Apprentissage non-supervisé d'images par hybridation génétique d'une chaîne de Markov cachée

Unsupervised learning of pictures by genetic hibrydization of hidden Markov chain

par Mohamed SLIMANE, Thierry BROUARD, Gilles VENTURINI et Jean-Pierre ASSELIN DE BEAUVILLE

Laboratoire d'Informatique- EA2101, Ecole d'Ingénieurs en Informatique pour l'Industrie,
Université de Tours, 64, Avenue Jean Portalis, 37200 Tours FRANCE
Tel : (+33) 2 47 361414, Fax : (+33) 2 47 3614 22,
Email : slimane, brouard, venturini, asselin e3i.univ-tours.fr

résumé et mots clés

Cet article présente un algorithme d'apprentissage non supervisé par chaînes de Markov cachées (CMC) et algorithmes génétiques (AG). Deux des problèmes rencontrés lors de l'utilisation des CMC sont de déterminer les probabilités de la CMC et le nombre d'états de cette chaîne. Bien souvent, ce nombre d'états est déterminé soit par expériences successives, soit à l'aide de connaissances *a priori* du domaine. L'algorithme présenté ici emploie un algorithme génétique afin de déterminer le nombre d'états cachés de la CMC ainsi que les différentes probabilités qui la constituent. Cet algorithme est couplé à l'algorithme de Baum-Welch qui permet une réestimation efficace des probabilités de la CMC. Différents algorithmes, hybrides ou non, sont comparés entre eux sur une application d'apprentissage et de reconnaissance d'images représentant des visages. Les résultats montrent la supériorité de l'approche génétique pour ce type de problème.

Chaînes de Markov cachées, algorithmes génétiques, hybridation, apprentissage non supervisé, reconnaissance d'images.

abstract and key words

This paper presents a learning algorithm using hidden Markov models (HMMs) and genetic algorithms (GAs). Two standard problems to be solved with HMMs are how to determine the probabilities and the number of hidden states of the learned models. Generally, this number of states is determined either by the trial-error method that needs experimentation, or by the background knowledge available. The presented algorithm uses a GA in order to determine at the same time both the number of states and the probabilities of learned HMMs. This hybrid algorithm uses the Baum-Welch algorithm to optimise precisely the probabilities of HMMs. Several algorithms, either hybrid or not, are compared in a face recognition task. The obtained results highlight the strength of our approach for the concerned problem.

Hidden Markov chains, genetic algorithms, hybridization, unsupervised learning, picture recognition.

introduction

Les chaînes de Markov cachées (CMC) [1] [2] sont des outils très utilisés dans de nombreux domaines, tels que la reconnaissance de la parole [3] [4] [5], la reconnaissance de l'écriture [6] [7] [8], le traitement du signal [9] [10] [11], l'analyse d'images [12] [13] [14] [15] ou encore la biologie [16]. Pour la suite de l'exposé nous utiliserons des CMC discrètes, *i.e.* le nombre d'états cachés ainsi que l'ensemble des symboles pouvant être générés par la CMC est fini ou dénombrable.

L'emploi de ces outils nous confronte souvent à un problème d'apprentissage de données (appelées observations) par la CMC. On entend ici par apprentissage d'une CMC la modification des paramètres (structurels ou non) de cette CMC dans le but d'accroître ses performances dans l'accomplissement d'une tâche précise. Nous allons chercher à faire apprendre à une CMC des observations, et l'objectif à atteindre est que cette CMC ait une probabilité la plus forte possible de générer ces observations. C'est la phase d'apprentissage qui va nous permettre d'améliorer cette CMC. La phase d'apprentissage va donc déterminer les différents éléments de la CMC (nombre d'états, probabilités de transition inter-états, probabilités de génération des symboles) qui maximisent la probabilité que la CMC génère les séquences de valeurs apprises. Généralement on considère au début de l'apprentissage que la connectivité de la chaîne est totale, *i.e.* aucune probabilité de transition n'est nulle. C'est alors à l'algorithme d'apprentissage employé de faire évoluer cette connectivité. Mais il peut arriver que l'on impose cette connectivité afin de respecter des contraintes du domaine étudié. Dans notre cas, la connectivité est supposée totale afin d'offrir l'espace de recherche le plus vaste possible à l'algorithme d'optimisation. De nombreux algorithmes d'apprentissage de CMC existent. Ils sont basés, généralement, sur des descentes de gradient tel que l'algorithme de Baum-Welch (BW) [17]. Deux problèmes majeurs se posent lors de leur utilisation. Il s'agit d'une part d'éviter les maxima locaux de la fonction d'évaluation des CMC et d'autre part de déterminer le bon nombre d'états cachés des chaînes susceptibles de mieux apprendre les données.

Le premier problème a fait l'objet de nombreux travaux [18] [19] [20] visant à éviter les maxima locaux de la fonction d'évaluation. L'algorithme SEM (Stochastic EM) [18] par exemple, est une version stochastique de EM. Cet algorithme vise à pallier les inconvénients de EM (dépendance du point de départ et convergence vers des points stationnaires qui ne sont pas des maxima locaux) en introduisant une perturbation aléatoire dans EM.

Dans nos travaux précédents, nous avons résolu efficacement ce premier problème grâce à une exploration parallèle de l'espace des CMC utilisant BW comme opérateur local de gradient dans un algorithme génétique (AG) [14] [21]. Ces algorithmes permettent un meilleur apprentissage en diminuant la probabilité que l'algorithme converge vers les maxima locaux situés dans l'espace de recherche. Les résultats obtenus ont montré que cet apprentissage hybride donnait de meilleurs résultats que l'apprentissage non hybride.

Le second problème correspond à un thème générique en reconnaissance des formes, c'est-à-dire à la détermination d'une bonne architecture (nombre d'états et transitions autorisées entre ces états), problème que l'on rencontre notamment avec les réseaux de neurones artificiels mais qui est encore peu étudié en particulier dans les applications des CMC en traitement d'images. On trouve quelques travaux dans ce domaine [22] [23], mais en général la détermination du nombre d'états cachés de la CMC est très souvent effectuée *a priori* en considérant des données du domaine d'application. Pourtant l'influence de ce nombre d'états est important, aussi bien sur la « qualité » du résultat final que sur le temps de calcul nécessaire pour l'obtenir.

L'algorithme que nous proposons ici, appelé GHOSP [24] (Genetic Hybrid Optimization and Search of Parameters), est issu de ceux présentés dans [14] [21]. Il optimise simultanément les paramètres structurels (architecture) et les paramètres non structurels (probabilités) d'une CMC alors qu'auparavant, l'optimisation ne portait que sur les paramètres non structurels de la CMC, le nombre d'états devant être fixé *a priori*. Pour cela, GHOSP maintient dans une même population des CMC ayant des nombres d'états cachés différents. Ce nouvel algorithme apporte donc une réponse au problème de l'apprentissage non supervisé (nombre d'états inconnu) d'une CMC.

1. les chaînes de Markov cachées (CMC)

1.1. approche des CMC par un exemple

Considérons un lancer de pièce de monnaie. Nous disposons de trois pièces (biaisées). Le lanceur ne nous informe pas de la pièce qu'il a choisie, et pour une même observation donnée, les symboles (pile ou face) peuvent être générés tour à tour par la pièce une deux ou trois. Une chaîne de Markov cachée modélise l'expérience de la manière suivante (figure 1). On considère qu'un état modélise une pièce, et que cet état peut produire deux symboles : pile et face. On admet alors que l'on peut passer d'un état à un autre (c'est-à-dire changer de pièce) suivant certaines lois de probabilités, et que chaque pièce possède sa propre distribution de probabilités de générer les symboles pile et face. La séquence

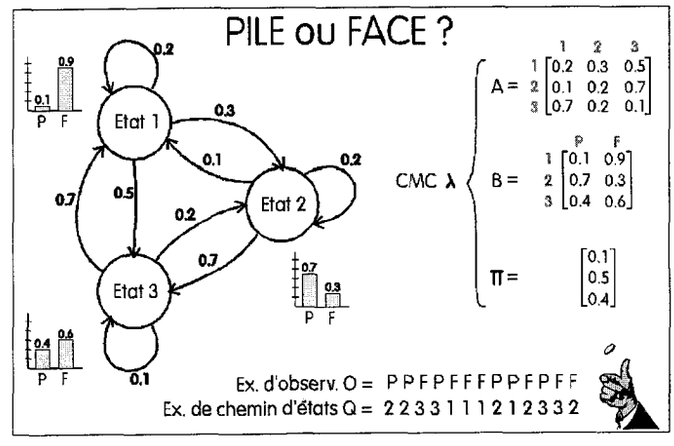


Figure 1. – Exemple de chaîne de Markov cachée : modélisation d'un lancer avec 3 pièces de monnaie.

de (pile-face) est toujours observable, mais la séquence d'états qui a engendré cette séquence n'est pas observable. On dit alors qu'elle est cachée. Les probabilités de transitions entre les états (non observables) sont stockées dans une matrice notée A, alors que les probabilités d'apparition des symboles (observables) sont mis dans une matrice notée B.

1.2. définitions formelles des CMC

Une chaîne de Markov cachée est une chaîne de Markov stationnaire générant un processus stochastique à deux composantes : une cachée, l'autre observable. La composante cachée est la réalisation $q = (q_1, q_2, \ldots, q_T)$ d'une suite de variables aléatoires $Q = (Q_1, Q_2, \ldots Q_T)$ où les q_t prennent leurs valeurs dans l'ensemble $S = s_1, s_2, \ldots, s_N$ de N classes (les états). La suite observable $o = (o_1, o_2, \ldots, o_T)$ est la réalisation du processus aléatoire $O = (O_1, O_2, \ldots O_T)$ où les o_t prennent leurs valeurs dans l'ensemble V de M classes (les symboles). La propriété de Markov est supposée vraie pour la composante cachée :

$\forall t \in [2 \ldots T]$,

$$P(Q_t = q_t | Q_1 = q_1 \ldots Q_{t-1} = q_{t-1}) = P(Q_t = q_t | Q_{t-1} = q_{t-1}),$$

On a pour cette composante les définitions suivantes :

- loi de probabilité initiale : $P(Q_1 = s_i) = \pi_i, 1 \leq i \leq N$
- matrice des probabilités de transition :

$$P(Q_t = s_j | Q_{t-1} = s_i) = a_{ij},\ 1 \leq i, j \leq N, t \in [2 \ldots T]$$

Les éléments d'une CMC sont donc :

- N : le nombre d'états du modèle.
- M : le nombre de symboles différents générés par la CMC. Ces symboles forment l'ensemble

$$V = \{v_1, v_2, \ldots, v_m\}$$

- $A = [\ldots a_{ij} \ldots]$: matrice de distribution des probabilités de transition. Cette matrice est de dimension $N \times N$. Le terme générique a_{ij} désigne la probabilité de transiter d'un état s_i vers un état s_j. La matrice est stochastique, *i.e.* la relation $\sum_{j=1}^{N} a_{ij} = 1$, $\forall i \in [1 \ldots N]$ est vérifiée. La définition même de cette matrice suppose que la connectivité est *a priori* totale entre tous les états. Certaines liaisons pouvant toutefois être affectées de probabilités de transition nulles.
- $B = [\ldots b_j(k) \ldots]$: matrice de distribution des probabilités de génération des symboles. Cette matrice est de dimension $N \times M$. Le terme générique $b_j(k) = P(O_t = v_k | Q_t = s_j)$, désigne la probabilité pour que la CMC se trouvant dans l'état s_j génère le symbole de rang k, v_k. La matrice B est stochastique, *i.e.*

$$\sum_{k=1}^{M} b_j(k) = 1,\ \forall j \in [1 \ldots N].$$

- $\Pi = [\ldots \pi_i \ldots]$ matrice de distribution des probabilités initiales. Cette matrice est de dimension $1 \times N$. Le terme générique π_i désigne la probabilité que la CMC se situe dans l'état s_i à l'instant initial. La matrice Π est stochastique, *i.e.* $\sum_{i=1}^{N} \pi_i = 1$

En résumé, la CMC est définie par le triplet de matrices noté $\lambda = (A, B, \Pi)$ qui suppose implicitement la connaissance de N (nombre d'états) et de M (nombre de symboles).

2. problèmes fondamentaux

On distingue trois principaux types de problèmes rencontrés lors de l'utilisation des chaînes de Markov cachées [1]. Nous utilisons les deux suivants :

2.1. premier problème : l'évaluation

Etant donnée une CMC λ et une observation o, on cherche à calculer la vraisemblance de l'observation o avec la CMC λ, c'est-à-dire avec quelle probabilité la CMC λ engendre l'observation o. Cette valeur est notée $P(O = o|\lambda)$. Un calcul direct de cette valeur est bien sûr possible, mais très coûteux en nombre d'opérations car il énumérerait tous les chemins q possibles tel que $P(O = o|\lambda) = \sum_q P(Q = q, O = o|\lambda)$. On préfère utiliser un algorithme de complexité moindre : l'algorithme Forward [1] qui réalise le calcul en N^2T opérations contre $2TN^T$ opérations pour le calcul direct (avec N le nombre d'états et T la longueur de l'observation).

Pour cela, on définit la variable Forward comme la probabilité d'observer la suite $o = (o_1, o_2, \ldots, o_t)$ à l'instant t en étant dans l'état s_i et connaissant la chaîne de Markov cachée λ.

$$\alpha_t(i) = P(o_1 \ldots o_t; q_t = s_i|\lambda)$$

A l'instant initial ($t = 1$), la variable Forward $\alpha_1(i)$ est la probabilité jointe d'observer le premier symbole de l'observation (o_1) et de se trouver dans l'état s_i. On a donc $\alpha_1(i) = \pi_i \times b_i(k_1)$ avec $o_1 = v_{k_1}$.

Puis, une phase d'induction calcule $\alpha_t(j)$ pour $t = 2 \ldots T$. A l'instant t ce calcul s'effectue comme le produit de la probabilité d'observer la suite partielle $o = (o_1, o_2, \ldots, o_{t-1})$ menant à l'état

s_j et de la probabilité que cet état s_j génère l'observation de rang t (o_t).

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) \times a_{ij}\right] \times b_j(k) \quad \begin{cases} 2 \leq t \leq T \\ 1 \leq j \leq N \\ o_t = v_k \end{cases}$$

La variable $\alpha_T(j)$ est par définition la probabilité d'avoir toute l'observation en terminant dans l'état s_j. La sommation de cette variable sur l'ensemble des états finaux possibles donne $P(O = o|\lambda)$

On peut considérer le problème d'une façon analogue. On obtient alors l'algorithme Backward, où le terme générique est $\beta_t(j) = P(o_{t+1} \ldots o_T | Q_t = s_j)$, la probabilité de générer l'observation $(o_{t+1} \ldots o_T)$ et d'être dans l'état s_j à l'instant t.

2.2. second problème : estimation des paramètres non structurels de la CMC

Etant donnée une CMC λ et une observation o, on cherche la CMC λ^* qui a la plus forte probabilité d'engendrer l'observation o. Ce problème est résolu par l'algorithme de Baum-Welch [1].

C'est une procédure de réestimation itérative des matrices A, B et Π d'une CMC. Etant donné un modèle $\lambda = (A, B, \Pi)$ quelconque et une séquence d'observations $o = (o_1, o_2, \ldots, o_T)$, l'algorithme de Baum-Welch réestime les valeurs des matrices A, B et Π de façon à maximiser la vraisemblance de l'observation O : $P(O = o\,\lambda)$. On obtient après exécution de l'algorithme la CMC λ^* telle que $\lambda^* = \text{argmax}_\lambda P(O = o|\lambda)$.

Pour réestimer les valeurs des matrices A, B et Π on définit deux matrices : Ξ et Γ. On note : $\Xi = [\ldots \xi_t(i,j) \ldots)]$ où $\xi_t(i,j)$ est la probabilité d'être dans l'état s_i à l'instant t et de passer dans l'état s_j à l'instant $t+1$, connaissant le modèle λ et la séquence d'observations o, ce qui peut s'écrire $\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O = o, \lambda)$. Cette matrice est de dimension $T \times N \times N$.

En utilisant les variables Forward (α) et Backward (β) précédemment vues, on établit la relation suivante :

$$\xi_t(i,j) = \frac{\alpha_t(i) \times a_{ij} \times b_j(o_{t+1}) \times \beta_{t+1}(j)}{P(O = o|\lambda)}$$

Ce qui peut s'interpréter par : c'est la probabilité d'observer $(o_1 \ldots o_t)$, d'être dans l'état s_i à l'instant t, de transiter dans l'état s_j à l'instant $t+1$, d'observer le symbole o_{t+1} étant dans l'état s_j et d'observer la séquence $(o_{t+2} \ldots o_T)$. Le dénominateur $P(O = o|\lambda)$ sert de « normalisateur ».

Rappel :

$$P(O|\lambda) = \sum_{i=1}^{N} \sum_{j=i}^{N} \alpha_t(i) \times a_{ij} \times b_j(o_{t+1}) \times \beta_{t+1}(j)$$

La seconde matrice (Γ) est définie par : $\Gamma = [\ldots \gamma_t(i) \ldots]$ où $\gamma_t(i)$ est la probabilité d'être dans l'état s_i à l'instant t sachant le modèle λ et la séquence o. Cette matrice est de dimension $T \times N$.

Si l'on reprend la définition de $\xi_t(i,j)$, on remarque que si l'on somme $\xi_t(i,j)$ sur j (considérer tous les chemins d'états permettant d'observer $o_1 \ldots o_{t+1}$, avec $q_{t+1} = s_j$) tous les états d'arrivée, on retrouve $\gamma_t(i)$. D'où la relation :

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

Intéressons-nous maintenant à la réestimation pour l'itération $k+1$ des paramètres non structurels du modèle obtenu à l'itération k :

$$\pi_i^{(k+1)} = \gamma_1^{(k)}(i) \quad 1 \leq i \leq N$$

$$a_{ij}^{(k+1)} = \frac{\sum_{t=1}^{T-1} \xi_t^{(k)}(i,j)}{\sum_{t=1}^{T-1} \gamma_t^{(k)}(i)} \quad \begin{cases} 1 \leq i \leq N \\ 1 \leq j \leq N \end{cases}$$

$$b_j^{(k+1)}(r) = \frac{\sum_{t=1 \cap o_t = v_r}^{T} \gamma_t^{(t)}(j)}{\sum_{t=1}^{T} \gamma_t^{(k)}(j)} \quad \begin{cases} 1 \leq i \leq N \\ 1 \leq j \leq M \end{cases}$$

Remarques :

- Le choix du modèle initial influe sur le résultat final,
- L'algorithme converge vers une CMC correspondant à un point critique (point d'inflexion ou maximum local) de la fonction de vraisemblance,
- La condition d'arrêt est en général le nombre d'itérations, qui est fixé empiriquement,
- La taille de la séquence d'observations influe sur le résultat final. Un manque de données entraînera un modèle trop général qui ne reconnaîtra rien de précis, et au contraire trop de données donneront un apprentissage très difflcile pour un modèle « trop ciblé »,
- D'après les formules de calcul de α et β, lorsque T croît, les α et, β tendent rapidement vers zéro. La probabilité devient alors trop petite pour être représentée en machine, problème connu sous le nom de dépassement de capacité (*underflow*).

Cet algorithme est de type gradient, appliqué à la fonction de vraisemblance. Il n'est donc optimal que localement. La solution qu'il trouve est fortement dépendante de la solution de laquelle il part. En fait partant d'une CMC λ il va trouver la solution optimale accessible depuis λ. L'aspect descente de gradient peut être facilement vu en posant la réestimation de le CMC en terme

d'optimisation sous contrainte de la vraisemblance. En notant $\mathcal{F}$ la vraisemblance, les formules de réestimation de la CMC sont :

$$\pi_i^{(k+1)} = \frac{\pi_i^{(k)} \times \frac{\partial \mathcal{F}}{\partial \pi_i^{(k)}}}{\sum_{j=1}^{N} \pi_j^{(k)} \times \frac{\partial \mathcal{F}}{\partial \pi_j^{(k)}}}$$

$$a_{ij}^{(k+1)} = \frac{a_{ij}^{(k)} \times \frac{\partial \mathcal{F}}{\partial a_{ij}^{(k)}}}{\sum_{r=1}^{N} a_{ir}^{(k)} \times \frac{\partial \mathcal{F}}{\partial a_{ir}^{(k)}}}$$

$$b_j(r)^{(k+1)} = \frac{b_j(r)^{(k)} \times \frac{\partial \mathcal{F}}{\partial b_j(r)^{(k)}}}{\sum_{x=1}^{M} b_j(x)^{(k)} \times \frac{\partial \mathcal{F}}{\partial b_j(x)^{(k)}}}$$

Nous étendons ce problème d'optimisation en distinguant deux classes : l'apprentissage de CMC dont l'architecture (nombre d'états) est connue et l'apprentissage de CMC dont l'architecture est inconnue *a priori*. Nous qualifierons le premier cas d'apprentissage supervisé, et le second cas d'apprentissage non supervisé.

Dans le cas de l'apprentissage supervisé, on suppose l'architecture (nombre d'états et transitions autorisées) de la CMC connue *a priori*. L'optimisation est réalisée par l'algorithme de Baum-Welch. Nous avons introduit de nouveaux algorithmes (AG + BW [21] et AG ∪ BW [14]) afin de pallier aux inconvénients signalés de l'utilisation de l'algorithme de Baum-Welch.

Dans le cas de l'apprentissage non-supervisé, on considère l'architecture de la CMC inconnue *a priori*, c'est l'algorithme d'optimisation qui va la déterminer. Nous apportons une solution à ce problème avec l'algorithme GHOSP (*cf.* section 4).

3. algorithme génétique pour la manipulation d'une CMC

Le premier problème qui se pose est de trouver un bon point de départ pour l'algorithme de Baum-Welch. Plusieurs solutions sont envisageables, hormis la méthode consistant à définir « manuellement » ce point de départ en fonction des connaissances *a priori* du domaine [1].

On pourrait appliquer plusieurs fois l'algorithme de Baum-Welch sur des CMC initialisées aléatoirement. Au moins une question se pose : combien faudra-t-il d'essais avant de trouver la meilleure solution au problème? Une seconde approche consiste à déterminer cette solution initiale avec un algorithme génétique. Ce principe mis en pratique dans l'algorithme AG + BW [21] offre l'avantage de trouver de meilleures solutions que l'approche précédente utilisant une génération aléatoire.

En poussant plus loin la coopération entre l'algorithme génétique et l'algorithme de Baum-Welch, on intègre l'algorithme de Baum-Welch comme un opérateur local d'optimisation dans l'algorithme génétique. L'algorithme résultant, AG ∪ BW [14] a des performances durant l'apprentissage bien supérieures à celles d'AG + BW.

Mais le principal problème qui reste est celui du choix de l'architecture de la CMC. Il n'existe pas de méthode universelle pour définir le nombre d'états optimal d'une CMC. Or cette valeur est très influente sur la qualité de la solution finale. L'idée est donc d'essayer de faire prendre en compte cette recherche par l'algorithme génétique. Cette recherche du nombre d'états optimal se réalise simplement en fournissant à l'algorithme génétique une population hétérogène, c'est-à-dire contenant des CMC avec des architectures (nombre d'états) diverses. La mise en application de ce principe a donné naissance à l'algorithme GHOSP (*cf.* section 4).

3.1. rappels sur les algorithmes génétiques

Les algorithmes génétiques (AG) font partie de méthodes de recherche énumératives et stochastiques. Ils ont été introduits par J.Holland en 1975 dans son livre « Adaptation in natural and artificial systems » [26]. Ils occupent aujourd'hui une place importante parmi les algorithmes d'évolution artificielle (AEA), et sont principalement utilisés comme des techniques d'optimisation. Les AG utilisent une population d'individus. Chaque individu est modélisé par une chaîne binaire, appelée chromosome, composée de 0 et de 1 de longueur fixe. L'AG cherche à déterminer la chaîne binaire qui maximise une certaine fonction d'évaluation f. D'une manière générale, l'algorithme génétique engendre une population $P\mathcal{P}^{(t+1)}$ à partir d'une population $\mathcal{P}^{(t)}$. Cela signifie que l'AG recherche de nouvelles solutions à partir de celles déjà explorées en utilisant des opérateurs génétiques. D'autre part, disposant d'un ensemble d'individus, l'AG explore l'espace des solutions d'une manière parallèle. Les nouvelles solutions (individus) de la population $\mathcal{P}^{(t+1)}$ sont ensuite « notées » par la fonction d'évaluation et un nouveau cycle peut commencer. L'algorithme général est décrit par l'algorithme 1 (voir page suivante).

Trois opérations sont fondamentales dans un AG :

1. la sélection : elle détermine une première population intermédiaire $\mathcal{P}_s$ (à partir de $\mathcal{P}^{(t)}$ constituée de « bons » individus au sens de la fonction d'évaluation f. Généralement, la sélection favorise les individus les meilleurs en leur accordant une probabilité de sélection proportionnelle à leur qualité. Une solution appartenant à une population de n solutions aura une probabilité $P(x_i) = f(x_i) \sum f(x_j)$ d'être sélectionnée;

Algorithme 1 Algorithme AG général

/ Initialisation */*
$t \leftarrow 1$
générer aléatoirement et évaluer une population initiale $\mathcal{P}^{/t/}$ de taille S

/ Phase d'évolution */*
A partir de $\mathcal{P}^{/t/}$, générer une nouvelle population $\mathcal{P}^{/t+1/}$ par sélection, croisement, mutation et recombinaison
Évaluer les individus de $\mathcal{P}^{/t+1/}$
$t \leftarrow t + 1$

/ Critère d'arrêt */*
Recommencer une phase d'évolution ou arrêt

2. la recombinaison : elle recombine entre elles les solutions de la population sélectionnée afin d'engendrer une seconde population de solutions $\mathcal{P}_r$, de même taille que $\mathcal{P}_s$;
3. la mutation : elle modifie aléatoirement les solutions de $\mathcal{P}_r$ pour donner une nouvelle population de solutions, venant remplacer les individus jugés « les moins bons » dans la population de départ $\mathcal{P}^{(t)}$ afin de donner la population $\mathcal{P}^{(t+1)}$.

3.2. définition d'un individu CMC

L'algorithme génétique va manipuler des individus, qui vont être des solutions au problème que l'on désire résoudre. Naturellement, dans le problème de l'apprentissage des CMC, les individus sont des CMC. Il faut maintenant coder les CMC en chromosomes, sur lesquels s'appliqueront les opérateurs génétiques. Le codage le plus naturel qui soit est de fabriquer le chromosome en réorganisant tous les coefficients de la CMC. Le plus simple est de juxtaposer toutes les lignes de toutes les matrices. On obtient ainsi un codage en nombres réels. Il n'est pas interdit de penser à d'autres codages, ainsi qu'à d'autres façons de réorganiser les coefficients dans le chromosome tout en respectant les contraintes liées aux CMC.

3.3. critère de qualité d'un individu

La qualité d'un individu (on dit aussi *fitness*) décrit l'adéquation de celui-ci avec son environnement. Plus précisément, l'individu aura une note d'autant plus élevée qu'il sera une bonne solution au problème. Dans le problème de l'optimisation d'une CMC, on cherche à quantifier l'aptitude d'une CMC à l'apprentissage d'une observation. On peut considérer que l'apprentissage est d'autant meilleur que la CMC à une forte probabilité de générer cette observation. On retrouve alors le problème de l'évaluation (*cf.* paragraphe 2.1). La qualité de l'individu sera donc calculée par l'algorithme Forward.

3.4. politique de reproduction des individus

Les critères de sélection sont dits « élitistes ». Cela signifie simplement que seule une partie des individus (les meilleurs) survit d'une génération à l'autre. En pratique, à une époque t, on sélectionne une partie de la population. Cette partie est constituée des meilleurs individus, au sens du critère recherché (la probabilité de générer l'observation dans notre cas). Le reste de la population sera remplacé par d'autres individus à l'époque $t + 1$. Ces nouveaux individus sont obtenus par combinaison des individus sélectionnés. Cette opération de combinaison est constituée de plusieurs sous-opérations, décrites aux paragraphes 3.6, 3.7 et 3.8. La politique élitiste de l'algorithme associée à l'emploi d'un opérateur d'optimisation locale (*cf.* paragraphe 3.8.2) lui confère une vitesse de convergence vers la solution élevée.

3.5. critère d'arrêt de l'algorithme génétique

L'algorithme génétique peut s'arrêter suivant deux conditions. La première est le nombre d'itérations. Cette valeur, fixée avant le lancement de l'algorithme, force celui-ci à s'arrêter au bout d'un certain temps de calcul. La seconde condition est que l'algorithme ne puisse pas trouver une solution meilleure. Ce cas est réalisé ici lorsque la probabilité que l'observation soit engendrée par le modèle est égale à l'unité, cas possible, bien que peu probable, d'apprentissage par cœur. Le critère employé par notre algorithme est composé d'un ET logique entre ces deux conditions. On force ainsi l'algorithme à s'arrêter dès le premier modèle optimal rencontré.

3.6. opération de croisement

Le croisement permet la combinaison de deux solutions afin d'en obtenir d'autres (une ou deux suivant l'opérateur choisi). Cet opérateur peut être vu comme une phase d'exploitation des solutions existantes. Dans le cas des CMC, différentes méthodes de croisement sont possibles. Nous ne présentons que celle qui actuellement offre les meilleurs résultats, d'autres opérateurs sont présentés dans [21].

L'opérateur de croisement à un point (1X) combine deux CMC pour en donner deux autres. On considère deux CMC λ_1 et λ_2 à croiser. Une position est déterminée aléatoirement sur le chromosome, appelée point de coupure. Ce point sépare le chromosome en deux parties (gauche et droite) donnant la notation $[\lambda_{1g}|\lambda_{1d}]$ et $[\lambda_{2g}|\lambda_{2d}]$. On obtient les deux nouvelles CMC en échangeant les deux parties gauches (ou les deux parties droites) des deux CMC d'entrée. Cela donne donc les CMC notées $[\lambda_{1g}|\lambda_{2d}]$ et $[\lambda_{2g}|\lambda_{1d}]$. Les CMC imposent une organisation particulière des

coefficients. Les matrices doivent être stochastiques. Si l'on veut prendre cette contrainte en compte dans l'opérateur de croisement, cela implique que le point de coupure doit se situer à des positions prédéfinies. Pour ne pas modifier la structure des lignes des matrices de la CMC, il suffit de n'autoriser la coupure qu'entre deux lignes. Si l'on ne désire pas appliquer la contrainte de stochasticité sur l'opérateur de croisement, il suffit de faire appel à un opérateur de normalisation (*cf.* paragraphe 3.8.1). Cet opérateur de croisement s'applique sur des CMC ayant le même nombre d'états.

3.7. opération de mutation

La mutation n'est pas un opérateur exclusif aux algorithmes génétiques. On le rencontre également dans le recuit simulé par exemple. Le but de cette action est d'introduire du hasard. Il faut entendre par là explorer l'espace des solutions dans d'autres directions. La mutation consiste à modifier la solution actuelle avec une faible probabilité (appelée taux de mutation). Concrètement, pour chacun des coefficients des matrices de la CMC on tire un nombre suivant la loi uniforme sur l'intervalle [0,1]. Si ce nombre est inférieur au taux de mutation, alors on modifie aléatoirement le coefficient en cours de test. Cette modification se traduit par un accroissement ou une réduction de la valeur du coefficient. La valeur du taux de mutation est généralement comprise entre un pour mille et quelques pour cent. La politique élitiste que nous employons nous oblige à porter ce taux jusqu'à dix ou quinze pour cent pour réellement explorer l'espace de recherche et sortir des optima locaux.

3.8. opérateurs spécifiques au problème

La manipulation des CMC par l'algorithme génétique nécessite l'introduction de certaines opérations. Parmi celles-ci on notera la vérification de la cohérence de la solution et une opération d'optimisation.

3.8.1. opérateur de normalisation

Après les opérations de croisement et de mutation, il faut s'assurer que l'individu répond encore aux contraintes des CMC. On doit vérifier que les matrices de la CMC sont stochastiques. Cet opérateur est appliqué après l'opération de mutation car les opérations postérieures travaillent impérativement sur des CMC.

3.8.2. opérateur d'optimisation locale : Baum-Welch

L'espace de recherche des CMC est très grand. Afin de trouver de meilleures solutions et de conserver des temps de calculs raisonnables on a recours à un opérateur d'optimisation locale. Cet opérateur est constitué par l'algorithme de Baum-Welch. Cet algorithme est optimal localement. C'est-à-dire qu'il va trouver la solution optimale accessible depuis un point de départ donné. Ce point de départ est la CMC support de l'optimisation.

Cet opérateur pourrait être appliqué avec une certaine probabilité sur les individus. Cependant la politique élitiste de l'algorithme combinée à la difficulté d'optimisation engendrée par l'espace de recherche pose une condition sur le taux d'application de l'algorithme. Les solutions obtenues après l'algorithme de Baum-Welch ont une qualité bien supérieure à celles obtenues avant. Il s'en suit que les solutions optimisées seront systématiquement toutes sélectionnées. Pour que des solutions non optimisées apparaissent dans la population, il faut donc employer un taux d'application inférieur à la proportion d'individus sélectionnés pour la régénération de la population.

4. algorithme GHOSP

L'algorithme GHOSP (Genetic Hybrid Optimization and Search of Parameters) apporte une solution au problème de l'apprentissage non supervisé de CMC. Il intègre la recherche des paramètres non structurels d'une CMC avec tous les opérateurs vus à la section précédente et effectue en plus la recherche du nombre d'états optimal.

4.1. principe de l'algorithme GHOSP

L'algorithme GHOSP recherche simultanément des valeurs pour l'architecture et pour les probabilités des matrices de la CMC. La population regroupe des CMC d'architectures différentes. C'est-à-dire que l'on peut rencontrer des CMC de $2, 3, \ldots, N$ états par exemple, (les bornes de recherche étant définies par l'utilisateur) dans la population initiale. Chaque architecture est représentée par un nombre minimum (à fixer au départ) de représentants afin de donner à chaque architecture une meilleure chance de survie. Les probabilités des CMC de cette population sont initialisées aléatoirement.

A chaque itération, on considère la population des CMC. Celle-ci est composée de CMC d'architectures diverses divisées en deux classes : les « parents » et les « enfants ». Chaque CMC qui n'est pas de type « parent » est optimisée par l'algorithme de Baum-Welch (la première fois elles sont toutes de type « enfant » pour être forcément optimisées), puis évaluée. L'évaluation va déterminer une note (probabilité de générer l'observation à apprendre dans notre cas) pour chaque CMC. En fonction des notes obtenues, on distingue une population de « parents » et une autre « d'enfants ». Les individus de type « enfant » sont éliminés et remplacés par de nouvelles CMC. Ces dernières sont obtenues par des opérations de croisement et de mutation sur les CMC de type « parent ».

Actuellement on ne croise que deux individus ayant une architecture « compatible » (les deux CMC possèdent le même nombre d'états). Il faut donc intégrer cette contrainte dans la méthode de sélection des deux CMC à croiser. On tire d'abord une CMC au hasard dans la population des CMC de type « parent », puis on recherche si une CMC est compatible. Si plusieurs candidates existent, on en tire une au hasard. Dans le cas où aucune CMC ne peut s'apparier avec la première choisie, on obtient une seule CMC « enfant » en recopiant la CMC « parent ». Celle-ci pourra éventuellement subir une ou plusieurs mutations.

Toutes ces opérations achevées, on s'assure par une phase de normalisation que chaque individu de la population est une CMC. On réitère alors en reprenant à la phase d'optimisation locale. En suivant cette démarche, à chaque itération on ne conserve que les CMC qui sont les plus adaptées au problème posé. Celles mal adaptées sont ainsi amenées à disparaître en quelques itérations, favorisant le développement (par un plus grand nombre de représentants dans la population) des CMC les plus adaptées. La qualité de la population va croissante puisqu'on n'élimine jamais les meilleures CMC.

4.2. algorithme détaillé

Les paramètres d'entrée de cet algorithme sont :

- Nombre d'itérations de l'algorithme
- Nombre d'itérations de l'algorithme de Baum-Welch pour l'optimisation locale
- Nombre d'états minimum pour une CMC
- Nombre d'états maximum pour une CMC
- Nombre S' de parents de l'itération t utilisés pour générer les enfants de l'itération $t+1$
- Probabilité de mutation
- Observation o pour laquelle on souhaite obtenir une CMC

Le paramètre principal de sortie est la CMC qui maximise la vraisemblance de l'observation o. L'implémentation actuelle permet en outre d'obtenir de nombreuses statistiques sur les CMC rencontrées durant la phase de recherche, l'évolution de certains paramètres, comme la distribution des CMC suivant le nombre d'états, les temps de calcul, l'évolution du maximum de vraisemblance, etc. L'algorithme 2 donne une version détaillée de GHOSP.

Algorithme 2 Algorithme GHOSP

/* *Initialisation* */
Créer une population de taille S, aléatoirement, contenant des architectures de CMC diverses (en nombre d'états).
Aucun individu n'est marqué « parent ».

/* *Optimisation locale* */
Appliquer sur chaque CMC non marquée « parent » de la population l'algorithme de Baum-Welch

/* *Évaluation* */
Appliquer sur chaque individu de la population non marqué « parent » l'algorithme Forward et mémoriser la probabilité renvoyée.
Pour chaque individu marqué « parent » enlever cette marque

/* *Condition d'arrêt* */
Si le nombre d'itérations maximum n'est pas atteint ou si $P.O/\ )\ =1$, alors continuer, sinon aller à « terminaison »

/* *Sélection* */
Parmi tous les individus de la population, en sélectionner un certain nombre $S'<S$, qui seront utilisés comme parents pour régénérer les $S-S'$ autres individus non retenus. La sélection se réalise suivant les meilleurs scores calculés à la phase « Evaluation ».
Chaque individu sélectionné est marqué « parent ».

/* *Croisement* */
Pour chaque individu non marqué « parent », le remplacer par un des fils obtenus après croisement de deux individus « parents » sélectionnés au hasard. La sélection d'un des deux fils obtenus après croisement se fait au hasard.

/* *Mutation & Normalisation* */
Sur chaque individu non marqué « parent » on applique l'opérateur de mutation. On lui applique ensuite l'opérateur de normalisation, afin que cet individu soit une CMC

/* *Recommencer une évolution* */
Retourner à l'étape « Optimisation locale »

/* *Terminaison* */
Renvoyer la meilleure CMC contenue dans la population en cours.

4.3. paramétrage de l'algorithme

Le paramétrage est empirique mais on constate que l'on trouve des valeurs qui sont correctes dans pratiquement 95 % des cas. En une cinquantaine d'itérations l'algorithme atteint son optimum. Un critère est à l'étude actuellement pour forcer la terminaison de l'algorithme si celui-ci n'améliore pas la solution. Au pire, l'algorithme trouve un optimum très tôt et consomme inutilement du temps de calcul. Il est très rare que l'augmentation du nombre d'itérations apporte quelque chose, surtout si l'on considère le rapport qualité du résultat/temps de calcul. Plus la quantité de données à apprendre est faible et plus on a de chances de trouver une CMC parfaite, ayant une probabilité de générer cette observation égale à 1. Si c'est le cas, l'algorithme se termine immédiatement. La même constatation est faite si les données contiennent un motif qui se répète indéfiniment (dans ce cas

l'observation peut être réduite au motif lui-même, le graphe d'états de la CMC contenant un cycle).

Le paramétrage du nombre d'états et du nombre de représentants pour chaque classe de chaîne est important. Si les bornes de recherche sont trop restreintes on risque de manquer des solutions intéressantes. Dans le cas contraire l'algorithme va perdre beaucoup de temps à concevoir des modèles avec beaucoup d'états, qui risquent de disparaître à l'itération suivante. En général, une plage de 3 à 10 états donne de bons résultats. En ce qui concerne le nombre de représentants pour chaque classe, il en faut suffisamment pour laisser une chance de survie à des modèles qui se révéleraient bons plus tard. Une seconde version de l'algorithme GHOSP en cours d'évaluation permet de contourner ce problème en utilisant une population de taille variable et un croisement différent. Cette valeur est en général de 5 individus par type d'architecture.

Le nombre d'itérations de l'algorithme de Baum-Welch influe grandement sur la convergence. Un trop grand nombre inhibe la capacité de recherche de l'algorithme génétique et GHOSP ne fera pas mieux que Baum-Welch seul. D'autant plus que Baum-Welch est coûteux en temps de calcul. Une plage de 3 à 10 itérations semble correcte. La valeur 10 a été choisie pour les expérimentations.

Le taux de mutation doit être assez fort, de l'ordre de 0.1 à 0.3. Cela est du à la politique élitiste de GHOSP, associé au fait que l'algorithme de Baum-Welch converge rapidement et donc tire la population vers les *optimum* locaux.

Enfin, le nombre de parents conservés d'une population sur l'autre est en général de 75 % du nombre d'individus total dans la population. Cela permet d'avoir un bon éventail de solutions possibles pour le croisement, donc de couvrir un espace de recherche plus grand. On garde un nombre minimum d'enfants et la phase d'optimisation locale ne portant que sur les enfants, l'algorithme est assez rapide. Les individus sont initialisés aléatoirement. Des tests avec une initialisation « équiprobable » (on accorde le même poids à chaque transition ou à chaque probabilité de générer un symbole) ont donnés des résultats sensiblement équivalents à ceux présentés ici. L'algorithme converge assez rapidement, grâce à l'influence de l'algorithme de Baum-Welch qui lui est convergent (sous certaines hypothèses).

Dans notre cas, l'apprentissage d'une image de 400 $\times$300 pixels, 256 niveaux de gris prend environ 30 minutes sur une station de travail SUN Sparc4 ayant 32 Mo de mémoire vive. Dans le cadre d'un travail sur la segmentation [27], la modélisation de l'image précédente par une CMC a permis de réduire le nombre de niveaux de gris de cette image à 11 tout en conservant un niveau de détails dans l'image segmentée atteignant très souvent 1 pixel. Un autre travail sur la prévision de séries temporelles [28] permet avec une observation de 30 valeurs d'effectuer des cycles (apprentissage, prévision, calcul d'erreur) au rythme de 4 par seconde.

5. système d'apprentissage d'images

On peut le décomposer en trois phases : codage, apprentissage, enregistrement.

Le rôle de la première phase est de constituer les observations à partir des données d'entrée. Le but est de faire correspondre les différents symboles, observables à des numéros de colonne de la matrice B de la CMC. Pour cela on code les différents niveaux de gris d'après leur numéro de classe comme indiqué ci-après. Le symbole adopté pour l'observation est alors le numéro de la classe d'appartenance de chaque pixel. Cela revient à faire une réduction du nombre de niveaux de gris dans l'image. La raison principale est de limiter le nombre de colonnes correspondant à des niveaux de gris absents de l'image, de manière à avoir le plus grand nombre de symboles utiles dans la CMC tout en couvrant toute la plage de niveaux de gris. Cette quantification permet de « faciliter » l'apprentissage en limitant la complexité.

Par exemple, travaillant avec des images comprenant 256 niveaux de gris, on peut choisir de faire deux classes : les niveaux inférieurs à 128 seront représentés dans l'observation par le numéro 0, et ceux supérieurs ou égaux à 128 seront représentés par le numéro 1. Cela revient à binariser l'image avec un seuil égal à 128. En pratique c'est l'utilisateur qui détermine le nombre de classes, mais un critère automatique est à l'étude actuellement. D'autre part, les plages de niveaux de gris qui constituent les classes sont uniformément réparties sur l'ensemble des niveaux de gris. Si l'on choisit de faire 4 classes sur 256 niveaux de gris, la première regroupera les niveaux 0 à 63, la seconde comprendra les niveaux 64 à 127 et ainsi de suite. Les test présentés plus tard utilisent 64 symboles pour coder les 256 niveaux de gris.

Ce codage achevé, on distribue les données codées suivant des vecteurs. La raison principale est qu'il est plus facile (pour des raisons calculatoires) d'apprendre une observation lorsqu'elle est découpée en plusieurs parties. Pour cela, on définit une fenêtre de taille fixe, qui va glisser sur l'image. A chaque position de cette fenêtre, on crée un vecteur en réorganisant les données contenues dans la fenêtre. Pour cette réorganisation, nous avons simplement considéré les données lignes par lignes. On pourrait utiliser d'autres parcours d'images tel que celui de Peano [18] par exemple, qui a la propriété de laisser proches dans la séquence unidimensionnelle des pixels qui sont voisins dans l'image. Le glissement de la fenêtre est défini par un pas sur chaque axe. Le pas peut être inférieur aux dimensions de la fenêtre, ce qui autorise un recouvrement et augmente artificiellement la quantité de données soumises à la CMC dans le but de minimiser l'effet du découpage arbitraire réalisé par les fenêtres. Ce système offre l'avantage de proposer plusieurs échantillonnages :

- un seul vecteur pour toute l'image (taille de la fenêtre identique à celle de l'image),

- ligne par ligne (largeur de la fenêtre identique à celle de l'image, hauteur unitaire),
- colonne par colonne (hauteur de la fenêtre identique à celle de l'image, largeur unitaire),
- bande par bande (idem ligne par ligne ou colonne par colonne avec l'autre dimension non unitaire),
- fenêtre par fenêtre (dimensions de la fenêtre inférieures à celles de l'image).

Les tests effectués dans la suite de l'article ont utilisé l'échantillonnage fenêtre par fenêtre, la taille de la fenêtre étant de 16×16 pixels, avec un déplacement de 16 pixels suivant les deux directions (on n'a pas de recouvrement).

La seconde phase est constituée par la mise en oeuvre de l'algorithme GHOSP. Il va rechercher dans tout l'espace des CMC celle qui a la probabilité maximum de générer l'ensemble d'observations constitué à la phase 1.

Afin de s'affranchir de certains problèmes de précision dans les calculs (problème de dépassement de capacité appelé « underflow ») nous avons utilisé des techniques de reéchelonnage (rescaling) [29] (*i.e.* modification des valeurs numériques dans le but d'éviter les dépassements de capacités de calcul de la machine). Ce problème survient dans l'algorithme Foward en particulier, à cause de la multiplication de nombreuses probabilités. L'utilisation des logarithmes ne peut pas résoudre ici ce problème car les produits en cause font partie de sommes. La technique utilisée consiste à normaliser les lignes de la matrice des résultats intermédiaires (les valeurs notées $\alpha_t(i)$). Cette technique est assez coûteuse en calculs, mais entraîne très peu de modifications sur les formules de réestimation des CMC.

Au terme de l'exécution de GHOSP, la meilleure chaîne de Markov cachée trouvée, ainsi que le paramétrage des algorithmes, (échantillonneur, GHOSP), sont enregistrés pour former une base de données. Cette base servira pour la phase de reconnaissance. Cet enregistrement constitue la troisième phase annoncée. Ainsi, pour chaque image I de la base d'apprentissage on associe dans la base de données une CMC de N_I états cachés et de paramètres $\lambda_I = (a_I, B_I, \Pi_I)$. Le schéma du système d'apprentissage est représenté sur la figure 2.

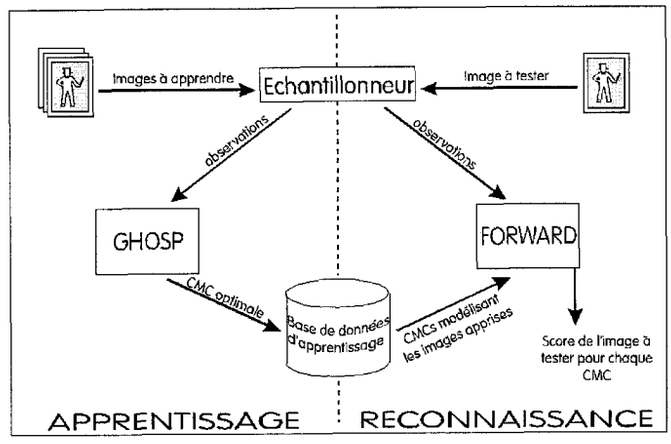


Figure 2. – Schématisation du système d'apprentissage.

6. système de reconnaissance d'images par CMC

On peut maintenant réutiliser les connaissances emmagasinées dans la base de données afin de construire un système de reconnaissance.

Le système d'apprentissage a permis de construire automatiquement une chaîne de Markov cachée pour chaque élément présenté au système (des images en l'occurrence). Lors de chaque apprentissage on a mémorisé le paramétrage de l'algorithme permettant de créer l'observation, c'est-à-dire la forme et la manière de présenter l'élément d'entrée à l'algorithme d'apprentissage. Ces valeurs sont réutilisées par l'algorithme de reconnaissance pour se placer dans les mêmes conditions que l'apprentissage.

Le déroulement du système de reconnaissance passe par deux phases : test de chacun des modèles obtenus après apprentissage, puis élection du meilleur (le plus représentatif pour l'observation à tester).

La phase de test des modèles est composée d'une boucle permettant d'évaluer chacun des modèles obtenus au terme de l'apprentissage. Pour chaque chaîne de Markov cachée trois étapes sont nécessaires :

- création de l'observation d'après l'image d'entrée : on paramètre l'algorithme qui convertit l'image d'entrée en observation avec les valeurs obtenues après apprentissage du modèle,
- évaluation de la correspondance entre le modèle et l'observation : on calcule avec quelle probabilité ce modèle peut générer l'observation,
- on mémorise ce score et on passe au modèle suivant de la base de données (s'il en reste).

La phase d'élection du meilleur modèle est très simple : on dispose d'un ensemble de modèles avec chacun un score. Le modèle élu est celui possédant le plus grand score. Dans le cas où il y aurait des *ex-aequo*, on ne se prononce pas et l'on donne tous les modèles *ex-aequo*.

Cette approche est dite « modèle discriminant », car on a construit durant l'apprentissage autant de CMC qu'il y avait d'images à apprendre. Le système de reconnaissance utilise le score du modèle (donné par l'algorithme Forward) pour discriminer la meilleure solution. Cette approche s'oppose à celle dite « chemin discriminant » où l'on utilise l'algorithme de Viterbi pour calculer le score de l'observation (on élit le modèle qui parcoure le chemin d'états le plus probable pour générer l'observation de test).

Figure 3. – Extrait de la base d'apprentissage.

7. tests et résultats

7.1. protocole de test

Les tests d'apprentissage ont porté sur les images de la base de données ORL [30]. Ces images sont de taille 92 × 112 pixels et comportent au maximum 256 niveaux de gris. Nous avons testé les algorithmes suivants :

- génération aléatoire des CMC (noté ALEA) : on fixe le nombre d'états puis chacune des probabilités des différentes matrices sont fixées aléatoirement,
- ALEA + algorithme de Baum-Welch (noté BW) : on génère aléatoirement une CMC puis on applique l'algorithme de Baum-Welch sur cette CMC,
- algorithme génétique seul (noté AG) : on fixe le nombre d'états de la CMC et c'est l'algorithme génétique qui détermine les coefficients des matrices de la CMC,
- algorithme génétique AG suivi de l'algorithme de Baum-Welch (noté AG + BW) : même algorithme que AG, et l'on applique sur le résultat obtenu (la CMC) l'algorithme de Baum-Welch,
- algorithme génétique comprenant une optimisation locale par l'algorithme de Baum-Welch (noté AG ∪ BW) : on fixe le nombre d'états, un algorithme génétique recherche les coefficients de la CMC et utilise en plus à chaque itération l'algorithme de Baum-Welch afin de travailler avec des CMC localement optimales,
- algorithme GHOSP (noté GHOSP) : l'algorithme génétique détermine le nombre d'états de la CMC ainsi que les coefficients des différentes matrices.

(note : tous les algorithmes génétiques utilisent l'opérateur de croisement à un point décrit au paragraphe 3.6).

Nous avons donné à chacun des algorithmes le même nombre d'essais ou le même nombre d'itérations afin de pouvoir les comparer entre eux. Le premier apprentissage est réalisé par GHOSP, afin de déterminer le meilleur nombre d'états. On reprend ensuite pour chaque image cette valeur pour chacun des cinq autres algorithmes. En parallèle, nous avons également balayé là même plage d'états que GHOSP pour chacun des algorithmes, afin de s'assurer que les autres algorithmes ne trouvaient pas mieux. C'est-à-dire que si GHOSP avait recherché des CMC entre 2 et 15 états, les autres algorithmes ont été appliqués sur des populations de CMC (à nombre d'états fixe pour chaque population) dont le nombre d'états est compris entre 2 et 15. Les valeurs utilisées pour l'apprentissage sont données dans le tableau 1.

La reconnaissance a porté sur l'ensemble des images non apprises. La première expérience (notée E1) consiste à réaliser l'apprentissage d'une image par personne (40 images au total, représentant 10 % de la base d'images) et de tester les ($9 \times 40 = 360$) images restantes de la base. La seconde expérience, notée E2, consiste à réaliser l'apprentissage de deux images par personne (80 images, 20 % de la taille de la base d'images) et de faire la reconnaissance sur les ($8 \times 40 = 320$) images restantes. Les expériences E3, E4 et E5 suivent la même logique en correspondant respectivement à des apprentissages de 30, 40 et 50 % de la base initiale d'images. Pour chacune des expériences, l'apprentissage associe toujours une image à une CMC. La reconnaissance s'effectue donc en terme de nombre de visages bien classés. Les taux de reconnaissance obtenus pour ces cinq tests sont donnés dans le tableau 3.

7.2. résultats

Le tableau 2 présente quelques résultats d'apprentissage relatifs à l'expérience E1. Pour chacun des algorithmes on y trouve la valeur

Tableau 1. – Paramètres d'apprentissage.

Paramètres	ALEA	BW	AG	AG + BW	AG ∪ BW	GHOSP
Nombre total de CMC	11500	11500	11500	11500	11500	11500
Nombre d'essais	11500	11500	50	50	50	50
Nombre d'itérations AG	-	-	20	20	20	20
Nombre d'itérations BW	-	10	-	10	10	10
Taille de la population	-	-	40	40	40	40
Nombre de CMC conservées	-	-	30	30	30	30
Taux de mutation	-	-	0.2	0.2	0.2	0.2
Taux de croisement	-	-	1	1	1	1

Figure 4. – Exemple d'images de test.

Tableau 2. – Résultats d'apprentissage relatifs à l'expérience E1.

Algorithmes	N moyen	Moyenne	Écart Type	Score comp.
ALEA	8	$2{,}14 \times 10^{-19}$	$1{,}35 \times 10^{-18}$	3.65×10^{-11}
BW	7	$1{,}42 \times 10^{-9}$	$5{,}86 \times 10^{-9}$	1
AG	8	$2{,}57 \times 10^{-14}$	$1{,}60 \times 10^{-13}$	4.38×10^{-6}
AG + BW	6	$7{,}55 \times 10^{-8}$	$4{,}71 \times 10^{-7}$	12,88
AG ∪ BW	6	$7{,}25 \times 10^{-7}$	$3{,}62 \times 10^{-6}$	123,7
GHOSP	5	$3{,}33 \times 10^{-6}$	$1{,}16 \times 10^{-5}$	568,3

Tableau 3. – Résultats en reconnaissance. Les valeurs sont arrondies à l'entier le plus proche.

Algorithmes	Taux E1	Taux E2	Taux E3	Taux E4	Taux E5
ALEA	1 %	1 %	1 %	1 %	1 %
BW	50 %	58 %	62 %	64 %	65 %
AG	33 %	38 %	41 %	41 %	45 %
AG + BW	52 %	59 %	63 %	66 %	65 %
AG ∪ BW	60 %	65 %	66 %	69 %	72 %
GHOSP	80 %	86 %	89 %	92 %	96 %

du nombre d'états moyen (noté N moyen) de la meilleure CMC trouvée. Cette moyenne est relative à l'ensemble des images d'apprentissage. La valeur la plus faible est obtenue avec l'algorithme GHOSP (5 états) puis viennent les algorithmes hybrides, l'algorithme de Baum-Welch, et enfin, *ex-aequo*, l'algorithme génétique et la génération aléatoire. La seconde ligne du tableau donne la moyenne des probabilités de chaque vecteur composant chaque observation, pour la meilleure CMC obtenue. La troisième ligne donne l'écart type des probabilités de chaque vecteur composant chaque observation, pour la meilleure CMC obtenue. Enfin la dernière ligne (notée Score comp.) donne le rapport entre la valeur présentée à la troisième ligne du tableau pour l'algorithme considéré et la valeur correspondante obtenue par l'algorithme de Baum-Welch. BW obtient donc un score comparé de 1, et 568.3 dans la colonne GHOSP signifie que la moyenne des scores des meilleurs modèles obtenus pour chaque image par GHOSP est près de 570 fois plus grande que celle obtenue avec BW.

Le tableau 3 donne les taux de reconnaissance obtenus sur la base des images qui n'ont pas servi pour l'apprentissage. Pour E1 on a appris 10 % des images (40 images) et effectué la reconnaissance sur les 90% restants (360 images). Dans le cas de E5 on a appris 50 % des images (200 images) et effectué la reconnaissance sur l'autre moitié. Les expériences E2, E3 et E4 sont relatives aux situations intermédiaires vues plus haut.

7.3. discussion sur les résultats

Les algorithmes hybrides conduisent systématiquement aux meilleurs résultats. D'une part la valeur moyenne du maximum de vraisemblance des observations se rapproche de 1 lorsqu'on utilise les méthodes hybrides, d'autre part l'écart type des valeurs de ce maximum de vraisemblance va en augmentant. Cette seconde constatation signifie qu'il y a une plus grande disparité des probabilités d'apprentissage des différents vecteurs. Sur l'image cela revient à dire qu'il y a des zones bien modélisées alors que d'autres le sont très mal. Des tests supplémentaires seraient nécessaires pour parvenir à mieux contrôler la variance. On peut conjecturer qu'il existe un jeu de paramètres pour lequel il doit être possible de ramener la variance des méthodes hybrides au moins au même niveau que celle des méthodes non hybrides. Le « niveau d'hybridation » influe grandement sur le résultat. AG + BW obtient déjà des résultats meilleurs que BW. Une hybridation plus forte, proposée par AG ∪ BW, améliore les résultats d'AG + BW d'un facteur proche de 10 (en terme de score comparé). Enfin, GHOSP améliore encore les résultats de l'apprentissage, en améliorant le score de l'algorithme AG ∪ BW d'un facteur proche de 5. Il ne faut pas oublier qu'une CMC apprend ici entre 10000 et 20000 données, suivant les valeurs utilisées pour l'apprentissage, et que les probabilités des observations sont alors très faibles. Cela explique les fortes valeurs des scores comparés dans certains cas.

Les taux de reconnaissance du tableau 3 sont en accord avec les résultats du tableau 2. GHOSP obtient les meilleurs scores. Notons l'évolution du taux de reconnaissance, inégal en fonction des algorithmes utilisés entre les expériences E1 et E2. La plus forte évolution est détenue par BW, GHOSP et AG + BW. Alors que ce score n'est que de 50 % pour BW dans l'expérience E1, il est déjà de 80 % pour GHOSP dans les mêmes conditions. Cette valeur passe à 86 % en doublant la quantité d'images apprises. L'introduction de nouvelles images augmente la quantité d'informations apprises, et par conséquent diminue le risque de mauvais classement.

En modifiant le paramétrage des algorithmes on ne bouleverse pas l'ordre de classement des algorithmes les uns par rapport aux autres. En vérifiant des résultats intermédiaires dans l'algorithme GHOSP, on se rend compte que l'optimum de la valeur du maximum de vraisemblance est rapidement atteint, et donc que le nombre d'itérations de cet algorithme est surévalué.

En ce qui concerne les temps de calcul, ceux-ci dépendent de plusieurs facteurs : la taille des modèles à manipuler, la longueur de l'observation à apprendre, la proportion de parents dans la population de CMC. Ce temps est difficilement compressible sans avoir recours à des techniques de parallélisme. Pour l'heure, si l'on prend comme unité de temps la durée d'exécution de l'algorithme de Baum-Welch, les temps d'exécution des autres algorithmes ont les poids donnés par le tableau 4 (avec les valeurs du tableau 1).

A titre indicatif, une station de travail SUN Sparc4 équipée de 32 Mo de mémoire vive traite une itération de chacun des algorithmes avec les durées présentées dans le tableau 5.

Tableau 4. – Rapport entre les temps d'exécution des différents algorithmes.

Algorithmes	ALEA	BW	AG	AG + BW	AG ∪ BW	GHOSP
Taux E2	0,25	1	0,5	1,5	10,5	10,5

Tableau 5. – Durée d'une itération, en secondes, sur une station de travail SUN Sparc4 possédant 32 Mo de mémoire vive.

Algorithmes	ALEA	BW	AG	AG + BW	AG ∪ BW	GHOSP
Temps	0,1	38,3	0,16	0,32	38,5	38,5

On remarquera au passage que le plus gros consommateur de temps de calcul n'est pas l'algorithme génétique mais les algorithmes de Baum-Welch et Forward.

F.S. Samaria [16] a présenté dans ses travaux un système de reconnaissance de visages semblable à celui décrit ici. Ce système est composé d'un système d'échantillonnage de l'image à fenêtre mobile, produisant des observations. L'architecture du modèle de CMC dit « pseudo-2D » est définie *a priori* par l'utilisateur (connectivité et nombre d'états). Il s'agit de CMC continues, et les probabilités initiales de la matrice B de la CMC sont définies suivant des lois Gaussiennes. L'apprentissage est réalisé par l'algorithme de Baum-Welch. Le système de reconnaissance est basé sur le même principe que celui que nous utilisons, à la différence près qu'il emploie l'algorithme de Viterbi pour déterminer le « score » d'un modèle, la valeur discriminante pour la classification. Rappelons que l'algorithme de Viterbi est identique à l'algorithme Forward, à la différence qu'il utilise des maxima alors que l'algorithme Forward utilise des sommes. Pour résumer, Forward prend en compte tous les chemins d'états possibles dans la CMC, alors que Viterbi ne retient que le plus probable.

Ses expérimentations l'on conduit à évaluer l'influence de divers paramètres sur la reconnaissance. Son système travaille sur une base d'images de l'Olivetti Research Laboratory [28]. L'apprentissage été réalisé sur la moitié des images de la base, et la reconnaissance a été effectuée avec l'autre moitié des images. Les résultats obtenus varient de 82 % à 94.5 % de reconnaissance, pour des CMC possédant de 16 à 30 états. Sur toutes les expériences présentées, environ la moitié présentent des taux de reconnaissance supérieurs ou égaux à 90 %. Les meilleurs résultats sont enregistrés pour des valeurs de recouvrement de la fenêtre d'échantillonnage importants (à chaque fois proche des dimensions de cette fenêtre.)

Notre système se positionne bien par rapport à celui qui vient d'être décrit. Il offre l'avantage de produire des CMC avec peu d'états (moins d'une douzaine en général, contre de 16 à 30 pour celui de F.S.Samaria) et d'être plus adaptatif en particulier pour le choix du nombre d'états. De plus, il présente un taux de reconnaissance un peu supérieur (96 % dans les mêmes conditions de test, contre 94.5 % pour celui de F.S.Samaria), en utilisant des modèles différents et plus simples (dans le sens où ils ont

moins d'états). Enfin, la phase d'apprentissage proposée par F.S. Samaria est plus rapide que GHOSP, car beaucoup moins coûteuse en calculs.

8. conclusion

Dans cet article nous avons présenté un nouvel algorithme d'apprentissage de CMC, illustré par une application de reconnaissance d'images. Les tests tendent à démontrer les divers avantages de cet algorithme : résolution du problème de détermination de l'architecture d'une CMC, meilleurs résultats d'apprentissage par rapport aux algorithmes traditionnels et aux algorithmes hybrides, tendance à trouver des CMC ayant un nombre d'états plus faible (par rapport aux autres algorithmes qui sont généralement surparamétrés sur ce plan). Ces caractéristiques permettent en outre d'apprendre de plus grandes quantités de données.

Parmi les améliorations éventuelles on peut citer par exemple :

• Tout d'abord une politique d'évolution moins brutale. Certaines architectures disparaissent très vite de la population. Il faudrait rendre cette disparition moins rapide. Cela peut être réalisé en autorisant une survie plus longue en assouplissant les règles de sélection, ou bien en introduisant une nouvelle forme de mutation qui porterait sur l'architecture.

• Le second point concerne l'opérateur de croisement. Un opérateur de croisement à 3 points a été développé dans le but de mieux respecter le lien entre les différents termes des matrices de la CMC. Car dans la politique actuelle, on peut très bien optimiser une CMC (mettre en accord les probabilités de génération des symboles avec les probabilités de transition par exemple) et la croiser avec une autre (changer sa matrice B avec une CMC n'ayant pas les mêmes transitions par exemple). Ce qui revient à défaire le travail de l'algorithme de Baum-Welch.

• On peut examiner la possibilité de croisement de CMC d'architectures différentes. Le terrain de recherche est très vaste. Comment croiser, où croiser, comment compléter les matrices qui se trouvent incomplètes après une réduction du nombre d'états ou du nombre de symboles... Les possibilités sont multiples.

• Enfin, on peut mettre en œuvre un opérateur de « restart ». Cet opérateur a pour rôle de régénérer toute la population (privée de son meilleur élément) aléatoirement. Il oriente les recherches dans de nouvelles directions. Avec notre politique élitiste on pourrait soit régénérer toute la population (privée de la meilleure CMC) soit toute la partie de la population non sélectionnée. Généralement on déclenche cette opération au bout d'un certain nombre d'itérations durant lesquelles la qualité moyenne de la population stagne.

9. remerciements

Nous tenons à remercier M. F.S. Samaria et M. A. Harter du « Olivetti Research Laboratory » pour avoir mis à notre disposition la base d'images utilisée pour ces tests.

BIBLIOGRAPHIE

[1] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. of IEEE*, vol. 77-2, pp. 257-286, 1989.

[2] M. Slimane, J.P. Asselin de Beauville, Les modèles de Markov cachés du premier ordre (1ère partie), *Rapport de recherches 171 - Laboratoire d'informatique, univ. de Tours*, 36p, 1994.

[3] A. Kriouille, La reconnaissance automatique de la parole et les modèles Markoviens cachés, *Thèse de doctorat*, Univ. de Nancy I, 149p, 1990.

[4] I. Guyon, F. Pereira, Design of a linguistic postprocessor using variable memory length Markov models, *International Conference on Document Analysis and Recognition (Montréal, Canada), IEEE Computer Society Press*, pp. 454-457, 1995.

[5] X. Huang, Y. Akiri, M. Jack, Hidden Markov models for Speech Recognition, *Edimburgh Unversity Press*, 1990.

[6] J.C. Anigbogu, Reconnaissance de textes imprimés multifontes à l'aide de modèles stochastiques et métriques, *Thèse de doctorat, Univ. de Nancy I*, 157p, 1992.

[7] S-S. Kuo, O. Agazzi, Keyword spotting in poorly printed documents using pseudo-2D hidden Markov models, *IEEE transactions on Pattern Analysis and Machine Intellegence*, vol. 16-8, pp. 842-848, 1994.

[8] A. Kundu, L. Bahl, Recognition of handwritten Script : a hidden Markov model based approach, *Proc. of International Conference on Acoustics, Speech and Signal Processing*, pp. 928-931, 1988.

[9] M. Saerens, Hidden Markov models assuming a continuous time dynamic emission of acoustic vectors, *Proc. of Eurospeech*, 1993.

[10] G. Celeux, J. Clairambault, Estimation de chaînes de Markov cachées : méthodes & problèmes, *GDR 134, Traitement du signal et images, journées thématiques «Approches Markoviennes en signal et images»*, pp. 5-19, 1992.

[11] V. Krishnamurthy, S.B. Moore, S-H. Chung, Hidden Markov model signal processing in presence of unknown deterministic interferences, *IEEE transactions on Automatic Control*, vol. 38-1, 1993.

[12] W.D. Mao, S.Y. Kung, An object recognition system using stochastic knowledge source and VLSI parallel architecture, *Proc. of International Conference on Pattern Recognition*, pp. 382-386, 1990.

[13] F. Salzenstein, Modèles Markoviens flous et segmentation statistique non supervisée d'image, *Thèse de doctorat, Univ. de Rennes* I, 142 p, 1996.

[14] M. Slimane, J.P. Asselin de Beauville, T. Brouard, G. Venturini, J.M. Sealelli, Reconnaissance d'image par chaîne de Markov cachée optimisée génétiquement, *28èmes Journées de Statistiques, (Québec, Canada)*, pp 683-687,1996.

[15] F.S. Samaria, F. Fallside, Face identification and features extract using hidden Markov models, Image processing : theory and applications, *Elsevier Science Publisher*, pp. 295-298, 1993.

[16] P. Baldi, Y. Chauvin, T. Hunkapiller, M. McClure, Hidden Markov models of biological primary Sequence information, *Proc. Natural Academic Sciences*, vol. 91-3, pp. 1059-1063, 1995.

[17] L.E. Baum, J.A. Eagon, An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology, *Bull. American Society*, vol. 73, pp. 360-363, 1967.

[18] B. Benmiloud, W. Pieczynski, Estimation des paramètres dans les chaînes de Markov cachées et segmentation d'images, *Traitement du Signal*, vol. 12-5, pp. 433-454, 1995.

[19] G. Celeux, J. Diebolt, The SEM algorithm : a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem, *Computational statistics quarterly*, vol. 2, pp. 73-82, 1985.

[20] M. Askar, B. Derin, A recursive algorithm for the Bayes solution of the smoothing problem, *IEEE transactions on Automatic Control*, vol. 26-2, pp. 558-561, 1981.

[21] M. Slimane, G. Venturini, J.P. Asselin de Beauville, T. Brouard, A. Brandeau, Optimizing HMM with a genetic algorithm, *Artificial Evolution, Lecture Notes in Computer Science, Springer Verlag*, vol. 1063, pp. 384-396, 1996.

[22] C. Dours, Contribution à l'étude du décodage accoustico-phonétique pour la reconnaissance automatique de la parole, *Thèse de doctorat*, Univ. de Toulouse, 1989.

[23] K.M. Ponting, A statistical approach to the determination of hidden Markov models structure, *Proc. of the 7th FASE Symposium*, Edimburgh, 1988.

[24] T. Brouard, M. Slimane, G. Venturini, J.P. Asselin de Beauville, Apprentissage du nombre d'états d'une chaîne de Markov cachée pour la reconnaissance d'images, *Colloque GRETSI sur le Traitement du Signal et des Images, Grenoble (France)*, pp. 845-848, 1997.

[25] A. J. Viterbi, Error bounds for convolutional codes and asymptotically optimum decoding algorithm, *IEEE transactions on Information Theory*, vol. 13, pp. 260-269, 1967.

[26] J. H. Holland, Adaptation in natural and artificial systems, *Ann Arbor, University of Michigan Press*, 1975.

[27] T. Brouard, M. Slimane, G. Venturini, J.P. Asselin de Beauville, Segmentation non-supervisée d'images par chaînes de Markov cachées, *5ème rencontre de la Société Francophone de Classification, Lyon (France)*, pp. 177-180, 1997.

[28] M. Slimane, G. Venturini, J.P. Asselin de Beauville, T. Brouard, Hybrid Genetic Learning of Hidden Markov Models for Time Series Prediction, *In «Biomimetic approaches in management science»*, pp. 179-196, KLUWER Academics Eds, 1998.

[29] T. Brouard, M. Slimane, J.P. Asselin de Beauville, G. Venturini, Apprentissage d'une chaîne de Markov cachée : problèmes numériques liés à l'application à l'image, *Revue de Statistique Appliquée*, vol. XLVI (2), pp. 83-108, 1998.

[30] F.S. Samaria, A. Harter, Paramatrisation of a stochastic model for human face identification, *2nd IEEE Workshop on application of computer vision, Sarasota (Florida)*, 1994.

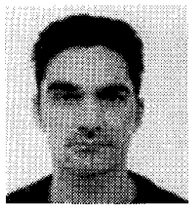
Manuscrit reçu le 26 janvier 1998.

LES AUTEURS

Mohamed SLIMANE

Mohamed Slimane est maître de conférences à l'Ecole d'Ingénieurs en Informatique pour l'Industrie (Université de Tours). Il travaille dans l'équipe Reconnaissance des Formes et Analyse d'Images au sein du Laboratoire d'Informatique (UPRES EA2101) de l'Université de Tours. Il étudie les techniques mixtes déterministes et stochastiques pour l'apprentissage et la reconnaissance à l'aide des chaînes de Markov cachées et des algorithmes génétiques. Les principales applications portent sur la reconnaissance d'images et la prévision de séries temporelles.

Gilles VENTURINI



Gilles Venturini est professeur à l'Université de Tours dans l'Ecole d'Ingénieurs en Informatique pour l'Industrie. Au sein du Laboratoire d'Informatique (UPRES EA 2101), il étudie les approches biomimétiques en informatique, et en particulier les algorithmes génétiques et les fourmis artificielles. Il est notamment rédacteur en chef de la revue électronique sur l'apprentissage par les données (READ) et participe au réseau d'excellence européen Evonet depuis sa création.

Thierry BROUARD

Thierry Brouard est maître de conférences à l'Université de Tours au Département d'Informatique. Il est membre de l'équipe Reconnaissance des Formes et Analyse d'Images du Laboratoire d'Informatique (UPRES EA 2101) de l'Université de Tours. Ses travaux portent sur l'apprentissage et la reconnaissance au moyen d'approches hybrides utilisant les chaînes de Markov cachées et les algorithmes génétiques. Les principales applications concernent la reconnaissance et la segmentation d'images, ainsi que la prévision de séries temporelles.

Jean-Pierre ASSELIN de BEAUVILLE

Jean-Pierre Asselin de Beauville est professeur d'informatique à l'Ecole d'Ingénieurs en Informatique pour l'Industrie (Université de Tours). Il a créé l'équipe Reconnaissance des Formes et Analyse d'Images au sein du Laboratoire d'Informatique (UPRES EA 2101) de l'Université de Tours. Il co-dirige aujourd'hui cette equipe avec le professeur Nicole Vincent. Ses principales publications concernent les approches stochastiques et neuronales en reconnaissance et apprentissage. Il est actuellement en détachement à Montréal (Canada-Québec).