Inférence d'automates et correction d'erreur pour la classification des protéines

Christopher Kermorvant et Pierre Dupont

EURISE,
Université Jean Monnet,
23 rue du Dr. Paul Michelon, Saint-Etienne
{kermorva, pdupont}@univ-st-etienne.fr

1 INTRODUCTION

Le champ d'investigation de la génomique couvre l'étude des gènes, de leur fonctionnement, des produits dérivés de leur fonctionnement (essentiellement des protéines) et des mécanismes de leur régulation. Les problèmes posés par la génomique sont très divers. On peut citer, entre autres, la localisation des gènes et l'identification de leur fonction, l'identification des fonctions des protéines et la prédiction de leur structure tridimensionnelle, la comparaison des génomes et l'étude de leur évolution.

La génomique est un domaine de recherche qui génère une masse considérable de données. En effet, les progrès récents des techniques automatiques d'analyses chimiques ont déjà permis de réaliser le séquençage complet de l'ADN de plusieurs organismes vivants. Il n'est donc plus envisageable de poursuivre l'interprétation des données disponibles sans techniques automatiques.

Un des problèmes qui se posent dans l'analyse des données génomiques est celui de la détermination des fonctions des protéines nouvellement séquencées. Dans une certaine mesure, des hypothèses concernant la fonction d'une protéine inconnue peuvent être faites en comparant sa structure à celles de protéines connues. En effet, deux protéines qui ont une séquence chimique "proche" ont une forte probabilité d'avoir une fonction identique. Il est cependant important de noter que la réciproque n'est pas toujours vraie.

Parmi les méthodes proposées pour la classification des protéines, on trouve des méthodes allant des approches déterministes basées sur des motifs locaux à des approches probabilistes prenant en compte les séquences entières[1]. Les premières cherchent à identifier des motifs caractéristiques de chaque famille de protéines et à les rechercher dans la séquence d'une protéine inconnue. Les secondes consistent à calculer une distance entre la séquence d'une protéine inconnue et

[1]Notons que ces modèles peuvent néanmoins être utilisés en prédiction locale de séquences.

un modèle de séquence de protéine correspondant à chaque famille. Pour les méthodes basées sur des motifs locaux, il existe des bases de données qui recensent les motifs caractéristiques des protéines (PROSITE (Bairoch, 1991)). Parmi les méthodes probabilistes, on peut citer les arbres de suffixes probabilistes (Bejerano & Yona, 2001), les profils (Gribskov & McLachlan, 1987) et les modèles de Markov (Krogh *et al.*, 1994)(Baldi *et al.*, 1994).

L'approche que nous proposons consiste à considérer les séquences d'acides aminés constituant les protéines comme des mots sur l'alphabet des acides aminés. Les familles de protéines constituent alors des langages, que l'on peut chercher à caractériser par des automates à états finis. Afin de prendre en compte les variations entre les séquences d'une même famille, les automates utilisés sont des automates à états finis déterministes stochastiques (SDFA). Le caractère probabiliste des automates nous permet de prendre en compte le "bruit" dans les séquences. Le caractère structurellement déterministe des automates nous permet d'utiliser des algorithmes d'apprentissage déjà proposés. De plus, afin de pallier le manque de données pour l'estimation des probabilités du SDFA, nous introduisons un modèle de correction d'erreur. Ce modèle permettrait aussi d'intégrer des phénomènes biologiques comme les substitutions communes d'un acide aminé par un autre (Dayhoff, 1978).

Dans une première partie, nous décrivons le modèle proposé, un automate à états fini stochastique déterministe couplé à un modèle de correction d'erreur, ainsi que les méthodes d'apprentissages du modèle. Dans une seconde partie, nous présentons des résultats de classification sur une base de données de protéines. Enfin, nous proposons des perspectives de développement et d'améliorations du modèle.

2 AUTOMATES À ÉTATS FINIS DÉTERMINISTES STOCHASTIQUES

2.1 Définition

Un automate à états fini stochastique déterministe (SDFA) $\mathcal{A}$ est un quintuplet $< Q, \Sigma, \delta, q_0, F >$, où Q est un ensemble (fini) d'états, Σ est un alphabet, δ est une fonction de $Q \times \Sigma \rightarrow Q \times [0..1]$ (fonction de transition probabiliste), q_0 est l'état initial et F une fonction de $Q \rightarrow [0..1]$ qui associe à chaque état sa probabilité d'être final. On définit δ_Q et δ_P les projections de δ sur respectivement Q et $[0..1]$. Afin de définir une distribution de probabilités sur Σ^*, les fonctions δ et F doivent vérifier $\forall q \in Q$, $\sum_{a \in \Sigma} \delta_P(q, a) + F(q) = 1$.

Un chaîne $a_0 \cdots a_{l-1}$ est engendrée par un automate $\mathcal{A}$ si et seulement si il existe une séquence d'états $e_0 \cdots e_l$ telle que

- $e_0 = q_0$
- $\forall i \in [0, l-1]$, $\delta_Q(e_i, a_i) = e_{i+1}$ et $\delta_P(e_i, a_i) \neq 0$
- $F(e_l) \neq 0$

entrée : I^+, ensemble d'exemples (séquences) et α, un paramètre
sortie : Automate à états fini stochastique déterministe

début
A ← *construire_PTA*(I^+)
tant que (q_i, q_j) ← *choisir_états*(A) **faire**
si *fusion_possible*(q_i, q_j, α) **alors** *fusionner*(A,q_i, q_j) **finsi**
fin tant que
retourner A
fin

FIG. 1: Algorithme générique d'inférence

La probabilité associée par l'automate à la chaîne est alors $P(a_0 \cdots a_{l-1}) = \prod_{i=0}^{l-1} \delta_P(e_i, a_i) * F(e_l)$. Un automate associe donc une probabilité (possiblement nulle) à toute chaîne de Σ^*, et on a la propriété de consistance : $\sum_{x \in \Sigma^*} P(x) = 1$. Le langage engendré par un SDFA est l'ensemble des chaînes de Σ^* de probabilité non nulle.

2.2 Apprentissage

Plusieurs algorithmes d'apprentissage ont été proposés pour les SDFA (Rulot & Vidal, 1988; Carrasco & Oncina, 1994; Ron *et al.*, 1995; Thollard & Dupont, 2000). Parmi ces algorithmes, seuls les algorithmes Alergia (Carrasco & Oncina, 1994) et MDI (Thollard & Dupont, 2000) s'appliquent sans restriction à tous les automates stochastiques déterministes. En première approche, nous avons choisi d'utiliser l'algorithme Alergia. Cet algorithme construit un SDFA à partir d'un échantillon d'exemples positifs I^+ de chaînes d'un langage. Le pseudo-code d'un algorithme générique d'inférence est présenté Figure 1.

L'algorithme commence par construire l'arbre accepteur des préfixes (PTA). Le PTA est l'automate qui accepte exactement les chaînes de I^+ et dont les états acceptant des préfixes identiques sont fusionnés (fonction *construire_PTA(I^+)*). On définit alors le PTA stochastique (SPTA) de la manière suivante : on étend la fonction de transition du PTA en associant à chaque transition une probabilité proportionnelle au nombre de fois que la transition est exercée dans I^+. De même, on définit la fonction F en associant à chaque état q une probabilité d'être final proportionnelle au nombre de fois que q est exercé comme état final dans I^+. Formellement, si on note $C(q)$ le nombre de fois que l'état q est exercé dans I^+, $C_f(q)$ le nombre de fois que l'état q est exercé comme état final dans I^+ et $C(q, a)$ le nombre de fois que la transition (q, a) est exercée dans I^+, on a $\delta_P(q, a) = \frac{C(q,a)}{C(q)}$ et $F(q) = \frac{C_f(q)}{C(q)}$.

On définit ensuite un ordre sur les états du SPTA, généralement l'ordre hiérarchique [2]. L'algorithme consiste alors à parcourir les états du SPTA dans l'ordre (fonction *choisir_états(A)*) et à tester si la fusion avec chacun des états d'ordre

[2] l'ordre hiérarchique correspond à un ordre par longueur des chaînes menant à chaque état, puis pour une longueur donnée, par ordre alphabétique des chaînes.

inférieur est possible (fonction *fusion_possible*(q_i, q_j, α)). Si la fusion est possible elle est effectuée (fonction *fusionner(A,*q_i, q_j*)*). Si cette fusion entraîne un non déterminisme, des fusions supplémentaires sont effectuées pour retrouver un automate déterministe. Le critère de compatibilité entre deux états est dérivé des bornes de Hoeffding (Hoeffding, 1963). Ce critère, qui dépend d'un paramètre α, exprime que deux états q_1 et q_2 sont compatibles si seulement si :

$$\forall a \in \Sigma, \qquad \left|\frac{C(q_1,a)}{C(q_1)} - \frac{C(q_2,a)}{C(q_2)}\right| < \sqrt{\frac{1}{2}\ln\frac{2}{\alpha}}\left(\frac{1}{\sqrt{C(q_1)}} + \frac{1}{\sqrt{C(q_2)}}\right) \quad (1)$$

$$\left|\frac{C_f(q_1)}{C(q_1)} - \frac{C_f(q_2)}{C(q_2)}\right| < \sqrt{\frac{1}{2}\ln\frac{2}{\alpha}}\left(\frac{1}{\sqrt{C(q_1)}} + \frac{1}{\sqrt{C(q_2)}}\right) \quad (2)$$

$$\forall a \in \Sigma,\ \delta_Q(q_1,a) \text{ et } \delta_Q(q_2,a) \qquad \text{sont compatibles} \quad (3)$$

La première condition exprime la compatibilité des probabilités des paires de transitions sortant des états q_1 et q_2, la deuxième condition exprime la compatibilité des probabilités d'être final des états q_1 et q_2, et la troisième condition exprime récursivement la compatibilité de toute paire d'états successeurs de q_1 et q_2.

2.3 Classification avec un SDFA

Les SDFA peuvent être utilisés dans un problème de classification de séquences où le nombre de classes est connu. L'approche consiste à apprendre pour chaque classe un SDFA sur un ensemble d'apprentissage. Après la phase d'apprentissage, la classification s'opère comme suit. Si on note $S = a_0 \cdots a_{l-1}$ une séquence à classer et $\mathcal{A}_i$ l'automate associé à la classe i, alors la probabilité que la séquence appartienne à la classe i est :

$$P(\mathcal{A}_i|S) \propto P(S|\mathcal{A}_i)P(\mathcal{A}_i) \quad (4)$$

où $P(S|\mathcal{A}_i)$ est appelée vraisemblance que S ait été générée par l'automate $\mathcal{A}_i$ et $P(\mathcal{A}_i)$ la probabilité a priori de la classe i. L'équation 4 se réécrit [3] :

$$-\log P(\mathcal{A}_i|S) = -\log P(S|\mathcal{A}_i) - \log P(\mathcal{A}_i) \quad (5)$$

où $-\log P(S|\mathcal{A}_i)$ est le coût associé à la séquence S par l'automate $\mathcal{A}_i$. Chercher la classe i la plus probable pour la séquence S revient donc à cherche la classe i qui minimise $-\log P(\mathcal{A}_i|S)$. La probabilité a priori de chaque classe peut être uniforme ou bien estimée par la proportion de chaque classe en nombre de séquences dans l'ensemble d'apprentissage.

2.4 Lissage par correction d'erreur

Le problème du lissage des paramètres (problème des fréquences nulles ou très petites) se pose fréquemment lors de l'estimation des paramètres d'un modèle

[3] le passage au logarithme est nécessaire pour éviter des problèmes de précision numérique

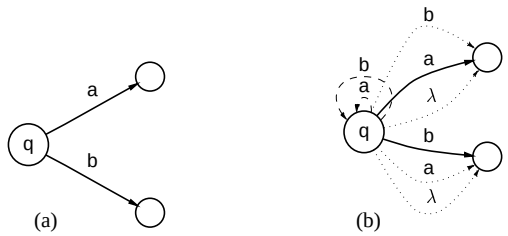

FIG. 2: Ajout des transitions d'erreur sur un SDFA

complexe. En effet, dans le cas de modèles à très grand nombre de paramètres, même si l'estimation s'effectue sur des corpora importants, de nombreux paramètres sont estimés avec peu ou pas de données. Certains évènements sont donc prédits avec une probabilité incorrecte par manque de données. En particulier, dans le cas d'estimation d'un automate stochastique, on ne peut pas garantir que toutes les chaînes possibles d'un langage se voient affecter une probabilité non nulle par l'automate. Une méthode de lissage particulière aux SDFA a été proposée par Dupont et Amengual (Dupont & Amengual, 2000) : le lissage par correction d'erreur. Elle nous semble pertinente pour le problème traité car elle modélise, dans un sens probabiliste, les opérations de mutation qui peuvent apparaître dans des séquences biologiques.

2.4.1 Définition du modèle

Le lissage par correction d'erreur consiste à estimer en plus du SDFA un modèle de correction d'erreur. Ce modèle permet de rendre possibles des insertions, suppressions et substitutions de lettres par rapport au SDFA. On étend donc le SDFA en ajoutant, pour chaque état et pour chaque transition, des transitions supplémentaires correspondant aux opérations élémentaires d'édition possibles (voir figure 2.4.1). Si ce modèle est complet, c'est-à-dire si une insertion, une suppression et une substitution sont possibles pour toute lettre et pour tous les états, alors l'automate étendu accepte toutes les chaînes de Σ^*.

Les paramètres du modèle d'erreur sont, pour chaque état q du SDFA et pour toutes les lettres de l'alphabet, les paramètres des distributions de probabilité suivantes : $p(\lambda \rightarrow a|q)$, la probabilité d'insérer la lettre a sur l'état q, $p(a \rightarrow b|q)$, la probabilité de substituer a par b en passant de l'état q à un état q', et $p(a \rightarrow \lambda|q)$, la probabilité de supprimer la lettre a en passant de l'état q à un état q'. Comme le nombre de paramètres à estimer peut être très important si le SDFA comporte un grand nombre d'états, on peut définir un modèle de correction d'erreur qui ne dépend pas de l'état courant. Les paramètres à estimer sont alors les paramètres des distributions de probabilité suivantes, indépendantes de l'état courant : $p(\lambda \rightarrow a)$, la probabilité d'insérer la lettre a, $p(a \rightarrow b)$, la probabilité de substituer a par b, et $p(a \rightarrow \lambda)$, la probabilité de supprimer la lettre a. Le nombre de paramètres à

estimer est alors indépendant du nombre d'états de l'automate.

2.4.2 Estimation initiale du modèle d'erreur

Les distributions du modèle d'erreur sont estimées sur un ensemble de validation, ensemble différent de l'ensemble d'apprentissage du SDFA. Le SDFA est utilisé pour reconnaître chaque séquence de cet ensemble de validation en utilisant la distance de Levenshtein (Levenshtein, 1966). Pour chaque séquence, le meilleur chemin est celui qui minimise la distance de Levenshtein. On compte alors $C(\lambda \to a)$, le nombre de fois où l'on a dû insérer le symbole a, $C(a \to \lambda)$ le nombre de fois où l'on a dû supprimer le symbole a et $C(a \to b)$, le nombre de fois où l'on a dû substituer le symbole b au symbole a. On compte aussi le nombre de fois où un état final a été exercé, C_f. Des informations biologiques concernant les mutations d'acides aminés peuvent être introduites dans le modèle à ce niveau, en utilisant par exemple les matrices de substitution (Henikoff & Henikoff, 1992).

2.4.3 Définition du SDFA étendu

Dans un SDFA étendu avec le modèle de correction d'erreur, les probabilités associées aux transitions sont définies de manière à assurer la consistance des états. Ces transitions sont donc définies de la manière suivante. On note

$$C_{ins} = \sum_{a \in \Sigma} C(\lambda \to a) \qquad C_{\overline{ins}} = \sum_{a \in \Sigma} \sum_{b \in \Sigma \cup \lambda} C(a \to b) + C_f$$

respectivement le nombre total d'insertions et son complémentaire. La probabilité d'opérer une insertion est donc $P_{ins} = \frac{C_{ins}}{C_{ins}+C_{\overline{ins}}}$. On a alors les estimations suivantes pour les probabilités de transition du modèle étendu :

– la probabilité d'insérer le symbole a :

$$P(\lambda \to a|q) = P_{ins} \cdot \frac{C(\lambda \to a)}{C_{ins}}$$

– la probabilité de substituer b à a :

$$P(a \to b|q) = (1 - P_{ins}) \frac{C(a \to b)}{\sum_{b \in \Sigma \cup \lambda} C(a \to b)} \cdot \delta_P(q, a)$$

– la probabilité de supprimer a :

$$P(a \to \lambda|q) = (1 - P_{ins}) \frac{C(a \to \lambda)}{\sum_{b \in \Sigma \cup \lambda} C(a \to b)} \cdot \delta_P(q, a)$$

– la probabilité d'arrêter la génération sur l'état q :

$$P_f(q) = (1 - P_{ins}) \cdot F(q)$$

2.4.4 Meilleur chemin et réestimation itérative

Le calcul de la probabilité attribuée à une chaîne par un SDFA s'obtient simplement par le produit des probabilités des transitions exercées le long du chemin d'acceptation. Par contre, un SDFA étendu par un modèle de correction d'erreur n'étant plus déterministe, il peut exister plusieurs chemins d'acceptation de la chaîne par cet automate étendu. Si cet automate est sans cycle, on peut alors calculer la probabilité associée au meilleur chemin par un algorithme de programmation dynamique, l'algorithme Viterbi. Le meilleur chemin est alors celui de probabilité maximale. Si l'automate étendu comporte des cycles, alors l'algorithme Viterbi n'est plus applicable. Amengual et Vidal (Amengual & Vidal, 1998) ont proposé une solution à ce problème.

Le modèle d'erreur peut être réestimé itérativement en recalculant les comptes, décrits à la Section 2.4.2, le long du chemin de probabilité maximale pour chaque séquence de l'ensemble de validation. Le processus de réestimation est conduit jusqu'à ce que le nombre maximum d'itérations soit atteint ou bien, jusqu'à ce que la variation des probabilités associés aux chaînes de l'ensemble de validation tombe sous un certain seuil.

3 EXPÉRIMENTATIONS

3.1 Données

Nous avons testé notre modèle sur la base de protéines PFAM (Sonnhammer *et al.*, 1997). Les protéines sont constituées de séquences d'acides aminés. Ces derniers étant au nombre de 20, les protéines sont codées sur un alphabet à 20 lettres. La base PFAM contient les séquences de protéines de divers organismes vivants classées par famille. Le problème consiste à apprendre un modèle afin de pouvoir classer une séquence inconnue dans une de ces familles. Nous avons utilisé la version 1.0 de la base car des études similaires ont été menées sur cette base. Cette version de la base contient les séquences de 22301 protéines classées en 175 familles. La base a été séparée aléatoirement en ensembles d'apprentissage, de validation et de test, contenant respectivement 13313, 4394 et 4594 séquences (soit environ 60%, 20% et 20% du total).

3.2 Apprentissage du modèle

Pour chaque famille, un SDFA est appris par l'algorithme Alergia sur l'ensemble d'apprentissage. La valeur optimale du paramètre α de l'algorithme Alergia est recherché sur l'intervalle $[0, 1]$ par différents tests sur l'ensemble de validation (voir Figure 3). Pour des raisons de capacité mémoire, nous n'avons pas pu mener les tests pour des valeurs du paramètre α supérieures à 0.8 . Nous avons choisi de garder $\alpha = 0.6$, cette valeur du paramètre offrant un bon compromis entre le taux de classification correcte et le temps de calcul.

α	0.2	0.4	0.5	0.6	0.8	PTA
nb états	136	315	439	652	1002	11034
nb trans.	787	2310	3068	3678	4209	11033
% classif. correcte	83.59	84.39	89.28	93.42	94.97	*

FIG. 3: Nombre d'états, de transitions et pourcentage de classification correcte sur l'ensemble de validation pour des SDFA étendus correspondant à différentes valeurs de α et pour le PTA. Pour des raisons de capacité mémoire, la classification n'a pas pu être réalisée pour $\alpha > 0.8$.

Un modèle de correction d'erreur initial est ensuite calculé sur l'ensemble de validation à partir de l'automate non stochastique de chaque famille en utilisant la distance de Levenshtein (Levenshtein, 1966). Tous les modèles de correction sont alors fusionnés en un modèle initial commun, simplement en sommant tous les comptes correspondant à la même opération d'édition. Ce modèle initial commun a l'avantage d'être plus général puisqu'appris sur l'ensemble des données de validation pour toutes les familles. Enfin, pour chaque modèle de famille, ce modèle initial commun est réestimé sur l'ensemble de validation.

3.3 Test du modèle

L'ensemble des modèles a été utilisé pour classer dans une famille de protéines chaque séquence d'acides aminés de l'ensemble de test selon la procédure décrite en Section 2.3 et en utilisant une distribution a priori uniforme sur tous les modèles de famille. Les taux moyens de classification correcte sur chaque ensemble sont présentés Figure 4 pour $\alpha = 0.6$. Sur l'ensemble de test, la classification d'une séquence parmi 175 classes est correcte dans 91.58% des cas. Ce résultat nous semble prometteur et des tests comparatifs sont en cours avec des méthodes à l'état de l'art comme les HMM.

Il est cependant important de noter que la comparaison avec les HMM sur la base PFAM est biaisée, en raison de la construction même de la base. En effet, la classification en famille de la base PFAM a été réalisée à l'aide de HMM. Une comparaison correcte devrait être réalisée sur une base indépendante des méthodes de classification. Malheureusement, aucune base de ce type ne semble disponible à ce jour.

Cependant, on peut d'ores et déjà comparer, pour la base PFAM, la taille des modèles HMM et celle des SDFA étendus. Le nombre de paramètres pour l'ensemble des 175 HMM de la base PFAM est de l'ordre de $1.9 \cdot 10^6$, alors que nos modèles ne nécessitent que $0.75 \cdot 10^6$ paramètres .

	Ens. apprent.	Ens. valid.	Ens. test
% classif. correcte	99.91	93.42	91.58

FIG. 4: Taux de classification moyen sur les ensembles d'apprentissage, de validation et de test pour $\alpha = 0.6$.

4 PERSPECTIVES DE DÉVELOPPEMENT

Dans cette première approche nous avons choisi l'algorithme Alergia pour l'inférence des automates stochastiques. Cependant, un autre algorithme, appelé MDI, permettant également l'inférence de tout type de SDFA, a récemment été proposé par Thollard et Dupont (Thollard & Dupont, 2000). Cet algorithme est basé sur un critère de fusion global et non plus local comme dans le cas d'Alergia. La supériorité de MDI sur Alergia a été montrée expérimentalement pour l'apprentissage de modèles de langage en reconnaissance de la parole. Nous pouvons donc envisager l'utilisation de MDI dans la cadre de la classification de protéines afin de renforcer l'aspect global de notre modèle et de nous rapprocher des techniques comme les profils (Gribskov & McLachlan, 1987) et les HMM (Krogh *et al.*, 1994).

D'autre part, toujours dans l'optique de renforcer l'aspect global du modèle, les algorithmes d'inférence comme Alergia ou MDI peuvent être modifiés afin d'introduire des contraintes sur la topologie du modèle à apprendre. On peut ainsi limiter l'apprentissage à des automates sans cycles (Rulot & Vidal, 1988; Ron *et al.*, 1995). L'apprentissage serait ainsi contraint vers des modèles présentant une topologie linéaire, proche des HMM utilisés dans ce cadre (Krogh *et al.*, 1994).

De plus, la similarité entre les HMM et les SDFA nous permet d'envisager l'adaptation des raffinements développés pour les premiers aux seconds, en particulier au niveau de l'estimation des distributions de probabilité avec l'utilisation de distributions a priori à la fois sur les modèles de famille et sur les probabilités de transition des modèles (Brown *et al.*, 1993; Sjolander *et al.*, 1996).

En ce qui concerne le modèle de correction d'erreur, celui-ci étant appris sur un ensemble de validation souvent de faible taille, les probabilités estimées sont peu fiables. Des méthodes de validation croisée peuvent être utilisées pour améliorer l'estimation de ces probabilités. D'autre part, on peut envisager d'introduire dans le modèle de correction d'erreur des informations biologiques concernant les mutations des acides aminés (Dayhoff, 1978). Enfin le modèle de correction peut être estimé par d'autres méthodes, comme celles utilisées pour les matrices de substitution (Henikoff & Henikoff, 1992).

RÉFÉRENCES

AMENGUAL J. & VIDAL E. (1998). Effi cient error correcting viterbi parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **20**(10), 1109–1116.

BAIROCH A. ("1991). PROSITE : A dictionary of sites and patterns in proteins. *Nucleic Acids Research*, **19**, 2241–2245.

BALDI P., CHAVIN Y., HUNKAPILLER T. & MCCLURE M. (1994). Hidden markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, **91**, 1059–1063.

BEJERANO G. & YONA G. (2001). Variations on probabilistic suffi x trees : statistical modeling and prediction of protein families. *Bioinformatics*, **17**(1), 23–43.

BROWN M., HUGHEY R., KROGH A., MIAN I., SJOLANDER K. & HAUSSLER D. (1993). Using dirichlet mixture priors to derive hidden markov models for protein families. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, p. 47–55 : AAAI Press.

CARRASCO R. & ONCINA J. (1994). Learning stochastic regular grammars by means of a state merging method. In *Proc. Int. Coll. on Grammatical Inference*, volume 862 of *Lecture Notes in Artifi cial Intelligence*, p. 139–152 : Springer Verlag.

DAYHOFF M. (1978). *Atlas of protein sequence and structure*, volume 5. National Biomedical Research Foundation.

DUPONT P. & AMENGUAL J.-C. (2000). Smoothing probabilistic automata : an error-correcting approach. In *Proc. Int. Coll. on Grammatical Inference*, Lecture Notes in Artifi cial Intelligence, p. 51–64 : Springer Verlag.

GRIBSKOV M. & MCLACHLAN M. (1987). Profi le analysis : detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, **84**, 4355–4358.

HENIKOFF S. & HENIKOFF J. (1992). Amino acid substitution martices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

HOEFFDING W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, **58**(301), 13–30.

KROGH A., BROWN M., MIAN I., SJOLANDER K. & HAUSSLER D. (1994). Hidden markov models in computational biology : Applications to protein modeling. *Journal of Molecular Biology*, **235**, 1501–1531.

LEVENSHTEIN V. (1966). Binary codes capable of correcting deletion, insertions, and reversals. *Cybernetics and Control Theory*, **10**, 707–710.

RON D., SINGER Y. & TISHBY N. (1995). On the learnability and usage of acyclic probabilistic automata. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, p. 31–40, Santa Cruz, CA : ACM Press.

RULOT H. & VIDAL E. (1988). An effi cient algorithm for the inference for pattern recognition. In G. FERRATÈ, T. PAVLIDIS, A. SANFELIU & H. BUNKE, Eds., *Advances in Structural and Syntactic Pattern Recognition*, p. 173–184 : NATO ASI Springer-Verlag.

SJOLANDER K., KARPLUS K., BROWN M., HUGHEY R., KROGH A., MIAN I. S. & HAUSSLER D. (1996). Dirichlet mixtures : a method for improving detection of weak but signifi cant protein sequence homology. *Computer Applications in the Biosciences*, **12**(4), 327–345.

SONNHAMMER E. L. L., EDDY S. R. & DURBIN R. (1997). Pfam : a comprehensive database of protein domain families based on seed alignments. *Proteins*, **28**(3), 405–420.

THOLLARD F. & DUPONT P. (2000). Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. Int. Conf. on Machine Learning*, p. 975–982 : Morgan Kaufmann, San Francisco, CA.