Estimation de paramètres pour HMM

Carlos Ramisch

Sources

Ces supports réutilisent du matériel de :

- Diapos d'Alexis Nasr - Statistique Inférentielle 2015-2016
- L. R. Rabiner, 1989, *A Tutorial on HMM and Selected Applications in Speech Recognition*,
- D. Jurafsky and J. H. Martin, 2009, *Speech and Language Processing*, chapters 5 & 6
- M. Collins, *Discriminative Training Methods for HMMs : Theory and Experiments with Perceptron Algorithms*

Programme

1. Estimation par maximum de vraissemblance
2. HMM en pratique : Lissage, logarithmes, $n > 2$, features
3. Apprentissage discriminant : perceptron
4. Algorithme EM - Baum Welch

Rappel : définition d'un HMM

Modèle de Markov à états cachés (HMM)

- N états S_i du modèle
- M symboles observables V_k
- Probabilités de transition $A = \{a_{ij}\} = P(q_{t+1} = S_j | q_t = S_i)$
- Probabilités d'émission $B = \{b_j(k)\} = P(O_t = V_k | q_t = S_j)$
- Probabilités initiales $\pi = \{\pi_i\} = P(q_1 = S_i)$

Un HMM est représenté sous forme compacte :

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

Problème trois - estimation de paramètres

Deux cas de figure :

1. Données complètes :
   - Estimation par maximum de vraissemblance (MLE)
   - Modèle entraîné comme VMM et utilisé comme un HMM
   - Alternative : perceptron - apprentissage discriminant
2. Données incomplètes : estimation avec algorithme EM
   - On initialise le modèle au hasard
   - On utilise ce λ_t pour générer un résultat Q pour O (maximisation)
   - On actualise les poids λ_{t+1} en fonction du nombre espéré de chaque paramètre (expectation)

Maximum de vraissemblance

- Méthode d'estimation qui a pour hypothèse :
  - ***échantillon = population***
- Maximiser la probabilité de l'ensemble d'entraînement
- Principe de base :

$$P(X) = \frac{\text{nombre d'occurrences de X}}{\text{nombre total d'événements}}$$

Données complètes I

états	$Q =$	q_1	q_2	q_3	$\dots$	q_T
observations	$O =$	O_1	O_2	O_3	$\dots$	O_T

On définit la fonction binaire δ :

$$\delta_t(i) = \begin{cases} 1, & \text{si } q_t = S_i \\ 0, & \text{sinon} \end{cases} \qquad \forall 1 \leq t \leq T, 1 \leq i \leq N$$

$$\delta_t(k) = \begin{cases} 1, & \text{si } O_t = V_k \\ 0, & \text{sinon} \end{cases} \qquad \forall 1 \leq t \leq T, 1 \leq k \leq M$$

Cette fonction équivaut à une boucle qui parcourt l'ensemble d'entraînement et renvoie 1 quand elle trouve la valeur S_i ou V_k

Données complètes II

On définit les variables suivantes (nombres d'occurrences) :

- $\mathcal{C}_q(i) = \sum_{t=1}^{T} \delta_t(i)$
- $\mathcal{C}_{O,q}(k,i) = \sum_{t=1}^{T} \delta_t(k) \times \delta_t(i)$
- $\mathcal{C}_{q,q}(i,j) = \sum_{t=2}^{T} \delta_{t-1} i \times \delta_t(j)$

Une façon naturelle d'estimer les paramètres du HMM :

$$\hat{b_i}(k) = \frac{\mathcal{C}_{O,q}(k,i)}{\mathcal{C}_q(i)} \qquad \hat{a_{ij}} = \frac{\mathcal{C}_{q,q}(i,j)}{\mathcal{C}_q(i)} \qquad \hat{\pi}_i = \frac{\delta_1(i)}{\sum_j \delta_1(j)}$$

Limites de l'estimation par maximum de vraisemblance

- Tout événement absent du corpus d'apprentissage a une probabilité nulle
- Exemples :
  - mot inconnu lors de l'étiquetage
  - combinaison (mot, catégorie)
  - *n*-gramme non observé en reconnaissance de la parole
  - structure syntaxique absente de la banque d'arbres
  - ...

Comment résoudre le problème ?

Lissage (smoothing)

- prélever une masse de probabilité des événements observés
- la redistribuer aux événements non observés
- combien prélever ?
- comment le redistribuer ?

Add-one smoothing

- On considère que les événements non observés ont été observés une fois.
- Les événements observés n fois l'ont été $n + 1$ fois.

Application aux unigrammes

- Tous les mots du vocabulaire n'ont pas été vus dans le corpus
- $P_{\text{MLE}}(O_t = V_k) = \frac{C(V_k)}{T}$, où T est la taille du corpus.
- $P_{\text{add-1}}(O_t = V_k) = \frac{C(V_k)+1}{T+M}$, où M est la taille du vocabulaire.

Si $\forall V_k$ on rajoute 1 au nombre d'occurrences, le nombre total d'événements devient $T + \sum_{k=1}^{M} 1 = T + M$

Limites du add-one smoothing

- Les bigrammes non observés se voient attribuer une probabilité très faible
- Mais ils sont très nombreux et absorbent une grande partie de la masse des probabilités

Lissage add-1 : alternatives

- ***Add-α smoothing***

$$P_{\text{add-}\alpha}(O_t = V_k) = \frac{C(V_k) + \alpha}{T + M \times \alpha} \qquad \alpha < 1$$

- ***Interpolation***

$$P_{interp}(q_{t-1} = S_i, q_t = S_j) = d \times \frac{C_{q,q}(i,j)}{C_q(i)} + (1-d) \times \frac{C_q(j)}{T}$$

- ***Back-off***
  - Considerer modèles d'ordre inférieur quand $\frac{C_{q,q}(i,j)}{C_q(i)} = 0$
- Chapitres 4.5 à 4.7 de *Speech and Language Processing*

HMM en pratique

- Produit de probabilités : underflow
  - Logarithme - transformation monotone (proba. $\rightarrow$ score)
- $\log(a \times b) = \log a + \log b$
- multiplication plus “chère” que somme

HMM pour l'étiquetage morphosyntaxique

- Adaptation aux 3-grammes :

$$P(Q) = \prod_{t=1}^{T} P(q_t | q_{t-2}, q_{t-1})$$

- Début de phrase : ajout de $n-1$ éléments bidons <s>
- Ajout d'autres features :
  - Majuscules/minuscules
  - Suffixes/préfixes (n-grammes de caractères)
  - Présence de nombres/caractères spéciaux

Exercice : modèle avec lissage

- Corpus annoté :
  - je/CL porte/V
  - je/CL la/P fais/V
  - la/D porte/N
- Phrase à étiqueter :
  - $O =$ je mange
- $V = \{$je, porte, fais, la, <ukn>$\}$

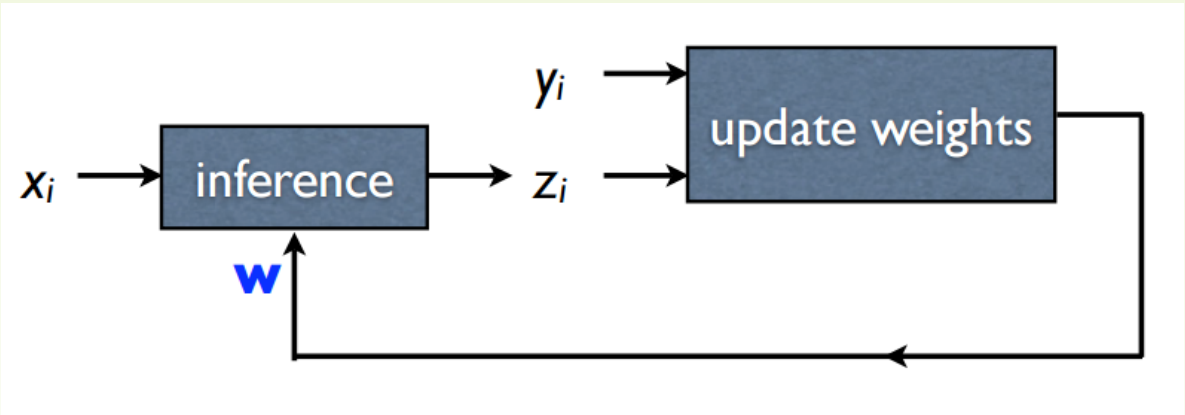
1. Calculer $P_{MLE}(O|\lambda)$ sans lissage
2. Estimer $b_j(k)$, a_{ij} et π_i avec lissage add-1
3. Calculer $P_{\text{add-1}}(O|\lambda)$ avec l'algorithme forward

Génératif - discriminant

- ***Modèle génératif*** : distribution jointe $P(Y, X)$
  - "Histoire générative" du processus $X \rightarrow Y$
  - Hypothèses fortes sur le modèle du processus
  - Requiert bonne estimation des probabilités
  - Bon pouvoir de généralisation
- ***Modèle discriminant*** : distribution conditionnelle $P(Y|X)$
  - Scores fondés sur les probabilités - pas de contrainte $P(\Omega) = 1$
  - Apprend à résoudre la tâche - minimisation de l'erreur
  - Modèle adaptable : il est faciles d'ajouter des features
  - Attention au sur-entraînement (overfit)
  - MaxEnt, perceptron, etc.

Perceptron

- Idée : on regarde 1 exemple à la fois
- Apprentissage incrémental : essai-erreur
  1. Trouver une solution étant donné les paramètres actuels
  2. Mettre les paramètres à jour en fonction des erreurs



Perceptron pour étiquetage morphosyntaxique

- Modèle HMM : scores de transition et d'émission
- Trouver une solution : algorithme de Viterbi
- Mise à jour des scores lorsque le modèle prédit une séquence d'étiquettes erronées :
  - `- -` paramètres qui mènent à l'erreur
  - `++` paramètres qui mènent à la bonne réponse

Perceptron : exemple

Phrase d'entraînement

la/DET porte/N est/V rouge/ADJ

Sortie du modèle λ_i

la/DET ***porte/V*** est/V rouge/ADJ

- Décrémenter les scores :
  - $a_{\texttt{DET},\texttt{V}}$
  - $a_{\texttt{V},\texttt{V}}$
  - $b_{\texttt{V}}(porte)$
- Incrémenter les scores :
  - $a_{\texttt{DET},\texttt{N}}$
  - $a_{\texttt{N},\texttt{V}}$
  - $b_{\texttt{N}}(porte)$

Algorithme du perceptron structuré I

1. Choisir $I \rightarrow$ nombre d'itérations de l'algorithme
2. Initialiser tous les paramètres (a_{ij},$b_i(k)$ et π_i) à zéro
3. Pour chaque itération et pour chaque phrase $O = O_1 \dots O_T$ du corpus d'apprentissage ayant pour séquence d'étiquettes $Q = q_1 \dots q_T$, trouver la meilleure séquence de catégories $\hat{Q} = \hat{q}_1 \dots \hat{q}_n$ à l'aide de l'algorithme de Viterbi
   Si $Q \neq \hat{Q}$, mettre les paramètres à jour :
   - $a_{ij} = a_{ij} + \sum_{t=1}^{T-1} \phi_{a_{ij}}(t, Q) - \sum_{t=1}^{T-1} \phi_{a_{ij}}(t, \hat{Q})$
   - $b_j(k) = b_j(k) + \sum_{t=1}^{T-1} \phi_{b_j(k)}(t, Q) - \sum_{t=1}^{T-1} \phi_{b_j(i)}(t, \hat{Q})$
   - $\pi_i = \pi_i + \phi_{\pi_i}(Q) - \phi_{\pi_i}(\hat{Q})$

Algorithme du perceptron structuré II

Les fonctions caractéristiques $\phi_{a_{ij}}(t, Q)$, $\phi_{b_j(k)}(t, Q)$ et $\phi_{\pi_i}(Q)$ sont définies de la façon suivante :

- $\phi_{a_{ij}}(t, Q) = \begin{cases} 1 & si\ q_t = S_i\ et\ q_{t+1} = S_j \\ 0 & sinon \end{cases}$
- $\phi_{b_j(k)}(t, Q) = \begin{cases} 1 & si\ q_t = S_i\ et\ O_t = V_k \\ 0 & sinon \end{cases}$
- $\phi_{\pi_i}(Q) = \begin{cases} 1 & si\ q_1 = S_i \\ 0 & sinon \end{cases}$

Perceptron moyenné

- Mise à jour “soft” des poids
- Convergence plus rapide
- Cf. article Collins

Exercice : perceptron

Corpus annoté :

- je/CL porte/V
- je/CL la/P fais/V
- la/D porte/N

1. Effectuez deux itération de l'algorithme perceptron pour entraîner un étiqueteur morphosyntaxique ce corpus

Données incomplètes

états	$x =$	?	?	?	...	?
observations	$o =$	o_1	o_2	o_2	...	o_T

- On ne dispose que des données d'apprentissage o et de la structure du HMM $\hat{\lambda}$.
- On ne connaît pas de méthode permettant de calculer directement $\hat{\lambda}$.
- Il existe une procédure, appelée algorithme de Baum-Welch ou algorithme *forward-backward* qui permet de s'en approcher.
- Procédure itérative : on calcule une suite de HMM $\lambda_0, \lambda_1, \ldots, \lambda_I$ où λ_{i+1} est construit à partir de λ_i et tel que :

$$P(O|\lambda_{i+1}) \geq P_(O|\lambda_i)$$

Baum-Welch : estimation de a I

- On veut estimer $\hat{a_{ij}}$ en termes du nombre moyen de transitions de S_i à S_j dans O :

$$\hat{a_{ij}} = \frac{\text{nombre moyen de transitions de } S_i \text{ à } S_j}{\text{nombre moyen de transitions depuis } S_i}$$

Baum-Welch : estimation de a II

- Le ***dénominateur*** peut être défini comme la somme, pour tous les instants t, de la probabilité d'être dans l'état i.
- Nous devons définir la probabilité d'être dans un état S_i au temps t sachant l'observation O et le modèle λ_0, notée $\gamma_t(i)$:

$$\begin{aligned}\gamma_t(i) &= P(q_t = S_i | O, \lambda_0) \\ &= \frac{P(q_t = S_i, O, \lambda_0)}{p(O, \lambda_0)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}\end{aligned}$$

- Plusieurs formulations équivalentes pour $P(O|\lambda_0)$ en fonction de $\alpha_t(i)$ et $\beta_t(i)$.

Baum-Welch : estimation de a III

- Le ***numérateur*** peut être défini comme la somme, pour tous les instants t, de la probabilité de traverser l'arc $S_i \rightarrow S_j$.
- Nous devons définir la probabilité d'être dans un état S_i au temps t et aller vers l'état S_j au temps $t+1$, sachant l'observation O et le modèle λ_0, notée $\xi_t(i,j)$:

$$\begin{aligned}
\xi_t(i,j) &= P(q_t = S_i, q_{t+1} = S_j | O, \lambda_0) \\
&= \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda_0)}{P(O|\lambda_0)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)}
\end{aligned}$$

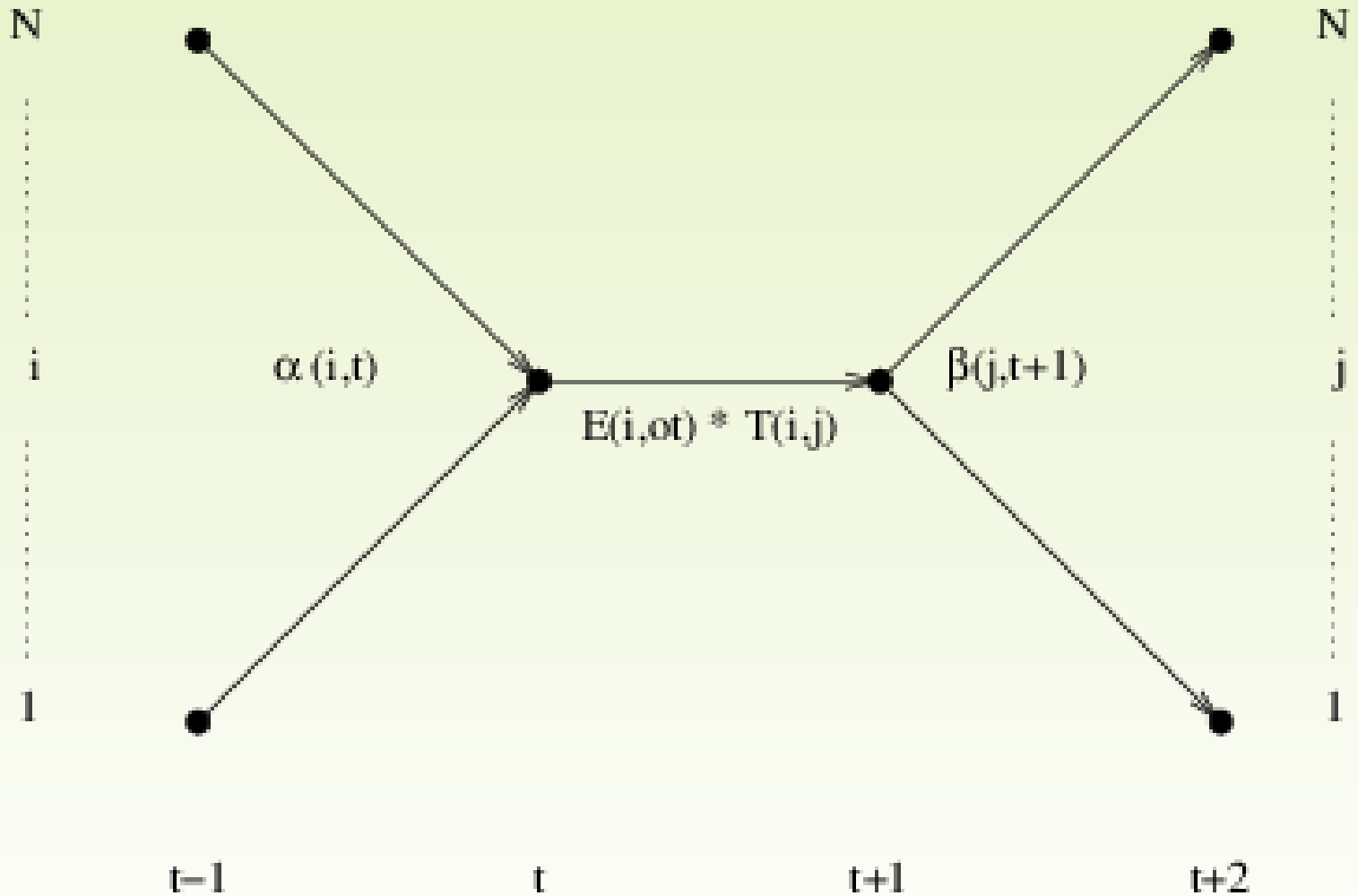
- Notez que $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$

Baum-Welch : estimation de a IV

- Maintenant, nous sommes prêts pour estimer $\hat{a_{ij}}$
- Nombre moyen de transitions $S_i \rightarrow S_j = \sum_{t=1}^{T-1} \xi_t(i,j)$
- Nombre moyen de transitions depuis $S_i = \sum_{t=1}^{T-1} \gamma_t(i)$

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

Baum-Welch : estimation de *a* V



Baum-Welch : estimation de b I

- On veut estimer $b_j(\hat{k})$ en termes du nombre moyen de d'émissions de V_k à partir de S_j dans O :

$$b_j(\hat{k}) = \frac{\text{nombre moyen d'émissions de } V_k \text{ depuis } S_j}{\text{nombre moyen de transitions depuis } S_j}$$

- Le ***numérateur*** peut être défini en fonction de $\gamma_t(j)$:

$$\sum_{t=1,\ \text{où}\ O_t=V_k}^{T} \gamma_t(j)$$

- Ainsi :

$$b_j(\hat{k}) \quad = \quad \frac{\sum_{t=1,\ \text{où}\ O_t=V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Baum-Welch : estimation de π

Les probabilités intiales peuvent être réestimées de la façon suivante :

$$\begin{aligned} \hat{\pi}_i &= \text{probabilité d'être en } i \text{ à l'instant } t = 1 \\ &= \gamma_1(i) \end{aligned}$$

Algorithme Baum-Welch I

1. ***Entrée*** : suite d'observations $O = O_1 \dots O_T$, ensemble d'états $S_1 \dots S_N$
2. Initialiser $\hat{\lambda} = (A_0, B_0, \pi_0)$
3. Itérer les étapes ci-dessous jusqu'à convergence :
   1. E-step (espérance) - utiliser $\hat{\lambda}$ pour calculer :

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \qquad \forall t \text{ et } \forall j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \qquad \forall t, \forall ti \text{ et } \forall j$$

Algorithme Baum-Welch II

2. M-step (maximisation) - reestimer les paramètres du modèle

$$\hat{\pi}_i = \gamma_1(i)$$

$$\hat{b_j}(k) = \frac{\sum_{t=1,\text{ où } O_t=V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

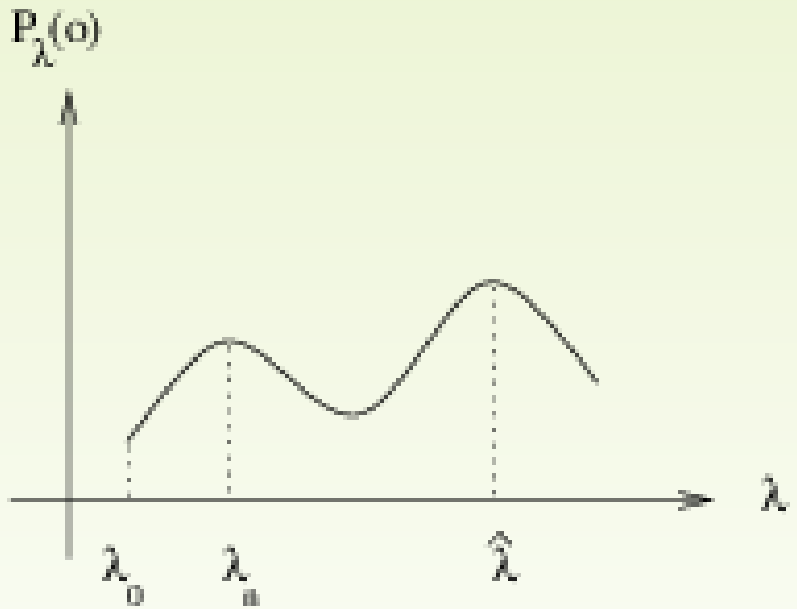
$$\hat{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{\lambda} = (\{\hat{\pi}_i\}, \{\hat{b_j}(k)\}, \{\hat{a_{ij}}\})$$

3. ***Sortie*** : les paramètres $\hat{\lambda}$ du modèle HMM

Baum-Welch - initialisation

λ_n n'est pas le meilleur possible, il peut s'agir d'un maximum local, qui dépend de λ_0 :



Souvent, des connaissances ou des données complètes sont utilisées pour initialiser λ_0