_{11}: on obtient 0,0543375
    - On normalise les 2 valeurs en divisant par la proba de la séquence 1 : 0,0334125
      - On obtient respectivement 0,83501684 pour a_{12} et 1,62626263 pour a_{11}
      - La contribution de la première séquence au numérateur pour a_{12} est 0,83501684,
      - et au dénominateur 0,83501684 + 1,62626263 = 2,46127946

Apprentissage Baum-Welch
Exemple

- Ré-estimation de la transition a_{12}
  - Contribution de la deuxième séquence : tous les chemins passant par a_{12}
    - tous les chemins passant par a_{12} entre t et t+1 : $\alpha_t(1) * a_{12} * b_{t+1}(2) * \beta_{t+1}(2)$
    - tous les chemins passant par a_{12} : $\sum_{t=1}^{t=2} \alpha_t(1) * a_{12} * b_{t+1}(2) * \beta_{t+1}(3)$ =0.154
    - On fait pareil pour a_{11}: on obtient 0,1715
    - On normalise les 2 valeurs en divisant par la proba de la séquence 2 : 0,19075
      - On obtient respectivement 0,80733945 pour a_{12} et 0,89908257 pour a_{11}
      - La contribution de la deuxième séquence au numérateur pour a_{12} est 0,80733945,
      - et au dénominateur 0,80733945 + 0,89908257 = 1,70642202

Apprentissage Baum-Welch
Exemple

- Rappel : la ré-estimation, c'est :
  - Somme de toutes les contributions au numérateur pour tous les échantillons
  - Somme de toutes les contributions au dénominateur pour tous les échantillons
  - La nouvelle probabilité est le rapport des deux valeurs
- Dans notre cas
  - Somme numérateur = 0,83501684 + 0,80733945 = 1,6423563
  - Somme dénominateur = 2,46127946 + 1,70642202 = 4,1677015
  - La nouvelle probabilité de a_{12} est donc : 1,6423563 / 4,1677015 = 0,394067641

Apprentissage Baum-Welch
Exemple

La probabilité a baissé, et cela s'explique logiquement par les séquences et les probabilités initiales d'observation :
Première séquence : soleil, pluie, pluie, soleil; 2ème séquence : pluie, pluie, soleil

$$P(soleil)=0.3, P(pluie)=0.7 \qquad P_2(soleil)=0.8, P_2(pluie)=0.2$$

Le premier état favorise la pluie, et le second le soleil. Or les 2 séquences favorisent le premier état puisque la pluie incite à rester dans le premier état. Il devient donc logiquement moins probable de passer dans le second état. Le calcul de la nouvelle probabilité de transition a_{11} montrera que celle-ci, au contraire, augmente ☺

Exemple Matlab

http://www.npt.nuwc.navy.mil/Csf/htmldoc/pdf/node37.html

- http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=1341