COURS DE DEUXIÈME ANNÉE
INGÉNIEUR ISIMA

Machine-Learning
Chaînes de Markov cachées

Vincent Barra

ISIMA

Table des matières

Les modèles de Markov cachés (Hidden Markov Models ou HMMs) ont été introduits par Baum *et al.* à la fin des années 60 [1]. Ce modèle est fortement apparenté aux automates probabilistes, définis par une structure composée d'états et de transitions, et par un ensemble de distributions de probabilité sur les transitions. À chaque transition est associée un symbole d'un alphabet fini. Ce symbole est générée à chaque fois que la transition est empruntée. Un HMM se définit également par une structure composée d'états et de transitions et par un ensemble de distributions de probabilité sur les transitions. La différence essentielle avec les automates probabilistes est que la génération de symboles s'effectue sur les états, et non sur les transitions. De plus, on associe à chaque état non pas un symbole, mais une distribution de probabilité sur les symboles de l'alphabet.
Les modèles de Markov cachés sont utilisés pour modéliser des séquences d'observations. Ces observations peuvent être de nature discrète (par exemples les caractères d'un alphabet fini) ou continue (fréquence d'un signal, température). Sans prétendre à une présentation exhaustive des modèles de Markov cachés (Rabiner [2] fournit par ailleurs une introduction très complète du sujet), l'objectif de ce chapitre est de dresser un portrait général de cet outil et de son utilisation en apprentissage.

1 CHAÎNES DE MARKOV CACHÉES

Nous nous limitons ici à la modélisation de séquences de symboles appartenant à un alphabet fini. Une courte introduction au cas continu est présentée en fin de cette section.

Définition 1
Un HMM est défini par un quintuplet (S, Σ, T, G, π)*, où :*

- S *est un ensemble de N états,*
- Σ *est un alphabet de M symboles,*
- $T = S \times S \to [0,1]$ *est la matrice de transition, indiquant les probabilités de transition d'un état à l'autre ; on note* $P(s \to s')$ *la probabilité de transition de l'état s vers l'état* s'*,*
- $G = S \times \Sigma \to [0,1]$ *est la matrice de génération, indiquant les probabilités de génération associées aux états ; on note* $P(o \mid s)$ *la probabilité de générer le symbole* $o \in \Sigma$ *à partir de l'état* $s \in S$*.*
- $\pi : S \to [0,1]$ *est un vecteur de probabilités initiales de visite.*

La procédure de génération d'une séquence $o_1 \cdots o_T$ de symboles à l'aide d'un HMM consiste à partir d'un état s en suivant la distribution π, de se déplacer d'état en état suivant les probabilités de transition, et générer un symbole sur chaque état rencontré en utilisant la distribution de probabilité de génération associée à l'état. Lorsqu'un symbole a été généré, on choisit une transition sortante suivant la distribution de probabilité de transition associée à l'état courant, et la procédure est réitérée jusqu'à la $T^{\text{ème}}$ génération de symbole. Les HMMs définissent donc un processus stochastique non déterministe, ce qui explique le nom donné à ce modèle.
On peut classer les principales applications des HMMs en deux catégories. La première traite des problèmes de reconnaissance ou de classification, la seconde a trait aux problèmes de segmentation de séquences, c'est-à-dire au découpage d'une séquence en sous-séquences de différents types.

Trois problèmes de bases sont soulevés par l'utilisation des HMM, qui vont être abordés dans les parties suivantes :

1. étant donnés une séquence d'observations $O = o_1 \cdots o_T$ et un HMM H, comment évaluer efficacement $P(O|H)$, la probabilité d'observation de O étant donné H ?
2. étant donnés une séquence d'observation $O = o_1 \cdots o_T$ et un HMM H, comment choisir une séquence d'états $s_1 \cdots s_T$ optimale dans un certain sens (*i.e.* expliquant au mieux O) ?
3. comment ajuster les paramètres d'un HMM H pour maximiser $P(O|H)$?

Le premier problème est un problème d'évaluation, qui peut également être vu comme un problème d'estimation de la capacité d'un modèle donné à reconnaître une séquence d'observations donnée. Le second problème se ramène à l'idée de dévoiler les états cachés S, sans y avoir accès directement. Dans la plupart des cas, le critère d'optimalité retenu influencera la séquence d'états calculée. Enfin, le troisième problème se ramène à l'entraînement d'un HMM par des séquences d'observations, en vue d'en optimiser les paramètres pour un problème spécifique donné.

1.1 Algorithme forward-backward

1.1.1 Position du problème

Si l'on veut calculer la probabilité de générer la séquence de symboles $O = o_1 \cdots o_T$ à l'aide du HMM $H = (S, \Sigma, T, G, \pi)$, l'approche directe consiste à calculer la probabilité de génération pour chaque chemin possible et faire la somme de ces probabilités. La probabilité P_s de générer O suivant le chemin $S = s_1 \cdots s_T$ est :

$$P_s = \pi(s_1)P(o_1|s_1)\cdots P(s_{T-1} \to s_T)P(o_T|s_T) \tag{1}$$

La probabilité de générer O avec H obtenue en sommant sur l'ensemble des séquences d'états possibles est donc

$$P(O \mid H) = \sum_{\{s_1 \cdots s_T\}} P_s \tag{2}$$

Si N est le nombre d'états du HMM, alors le nombre de chemins possibles pour générer une séquence de symboles de longueur T est de l'ordre de N^T. Comme, pour chaque chemin, le calcul de (1) demande de l'ordre de T opérations, le calcul de la probabilité de génération de O par H suivant (2) est en $O(TN^T)$. Ainsi, même pour des valeurs de T et N peu élevées, l'approche directe n'est clairement pas acceptable : avec $N = 5$ et $T = 100$ le calcul de $P(O \mid H)$ demande approximativement 10^{72} opérations. C'est pourquoi des alternatives doivent être mises en place.

1.1.2 Algorithme forward-backward

Soit $\alpha_t(s) = P(o_1 \cdots o_t, s_t = s \mid H)$ la probabilité d'avoir généré la séquence $O = o_1 \cdots o_t$ et d'être arrivé sur l'état s à l'instant t. Cette variable peut être calculée de manière inductive :

1. initialisation : $\alpha_1(s) = \pi(s).P(o_1 \mid s)$
2. induction : $\alpha_t(s) = \left(\sum_{s' \in S} \alpha_{t-1}(s').P(s' \to s) \right) P(o_t \mid s)$

Connaissant $\alpha_T(s)$ la probabilité d'avoir généré la séquence O et d'être arrivé sur s pour tout $s \in S$, le calcul de $P(O \mid H)$ est immédiat :

$$P(O \mid H) = \sum_{s \in S} \alpha_T(s) \tag{3}$$

La phase d'initialisation requiert une opération pour chaque état du HMM, donc au total $O(N)$ opérations. Pour la phase d'induction, pour chaque instant et chaque état, on réalise $O(N)$ opérations. Sommé sur l'ensemble des états et la totalité des instants, la phase d'induction requiert donc $O(N^2T)$ opérations. Une fois les $\alpha_T(s)$ calculés, le calcul de $P(O \mid H)$ suivant (2) demande une opération pour chaque état, donc au total $O(N)$ opérations. Le calcul de $P(O \mid H)$ à l'aide de cet algorithme, dit forward, ne requiert donc au total que $O(N^2T)$ opérations. En reprenant la même application numérique que précédemment (N = 5 et T = 100), le calcul nécessite approximativement 3000 opérations.
Cet algorithme est appelé forward car l'induction est réalisée en avant : on calcule tout

d'abord la probabilité de générer le premier symbole de la séquence, puis à chaque étape de l'induction on rajoute un symbole et on réitère la procédure jusqu'à ce que l'on ait calculé la probabilité de génération de la séquence entière.
Un algorithme similaire, l'algorithme backward, peut être utilisé pour réaliser ce calcul à l'envers. On utilise alors la variable backward $\beta_t(s) = P(o_{t+1} \cdots o_T \mid s_t = s, H)$ qui exprime la probabilité de générer la séquence $O = o_{t+1} \cdots o_T$ en partant de l'état s. L'induction suit alors le schéma :

1. initialisation : $\beta_T(s) = 1$
2. induction : $\beta_t(s) = \left(\sum_{s' \in S} \beta_{t+1}(s') P(s \to s') P(o_{t+1} \mid s') \right)$

Connaissant la probabilité de générer la séquence O en partant de l'état s, le calcul de $P(O \mid H)$ peut alors être réalisé suivant la formule

$$P(O \mid H) = \sum_{s \in S} \pi(s) \beta_1(s) \tag{4}$$

avec la même complexité.

1.2 Algorithme de Viterbi

1.2.1 Position du problème

Le problème posé ici est de trouver, étant donnés une séquence de symboles $O = o_1 \cdots o_T$ et un HMM $H = (S, \Sigma, T, G, \pi)$ la séquence d'états du HMM qui a la probabilité maximale de générer O. Ce qui nous préoccupe n'est pas la valeur de la probabilité maximale mais le chemin, appelé **chemin de Viterbi**, qui permet de générer la séquence O avec cette probabilité. De manière similaire à l'approche utilisée pour le calcul de $P(O \mid H)$, l'approche directe pour résoudre ce problème consiste à calculer la probabilité de génération suivant tous les chemins possibles et de choisir parmi ces chemins celui qui a la probabilité la plus élevée. Cette approche a également une complexité en $O(TN^T)$ et est donc également inapplicable. L'algorithme de Viterbi est un algorithme de programmation dynamique très similaire à l'algorithme forward et qui permet de résoudre efficacement ce problème.

1.2.2 Algorithme de Viterbi

Soit $\delta_t(s) = \max_{s_1 \cdots s_{t-1}} P(s_1 \cdots s_t = s, o_1 \cdots o_t \mid H)$ la probabilité maximale de générer la séquence $O = o_1 \cdots o_t$ suivant un unique chemin arrivant sur l'état s à l'instant t. De la même manière que pour $\alpha_t(s)$, cette variable peut être calculée de manière inductive :

1. initialisation : $\delta_1(s) = \pi(s) P(o_1 \mid s)$
2. induction : $\delta_t(s) = \max_{s' \in S} \left(\delta_{t-1}(s') P(s' \to s) \right) P(o_t \mid s)$

Connaissant $\delta_T(s)$ pour tous les états s, on peut calculer la probabilité maximale $P(O \mid H, V)$ de générer O avec H suivant un simple chemin V par :

$$P(O \mid H, V) = \max_{s \in S} \left(\delta_T(s) \right) \tag{5}$$

En fait, ce n'est pas la valeur de cette probabilité qui nous intéresse mais réellement le chemin qui permet de générer O avec cette probabilité. On doit donc, à chaque étape t de l'induction et pour chaque état s, mémoriser l'état $s' = \psi_t(s)$ qui maximise l'équation d'induction. Alors :

1. initialisation : $\psi_1(s) = 0$
2. induction : $\psi_t(s) = arg\max_{s' \in S} \left(\delta_{t-1}(s')P(s' \to s)\right)$

Une fois les variables $\delta_t(s)$ et $\psi_t(s)$ calculées pour chaque étape de l'induction et pour chaque état, il ne reste plus qu'à lancer une procédure inductive de rétro-propagation pour "dérouler" le chemin de Viterbi $s_1^* \cdots s_T^*$:

1. initialisation : $s_T^* = arg\max_{s \in S}(\delta_T(s))$
2. induction : $s_t^* = \psi_{t+1}(s_{t+1}^*), \quad t \in \{T-1 \cdots 1\}$

Mise à part la phase de rétro-propagation, l'algorithme de Viterbi est très similaire à l'algorithme forward. La principale différence résulte de la maximisation des probabilités attachées aux états précédents au lieu du calcul de la somme de ces probabilités.

1.3 Apprentissage d'un HMM

Les deux algorithmes précédent supposent disposer d'un HMM construit et paramétré de manière à modéliser de façon satisfaisante les séquences à traiter. La question au centre de ce paragraphe est celle de la construction d'un tel HMM. Dans le cas le plus favorable, le HMM recherché peut être construit directement à partir des connaissances *a priori* dont on dispose sur le domaine. Dans la plupart des applications, le HMM doit être construit à l'aide d'un algorithme d'apprentissage.
Ces algorithmes sont appliqués sur un ensemble de séquences représentatives des séquences que l'on souhaite modéliser et appelées **séquences d'apprentissage**. On peut distinguer dans le problème de l'apprentissage d'un HMM deux cas de figure distincts, suivant que la structure (nombre d'états du HMM et transitions autorisées) est connue ou ne l'est pas. Lorsque la structure est connue, le problème se réduit à un problème d'entraînement consistant à estimer les paramètres numériques (distributions de probabilité de première visite, de transition et de génération) de manière à expliquer au mieux les séquences d'apprentissage. Pour certaines applications, on ne dispose pas de connaissances suffisantes pour inférer naturellement la structure du HMM. L'apprentissage devient alors encore plus difficile. Il ne suffit plus de paramétrer une structure mais il faut également déduire cette structure des exemples fournis.

1.3.1 Apprentissage d'un HMM à structure connue

On souhaite ici apprendre, ou entraîner, un HMM à partir d'une structure connue. On dispose pour cela d'un ensemble d'apprentissage composé de séquences supposées représentatives des séquences que l'on souhaite modéliser. Une approche possible est, suivant le principe du maximum de vraisemblance, de chercher les paramètres $\lambda =< T, G, \pi >$ du HMM qui maximisent la probabilité de génération des séquences d'apprentissage.

Soit $O = \{O^1 \cdots O^K\}$ l'ensemble des séquences d'apprentissage. On suppose que ces séquences sont indépendantes et donc que la probabilité de générer l'ensemble d'apprentissage est simplement le produit des probabilités de génération de chacune des séquences. L'objectif est alors alors de trouver les paramètres $\lambda =< T, G, \pi >$ maximisant :

$$P(O \mid H) = \prod_{k=1}^{K} P(O^k \mid H) \tag{6}$$

Une approche alternative au principe du maximum de vraisemblance est de chercher à maximiser la probabilité de génération des séquences d'apprentissage suivant leur chemin de Viterbi. Le but est alors de trouver les paramètres $\lambda =< T, G, \pi >$ maximisant

$$P(O \mid H, \mathcal{V}) = \prod_{k=1}^{K} P(O^k \mid H, V^k) \tag{7}$$

où V^k est le chemin de Viterbi de la séquence O^k dans H , et $\mathcal{V} = \{V^1 \cdots V^K\}$. Les paramètres ne sont donc pas estimés en maximisant la vraie probabilité de génération des séquences d'apprentissage mais la probabilité de génération suivant les chemins les plus probables. Cette approche peut sembler moins rigoureuse que l'approche basée sur le principe du maximum de vraisemblance, mais elle est très utilisée en pratique et possède quelques arguments en sa faveur, en particulier celui selon lequel la probabilité de génération d'une séquence de symboles suivant son chemin de Viterbi est en général beaucoup plus élevée que suivant n'importe quel autre chemin. Cette observation a conduit à l'hypothèse, appelée hypothèse de Viterbi, que tous les chemins, excepté le chemin de Viterbi, ont une probabilité nulle ou négligeable d'engendrer la séquence.
Sous cette hypothèse, la probabilité de génération d'une séquence O^k par un HMM H peut être approximée par la probabilité de génération de O^k suivant son chemin de Viterbi dans H . On a alors :

$$P(O \mid H) \approx P(O \mid H, \mathcal{V})$$

Suivant la méthode d'estimation des paramètres choisie, il existe deux heuristiques permettant l'entraînement des HMMs. Ces deux algorithmes, très similaires, sont tous deux issus d'une méthode générale servant à l'estimation des paramètres d'une grande famille de modèles probabilistes et appelée **algorithme d'Expectation-Maximization (EM)**.

Entraînement de Viterbi Il existe un cas où il est possible de paramétrer la structure du HMM de manière à maximiser l'équation (7) de manière optimale. Ce cas est celui rencontré lorsque l'on connait les chemins de Viterbi des séquences d'apprentissage dans la structure. En effet, on peut alors associer à chaque état, chaque transition et chaque symbole attaché aux états du HMM le nombre de fois où ils sont utilisés pour générer l'ensemble d'apprentissage.
Soit $n_s, n_{s \to s'}$et n_s^o respectivement le nombre de fois où l'état s est utilisé, le nombre de fois où la transition $s \to s'$ est utilisée et le nombre de fois où le symbole o est généré par l'état s dans les chemins de Viterbi. Alors (7) peut être réécrite :

$$P(O \mid H, \mathcal{V}) = \prod_{s \in S} \left(\prod_{o \in \Sigma} P(o \mid s)^{n_s^o} \prod_{s' \in S} P(s \to s')^{n_{s \to s'}} \right) \tag{8}$$

Maximiser cette formule revient à maximiser indépendamment chacun de ses sous-produits. On peut donc estimer les paramètres T en maximisant l'expression $\prod_{o\in\Sigma} P(o \mid s)^{n_s^o}$ et les estimateurs de T et G recherchés se calculent donc par :

$$\hat{P}(s \to s') = \frac{n_{s\to s'}}{n_s} \tag{9}$$

et

$$\hat{P}(o \mid s) = \frac{n_s^o}{n_s} \tag{10}$$

Cette méthode d'estimation n'est possible que dans le cas favorable où les chemins de Viterbi sont connus. Lorsque ce n'est pas le cas, le problème est évidemment plus difficile et l'algorithme d'entraînement de Viterbi peut alors être une solution : c'est une méthode de réestimation itérative qui consiste, à partir d'un paramétrage initial du HMM, à calculer les chemins de Viterbi des séquences d'apprentissage à l'aide de l'algorithme de Viterbi. Les chemins de Viterbi sont utilisés pour calculer le nombre de fois où chaque transition, chaque état et chaque symbole attaché aux états est utilisé. On réestime alors à l'aide des formules (9) et (10) les paramètres du HMM, on reparamètre la structure à l'aide de ces estimations et on réitère la procédure jusqu'à stabilité. On peut montrer que $P(O \mid H, \mathcal{V})$ augmente à chaque itération et que l'algorithme converge vers un optimum local. Malheureusement, il existe généralement un grand nombre d'optima locaux et le paramétrage obtenu par ce processus dépend fortement du paramétrage initial choisi.

Entraînement de Baum-Welch L'algorithme d'entraînement de Baum-Welch est un algorithme qui cherche à estimer les paramètres $\lambda =< T, G, \pi >$ du HMM en maximisant (6). C'est un algorithme de réestimation itératif qui fonctionne sur le même principe que l'algorithme d'entraînement de Viterbi. Dans ce dernier, on compte le nombre de fois où chaque état, chaque transition et chaque symbole attaché aux états est utilisé dans les chemins de Viterbi. Ici on veut maximiser les probabilités de génération réelles, et non celles des chemins les plus probables. On associe aux états, aux transitions et aux symboles le nombre de fois où ils sont utilisés pour toutes les séquences et tous les chemins susceptibles de générer les séquences, pondéré par la probabilité du chemin. Ces comptes pondérés sont alors utilisés pour réestimer les paramètres du modèle de la même manière que pour l'algorithme d'entraînement de Viterbi.
De manière plus formelle, considérons pour commencer une unique séquence d'observations $O = o_1 \cdots o_T$. Il s'agit alors de trouver les paramètres $\lambda =< T, G, \pi >$ maximisant la probabilité $P(O \mid H)$ de générer O avec H . Nous verrons ensuite comment étendre l'algorithme à un ensemble de séquences d'observations. Soit $\xi_t(s,s') = P(s_t = s, s_{t+1} = s' \mid O, H)$ la probabilité qu'en générant O avec H on passe par l'état s à l'instant t et par l'état s' à l'instant $t+1$. On a alors :

$$\xi_t(s,s') = \frac{P(s_t = s, s_{t+1} = s', O \mid H)}{P(O \mid H)}$$

et en utilisant les variables forward et backward :

$$\begin{aligned}\xi_t(s,s') &= \frac{\alpha_t(s)P(s \to s')P(o_{t+1}|s')\beta_{t+1}(s')}{P(O \mid H)} \\ &= \frac{\alpha_t(s)P(s \to s')P(o_{t+1}|s')\beta_{t+1}(s')}{\sum_{q\in S}\sum_{r\in S}\alpha_t(q)P(q \to r)P(o_{t+1}|r)\beta_{t+1}(r)}\end{aligned}$$

Si $\gamma_t(s) = P(s_t = s \mid O,H)$ est la probabilité qu'en générant O avec H on se trouve sur l'état s à l'instant t, on a :

$$\gamma_t(s) = \sum_{s'\in S}\xi_t(s,s')$$

Si l'on somme $\gamma_t(s)$ sur l'ensemble des instants t, on obtient une quantité que l'on peut interpréter comme l'espérance du nombre de fois où l'état s est utilisé pour générer la séquence O. De même, si on somme $\xi_t(s,s')$ sur l'ensemble des instants t, on obtient une quantité que l'on peut interpréter comme l'espérance du nombre de fois où la transition $s \to s'$ est utilisée pour générer la séquence O. On a donc un estimateur $\hat{H}$ du HMM défini par les expressions suivantes :

$$\begin{aligned}\hat{\pi}(s) &= \gamma_1(s) \\ \hat{P}(s \to s') &= \frac{\sum_{t=1}^{T-1}\xi_t(s,s')}{\sum_{t=1}^{T-1}\gamma_t(s')} \\ \hat{P}(o \mid s) &= \frac{\sum_{t=1,o_t=o}^{T}\gamma_t(s)}{\sum_{t=1}^{T}\gamma_t(s)}\end{aligned}$$

En utilisant ces estimateurs, Baum a montré que

1. soit le modèle initial H définit un point critique de la fonction de vraisemblance, auquel cas $\hat{H} = H$
2. soit $\hat{H}$ est tel que $P(O|\hat{H}) > P(O|H)$, c'est à dire que $\hat{H}$ est plus à même de produire O que H.

Dans le cas où l'on dispose d'un ensemble de séquences d'observations $O = \{O^1 \cdots O^K\}$, ces estimateurs se généralisent simplement en sommant numérateur et dénominateur sur l'ensemble des séquences d'apprentissage. On peut alors proposer une version complète de l'algorithme de Baum-Welch, en appliquant itérativement cette procédure d'estimation des paramètres de H sur O. Comme pour l'algorithme d'entraînement de Viterbi, on peut montrer que la probabilité de génération des séquences $P(O \mid H)$ augmente à chaque itération et que l'algorithme converge vers un optimum local. Néanmoins, dans ce cas là, l'optimum local n'est jamais atteint et il est nécessaire de définir un critère d'arrêt.

1.3.2 Apprentissage d'un HMM à structure inconnue

Pour certaines applications, aucune connaissance ne permet d'inférer naturellement la structure du modèle de Markov caché. Le problème est alors, non seulement de paramétrer une structure, mais encore d'inférer cette structure à partir des exemples d'apprentissage. Plusieurs méthodes ont été proposées pour résoudre ce problème. Remarquons tout d'abord qu'une approche du type maximum de vraisemblance n'est clairement pas suffisante. En effet, il n'est pas difficile de construire un HMM qui a la probabilité maximale de générer les séquences d'apprentissage. Il suffit de construire un HMM équivalent à une disjonction de ces séquences : chaque chemin représente exactement une séquence d'apprentissage ; chaque état ne génère qu'un des symboles de l'alphabet et n'a qu'une transition sortante. Le HMM construit de cette manière est spécifique aux séquences d'apprentissage (il ne peut générer que ces séquences), et est donc inutilisable en pratique puisque incapable de reconnaître d'autres séquences. On se trouve alors face à un problème classique en apprentissage, celui de la recherche d'un biais inductif, c'est-à-dire d'une règle de généralisation *a priori* permettant d'inférer des HMMs qui puissent être utilisés sur de nouvelles séquences.

Apprentissage par généralisation L'apprentissage d'un HMM par généralisation procède de manière gloutonne et ascendante. Cette technique consiste à construire un HMM H spécifique comme exposé ci-dessus,et, par une opération élémentaire de fusion d'états, à transformer progressivement ce HMM en un modèle plus simple et plus général. Ce gain en généralité s'accompagne généralement d'une baisse de la vraisemblance des séquences d'apprentissage.
Cette méthode construit ainsi une suite de HMMs $(H_i)_{1\leq i\leq n}$, où H_n est le HMM obtenu en fusionnant deux états de H_{n-1}. Après chaque fusion, les paramètres de la structure sont réestimés de manière à maximiser les chemins de Viterbi des séquences d'apprentissage. Plutôt que d'utiliser l'algorithme d'entraînement de Viterbi à chaque itération, les chemins de Viterbi sont supposés préservés par les fusions. En d'autres termes, on suppose que les chemins de Viterbi d'après la fusion peuvent être déduits des chemins de Viterbi d'avant la fusion en remplaçant simplement les occurrences des états fusionnés par l'état résultant de la fusion. Sous cette hypothèse, les paramètres du HMM se réestiment aisément grâce à (9) et (10). De plus, seuls les paramètres attachés au nouvel état et à ses transitions adjacentes ont à être réestimés puisque la fusion n'a pas d'effet sur les autres états et transitions.
Différentes techniques peuvent être utilisées pour guider la fusion au cours de l'apprentissage. Il faut d'abord choisir un couple d'états à fusionner à chaque itération, puis définir un critère d'arrêt pour l'algorithme. Par exemple, on peut utiliser un biais s'exprimant comme une probabilité *a priori* sur l'ensemble des HMMs possibles. Le critère de fusion est alors la maximisation d'une probabilité *a posteriori* construite suivant le théorème de Bayes :

$$P(H \mid O) = \frac{P(O \mid H)P(H)}{P(O)}$$

où $P(H)$ est la probabilité *a priori* du HMM H,$P(O \mid H)$ est la probabilité de générer les séquences d'apprentissage O avec H (*cf.* (6) ou (7)) et $P(O)$ est la probabilité d'observer

les séquences O. De manière à favoriser les structures de tailles réduites, on peut exprimer la probabilité *a priori* du HMM à l'aide d'une fonction du type $P(H) = e^{-l(H)}$, où $l(H)$ est la taille du HMM calculée en fonction du nombre d'états et de transitions. À chaque itération, on choisit alors le couple d'états dont la fusion engendre l'augmentation la plus importante de la probabilité *a posteriori* $P(H \mid O)$ ou plus simplement du produit $P(O \mid H)P(H)$, puisque $P(O)$ est une constante indépendante du HMM. La procédure est stoppée lorsque plus aucun couple ne permet d'augmenter cette probabilité. Le critère de fusion et d'arrêt définit donc un compromis entre généralisation et vraisemblance des séquences d'apprentissage.

Apprentissage par spécialisation À l'inverse de l'apprentissage par généralisation, l'apprentissage par spécialisation consiste à partir d'un HMM très général (par exemple composé d'un unique état qui boucle sur lui-même), et par une opération de spécialisation (création d'un nouvel état, ajout d'une transition entre deux états existants), de construire progressivement un HMM plus spécifique de manière à augmenter la vraisemblance des séquences d'apprentissage.

1.4 Différents types de HMM

1.4.1 Quelques structures de modèles

Différentes topologies de HMM peuvent être envisagées, à commencer par le modèle **ergodique**, celui où chaque état peut être atteint à partir de n'importe quel autre en un nombre fini de transitions (fig 1). Le modèle est donc caractérisé par une matrice T aux coefficients strictement positifs. Une structure couramment utilisée est celle des **modèles**

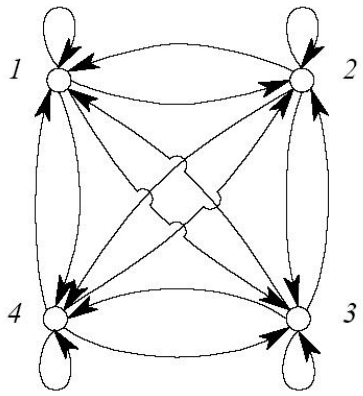

Fig. 1: modèle de chaîne de Markov ergodique

de Bakis, dits également **modèles gauche-droite** (figure 2), ainsi appelés parce qu'ils n'autorisent aucune transition d'un état vers un autre d'indice inférieur : les états qui se succèdent ont donc des indices égaux ou supérieurs aux précédents. Une fois dans le dernier état, le système est condamné à y rester : c'est pourquoi la probabilité initiale du premier état est posée égale à 1, les autres étant égales à 0. Dans le même ordre d'idées,

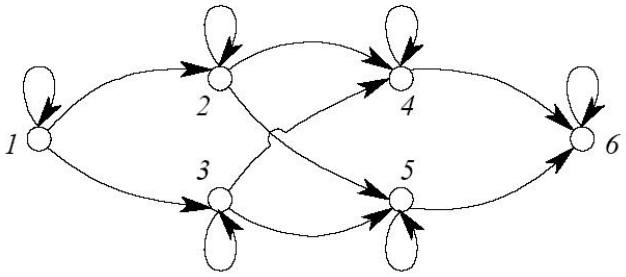

Fig. 2: modèle de Bakis

les sauts de nombreux états consécutifs sont souvent interdits ($T(i,j) = 0, j - i > \Delta$). Une variante du modèle gauche-droite est le modèle gauche-droite parallèle, utile, par exemple, en reconnaissance de la parole, où il permet de prendre en compte la possibilité de multiples prononciations d'un même mot.

Les procédures de ré-estimation des paramètres d'un modèle de Markov sont telles que, si certains de ces paramètres sont initialement mis à zéro, ils resteront nuls tout au long de la phase d'entraînement. Ceci permet donc de conserver les contraintes imposées à la structure du modèle. Dans le cas spécifique d'un modèle de Bakis, l'entraînement ne peut toutefois pas être réalisé à partir d'une seule séquence d'observations car, étant donné la nature du modèle, trop peu d'observations dans une séquence sont associées à un état particulier. Afin de disposer de données suffisantes pour l'estimation des paramètres, la procédure de ré-estimation doit donc être modifiée de façon à prendre en compte des séquences d'observations multiples.
BIen sûr, il existe évidemment encore bien d'autres topologies de Modèles de Markov Cachés.

1.4.2 Durée des états

On peut montrer que la probabilité qu'un état s_i produise d observations élémentaires est une fonction exponentielle décroissante de d. Ceci constitue l'une des plus grandes faiblesses des modèles de Markov car cette modélisation s'avère inappropriée pour représenter la plupart des signaux physiques. Une amélioration possible du modèle est d'inclure explicitement les distributions $p_i(d)$ de la durée des états, au prix cependant d'une augmentation du nombre de paramètres et, surtout, du volume de calcul. Une autre solution, beaucoup plus avantageuse, consiste à introduire une condition de durée minimale des états du modèle de Markov caché. Ceci s'avère particulièrement utile lorsque les séquences comportent davantage de boucles que de transitions d'un état vers un autre, les exponentielles prenant alors des valeurs beaucoup trop faibles.

1.4.3 Modèles de Markov à temps continu

Jusqu'à présent, seuls des Modèles de Markov Cachés modélisant des séquences d'observations discrètes ont été envisagés : ces observations prenaient des valeurs appartenant à

un alphabet fini, et leurs probabilités étaient définies dans chaque état par un ensemble fini de valeurs. Le problème est que les observations sont souvent des vecteurs continus : l'utilisation de modèles à distributions discrètes implique donc une phase préalable de quantification de ces vecteurs, avec les dégradations qui en résultent. Il est dès lors intéressant d'inclure des densités d'observations continues dans les modèles de Markov. Néanmoins, afin de limiter le nombre de paramètres de ces distributions, il faut apporter quelques restrictions à la forme de ces densités de probabilité. La forme la plus commune de ces fonctions se présente comme une somme finie de gaussiennes, ou multigaussiennes :

$$P(O|s) = \sum_{m=1}^{M} c_{sm} \mathcal{N}_O(\mu_{sm}, \Sigma_{sm})$$

Les paramètres du modèle à ajuster sont alors, en plus de T, G et π, les coefficients de pondération c_{sm}, les vecteurs de moyenne μ_{sm} et les matrices de covariance Σ_{sm}. Les formules de réestimation nécessitent alors la définition de :

$$\gamma_t(s,m) = \left[\frac{\alpha_t(s)\beta_t(s)}{\sum_{s' \in S} \alpha_t(s')\beta_t(s')} \right] \left[\frac{c_{sm}\mathcal{N}_O(\mu_{sm}, \Sigma_{sm})}{\sum_{n=1}^{M} c_{sn}\mathcal{N}_O(\mu_{sm}, \Sigma_{sn})} \right]$$

qui généralise $\gamma_t(s)$ au cas continu. Ce coefficient s'interprête comme la probabilité de se trouver en s au temps t, en ne prenant en compte que la $m^{\text{ème}}$ composante de la multigaussienne de l'état considéré en o_t. Les formules de réestimation deviennent alors

$$c_{sm} = \frac{\sum_{t=1}^{T} \gamma_t(s,m)}{\sum_{t=1}^{T} \sum_{n=1}^{M} \gamma_t(s,n)}$$

$$\mu_{sm} = \frac{\sum_{t=1}^{T} \gamma_t(s,m) o_t}{\sum_{t=1}^{T} \gamma_t(s,m)}$$

et la procédure demeure inchangée pour la réestimation des autres paramètres.
La représentation continue par multigaussienne des probabilités d'émission de chaque état permet de bien mieux estimer les densités d'observations réelles qu'avec un modèle à temps discret. La grande quantité de paramètres qui doivent alors être déterminés représente cependant un lourd handicap du modèle : si d est la dimension des vecteurs de primitives, l'estimation des probabilités de génération $P(o|s)$ nécessite, pour chaque état, la connaissance de $(d^2 + d + 1)M$ paramètres, contre $|\Sigma|$ dans le cas d'un alphabet fini. L'entraînement d'un modèle de Markov à temps continu nécessite donc la disponibilité d'une très grande quantité de données.

Ces difficultés peuvent être limitées en supposant que les éléments non diagonaux des matrices de covariance sont nuls, c'est-à-dire en admettant que les composantes du vecteur caractéristique sont non corrélées. Ceci permet de réduire de d^2 à d le nombre de paramètres de chaque matrice de covariance. Une autre solution est aussi de limiter le nombre de gaussiennes : l'utilisation d'une seule gaussienne, même si cela restreint la qualité d'estimation des densités d'observations, peut conduire à de meilleurs résultats que l'utilisation d'un modèle de Markov à temps discret.

1.4.4 Modèles hybrides

Le grand intérêt des modèles de Markov cachés pour la classification est leur aptitude à traiter la nature à la fois statistique et séquentielle des observations. Néanmoins, ils présentent certaines faiblesses, notamment à cause de l'algorithme d'entraînement, qui optimise indépendamment les vraisemblances des observations étant donné le modèle $P(O|H_k)$, ne réalisant ainsi aucune discrimination entre les classes. Ceci justifie l'intérêt de reformuler le formalisme des modèles de Markov cachés en termes de probabilités *a posteriori* qui sont, quant à elles, discriminantes, puisque :

$$\sum_{k=1}^{C} P(H_k|O) = 1 \tag{11}$$

où C est le nombre total de classes. Ainsi, si l'un de ces termes augmente, tous les autres doivent, en moyenne, diminuer : la probabilité de l'adéquation d'un modèle avec l'observation est d'autant plus forte que celles des autres modèles sont faibles.
Les probabilités *a posteriori* sont de la forme

$$P(H_k|O) = \sum_{Q} P(Q, H_k|O)$$

où Q est une séquence d'états quelconque. La formule de Bayes permet alors d'écrire

$$P(Q, H_k|O) = \frac{P(O, Q|H_k)P(H_k)}{P(O)}$$

et ces probabilités sont donc, à un facteur d'échelle près, les vraisemblances utilisées dans les modèles de Markov conventionnels.
La maximisation de $P(H_k|O)$ conduit donc aux mêmes formules d'évaluation que dans le cas des modèles de Markov cachés. Il en va cependant tout autrement de la procédure de réestimation : (11) implique que toute modification des paramètres d'un modèle H_k nécessite la modification des paramètres des autres modèles. La procédure de réestimation des modèles de Markov cachés devra dès lors être modifiée, de façon à prendre en compte cette contrainte. On peut montrer que cette contrainte peut être approchée grâce à l'utilisation d'un perceptron multicouches en association avec le modèle de Markov caché.

2 PARTIE PRATIQUE

Scikit-learn ne maintient plus le module hmm, mais hmmlearn propose un module "à la scikit" qui gère les chaînes de Markov cachées. L'aide de ce module est disponible ici.

2.1 Génération à partir d'un HMM entraîné

Un système automatique composé d'une lampe et d'un mécanisme d'arrosage automatique est placé dans un pot de fleurs, l'objectif étant d'améliorer la croissance de la plante s'y trouvant. Chaque jour ce système est sensé fonctionner de 12h à 15h. Étant au travail, vous ne pouvez observer si le mécanisme a été opérationnel ou défectueux, vous observez juste l'effet indirect produit par son bon (ou mauvais) fonctionnement : la plante peut être en bonne santé, ou présenter des signes de carences lunineuse et hydrique. Vous faites l'observation de l'état de la plante chaque jour.
Le mécanisme étant directement relié à l'électricité, EDF assure que la probabilité qu'il soit en panne deux jours consécutifs est de 0.3, et qu'avec probabilité 0.2 il ne fonctionne pas le jour $j+1$ alors qu'il fonctionnait le jour j.
Si le système fonctionne la plante semble en bonne santé avec probabilité 0.8, et en cas de dysfonctionnement elle semble en carence avec probabilité de 0.7.
Enfin, on a aucun a priori sur l'état de fonctionnement du système automatique le premier jour.

1. Modélisez ce problème sous la forme d'une HMM $H = (S, \Sigma, T, G, \pi)$, où vous préciserez les valeurs des matrices G, T et π
2. Vous suivez pendant 6 jours l'état de la plante et vous relevez les observations suivantes : "bonne santé", "bonne santé", "carence", "carence", "carence", "bonne santé". Coder le HMM sous `hmmlearn` pour prédire l'état le plus probable du système automatique pendant cette période.

2.2 Challenge

On se donne un corpus de mots dans une langue donnée (l'anglais en l'occurrence, les corpus étant importants et nombreux), et on cherche s'il existe des propriétés élémentaires sur le langage écrit correspondant. Dans une première approche, on va chercher à déterminer si les caractères composant ce langage peuvent être partitionnés en N sous-ensembles ayant des caractéristiques distinctes. On commencera l'étude avec $N = 2$.

1. Récupérer le fichier `brown.txt`. Celui-ci contient le corpus Brown, "représentatif" de l'anglais parlé aux Etats-Unis au début des années 1960. Chaque ligne est de la forme `Xnn offset1 offset2`, suivis de 4 espaces minimum et d'un morceau de texte. `X` indique le genre du texte associé (A pour reportage, B pour éditorial, ...R pour humour)

2. isoler les mots des textes : retirer les ponctuations, chiffres, caractères spéciaux et convertir tous les caractères en minuscule. Cela donne ainsi M=27 symboles distincts (les 26 lettres de l'alphabet et l'espace).
3. On teste l'hypothèse que les caractères en anglais sont régis par un processus markovien caché à N=2 états, et que pour chacun des états cachés, les 27 symboles peuvent être générés par des distributions de probabilité stationnaires. On définit donc une chaîne de Markov cachée $H = (S, \Sigma, T, G, \pi)$, où $S = (S_0, S_1)$, $\Sigma = \{a, b, c \cdots z, \_\}$, $T \in \mathcal{M}_{2,2}([0,1])$,$G \in \mathcal{M}_{2,27}([0,1])$,$\pi \in \mathbb{R}^2$
4. déclarer un HMM et l'entraîner pour générer les matrices T, G et le vecteur π. On utilisera la méthode fit sur les données du corpus. On prendra garde à une initialisation pertinente des matrices de H.
5. Observer la matrice G à convergence. Qu'en déduire sur le partitionnement des lettres de l'alphabet anglais ? Est-ce étonnant ?
6. Même question pour π.
7. Utiliser H pour générer 50 mots.

RÉFÉRENCES

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41 :164–171, 1970.

[2] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.