Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Amélioration de l'apprentissage des Modèles de Markov Cachés pour l'estimation de l'état de santé d'un système industriel

Improved learning of Hidden Markov Model for estimating the state of health of an industrial system

KHODJA Nesrine Keltoum
Université d'Orléans, Laboratoire PRISME, IUT de l'Indre,
Châteauroux
Nesrine.khodja@univ-orleans.fr

DUCULTY Florent
Université d'Orléans,Laboratoire PRISME, IUT de l'Indre
Châteauroux
florent.duculty@univ-orleans.fr

BEGOT Stéphane
Université d'Orléans, Laboratoire PRISME, IUT de l'Indre
Châteauroux
stephane.begot@univ-orleans.fr

AVILA Manuel
Université d'Orléans, Laboratoire PRISME, IUT de l'Indre
Châteauroux
manuel.avila@univ-orleans.fr

I. Introduction

La production de données numériques (puissance, courant, tension, vibration, régime, images, vidéo) par des capteurs, des machines ou des êtres humains augmente considérablement de jour en jour. Le traitement, l'analyse et l'interprétation des données, ainsi que l'acquisition de connaissances à partir des données, sont des éléments essentiels dans notre société. Dans un tel contexte technologique, l'utilisation de la reconnaissance de formes (RP) et de l'intelligence artificielle (IA) est devenu importante. Les données des capteurs, la parole, les images, le langage et les documents peuvent être analysées grâce aux domaines du traitement du signal, de la reconnaissance des formes et de l'intelligence artificielle. Il est donc plus que jamais nécessaire de mettre en évidence les spécificités et les complémentarités de ces domaines afin de résoudre ces problèmes.

La plupart des chercheurs se sont consacrés au développement et à l'amélioration d'algorithmes d'identification et de classification. Pour prédire à partir d'énormes quantités de données, les méthodes de modélisation sont fondées sur l'apprentissage automatique. Les méthodes de classification comprennent les réseaux neuronaux (NN) (Gong et al.,2021), la régression à vecteur de support (SVR) (Qiu et al. ,2020) et les machines à vecteur de support (SVM) (Li et al.,2021).

Dans le cas d'événements séquentiels (dans le temps ou dans l'espace), l'utilisation de modèles de Markov cachés (MMC) est efficace et a été appliquée avec succès dans le domaine de l'identification. En raison de leurs puissantes capacités de reconnaissance des formes, de nombreux chercheurs ont appliqué les MMC dans plusieurs domaines : reconnaissance de caractères, reconnaissance vocale, reconnaissance faciale (Kim et al.,2003), caractérisation en médecine (toux) (Liu et al. ,2015), reconnaissance du comportement, ainsi que dans le diagnostic et le pronostic des défaillances (Lee et al.,2004). Dans certains travaux, les chercheurs ont contribué à améliorer l'utilisation des modèles de Markov cachés (MMC) dans les systèmes d'intelligence artificielle, comme dans les travaux de (Aupetit et al.,2007), qui ont introduit les algorithmes génétiques, l'algorithme de fourmis artificielles pour améliorer l'apprentissage des modèles de Markov cachés. Les MMC ont également été utilisés dans des problèmes multi-classes en les combinant avec d'autres classificateurs, comme dans (Martín-Iglesias et al.,2005), pour effectuer une reconnaissance vocale multi-classes fondée sur un SVM avec une segmentation guidée par un MMC, ou (Wang et al., 2022) qui ont fusionné un réseau neurone profond et un MMC pour reconnaître le comportement multi-classes anormal des personnes âgées. Les MMC ont également été utilisés dans les problèmes de clustering, comme dans le travail de (Smyth,1997), mais dans notre étude, l'objectif n'est pas de faire du clustering à l'aide de MMC, mais de résoudre des problèmes multi-classes. Pour ce faire, nous introduirons une nouvelle stratégie d'apprentissage. Cette stratégie s'inspire des principes de non-comportement (NBA) (Cao et al.,2015).

II. Les Models de Markov Caches (MMC)

D'abord utilisées pour résoudre le problème de la reconnaissance de la parole (Rabiner,1989), puis pour la reconnaissance de caractères, les MMC ont la capacité d'assimiler la variabilité des observations (ou des caractéristiques) à une structure (ou une topologie) spécifique. Les MMC, avec la propriété de Markov, ont la capacité de détecter la forme ou la signature spécifique d'une séquence d'événements (ou d'observations). La propriété de Markov signifie que les états futurs d'un processus dépendent uniquement de l'état actuel. Depuis des dizaines d'années, de nombreux travaux ont été publiés dans plusieurs domaines sur les différentes utilisations des MMC. Dans cet article, nous nous concentrons sur le problème de l'apprentissage de MMC discrets dans le cas d'une mise en compétition de plusieurs classes. Un modèle de Markov est caractérisé par les éléments suivants :

- $S=\{S_1,S_2,\ldots S_N\}$: État individuel avec N le nombre d'états dans le modèle, ces états sont cachés.
- $V = \{v_1,v_2,\ldots,v_M\}$: Symboles individuels avec M le nombre de symboles d'observation distincts par état.
- $Q=\{q_1,q_2,\ldots q_T\}$: Séquence d'états avec T la longueur de la séquence, l'état à l'instant t est q_t.
- $O = \{O_1,O_2,\ldots,O_T\}$: Séquence d'observations où chaque O_t est une des formes de symboles suivantes forme de symbole V.
- La distribution de probabilité des transitions d'état $A = \{a_{ij}\}$ avec :

$$a_{ij}=P\left[q_{t+1}=S_j \mid q_t = S_i\right] \quad 1\leq i,j\leq N \tag{1}$$

- La distribution de probabilité du symbole d'observation dans l'état j, $B= \{b_j(k)\}$ est:

$$\begin{cases} b_j(k)=P[v_k \text{ à } t \mid q_t=S_j] \\ 1\leq j\leq N \text{ et } 1\leq k\leq M \end{cases} \tag{2}$$

- π le vecteur d'état initial où $\pi = \{\pi i\}$.

Un modèle de Markov caché discret peut être défini comme suit:

$$\lambda = (\pi, A, B) \tag{3}$$

Dans cet article, nous considérons qu'il y a plusieurs modèles en compétition. Pour chaque catégorie ou classe, nous utilisons un modèle de 1 à R, avec R le nombre de classes.

$$\lambda_r= (\pi_r, A_r, B_r),\ r \in [1,R] \tag{4}$$

A. Type de MMC

La topologie des MMC peut être adaptée en fonction de la spécificité du système à modéliser. Plusieurs types de topologies peuvent être utilisés. Par exemple, un modèle ergodique (Fig. 1 (a)) ou un modèle gauche-droite (Fig. 1 (b)). Le modèle ergodique ne comporte aucune restriction. Il offre au modèle de nombreuses possibilités d'assimiler des observations ou des événements au cours de la phase d'apprentissage. Les modèles gauche-droite sont mieux adaptés aux événements orientés. Ces types de topologies sont souvent utilisés dans le domaine de la reconnaissance de caractères ou de mots. Définir une topologie signifie seulement mettre certaines probabilités de transition à zéro.

B. Trois problèmes fondamentaux pour les MMC

Pour exploiter les MMC de nos jours, nous disposons de diverses boîtes à outils. Nous utilisons l'une d'entre elles disponible dans Matlab. Les algorithmes habituels sont utilisés pour résoudre les trois problèmes de base des MMC: l'évaluation de $P(O|\lambda)$, le décodage et l'apprentissage.

- **Évaluation** : étant donné un MMC, il est essentiel de déterminer la probabilité qu'une séquence observée soit générée par ce modèle. Les algorithmes Forward et Backward sont utilisés pour cette tâche. Cependant, dans le cadre d'un dispositif de suivi de l'état de santé d'un système, seule la variable Forward sera utilisée, la variable Backward étant calculée depuis la fin de la chaîne. Cela suppose que l'on a déjà atteint l'état de panne.

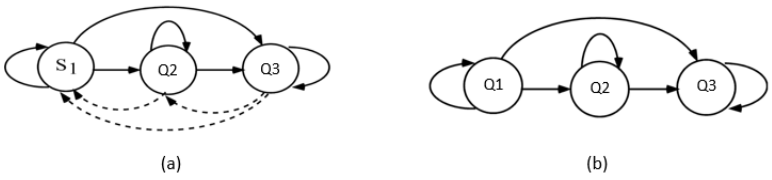

Fig. 1. Type de MMC (a : ergodique, b : gauche-droite).

- **Décodage** : les états d'un MMC ne sont pas observables. Le problème du décodage consiste à chercher la séquence d'états Q la plus probable, étant donné une séquence d'observations O. L'algorithme de Viterbi est généralement utilisé pour résoudre ce problème.

- **Apprentissage** : le but est d'ajuster les paramètres du MMC, afin de maximiser la probabilité d'observation P(O|λ). L'algorithme Baum Welch est souvent utilisé pour cette tâche. En débutant le processus d'apprentissage avec un ensemble de paramètres (π, A, B) tiré aléatoirement, un processus récursif va permettre de réestimer les paramètres du modèle pour converger vers un modèle $\bar{\lambda}$ qui maximise la vraisemblance des séquences O du corpus d'apprentissage sachant le modèle P(O|λ). L'apprentissage peut également s'effectuer avec l'algorithme de Viterbi. Dans ce cas, seule la séquence d'états la plus probable est prise en compte dans le processus récursif de réestimation des paramètres, contrairement à l'algorithme de Baum Welch qui considère l'ensemble des séquences d'états possibles.
  D'autres variantes ont été développées (Aupetit,2007), comme les algorithmes génétiques ou des fourmis. L'idée générale, pour éviter de converger vers un minima local, est de remettre en question l'apprentissage en cours pour repartir avec de nouveaux paramètres obtenus par assemblage de différents paramètres d'apprentissage.

III. Les Algorithmes De Base

Dans cette section, nous décrirons les différents algorithmes utilisés dans les MMC.

A. Algorithme de Forward-Backward

Si nous voulons calculer la probabilité de générer la séquence de symboles $O=\{O_1,O_2,...,O_T\}$ à l'aide d'un MMC, nous utilisons l'algorithme Forward (Rabiner ,1989) pour lequel nous définissons la variable suivante:

$$\alpha_t(i) = P(O_1, O_2 \dots O_t, q_t = S_i|\lambda) \quad (5)$$

Cette variable exprime la probabilité d'avoir généré la séquence $O = \{O_1,O_2,...,O_T\}$ depuis le début (start) jusqu'à l'état S_i à l'instant t (voir l'équation 6).

$$P(O|\lambda)= \left(\sum_{i=1}^{N} \alpha_t(i)\right) \quad (6)$$

Cet algorithme est appelé "Forward" car la récursivité est effectuée en avant : on calcule d'abord la probabilité de générer le premier symbole de la séquence, puis on ajoute un symbole à chaque étape de la récursivité et on répète la procédure jusqu'à ce que la probabilité de générer l'ensemble de la séquence ait été calculée. Un algorithme inverse similaire « Backward» peut être utilisé pour effectuer le calcul en sens inverse. La variable $\beta_t(i)$ est défini comme suit :

$$\beta_t(i)=P(O_{t+1},O_{t+2}...O_T,q_t=S_i|\lambda) \quad (7)$$

Elle exprime la probabilité d'une séquence d'observations partielles $O_{t+1}, O_{t+2} \dots O_T$ compte tenu de l'état S_i à l'instant t et du modèle λ (8)

$$\beta_t(i) = \left(\sum_{j=1}^{N} a_{ij} b_j O_{t+1} \beta_{t+1}(j)\right) \quad (8)$$

B. Algorithme de Viterbi

L'algorithme de Viterbi est une solution optimale de maximum de vraisemblance pour l'estimation d'une séquence d'états d'un processus de Markov avec des temps discrets et un nombre d'états fini. Rappelons que l'algorithme de Viterbi est une solution au problème 2 abordé dasns la section précédente. Avec l'algorithme de Viterbi, nous recherchons le chemin le plus probable $Q = \{q_1,q_2,...q_T\}$ pour lequel la vraisemblance est maximale, c'est-à-dire:

$$P\big((q_1,q_2,...q_T)\big|O,\lambda\big) \quad (9)$$

Pour cela, nous définissons:

$$\delta_t(i)= \max_{q_1,\, q_2 \dots,\, q_{t-1}} P(\, q_1,q_2,...q_t=i,\, O_1,O_2,...,O_t|\lambda) \quad (10)$$

$\delta_t(i)$ est la probabilité maximale des chemins se terminant dans l'état i. Par induction, nous obtenons:

$$\delta_{t+1}(j)= \max_i\big[\delta_t(i)a_{ij}\big].b_j(O_{t+1}) \quad (11)$$

Description de l'algorithme

Initialisation :

$$\begin{cases} \delta_1(i)=\pi_i b_i(O_1),\ 1\le i\le N \\ \Psi_1(i)=0 \end{cases} \quad (12)$$

Récursion :

$$\delta_t(j)= \max_{1\le i\le N} [\,\delta_{t-1}(i)a_{ij}].b_j(O_t) \quad (13)$$

$$\begin{cases} \Psi_t(j)=\arg \max_{1\le i\le N} [\,\delta_{t-1}(i)a_{ij}] \\ \text{avec} \\ 2\le t\le T, 1\le j\le N \end{cases} \quad (14)$$

Évaluation du meilleur score (c'est-à-dire du chemin le plus probable) :

$$P^*= \max_{1\le i\le N}[\delta_T(i)] \quad (15)$$

$$Q_T^*= \operatorname{argmax}_{1\le i\le N}[\delta_T(i)] \quad (16)$$

$$Q_t^* = \Psi_{t+1}(Q_{t+1}^*), \quad t=T-1,T-2,\ldots,1 \quad (17)$$

Dans la pratique, nous prenons le logarithme de chacune des probabilités impliquées, de sorte que tous les produits de probabilité sont remplacés par des sommes.

IV. Apprentissage des MMC

Dans cette section, nous nous concentrons sur l'apprentissage des MMC à partir d'une structure connue et d'un ensemble d'apprentissage.

Soit $O_{app} = \{O_{app\,1}, O_{app\,2}, \ldots O_{app\,L}\}$ l'ensemble des séquences d'entraînement. Ces séquences sont supposées indépendantes, de sorte que la probabilité de générer l'ensemble d'apprentissage est simplement le produit des probabilités de générer chaque séquence. Notre objectif est alors de trouver les paramètres $\lambda = (\pi, A, B)$ qui maximisent :

$$P(O_{app}|\lambda)= \prod_{l=1}^{L} P(O_{app\,l}|\lambda) \quad (18)$$

La plupart des méthodes d'apprentissage des MMC sont dérivées de l'algorithme d'Esperance-Maximisation (EM). L'algorithme EM général est attribué à (Dempster et al.,1997), cet algorithme conduit à une séquence de paramètres dont les valeurs de vraisemblance augmentent de façon monotone. Ils ont également inventé le terme "algorithme EM". Il est important de noter que l'algorithme EM pour les MMC a été décrit comme l'algorithme de Baum-Welch (Baum et al.,1970).

A. Algorithme de Baum_Welch

L'algorithme de Baum-Welch estime les paramètres du MMC λ en maximisant $P(O|\lambda)$.

$$\xi_t(i,j)=P(q_t=S_i,q_{t+1}=S_j|O,\lambda) \quad (19)$$

La probabilité que nous générions O en passant par l'état i au temps t et par l'état j au temps t+1, cette équation peut être réécrite comme suit :

$$\xi_t(i,j)= \frac{P(q_t=S_i,q_{t+1}=S_j|O,\lambda)}{P(O|\lambda)} \quad (20)$$

Et en utilisant les variables Forward et Backward décrites dans la section précédente :

$$\begin{cases} \xi_t(i,j)= \dfrac{\alpha_t(i)a_{ij}b_{ij}(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \\ = \\ \dfrac{\alpha_t(i)a_{ij}b_{ij}(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \end{cases} \quad (21)$$

En considérant la probabilité a posteriori, $\boldsymbol{\delta}_t(i)$:

$$\delta_t=P(q_t=S_i|O,\lambda) \quad (22)$$

Elle exprime la probabilité qu'en générant de O avec λ on se trouve à l'état i à l'instant t. Exprimée comme une fonction de $\xi_t(i,j)$, $\delta_t(i)$:

$$\delta_t = \sum_{j=1}^{N} \xi_t(i,j) \tag{23}$$

Si l'on fait la somme de $\delta_t(i)$ sur l'ensemble des instants t, on obtient une quantité qui peut être interprétée comme l'espérance du nombre de fois où l'état q est utilisé pour générer la séquence O. De même, si l'on fait la somme de $\xi_t(i,j)$ sur l'ensemble des instants t, on obtient des quantités qui peuvent être interprétées comme l'espérance du nombre de fois où la transition i→ **j** est utilisée pour générer la séquence O. On a donc :

$$\sum_{t=1}^{T-1} \delta_t(i) = \text{Espérance du nombre des passages } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{Espérance du nombre de transition de } i \rightarrow j$$

B. Algorithme d'apprentissage par Viterbi

Il existe une autre façon d'entraîner les MMC à l'aide de l'algorithme de Viterbi : si les chemins de Viterbi des séquences d'apprentissage sont connus, chaque état, chaque transition et chaque symbole attaché aux états des MMC peut être associé au nombre de fois qu'il a été utilisé pour générer ces séquences.
L'algorithme d'apprentissage de Viterbi est fondé sur le même principe que l'algorithme EM, sauf qu'ici nous remplaçons la partie E par le chemin de Viterbi.

$$P(O_{app}, Q^* | \lambda) = \prod_{l=1}^{L} P(O_{app\,l}, q^{*l} | \lambda) \tag{24}$$

Avec q^{*l}, le chemin de Viterbi dans la séquence $O_{app\,l}$ est λ $Q^* = \{q^{*1}, q^{*2}, \ldots q^{*L}\}$. Les paramètres ne sont donc pas estimés en maximisant la probabilité réelle de génération des séquences d'apprentissage, mais plutôt la probabilité de génération le long du chemin le plus probable.

Soit $n_i, n_{i\rightarrow j}$ et n_i^o respectivement le nombre de fois où l'état i est utilisé, le nombre de fois où la transition i→j est utilisée et le nombre de fois où le symbole O est généré par l'état q dans le chemin de Viterbi :

$$\bar{a}_{ij} = \frac{n_{i\rightarrow j}}{n_i} \tag{25}$$

Et

$$\bar{b}_j = \frac{n_i^o}{n_i} \tag{26}$$

Nous avons illustré le chemin de Viterbi sur la (Fig. 2), et le calcul des nombres de transition d'un état à l'autre sur la (TABLE.I).

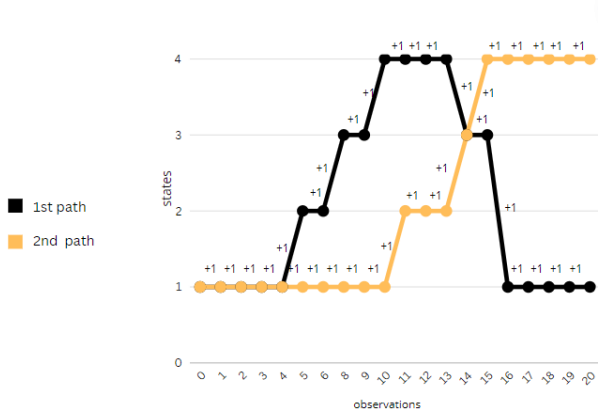

Fig. 2. Illustration des meilleurs chemins de Viterbi pour deux séquences

Chaque fois que nous passons d'un état à un autre, nous incrémentons de + 1, la cellule de n i→ *j* du chemin de Viterbi est indiquée dans la (TABLE. I) :

TABLE I. 1ER ET 2EME CHEMIN DE VITERBI

VERS j / De i	1	2	3	4
1	+4+10	+1+1	0	0
2	0	+1+2	+1+1	0
3	1	0	+2	+1
4	0	0	+1	+3+5

Les chemins de Viterbi sont utilisés pour calculer le nombre de fois que chaque symbole, état et transition d'état est utilisé. Les paramètres A et B sont ensuite réestimés jusqu'à ce que la stabilité soit atteinte. Cet algorithme d'apprentissage est résumé dans l'algorithme.1.

ALGORITHME 1. Apprentissage par Viterbi

Corpus apprentissage
O_{app} = {$O_{app\ 1}$, $O_{app\ 2}$, … $O_{app\ L}$}
Choisir un modèle initial λ= (A B, π)
Répéter
  Initialisation de toutes les variables : $n_i, n_{i\to j}$ et n_i^o à 0
  POUR chaque $O_{app\ l} \in O_{app}$ **Faire**
    Calculer le chemin de Viterbi de $O_{app\ l}$
    Ajuster les variables $n_i, n_{i\to j}$ et n_i^o la contribution de $O_{app\ l}$
  FIN Pour

  Estimer les nouveaux paramètres λ = (A B, π) en utilisant les Formules (25) et (26).
Jusqu'à la stabilité

V. UTILISATION DES MMC DANS UN PROBLÈME MULTICLASSES

En général, dans la classification MMC standard, chaque modèle est estimé à l'aide d'échantillons spécifiques à sa classe. Par exemple, nous considérons 4 classes (R=4), $O^r = \{O_1^r, O_2^r, ..., O_L^r\}$ avec r ∈ [1, R].

Pour classer des échantillons inconnus de la classe 1, seules les données d'apprentissage de la classe 1 ($O^1_{app} = \{O^1_{app\ 1}, O^1_{app\ 2}, \dots, O^1_{app\ L}\}$) doivent être utilisées mais dans notre étude, nous voulons utiliser l'information potentiellement contenue dans les autres classes dans ce que nous avons appelé non-modèle. Comme le montre la figure 3, l'opposé du modèle 1 serait la combinaison d'échantillons provenant des autres classes pour estimer le non modèle1.

Dans notre étude, l'objectif est d'intégrer l'information contenue dans le non-modèle pour améliorer l'apprentissage du modèle. Afin de valider nos hypothèses, nous utilisons la boîte à outils Matlab pour produire des données. Nous avons choisi une topologie gauche-droite et divers paramètres ont été calculés aléatoirement. Avec cette stratégie, plusieurs tests peuvent être effectués pour vérifier la pertinence des résultats. Comme nous avons utilisé des modèles synthétiques, nous pouvons avoir des séquences d'états

qui produisent des séquences d'observations. L'objectif de notre recherche est d'intégrer l'information contenue dans les non-modèles pour améliorer l'apprentissage des modèles (Khodja et al.,2022).

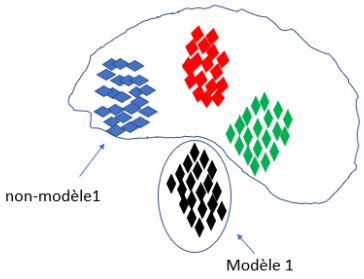

Fig. 3. Principe du modèle et du non-modèle

VI. Méthode proposée

R ensembles de paramètres de modèle ont été sélectionnés au hasard pour produire des données. L'ensemble des états et des observations peut être configuré (N et M). Le nombre de classes R est également choisi avant de générer des données.

A. Production des données avec des modèles synthétiques

Les différentes étapes de la production de données peuvent être résumées comme suit:

Étape 1: Sélection aléatoire des modèles

La première étape consiste à générer λ_r modèles avec Matlab. Ces modèles représentent R classes différentes.

***Étape 2*:** Génération de données

Des observations synthétiques sont générées à partir de ces R modèles, en définissant un nombre de séquences d'observations O, de longueur T.

***Étape 3*:** Corpus de formation et de test

Ces observations sont divisées en deux corpus: les données d'apprentissage et les données de test. Nous avons opté pour 2/3 des données utilisées pour l'apprentissage du modèle et 1/3 pour les données de test.

***Étape 4*:** phase d'apprentissage

Pour la phase d'apprentissage, nous avons utilisé les algorithmes de Viterbi et de Baum-Welch pour la réestimation du modèle de chaque λr.

***Étape 5*:** Phase d'identification et de classification

Pour vérifier la capacité des modèles à identifier chaque classe, nous avons utilisé le maximum de vraisemblance. Les cinq étapes sont présentées dans la (Fig. 3). Les résultats de cette classification fondés sur l'approche classique, fourniront les taux de reconnaissance de référence pour la suite de l'étude.

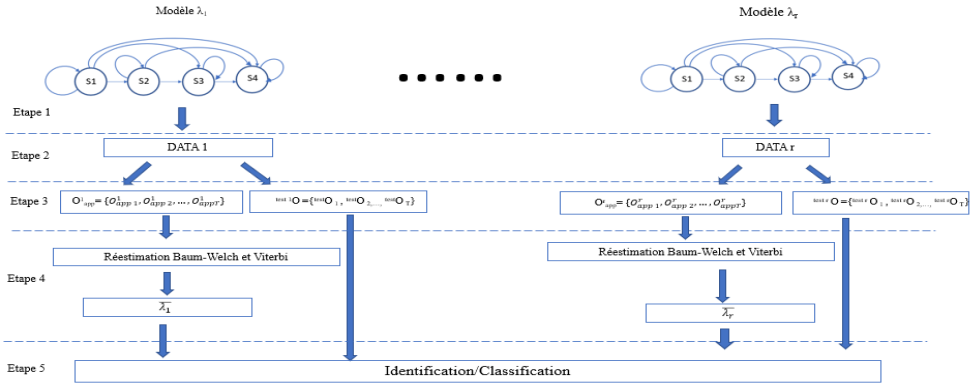

Fig. 4. Classification usuelle en utilisant les MMC

B. Discrimination par non-modèle

Dans la phase suivante de notre étude, nous avons examiné des échantillons des autres classes pour apprendre ce que nous appelons le non-modèle (ce que nous ne sommes pas). Dans cette partie, nous avons choisi R = 4 classes avec N = 4 états et M =16 symboles. Les données d'apprentissage du non-modèle sont présentées dans la (Fig. 5).

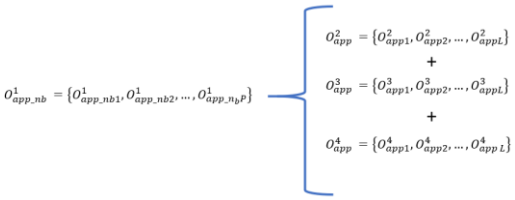

Fig. 5. Ensemble d'apprentissage du non-modèle 1

Dans la phase de test qui vise à identifier des échantillons inconnus, nous calculons la probabilité de chaque séquence avec chacun des non-modèles qui ont été estimés. Puisque nous avons appris le "contraire" des séquences (ou formes) que nous essayons d'identifier, nous recherchons le non-modèle qui ressemble le moins à la séquence à identifier. Le non-modèle ayant la probabilité la plus faible est celui qui identifie la non-classe. (La Fig. 6) résume la méthode.

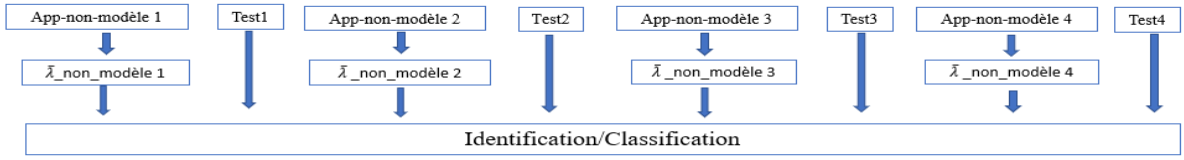

Fig. 6. Méthode de classification par non-modèles

VII. Performance de l'approche classique

Après avoir défini la méthodologie, nous présentons les résultats obtenus par la modélisation "classique", qui serviront de référence pour quantifier la capacité de discrimination des non-modèles. Pour éviter de juger une situation qui pourrait être anecdotique, liée au tirage aléatoire des paramètres du modèle, les tests ont été répétés plusieurs fois. C'est la moyenne des résultats obtenus qui est fournie.

A. Résultats pour les modèles synthétiques

Nous avons généré quatre modèles (R=4). La topologie de chaque modèle est de type gauche-droite à quatre états (N=4) avec seize symboles différents (M=16) émis. La longueur des séquences est de T= 20 observations. Chaque base de données contient 1000 séquences, les corpus d'apprentissage contiennent 700 séquences, tandis que les corpus de test contiennent 300 séquences. Nous avons d'abord effectué l'apprentissage à l'aide de l'algorithme de Baum-Welch. Ensuite, nous avons calculé la vraisemblance de chaque séquence avec chacun des modèles estimés. Pour la phase d'identification, nous avons calculé le maximum de vraisemblance. Les résultats sont présentés sous la forme d'une moyenne des résultats des 4 classes (TABLE. II).

TABLE II. Résultat de la classification par les modèles

Algorithmes	Résultats (%)
Baum-Welch	81.45
Viterbi	71.52

Nous constatons que les résultats obtenus par l'apprentissage de Baum-Welch sont meilleurs que ceux obtenus par l'apprentissage de Viterbi.

B. Résultats pour les non-modèles

Dans la partie non-modèle, nous avons d'abord construit nos corpus d'apprentissage non-modèle comme le montre la (Fig. 5). Chaque corpus d'apprentissage non-modèle contient 700 séquences*(R-1), dans notre cas R=4, soit P= 2100 séquences d'observations d'apprentissage. Nous avons ensuite effectué la même approche pour les modèles (d'abord apprentissage par Baum-Welch, puis Viterbi). Nous avons calculé la vraisemblance de chaque séquence pour chacun des non-modèles estimés. Ensuite, nous avons recherché le minimum de vraisemblance. Les résultats sont présentés dans la TABLE III.

TABLE III. RÉSULTAT DE LA CLASSIFICATION PAR LES NON- MODÈLES

Algorithmes	Résultats des non-modèles (%)	Résultats des Modèles (%)
Baum-Welch	70.75	81.45
Viterbi	68	71.52

Ces résultats montrent la capacité de ces non-modèles à identifier les classes. Mais l'information contenue dans ces "non-modèles" est-elle complémentaire de celle du modèle ? Pour répondre à cette question, nous avons étudié la nature des erreurs commises par les modèles et les non-modèles, nous présentons un bref aperçu des erreurs (échantillons appartenant à la classe1 et à la non-classe1, mais mal identifiés par le modèle ou le non-modèle) :

- Échantillon 1 : identification correcte pour le modèle et le non-modèle.
- Échantillons 10 et 32 : mêmes erreurs pour le modèle et le non-modèle.
- Échantillons 2 et 27 : erreur pour le modèle mais bonne identification pour le non-modèle.

A l'aide des différents tests effectués, nous avons constaté la capacité des non-modèles à identifier correctement les échantillons. Nous avons ensuite développé un nouvel algorithme d'apprentissage qui exploite l'information contenue dans les non-classes pour améliorer l'apprentissage des modèles.

VIII. ALGORITHME NON-BEHAVIOR VITERBI (NB VITERBI)

Afin d'améliorer l'apprentissage des modèles de Markov cachés, nous avons développé un algorithme qui exploite à la fois l'information contenue dans les classes et l'information contenue dans les autres classes (non-modèles). Dans l'algorithme d'apprentissage habituel de Viterbi, les meilleurs chemins sont utilisés pour améliorer les probabilités en incrémentant chaque transition. Plus une séquence d'états est fréquente, plus la probabilité associée est élevée.

Le principe de ce nouvel algorithme est de prendre en compte à la fois l'information contenue dans les modèles et celle des non-modèles. Pour tenir compte de l'influence des données de non-modèles, les meilleurs chemins calculés. Afin de transposer l'effet des séquences d'états de la base 'modèle' qui consiste à incrémenter les compteurs, les séquences d'états pour les' non-modèles' vont décrémenter les compteurs. Dans les données non modèles, plus une séquence d'états est fréquente, plus la probabilité associée est faible.

Nous illustrons le chemin de Viterbi dans la Fig. 7, et le calcul des nombres de transition d'un état à un autre dans la

TABLE IV, L'algorithme d'apprentissage de NB Viterbi est illustré dans l'algorithme 2. Les résultats de cet algorithme sont présentés dans la TABLE V

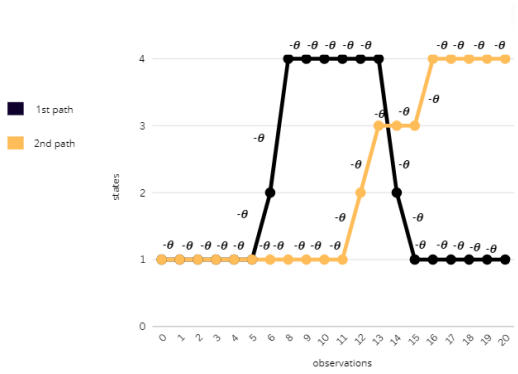

Fig. 9. Illustration des meilleurs chemins de Viterbi pour les non-modèles pour deux séquences

TABLE IV. 1ER ET 2ÈME CHEMIN DE VITERBI POUR LES NON-MODÈLES POUR DEUX SEQUENCES

Vers j / De i	1	2	3	4
1	-5θ-10 θ	-θ-θ	0	0
2	- θ	0	-θ	- θ
3	1	0	- 2θ	- θ
4	0	-θ	0	-5θ-4θ

ALGORITHME 2. Apprentissage par NB Viterbi

Données d'apprentissage :
O_{app} = {$O_{app\ 1}$, $O_{app\ 2}$, … $O_{app\ L}$}
$O_{app\ nb}$ = {$O_{app\ nb1}$, $O_{app\ nb2}$, … , $O_{app\ nb\ P}$}
Choisir un modèle initial λ= (A B, π)
Répéter
  Initialisation de toutes les variables : $n_i, n_{i\to j}$ et n_i^O à 0
  POUR chaque $O_{app\ l} \in O_{app}$ **Faire**
    Calculer le chemin de Viterbi de $O_{app\ l}$
    Ajuster les variables $n_i, n_{i\to j}$ et n_i^O contribution
    de $O_{train\ l}$
  Fin Pour
  FOR chaque $O_{app\ nb\ P} \in O_{app\ nb}$
    Calculer le chemin de Viterbi de $O_{app\ nb}$
    Ajuster les variables $n_i, n_{i\to j}$ et n_i^O la contribution
    de $O_{app\ nb\ P}$
  END
  [1]Ajouter un offset pour éliminer les éléments négatifs dans $n_i, n_{i\to j}$ et n_i^O
  Réestimer les nouveaux paramètres λ= (A B, π)
Jusqu'à la stabilité

TABLE V. RÉSULATS DE LA CLASSIFICATION POUR LES TROIS ALGORITHMES

Algorithmes	Résultats (%)
Baum-Welch	81.45
Viterbi	71.52
NB Viterbi	79.55

Nous notons que le résultat obtenu avec l'algorithme d'apprentissage NB Viterbi est meilleur que celui obtenu avec l'algorithme d'apprentissage de Viterbi. Les tests ayant été effectués sur plusieurs jeux de données, nous montrons que les informations contenues dans les séquences de non-modèles peuvent améliorer l'apprentissage du modèle.

IX. CONCLUSION

Dans ce travail, nous avons utilisé des Modèles de Markov Cachés (MMC) pour la classification et l'identification. Nous avons montré que dans les multi-classes, chaque modèle est estimé en utilisant les échantillons spécifiques à sa classe. Mais dans notre étude, nous avons utilisé une nouvelle approche qui consiste à estimer le contraire d'une forme (non-modèle). Pour ce faire, des données synthétiques pour différentes classes sont générées. En utilisant les résultats des différents tests, nous avons démontré la capacité des non-modèles, en utilisant des échantillons d'autres classes (non-classes), à identifier correctement la classe.

Sur la base des résultats obtenus pour les non-modèles, nous avons développé l'algorithme d'apprentissage NB Viterbi qui exploite à la fois l'information contenue dans les modèles et l'information contenue dans les non-modèles pour améliorer l'apprentissage des MMC. Nous avons montré que l'algorithme d'apprentissage NB Viterbi donne de meilleurs résultats que ceux obtenus par l'algorithme d'apprentissage de Viterbi.

Dans cet article, nous montrons qu'il est possible d'exploiter les informations non liées à la classe pour améliorer l'étape d'apprentissage.

Dans nos prochains travaux, nous étudierons différentes stratégies d'utilisation des informations non liées au modèle. Les données du modèle pourraient être utilisées en premier lieu dans une étape de formation initiale, puis les données non liées au

[1] Offset pour n_i = abs (min (n_i)) et Offset de $n_{i\to j}$=abs (min ($n_{i\to j}$))

modèle pourraient être prises en compte. Cela permettait de donner plus d'importance à la forme du modèle. Le poids du non-modèle pourrait également être ajusté.

X. LES RÉSUMÉS

***Résumé* —** Dans cet article, nous proposons une nouvelle stratégie pour l'apprentissage des modèles de Markov cachés dans le cas de problèmes multi-classes. Nous voulons contribuer à une autre approche de l'apprentissage des MMC. L'approche habituelle consiste à trouver le modèle qui correspond le mieux aux données de la classe considérée. Dans le cas d'un problème multi-classes, nous voulons démontrer qu'il existe des informations pertinentes dans les autres classes qui peuvent être utilisées pour améliorer les paramètres du modèle de la classe. Il s'agit d'apprendre le "contraire" des séquences (ou formes) que nous essayons d'identifier. Nous montrons que les non-modèles ont la capacité d'identifier des classes, nous avons donc développé un nouvel algorithme d'apprentissage des MMC (apprentissage par NB Viterbi) qui utilise à la fois l'information contenue dans les modèles et l'information contenue dans les non-modèles pour améliorer l'apprentissage des MMC. Nous comparons les résultats de l'algorithme "classique" de formation de Viterbi avec le nouvel algorithme NB Viterbi et démontrons que les résultats peuvent être améliorés.

***Mots-clefs* —Modèle de Markov Caché, algorithmes d'apprentissage, classification, simulation**

***Abstract* —** In this paper, we propose a new strategy for training Hidden Markov Models in case of multiclass problems. We want to contribute to a new approach to perform training of HMM. The usual approach consists in finding the model that better match the data of the considered class. In case of multi-classes problem, we want to demonstrate that there is pertinent information in the others classes that can be used to enhance parameters of the model of the class. This consists in learning the" opposite of the sequences (or shapes) we're trying to identify. We show that non-model has the ability to identify classes, so we've developed a new HMM learning algorithm (NB Viterbi training) which uses both the information contained in models and the information contained in non-models to improve HMM learning. We compare results of "classical" Viterbi training algorithm with the new NB Viterbi algorithm and demonstrate that results can be enhanced.

***Keywords* — Hidden Markov Model, Training algorithm, Classification**.

REFERENCES

Aupetit,S., Monmarché,N.,& Slimane,M. (2007). Utilisation des Modèles de Markov cachés pour la reconnaissance robuste d'images : apprentissage par colonie de fourmis, algorithme génétique et essaim particulaire. *Optimisation en traitement du signal et de l'image (Traité IC2 série traitement du signal et de l'image)*, 245–269.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. Annals Of Mathematical Statistics, 41(1), 164-171. https://doi.org/10.1214/aoms/1177697196.

Cao, L., Yu, P. S., & Kumar, V. (2015). Nonoccurring Behavior Analytics : a new area. *IEEE Intelligent Systems*, *30*(6), 4-11. https://doi.org/10.1109/mis.2015.105.

Dempster, A. P., Laird, N. M., & Rubin, D.B.(1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) (1977),1-38.

Gong, C., Wang, X., & He, Y. (2021). Remaining useful life and state of health prediction for lithium batteries based on empirical mode decomposition and a long and short memory neural network. *Energy*, *232*, 121022. https://doi.org/10.1016/j.energy.2021.121022.

Khodja, N.K., Duculty, F., Begot, S., & Avila. M. (2022). Contribution à l'apprentissage de Modèles de Markov Cachés pour l'estimation de l'état de santé de systèmes industriels. *Le congrès lambda mu 23.*

Kim, M. S., Kim, D., & Lee, S. Y. (2003). Face recognition using the embedded HMM with second-order block-specific observations. *Pattern Recognition*, *36*(11), 2723-2735. https://doi.org/10.1016/s0031-3203(03)00137-7.

Lee, J. M., Kim, S., Hwang, Y. M., & Song, C. H. (2004). Diagnosis of mechanical fault signals using continuous hidden Markov model. *Journal Of Sound And Vibration*, *276*(3-5), 1065-1080. https://doi.org/10.1016/j.jsv.2003.08.021.

Li, X., Ma, Y., & Zhu, J. (2021). An online dual filters RUL prediction method of lithium-ion battery based on unscented particle filter and least squares support vector machine. *Measurement*, *184*, 109935. https://doi.org/10.1016/j.measurement.2021.109935

Liu, J., You, M., Wang, Z., Li, G., Xu, X., & Qiu, Z. (2015). Cough event classification by pretrained deep neural network. *BMC Medical Informatics And Decision Making*, *15*(S4). https://doi.org/10.1186/1472-6947-15-s4-s2.

Martín-Iglesias, D., Bernal-Chaves, J., Peláez-Moreno, C., Gallardo-Antolín, A., & Díaz-De-María, F. (2006). A Speech Recognizer Based on Multiclass SVMs with HMM-Guided Segmentation. Dans *Lecture Notes in Computer Science* (p. 257-266). https://doi.org/10.1007/11613107_22.

Qiu, G., Gu, Y., & Chen, J. (2020). Selective health indicator for bearings ensemble remaining useful life prediction with genetic algorithm and Weibull proportional hazards model. *Measurement*, *150*, 107097. https://doi.org/10.1016/j.measurement.2019.107097.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings Of The IEEE*, *77*(2), 257-286. https://doi.org/10.1109/5.18626.

Smyth, P. (1997). Clustering sequences with hidden Markov models. *Adv Neural Inf Process Syst* 9, 648–654.

Wang, L., Zhou, Y., Li, R., & Ding, L. (2022). A fusion of a deep neural network and a hidden Markov model to recognize the multiclass abnormal behavior of elderly people. *Knowledge-Based Systems*, *252*, 109351. https://doi.org/10.1016/j.knosys.2022.109351.