Contribution au développement d'une méthode hybride discriminante-générative de prédiction de la structure secondaire des protéines

Julien Vannesson

Nancy Université – Université Henri Poincaré

Contribution au développement d'une méthode hybride discriminante-générative de prédiction de la structure secondaire des protéines

MÉMOIRE

soutenu le 05 septembre 2007

pour l'obtention du

Diplôme Master 2 recherche

(Spécialité Perception Raisonnement et Intéractions Multimodales)

par

Julien Vannesson

Encadrant : Yann Guermeur

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UI

Nancy-Université
Université Henri Poincaré

Loria

Remerciements

Je tiens à remercier tout d'abord le responsable scientifique de l'équipe ABC : Yann Guermeur, qui m'a encadré durant ces 6 mois de stage, qui m'a fourni une aide précieuse en prenant le temps de m'expliquer les notions qui m'échappaient et qui m'a guidé pour mener à bien les objectifs fixés. Ensuite je voudrais également remercier Fabienne Thomarat pour ses conseils. Et enfin, toute l'équipe ABC pour son accueil.

Table des matières

Introduction

La prédiction de la structure spatiale des protéines est un problème central de biologie structurale qui a été abordé sous de nombreux angles différents. Un premier pas dans le sens d'une prédiction de la structure 3D totale d'une protéine est de prédire les conformations locales de la chaîne polypeptidique constituant la protéine. On appelle structure secondaire l'ensemble de ces conformations. De nombreux travaux ont été réalisés depuis ces 30 dernières années et les méthodes de prédiction on beaucoup gagné en précision. Les premiers travaux pour déterminer la structure secondaire étaient basés sur des méthodes statistiques dans lesquelles la vraisemblance de chaque acide aminé par rapport à tel ou tel état conformationnel était déterminée par rapport aux structures des protéines déjà connues [1]. Ce n'est qu'aux alentours de 1988 que les premiers réseaux de neurones dédiés à la prédiction des structures secondaire furent développés [2]. Bien qu'utilisant des protocoles expérimentaux basés sur des protéines différentes, la précision des prédictions faites par Qian et Sejnowski semblait meilleure. Ainsi, de nombreuses autres applications des réseaux de neurones à ce problème furent mises en œuvre. L'utilisation des réseaux de neurones est alors une grande avancée pour le domaine. Il faudra attendre 1993 et l'article de Rost et Sander [3] pour qu'un nouveau pas majeur soit franchi. L'idée majeure de Rost et Sander fut l'utilisation des alignements multiples des protéines afin d'améliorer la prédiction[1]. C'est ainsi que la barre de 71% de taux de reconnaissance de la structure secondaire des protéines fut dépassée.

De nos jours, les réseaux de neurones couplés aux alignements, plus d'autres techniques (comme les méthodes d'ensembles qui sont l'utilisation conjointe de centaines de réseaux de neurones) détiennent les meilleures performances avec environ 80% de taux de reconnaissance. Cependant, depuis une dizaine d'année, une nouvelle méthode d'apprentissage a été employée avec succès dans la prédiction de la structure secondaire des protéines : les machines à vecteurs support. Il s'agit d'une autre méthode d'apprentissage numérique qui est la spécialité de l'équipe Apprentissage et Biologie Computationnelle (ABC) du LORIA. Cela fait bientôt 10 ans que cette équipe développe des SVM mutliclasses, et notamment une dédiée au traitement des séquences protéiques. Les résultats obtenus avec cette SVM étant très encourageants, il était naturel d'y appliquer les méthodes qui ont fait le succès des réseaux de neurones : c'est l'objectif de mon stage. L'idée

[1] on fournit en entrée de réseau de neurones, non plus une seule protéine, mais un profil d'alignement de protéines homologues.

principale est tirée de la thèse de mon encadrant Yann Guermeur [4], à savoir, utiliser les alignements multiples [5] avec la SVM de l'équipe, puis transformer les sorties de la SVM afin qu'elles puissent être directement exploitées par un HMM (modèle de Markov caché) [2]. Il s'agit donc de marier une méthode discriminante (la SVM) à une méthode générative(le HMM). Ce stage se situe en aval des travaux de l'équipe, afin de développer une méthode hybride de discrimination s'appuyant sur les résultats théoriques produits par ailleurs.

Ainsi, nous verrons dans le chapitre 1 ce qu'est une SVM et plus précisément une M-SVM [3]. Puis dans le chapitre 2 nous présenterons le problème du traitement des séquences protéiques et comment l'adapter à une M-SVM (section 2.2). Nous verrons ensuite comment exploiter les alignements (section 2.3) et enfin nous présenterons les résultats obtenus par la SVM avec les alignements (section 2.5).

[2] le couplage de la M-SVM avec un HMM ne sera que brièvement abordé car ces travaux feront l'objet de 3 mois d'ingénieur que j'effectuerai dans l'équipe suite à mon stage

[3] M-SVM : Machine vecteurs support multiclasse

Chapitre 1

Contexte scientifique

1.1 Présentation

Une protéine est une macro-molécule biologique composée d'une suite d'acides animés. Cette suite d'acides aminés se replie en une structure tridimensionnelle (nommée structure tertiaire) qui va déterminer sa fonction. Des techniques comme la cristallographie par rayons X existent pour déterminer la structure des protéines. Cependant elles sont lentes et n'ont pas de garantie de résultat. A l'inverse, le nombre de séquences d'acides aminés connues augmente très rapidement si bien qu'un fossé se creuse entre le nombre de séquences connues et le nombre de structures connues [3]. Il y a donc un réel besoin de pouvoir prédire la structure spatiale des protéines, en partant de sa séquence d'acides aminés. Le problème de la prédiction de la structure tertiaire étant très complexe, on peut se ramener dans un premier temps à un objectif plus simple : prédire la structure secondaire. Cette structure secondaire est constituée par la succession d'éléments structuraux de trois types possibles appelés également états conformationnels : 2 structures périodiques (hélice alpha et brin bêta) et une structure apériodique (coïl).

L'équipe ABC développe une méthode de prédiction de la structure secondaire des protéines basée sur l'utilisation de machines à vecteurs support *multiclasses* (M-SVM). Il s'agit d'une architecture hybride et modulaire associant systèmes discriminants et génératifs (voir figure 1.1). Dans ce contexte, mon travail consiste à compléter la méthode de prédiction de l'actuelle M-SVM développée par l'équipe afin qu'elle puisse prendre en compte les alignements de protéines et qu'elle puisse être couplée à un HMM afin d'apporter plus de connaissances biologiques à la prédiction.

1.2 Machine à vecteurs support, le cas biclasse

Une machine à vecteur support (SVM pour "Support Vector Machine") est un couple constitué d'une classe de fonction et d'un algorithme d'apprentissage sur cette classe [6]. Une SVM apprenant à partir d'exemples étiquetés, il s'agit d'apprentissage supervisé. Il existe plusieurs modèles de SVM. Celles que nous

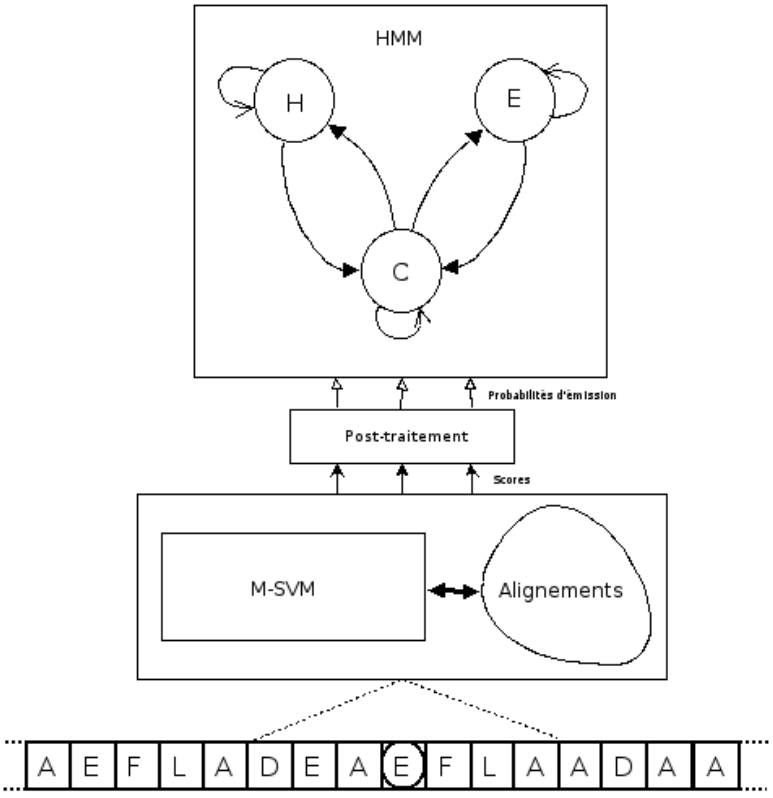

FIG. 1.1 – Architecture hybride mise en œuvre pour prédire la structure secondaire

considérerons ici sont les SVM biclasses et les SVM multi-classes (M-SVM) spécialité de l'équipe ABC.

Soit S un ensemble d'exemples étiquetés :

$$S = \{(\overrightarrow{x_l}, y_l)\}_{1 \leq l \leq p} \quad \forall l, y_l \in \{-1, 1\},$$

y_l étant donc l'étiquette associée à l'exemple $\overrightarrow{x_l}$

Notons que y ne peut prendre que deux valeurs, fixées ici à -1 et 1 car il s'agit d'une SVM biclasse. Ainsi, l'appartenance à une classe sera symbolisée par la valeur -1 ou 1 de l'étiquette y.

1.2.1 Le cas linéaire

Hyperplan et Séparabilité

Pour $\vec{x} \in \mathbb{R}^n$, on peut définir un séparateur linéaire de la forme :
$f_{\vec{w},b}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b$ où $f : \mathbb{R}^n \to \mathbb{R}$

Quand le résultat de cette fonction est positif, par convention on affecte au vecteur $\vec{x}$ la classe d'étiquette 1, et quand il est négatif, la classe d'étiquette -1. Lorsqu'il est nul, il convient de choisir comme convention l'affectation à l'une ou l'autre des classes.

On peut noter que $f_{\vec{w},b}(\vec{x}) = 0$ définit un hyperplan séparateur entre les deux classes d'exemples.

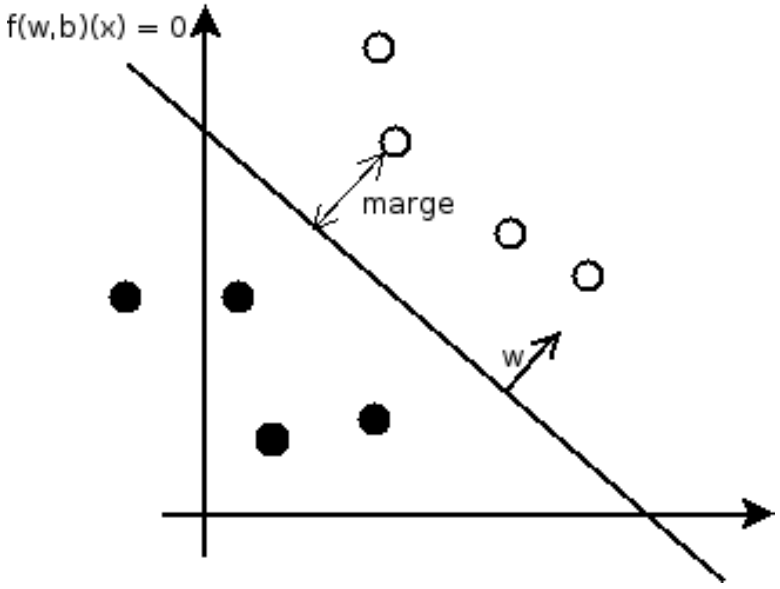

FIG. 1.2 – Hyperplan séparateur

Pour exprimer la notion de séparabilité linéaire, découpons l'ensemble des exemples étiquetés en deux sous ensembles :
$S^+ = \{\vec{x} : (\vec{x}, y) \in S \text{ et } y = 1\}$
$S^- = \{\vec{x} : (\vec{x}, y) \in S \text{ et } y = -1\}$
Ainsi si S est linéairement séparable :
$f_{\vec{w},b}(\vec{x}) > 0 \ \forall \vec{x} \in S^+$
$f_{\vec{w},b}(\vec{x}) < 0 \ \forall \vec{x} \in S^-$
Ces équations signifient que tous les exemples peuvent être séparés par $f_{\vec{w},b}$. Ceux d'une même catégorie seront du même côté de l'hyperplan séparateur. On peut illustrer cette notion dans le cas de données à 2 dimensions. Par exemple, prenons 4 points du plan. Donnons leur l'étiquette noire ou blanche en fonction de leur classe (noire : -1 et blanche : +1). L'hyperplan séparateur est une droite séparant les points blancs d'un côté et noirs de l'autre.

On voit bien que sur la figure 1.3, les points des deux catégories ne sont pas linéairement séparables (on ne peut pas tracer de droite les séparant). Les différentes solutions pour gérer le cas non séparable seront abordées par la suite.

Marge

La notion de marge est la clef des SVM. Il y a deux types de marges, une marge relative à chaque exemple, qui est la distance entre l'exemple et l'hyperplan séparateur et une marge relative au séparateur, qui est égale au minimum des marges relatives aux exemples.

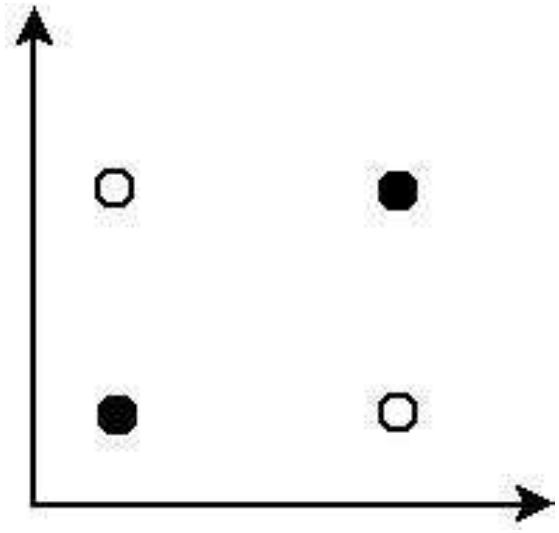

FIG. 1.3 – Cas non séparable linéairement

Dans les définitions qui vont suivre, il faut que les exemples soient biens classés, c'est-à-dire que tous les exemples d'une même classe soient d'un même côté de l'hyperplan et tous ceux de la classe opposée soient de l'autre côté. Nous avons expliqué au début de cette section que la contrainte de bon classement pouvait se traduire ainsi :

$$f_{\vec{w},b}(\vec{x}) > 0, \forall \vec{x} \in S^+$$
$$f_{\vec{w},b}(\vec{x}) < 0, \forall \vec{x} \in S^-,$$

Ce qui nous donne les deux types de contraintes suivantes :

$$\langle \vec{w}, \vec{x_l} \rangle + b > 0, \forall (\vec{x_l}, y_l) \in S \text{ où } y_l = 1$$

$$\langle \vec{w}, \vec{x_l} \rangle + b < 0, \forall (\vec{x_l}, y_l) \in S \text{ où } y_l = -1,$$

qui peuvent être fusionnés en un seul type de contrainte :

$$y_l(\langle \vec{w}, \vec{x_l} \rangle + b) > 0, \forall (\vec{x_l}, y_l) \in S$$

Ainsi, la marge peut s'écrire de la sorte :

DÉFINITION 1 (Marge d'un exemple, cas biclasse) *Soit $M^f_{(\vec{x},y)}$ la marge d'un exemple $(\vec{x}, y)$ associé au séparateur f, alors :*

$$M^f_{(\vec{x},y)} = \frac{y\left(\langle \vec{w_f}, \vec{x} \rangle + b_f\right)}{\|w_f\|} = \frac{|\langle \vec{w_f}, \vec{x} \rangle + b_f|}{\|w_f\|}$$

On peut donc définir la marge d'un séparateur comme :

Définition 2 (Marge d'un séparateur, cas biclasse) *Soit M_S^f La marge du séparateur f sur la base d'exemples S, alors :*

$$\begin{aligned} M_S^f &= \min_{(\vec{x},y)\in S} M_{(\vec{x},y)}^f \\ &= \min_{(\vec{x},y)\in S} \frac{y\left(\langle \vec{w_f}.\vec{x}\rangle + b_f\right)}{\|w_f\|} \end{aligned} \tag{1.1}$$

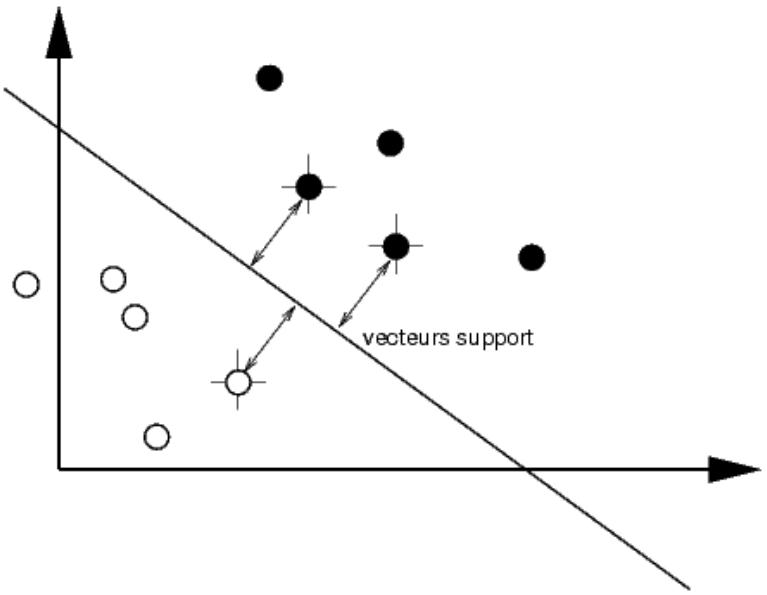

Fig. 1.4 – Vecteurs support d'une SVM

Lorsque les données sont linéairement séparables, il y a un nombre infini d'hyperplans séparant les exemples sans erreur. L'hyperplan qui nous intéresse va être celui qui a la marge la plus élevée. On se rend compte sur la figure 1.4 que si le séparateur est de marge maximale, il y a déjà au minimum deux exemples (l'un de chaque classe) dont la marge est la même que celle du séparateur. On appelle ces exemples les vecteurs support, car l'on peut retirer tous les autres exemples de la base d'apprentissage, seuls les vecteurs support vont spécifier la fonction calculée par la SVM (ici une droite).

Problème d'optimisation

Le but d'une SVM lors de la phase d'apprentissage est la maximisation de la marge, afin de séparer *au mieux* les exemples donnés. Pour cela, la SVM doit donc trouver les paramètres de l'hyperplan $\vec{w}$ et b qui maximisent la marge.

Ainsi l'hyperplan f^* de marge maximale peut s'exprimer :

$$f^* = \operatorname*{argmax}_f M_S^f$$

où M_S^f est donné par l'équation (1.1).

Il faut donc déterminer le couple $\vec{w}$ et b correspondant à l'hyperplan de marge maximale.

La *formulation canonique* d'un hyperplan séparateur s'écrit de la sorte :

$$\min_{(\vec{x},y)\in S} y\left(\langle \vec{w_f}, \vec{x}\rangle + b_f\right) = 1$$

Ceci est possible car l'hyperplan $f_{\vec{w},b}(\vec{x}) = 0$ est défini à un coefficient multiplicateur près. En effet, $f_{\vec{w},b}(\vec{x}) = \langle \vec{w}, \vec{x}\rangle + b = 0 \Rightarrow K f_{\vec{w},b}(\vec{x}) = 0$.

L'hyperplan étant exprimé sous forme canonique, sa marge est donnée par l'équation :

$$M_S^f = \frac{1}{\|w_f\|}$$

.

La recherche de l'hyperplan de marge maximale revient alors à trouver parmi les hyperplans exprimés sous forme canonique, f^* tel que la norme du vecteur orthogonal $\|w_f\|$ soit minimale. Il faut donc jouer sur $\vec{w}$ et sur b pour minimiser $\frac{1}{2}\langle \vec{w}, \vec{w}\rangle$. (Le 1/2 n'est présent que pour faciliter les futurs calculs).

Nous avons vu jusqu'ici quels étaient les critères à maximiser pour obtenir la marge maximale et comment exprimer la contrainte de bon classement par rapport à un hyperplan. Travaillant avec des hyperplans canoniques, $\min_{(\vec{x},y)\in S} |\langle \vec{w_f}.\vec{x}\rangle + b_f| = 1$. Ainsi la contrainte de bon classement peut se récrire de la manière suivante :

$$y_l(\langle \vec{w}, \vec{x_l}\rangle + b) \geq 1,\ \forall(\vec{x_l}, y_l) \in S$$

Le problème d'optimisation de la SVM est donc pour résumer :

Problème 1

$$\begin{aligned} \textit{Minimiser} \quad & \frac{1}{2}\langle \vec{w}, \vec{w}\rangle \\ s.c. \quad & y_l(\langle \vec{w}.\vec{x_l}\rangle + b) \geq 1,\ \forall(\vec{x_l}, y_l) \in S \end{aligned}$$

Données non linéairement séparables

En pratique, les données de S sont rarement linéairement séparables. La solution consiste alors à autoriser certains exemples à avoir une marge plus petite que 1, voire négative, c'est-à-dire autoriser les exemples mal classés. Pour ce faire, on assouplit la contrainte $y_l(\langle \vec{w}.\vec{x_l}\rangle + b) \geq 1,\ \forall(\vec{x_l}, y_l) \in S$ en introduisant des variable ξ_l. Ceci nous donne : $y_l(\langle \vec{w}.\vec{x_l}\rangle + b) \geq 1 - \xi_l,\ \forall(\vec{x_l}, y_l) \in S$ avec $\xi_l \geq 0$.

Ainsi, des exemples peuvent transgresser la limite symbolisée par l'hyperplan et se retrouver de l'autre côté. Cependant, si l'on se contente de modifier la contrainte sans contrepartie, il est alors possible de minimiser $\langle \vec{w}, \vec{w}\rangle$ jusqu'à le rendre arbitrairement petit (il suffit de prendre des ξ_l assez grand). Il faut donc que l'utilisation des ξ_l ait un coût sur la fonction à minimiser. Plus il y a d'exemples qui transgressent la frontière, plus il doit être difficile de minimiser

$\langle \vec{w}, \vec{w} \rangle$. Pour ce faire, il suffit d'ajouter les ξ_l à la fonction objectif, en tant que termes de pénalisation. De telles variables permettant d'assouplir les contraintes sont appelées des variables d'écart.

Avec l'ajout des variables d'écart on obtient :

Problème 2

$$\begin{aligned} &\textit{Minimiser} && \frac{1}{2}\langle \vec{w}, \vec{w} \rangle + C \sum_l \xi_l \qquad \textit{où } C > 0 \\ &\textit{s.c.} && \begin{cases} y_l(\langle \vec{w}, \vec{x_l} \rangle + b) \geq 1 - \xi_l, \ \forall (\vec{x_l}, y_l) \in S, \\ \xi_l \geq 0, \ \forall l \end{cases} \end{aligned}$$

où C est appelée la *constance de marge douce*. Elle permet de moduler l'influence des variables d'écart pour pouvoir garder un bon classement des données. En effet, plus C est élevée et plus l'impact d'un exemple mal classé sur la fonction objectif sera important et plus il sera difficile pour les exemples de violer les contraintes.

L'utilisation de tels paramètres est une procédure habituelle en programmation mathématiques. Cela rejoint un problème bien connu des personnes faisant de l'apprentissage : la recherche du compromis entre adaptation aux données et complexité (pour éviter le surapprentissage).

1.2.2 Le cas non linéaire

Jusqu'ici nous avons utilisé un séparateur linéaire. En ayant recours à des variables d'écart ξ_l afin de traiter les cas non linéairement séparables. Cependant, dans de nombreux cas, un tel séparateur n'est pas adapté.

En introduisant la notion de noyau et de séparateurs non linéaires, on obtient des SVM bien plus puissantes, sans remettre en cause ce qui a été établi dans le cas linéaire, sous réserve de respecter certaines règles.

Noyaux

L'idée qui fonde l'utilisation des noyaux est la suivante. Pour un ensemble de données S décrites dans un espace $\mathcal{X}$ de dimensions n, il peut être difficile de les séparer linéairement dans cet espace. Cependant, si l'on projette les données dans un espace de dimensions supérieur $n' > n$, il est souvent plus aisé de réaliser la séparation.

Définition 3 (Noyau symétrique semi-défini positif ou noyau de Mercer) *Un noyau de Mercer est une fonction qui réalise un produit scalaire dans un espace de dimension qu'il définit implicitement [7]. Soit κ un noyau, il existe alors une projection Φ telle que :*

$$\forall (x, x') \in \mathcal{X}^2, \kappa(x, x') = \langle \Phi(\vec{x}), \Phi(\vec{x'}) \rangle \tag{1.2}$$

Où,

$$\Phi(\vec{x}) = \begin{pmatrix} \phi_1(\vec{x}) \\ \phi_2(\vec{x}) \\ \vdots \\ \phi_{n'}(\vec{x}) \end{pmatrix}, \text{ où } 1 \leq n' \leq \infty$$

La manière intuitive de procéder serait de travailler directement sur l'ensemble des données *projetées* :

$$S' = \{(\Phi(\vec{x_l}), y_l)\}_{1 \leq l \leq p} \text{ avec } \forall l, y_l \in \{-1, 1\}$$

On peut alors essayer de séparer linéairement les exemples du nouvel ensemble S'.

Illustrons ceci avec l'exemple suivant : prenons des données vivant dans $\mathbb{R}^2$ et étiquetées $+$ ou $-$. Les exemples négatifs sont inscrits dans le cercle unité, et les positifs sont à l'extérieur (voir le schéma 1.5). Il est impossible de séparer linéairement ces données.

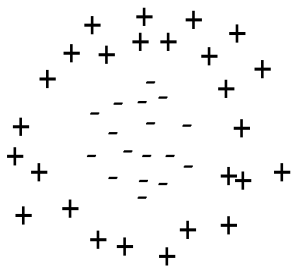

FIG. 1.5 – Données non linéairement séparables dans $\mathbb{R}^2$

Intéressons-nous alors à la projection Φ suivante :

$$\Phi(\vec{x}) = \begin{pmatrix} \phi_x(\vec{x}) = x \\ \phi_y(\vec{x}) = y \\ \phi_z(\vec{x}) = x^2 + y^2 \end{pmatrix}$$

On projette les données dans un espace de dimension 3 où $\vec{x}_z$ s'exprime comme une combinaison des deux autres composantes. Dans ce cas présent, il s'agit d'une équation de cercle. Ainsi, plus les exemples sont éloignés de l'origine dans $\mathbb{R}^2$ et plus $\vec{x}_z$ sera élevé.

On peut voir sur la figure 1.6 que l'on obtient une répartition en *volant de badminton* des données dans $\mathbb{R}^3$. Il est alors aisé de séparer linéairement les exemples avec un plan horizontal de hauteur $z = 1$.

Cependant l'utilisation des noyaux permet d'éviter de projeter les données de manière explicite. En effet, la projection est réalisée de manière totalement

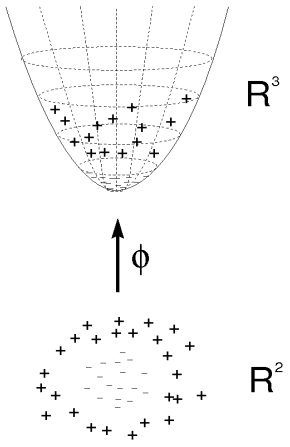

FIG. 1.6 – Projection dans un espace de dimension supérieure

implicite par le noyau. L'espace de projection pouvant être de dimension infinie, c'est un critère essentiel.

La manière de résoudre le problème 2 est de passer par sa forme duale. Nous ne détaillerons pas ce processus ici mais la chose à retenir est que, lors de la résolution du problème en passant par la forme duale, les données n'interviennent qu'au sein de produits scalaires. L'astuce des noyaux réside alors dans le fait qu'au lieu de projeter les données dans un espace de dimension n', de faire l'optimisation, et de revenir dans l'espace de dimensions n, on reste dans l'espace de dimension n en utilisant un noyau à la place de chaque produit scalaire dans la forme duale. Cette astuce est connue sous le nom de *kernel trick*.

Il va de soi que ne peuvent être noyau que des fonctions réalisant effectivement un produit scalaire dans un espace de hilbert. En fait, pour qu'une fonction puisse être qualifiée de noyau, il faut qu'elle respecte les conditions définies pas le théorème de Mercer [8, 7].

Prenons un exemple. Posons :

$$k(\vec{x}, \vec{x'}) = exp\left(-\frac{\|\vec{x} - \vec{x'}\|^2}{2\sigma}\right)$$

Cette fonction correspond au produit scalaire des projetés de $\vec{x}$ et $\vec{x'}$ dans un espace de dimension infinie. Elle se nomme le noyau gaussien.

On peut définir nos propres fonctions noyau (à condition qu'elles soient symétriques semi-définies positives) ceci dans le but d'adapter la SVM à un certain type de problème comme nous le verrons par la suite.

1.3 Machine à vecteurs support, le cas multiclasse

Le cas biclasse peut se généraliser au cas multiclasse en gardant certaines propriétés et résultats [9, 10].

L'ensemble des données sur lesquelles nous travaillons change quelque peu. Dans le cas biclasse, nous avions deux catégories, désormais nous en avons plus. Soit Q classes différentes et k une catégorie où $1 \leq k \leq Q$. L'ensemble des données peut s'écrire :

$$S = \{(\overrightarrow{x_l}, y_l)\}_{1 \leq l \leq p} \quad x_l \in \mathcal{X} \quad \forall l, y_l \in \{1, \ldots, Q\}$$

Le classifieur calcule des fonctions h qui sont des fonctions vectorielles telles qu'à chaque catégorie k est associée une fonction composante h_k. Ainsi, $h_k(\vec{x_i})$ retourne une note de l'exemple $\vec{x_i}$ pour la classe k.

La discrimination s'obtient de la manière suivante :

La M-SVM assigne $\vec{x_i}$ à la catégorie k tel que :

$$h_k(\vec{x_i}) = \underset{l}{\operatorname{argmax}}\, h_l(\vec{x_i})$$

Ainsi la classe de $\vec{x_i}$ est celle de la fonction composante qui lui donne le meilleur score. On peut noter qu'il y a donc autant de notes par exemple que de classes à prédire. Ceci se traduit par le problème d'optimisation suivant :

Problème 3

$$\begin{array}{ll} \min\limits_{w_k, b_k} & \dfrac{1}{2}\sum\limits_{k=1}^{Q}\langle \vec{w_k}, \vec{w_k}\rangle + C\sum\limits_{i=1}^{l}\sum\limits_{k \neq y_i}\xi_{ik} \\ s.c. & \begin{cases} \langle w_{y_i}, x_i\rangle + b_{y_i} \geq \langle w_k, x_i\rangle + b_k + 1 - \xi_{ik} \\ \xi_{ik} \geq 0, \qquad (1 \leq i \leq l)(1 \leq k \neq y_i \leq Q) \end{cases} \end{array}$$

$\sum_{i=1}^{l}\sum_{k \neq y_i}\xi_{ik}$ représente la somme sur tous les exemples $\{1, \ldots, l\}$ des variables d'écart pour chaque hyperplan.

Chapitre 2

Travail effectué

2.1 Problème posé

Nous avons expliqué précédemment qu'il y avait deux buts principaux à ce stage : utiliser les alignements multiples, et modifier les sorties de la SVM pour pouvoir ajouter un HMM en post-traitement. Utiliser les alignements peut signifier qu'au lieu de présenter une seule séquence protéique à la SVM, on présente en apprentissage les séquences homologues. Ainsi on ajoute des informations évolutives sur les données, avant même de les introduire dans la SVM. Cela peut également signifier qu'il faut traiter l'alignement de manière indirecte, non plus en les ajoutant dans la base d'apprentissage mais lors de la phase de prédiction, en regroupant les sorties provenant des séquences d'un même alignement. C'est cette option que nous avons retenue dans un premier temps pour le stage, la deuxième sera traitée par la suite lors de mes 3 mois d'ingénieur au sein de l'équipe.

Voyons maintenant comment adapter ce problème au cas de la SVM multiclasses.

2.2 M-SVM et structure secondaire

Une M-SVM est un couple formé d'une classe de fonctions et d'un algorithme d'apprentissage sur cette classe. Dans le cas de la prédiction de la structure secondaire des protéines, ce qui nous intéresse, c'est de prévoir l'état conformationnel (de quelle structure il s'agit) d'un résidu (acide aminé).

On voit apparaître un problème de discrimination à 3 catégories (hélice alpha, brin bêta et coïl) qui appelle donc l'utilisation d'une SVM multiclasses. Ainsi, lors de la phase de prédiction, chaque acide aminé présenté se verra attribuer 3 scores, un pour chaque classe (voir schéma 2.1).

Une protéine est une séquence d'acides aminés reliés entre eux par des liaisons chimiques. On pourrait imaginer une M-SVM qui prendrait en entrée un acide aminé de la séquence et qui essaierait de prédire l'état conformationnel associé. Cependant, un acide aminé seul ne fournit pratiquement aucune information sur la structure secondaire. C'est pourquoi, on augmente la dimension de l'espace

d'entrée de la M-SVM en lui présentant non pas un résidu, mais une suite de résidus, correspondant au contenu d'une fenêtre glissante sur la séquence de la protéine correspondante (fig 2.2). Le résidu pour lequel est effectué la prédiction est en général celui qui se trouve au centre de la fenêtre.

Il y a 20 acides aminés différents pouvant constituer une protéine, chacun représenté par une lettre de l'alphabet latin. Ajoutons à cela 1 place pour les acides aminés inconnus et une pour les vides (dont nous expliquerons plus bas l'utilité), cela nous donne un alphabet de 22 symboles.

Il faut ensuite déterminer la taille de la fenêtre d'entrée.

Comme on peut le voir sur la figure 2.2, la fenêtre d'observation est de taille $|W| = 2n + 1$ où n correspond à la longueur du contexte entourant le résidu central.

Revenons sur le cas de l'acide aminé *vide*. Le cas des résidus de début et de fin de chaîne pose problème car on utilise une fenêtre glissante sur la séquence d'acides aminés. En effet, lorsque la fenêtre est centrée sur le premier résidu de la protéine, que faut-il mettre dans les n positions de la fenêtre se trouvant au début ? La réponse est l'acide aminé *vide*. Ainsi, chaque protéine se voit complétée par n acides aminés vides à ses extrémités N-terminale (le début) et C-terminale (la fin).

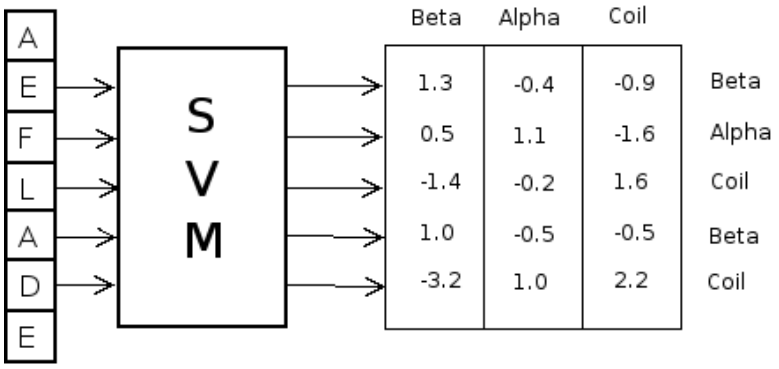

FIG. 2.1 – Notes et prédiction

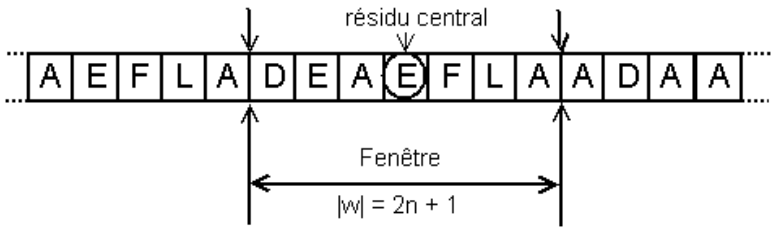

FIG. 2.2 – Fenêtre d'entrée

symbole	codage
a	00001
c	00010
d	00100
e	01000
¬	10000

TAB. 2.1 – Exemple de codage orthogonal

2.2.1 Codage générique

Maintenant que l'on sait comment adapter les entrées de la SVM au traitement des séquences protéiques, il faut préciser comment adapter les données à la SVM.

Une première manière de formater les données d'entrée pour la SVM est l'utilisation d'un codage orthogonal, permettant d'employer une SVM générique pour traiter le problème.

La notion de distance est au cœur des SVM car elles réalisent des produits scalaires (euclidiens) pour comparer les données.

Lorsque l'on travaille avec une SVM biclasse classique, avec des données représentant des points dans le plan, il n'y a pas de problème, on fournit en entrée deux réels qui représentent les coordonnées d'un point, et la classe associée. Ainsi, deux couples de deux réels aux valeurs proches auront plus de chance d'être dans la même classe que deux couples aux valeurs plus éloignées.

Maintenant regardons le problème avec les séquences d'acides aminés. Si l'on code simplement le A par 1, le C par 2, le D par 3, etc il y aura un problème au niveau de la notion de distance. En effet l'acide aminé A est peut-être beaucoup plus similaire à D que ne l'est C.

Le codage orthogonal propose de résoudre ce problème en rendant les acides aminés indépendants entre eux. L'idée est la suivante. Chaque acide aminé est codé sur un vecteur binaire de 22 bits, où tous les bits sont à 0 sauf celui de rang n où n est l'indice de l'acide aminé dans l'alphabet.

Prenons un exemple sur un alphabet de 5 lettres : $\alpha = \{a, c, d, e, \neg\}$. Alors chaque acide aminé, de même que le vide ¬ sera codé de la sorte (voir tableau 2.1).

Ainsi, $\langle a, a\rangle = 1$ et $\langle a.x\rangle = 0$, $\forall x \in \alpha \setminus \{a\}$.

Ce codage permet de capturer correctement la notion de différence mais pas celle de proximité. En effet $\langle a, b\rangle = 0$ et l'on ne peut rien tirer sur la dissimilarité entre a et b sans rajouter de connaissance biologique. Nous verrons dans la section suivante comment adapter la SVM au cas des séquences biologiques en ajoutant la notion de distance entre acides aminés.

Il faut maintenant adapter l'entrée de la SVM. La fenêtre glissante est de taille $|W| = 2n + 1$. Or, chaque acide aminé correspond à autant d'entrées dans la SVM qu'il y a de symboles dans l'alphabet.

Notons $l = |\alpha|$ le nombre de lettres de l'alphabet et $\mathcal{X}$ l'espace d'entrée alors :

$$dim(\mathcal{X}) = (2n+1).l$$

Ce codage orthogonal permet d'utiliser la distance de Hamming pour mesurer la similarité entre une fenêtres d'acides aminés et une autre. Le produit scalaire entre deux exemples (fenêtres) est égal à la somme des produits scalaires des acides aminés des deux fenêtres calculés position à position. En d'autres termes cela signifie que l'on ajoute 1 lorsque l'acide aminé d'indice i de la première fenêtre est le même que celui d'indice i de la deuxième.

2.2.2 Codage biologique

Le problème du codage orthogonal vient de l'utilisation du produit scalaire euclidien dans le noyau. D'une part ceci empêche de donner des entiers, un par acide aminé, en entrée de la SVM et d'autre part le produit scalaire ne renvoit que 0 ou 1. Comme expliqué précédemment, la notion de proximité n'est alors pas utilisée.

En ajoutant des connaissances biologiques dans le noyau, on retire le produit scalaire euclidien pour le remplacer par une notion de distance biologique. Un tableau fourni par les biologistes permet d'associer à chaque couple d'acide aminé présenté une distance ayant un sens biologique réel.

Le SVM résultante n'est alors plus générique mais est dédiée au traitement des séquences protéiques et elle capture cette fois-ci non seulement la différence mais également la proximité. Pour être plus précis, il s'agit du noyau de la SVM qui est alors dédié au traitement des séquences protéiques [4] [11]. La dimension de l'espace d'entrée reste alors : $dim(\chi) = |W| = 2n+1$

Il s'agit du type de codage qui sera employé par la suite dans la M-SVM autour de laquelle nous avons effectué nos modifications.

2.3 Utilisation des alignements multiples

Une des techniques pour améliorer la précision des algorithmes de prédiction de la structure secondaire est l'utilisation d'alignements multiples [5, 3]. Un alignement est un regroupement de séquences homologues (ayant une structure secondaire identique).

Résumons grossièrement la manière dont un biologiste va s'y prendre pour déterminer la structure tertiaire d'une séquence protéique. En premier lieu, il va regarder si la séquence est homologue d'une protéine dont la structure aura déjà été déterminée auparavant. Si c'est le cas, le travail s'arrête ici et il a trouvé la structure tertiaire correspondante. Si ce n'est pas le cas il va regrouper les séquences homologues à celle dont il veut prédire la structure tertiaire : il construit l'alignement correspondant. Il le soumet ensuite à un système de prédiction. Enfin,

[4] la spécification d'un noyau à un problème donné a été évoqué dans la section sur les noyau 1.2.2

s'il n'y a pas de séquences homologues, il soumet la séquence seule au système de prédiction.

Ainsi, on ne présente plus à la SVM une séquence d'acides aminés seule mais plusieurs similaires (celles de l'alignement). Les séquences similaires sont donc regroupées avant même que la SVM ne commence la prédiction. Ainsi il y a un travail d'enrichissement de l'information grâce à une source de données de nature évolutive. Ce n'est donc pas étonnant que les résultats de prédiction soient meilleurs car plus d'information est fournie à la SVM. En fait, les alignements de protéines homologues contiennent plus d'informations qu'une séquence seule car au cours de l'évolution la structure secondaire est beaucoup plus conservée que la séquence d'acides aminés [5]. Ainsi on peut identifier plus facilement les acides aminés qui jouent un rôle important dans la structure secondaire (au sein d'un même alignement, ils sont conservés d'une protéine homologue à l'autre).

Nous avons intégré en premier lieu les alignements en post-traitement de la SVM afin de ne pas avoir à modifier son noyau. Dans un premier temps, les alignements n'interviennent que dans la phase de prédiction, il faut les ajouter également dans la phase d'apprentissage.

Lorsque l'on veut prédire la structure secondaire d'une protéine (appelée protéine de *base*), on va constituer l'alignement. Il est alors utilisé pour la prédiction de la structure secondaire de la manière suivante :

1. Pour chaque protéine de l'alignement, on prédit sa structure secondaire.
2. On fait une moyenne pondérée des prédictions afin d'obtenir une *prédiction consensus*.

Plus précisément cela signifie que lors de la phase de prédiction d'une séquence, la fenêtre en cours de prédiction se voit attribuer trois scores, un pour chacune des 3 classes représentant les 3 structures possibles correspondant au résidu central. Lorsque l'on n'utilise pas les alignements, la classe associée est alors celle qui a le score le plus élevé. En utilisant les alignements, c'est un peu plus compliqué.

Soit une séquence protéique S. Soit n le nombre de séquences de l'alignement correspondant. Soit W la fenêtre centrée sur le résidu d'indice i de la séquence S dont on veut déterminer la structure secondaire.

Au lieu de présenter la fenêtre W de la séquence protéique, nous allons regarder dans l'alignement N et allons récupérer les séquences protéiques qu'il contient. Nous prenons alors n fenêtres, une sur chaque séquence protéique de l'alignement, toutes centrées sur le résidu d'indice i, on ne présente plus à la SVM la fenêtre centrée sur le résidu d'indice i de la protéine de base mais toutes celles issues de l'alignement.

Sans alignement, nous aurions avec un triplet de scores, un pour chaque classe possible du résidu central de la fenêtre sur la protéine de base. Cependant ici, comme nous avons présenté n fenêtres à la SVM, nous nous retrouvons avec n triplets de scores. Nous effectuons alors une moyenne pondérée afin d'obtenir un score par catégorie pour le résidu central. La catégorie choisie est de nouveau celle qui a le meilleur score.

Si l'on regarde l'exemple d'alignement du tableau 2.2. On se rend compte que

fenêtre	bêta	alpha	coil
EAATYYE	0.7	0.2	-0.9
E_ATYYE	0.2	0.6	-0.8
ECATYYE	0.3	0.9	-1.2
Σ	1.2	1.7	-2.9

TAB. 2.2 – Exemple de scores avec alignement

lorsque l'on donne à la SVM la séquence protéique EAATYYE seule, la structure bêta est celle qui a la meilleure note, ce sera donc celle qui sera prédite. Cependant, si on utilise les alignements, la structure ayant le meilleur score devient la structure alpha.

On peut noter la présence du caractère "_" qui est en fait un saut. Ces sauts sont dus aux mutations génétiques telles que les insertions ou les délétions.

2.3.1 Algorithme de pondération de Henikoff

Le choix de la pondération des protéines constituant l'alignement a un impact direct sur la prédiction. En effet, un des problèmes majeurs avec l'utilisation des alignements est le fait que certains groupes de séquences dominent très largement l'alignement. Sans algorithme de pondération particulier (pondération uniforme), ces groupes prédominants *masquent* les séquences les moins similaires aux autres. Cependant, l'information contenue dans ces séquences peu similaires doit avoir autant d'importance que celle des groupes prédominants. Prenons un exemple où quelques homologues forment un alignement. Puis avec les mutations génétiques, une séquence donne naissance à toute une famille d'homologues qui lui seront donc très similaires et qui seront alors classés dans le même alignement. Les autres séquences homologues ne doivent pas être pour autant moins considérées [12].

Pour résoudre ce problème, il faut donc donner un poids d'autant plus important à une séquence que celle-ci à peu d'homologues très similaires et inversement.

Une méthode très simple a été proposée par Henikoff [12] afin de s'assurer d'une telle pondération.

Soit n le nombre de protéines constituant l'alignement et n_j l'indice de séquence courante. Soit i l'indice de la colonne courante dans l'alignement. Elle contient donc n acides aminés. S'il y a m acides aminés identiques dans la colonne i à celui d'indice (n_j, i), on ajoute $1/(mn)$ au poids de la protéine n_j initialisé à 0.

Considérons que dans un alignement de 3 séquences protéiques, dans la première colonne, l'acide aminé de la séquence protéique 2 est un A. Admettons qu'il y ait également un A dans la première colonne de la première séquence, mais pas dans la troisième. Alors on ajoute $1/(mn) = 1/(2 \times 3)$.

On itère ainsi sur tous les acides aminés de toutes les séquences homologues.

On se rend alors compte sur des exemples très simples comme celui du tableau

Séquence	Henikoff
AAAAAA	0.1667
AAAAAA	0.1667
CCCCCC	0.1667
CCCCCC	0.1667
GGGGGG	0.3333
Σ	1

TAB. 2.3 – Pondération de Henikoff sur un exemple d'alignement simple

2.3, qu'une séquence qui apparaît deux fois moins se voit attribuer un poids deux fois plus élevé.

Ainsi, grâce à l'algorithme de Henikoff, on obtient une pondération qui ne va pas défavoriser les séquences peu similaires d'un alignement. Cependant, dans des cas réels, à savoir lorsque les séquences homologues ne sont pas constituées que d'un seul acide aminé identique mais d'une suite d'acides aminés codant la structure secondaire, la pondération de Henikoff ne garantit plus qu'une séquence peu similaire sera pondérée correctement. C'est pourquoi nous avons également mis en place une pondération maximisant l'entropie.

2.3.2 Maximisation de l'entropie

Avant tout, rappelons ce qui fait dans notre cas une bonne pondération. Il faut qu'une séquence protéique ait un poids fort si elle est peu similaire à ses homologues, et inversement. Il faut donc se rapprocher, en quelque sorte, d'une répartition uniforme des séquences protéiques en agissant sur leurs poids afin que l'information apportée par chaque protéine soit maximale. Il existe une notion permettant de mesurer à quel point une loi se rapproche d'une loi uniforme, il s'agit de l'*entropie*. L'entropie est maximale lorsque la loi est uniforme.

Nous avons vu que l'algorithme proposé par Henikoff est une bonne solution pour aborder le problème, cependant on peut aller plus loin, en s'assurant que l'entropie de l'alignement soit maximale.

Pour comprendre cela, il faut préciser la notion d'entropie. Par entropie nous entendons plus particulièrement l'entropie de Shannon.

DÉFINITION 4 (Entropie de Shannon) *Soit X une variable aléatoire discrète, avec n états possibles, son entropie de Shannon se définit comme :*

$$H(X) = -\sum_{i=1}^{n} p(i) \ln p(i)$$

Une manière habituelle tirée de la thermodynamique de définir pour définir l'entropie est de la considérer comme une mesure de quantification du désordre. Plus l'entropie est élevée et plus la distribution suit une loi uniforme. En effet, si le désordre est maximal, l'entropie est maximale également.

Shannon définit plus l'entropie comme une quantité d'information apportée. Plus l'incertitude du prochain tirage est grande, et plus il va apporter d'information. En outre, plus il existe de tirages différents, plus l'incertitude sera grande, et plus l'information apportée sera riche.

L'entropie de Shannon vérifie toutes ces propriétés et $H(X)$ est maximale lorsque X suit une loi uniforme. De plus, lorsque le nombre d'états possibles de X augmente, $H(X)$ augmente également.

Entropie et alignement

Avec l'entropie, nous avons un moyen de caractériser la redondance d'une source d'information. Cet outil peut être utilisé pour *évaluer* le vecteur de poids w associé à un alignement.

Considérons un alignement de n séquences protéiques de L acides aminés de longueur.

Prenons une colonne i au hasard. Elle est constituée de n acides aminés. Nous pouvons voir cette colonne comme le résultat de n tirages consécutifs parmi un alphabet α de $|\alpha| = 23$ symboles [5]. La fréquence pondérée de chaque acide aminé j dans la colonne i peut se définir comme :

$$p_{ij} = \frac{\sum_n w_n m_{ij}^n}{\sum_n w_n},$$

où m_{ij}^n est une fonction indicatrice qui vaut 1 lorsque l'acide aminé j est le même que celui de la protéine d'indice n à la colonne i, et 0 sinon.

p_{ij} est donc la probabilité d'avoir l'acide aminé j dans la colonne i. Cette probabilité dépend directement du vecteur de poids w associé à l'alignement et dont chaque composante w_n pondère une protéine.

Définition 5 (Entropie d'un alignement) *Soit $H_i(w)$ l'entropie d'une colonne de l'alignement :*

$$H_i(w) = -\sum_{j \in \alpha} p_{ij} \, ln \, p_{ij}$$

L'entropie d'un alignement est la somme des entropies de ses colonnes :

$$H(w) = \sum_{i=1}^{L} H_i(w)$$

Nous avons expliqué que la distribution uniforme est celle qui avait l'entropie la plus élevée. Ainsi en maximisant l'entropie, la distribution des probabilités p_{ij} sera aussi proche que possible d'une distribution uniforme. C'est exactement ce que nous souhaitons, à savoir, que chaque protéine de l'alignement ait autant d'impact que les autres sur la prédiction.

[5] 22 symboles pour les acides aminés, le vide, et l'inconnu plus un symbole pour prendre en compte les sauts.

De cette manière on peut définir le vecteur de poids w comme étant celui qui maximise l'entropie de l'alignement :

$$w^* = \underset{w}{\operatorname{argmax}}\, H(w)$$

Il se trouve que la fonction H est une fonction semi-concave. Cela signifie que tout maximum local est maximum global. De plus, lorsque les poids w_n sont normalisés, H devient une fonction concave, c'est-à-dire que tout maximum local est global et est unique.

En prenant $H_c = -H(w)$ on peut ramener ce problème à un problème de programmation convexe.

Problème 4

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{L} \sum_{j \in \alpha} p_{ij} \ln p_{ij} \\ s.c. \quad & \begin{cases} w_n \geq 0 & 1 \leq n \leq N \\ \sum_{n=1}^{N} w_n = 1 \end{cases} \end{aligned}$$

Résolution du problème

Afin de déterminer les poids optimaux maximisant l'entropie, il faut que nous minimisions une fonction convexe sous contraintes linéaires. La fonction étant convexe, un algorithme de descente en gradient suffit pour la minimiser, le tout en tenant compte des contraintes. C'est la raison pour laquelle nous avons utilisé l'algorithme du gradient projeté [13].

Lors d'une descente en gradient simple, à chaque itération, on calcule le gradient de la fonction. Il nous indique alors la direction de plus forte pente. Il suffit alors de se déplacer dans cette direction, dans le bon sens et d'un pas adapté [6]. Lorsque la norme du gradient est nulle, la pente est nulle, nous sommes au minimum de la fonction.

Maintenant, nous nous plaçons dans le cas sous contrainte. A chaque itération, on calcule toujours le gradient. Cependant, au lieu de se déplacer dans la direction indiquée par le gradient, on projette le vecteur de pente dans le sous espace vectoriel formé par les contraintes. On peut dire que l'on modifie la pente pour qu'elle respecte les contrainte. Ainsi, on descend la pente projetée qui représente donc une direction de descente satisfaisant les contraintes.

Soit d_{old} la direction de la pente pour un jeu de poids donné w_{old} :

$$d_{old} = \nabla H_c(w_{old})$$

[6] Dans notre cas il s'agit d'un pas variable.

Description et application de l'algorithme du gradient projeté

Pour calculer le gradient il faut calculer la dérivée partielle selon chaque composante :

$$\begin{aligned}\frac{\partial H_c}{\partial w_n} &= \frac{\partial \sum_{i=1}^{L} \sum_{j\in\alpha} p_{ij} \ln p_{ij}}{\partial w_n} \\ &= \sum_{i=1}^{L} \sum_{j\in\alpha} \left[\frac{\partial p_{ij}}{\partial w_n} \ln p_{ij} + p_{ij} \frac{\partial \ln p_{ij}}{\partial w_n} \right] \\ &= \sum_{i=1}^{L} \sum_{j\in\alpha} \left[(1 + \ln p_{ij})) \frac{\partial p_{ij}}{\partial w_n} \right],\end{aligned}$$

où

$$\begin{aligned}\frac{\partial p_{ij}}{\partial w_n} &= \frac{\partial \frac{\sum_n w_n m_{ij}^n}{\sum_n w_n}}{\partial w_n} \\ &= \frac{\left[\frac{\partial \sum_n w_n m_{ij}^n}{\partial w_n} \right] \sum_n w_n - \sum_n w_n m_{ij}^n \frac{\partial \sum_n w_n}{\partial w_n}}{\left(\sum_n w_n\right)^2} \\ &= \frac{m_{ij}^n \sum_n w_n - \sum_n w_n m_{ij}^n}{\left(\sum_n w_n\right)^2}.\end{aligned}$$

Une fois la valeur du gradient connue nous pouvons calculer le gradient projeté :

$$\overline{y} = -P^0 . d_{old},$$

où $P^0 = I - A^{0^T} \left[A^0 . A^{0^T} \right]^{-1} . A^0$

P^0 est la matrice de projection de d_{old} dans le sous espace des contraintes. La matrice A^0 est la matrice des contraintes saturées au point w_{old}. Une contrainte est dites saturée pour une solution admissible si la contrainte est vérifiée avec égalité. Ainsi, dans notre problème la contrainte de sommation à 1 étant une contrainte d'égalité, elle est constamment saturée. De plus, lorsqu'une des contraintes de positivité des poids w_n se vérifie avec l'égalité ($w_n = 0$), alors elle devient elle même saturée.

Soit A la matrice des contraintes :

$$A = \begin{pmatrix} -1 & & & & \\ & \ddots & & 0 & \\ & & \ddots & & \\ & 0 & & \ddots & \\ & & & & -1 \\ 1 & \dots & \dots & \dots & 1 \end{pmatrix}$$

La matrice A^0 est constituée des lignes de la matrice A dont les indices correspondent aux indices des contraintes saturées.

Une fois la matrice de projection calculée, on peut définir le nouveau jeu de poids w_{new} :

$$w_{new} = w_{old} + \beta\,\overline{y}.$$

α représente le pas de gradient. Il est déterminé dynamiquement en fonction des contraintes.

Soit β_{max} le pas maximum autorisé par les contraintes. Nous avons choisi de calculer le pas de gradient de la manière suivante : $\beta = min(\beta_{max}, s_{max})$ où s_{max} est le pas maximal fixé arbitrairement en fonction du problème.[7]

La descente s'arrête lorsque la norme du gradient est en dessous d'un certain seuil qui définit la précision de l'algorithme.

Ainsi, en minimisant H_c, on a trouvé les poids w^* qui maximisent H. Nous pondérons donc l'alignement avec ces poids et nous sommes certains que l'entropie sera maximale.

2.3.3 Alignements et apprentissage

Utiliser les alignements lors de la phase de prédiction permet d'améliorer le taux de reconnaissance des états conformationnels. Cependant la phase d'apprentissage peut elle aussi bénéficier de l'utilisation des alignements.

Pour chaque séquence protéique donnée en apprentissage, on peut trouver un ensemble de séquences homologues et faire apprendre la M-SVM sur cet ensemble. Ceci revient à augmenter la taille de l'ensemble d'apprentissage en y ajoutant des séquences alignées.

Une façon simple de procéder est d'ajouter les alignements dans la base d'apprentissage. Cela permet de ne pas changer la M-SVM qui va apprendre sur cette nouvelle base.

Cependant, en procédant de la sorte on ne prend plus en compte la pondération des alignements. En effet, lors de la phase de reconnaissance, il est possible de faire analyser plusieurs séquences et de faire une moyenne pondérée en sortie. Or lors de la phase d'apprentissage, chaque séquence est présentée à la M-SVM de manière indépendante et le seul moyen de pondérer plus une séquence est de la dupliquer dans la base afin de la présenter plusieurs fois.

[7] Dans notre cas, nous avons fixé $s_{max} = 0.1$.

Le fait que les alignements ne soient pas pondérés en apprentissage pose le même problème que lors de la phase de prédiction. Les séquences alignées très similaires vont avoir un impact plus important sur ce que va apprendre la M-SVM que les séquences plus rares.

Un moyen de palier ce problème, que nous n'avons pas encore mis en œuvre est d'avoir recours aux matrices BLOSUM [14]. Au lieu d'ajouter toutes les séquences alignées à la base d'apprentissage, nous allons *regrouper* les séquences les plus similaires et n'ajouter qu'un représentant de chaque groupe.

Prenons un alignement et fixons le taux de regroupement à 80%. Si deux séquences A et B sont similaires à plus de 80%, alors elles sont regroupées. Si une séquence C est similaire à plus de 80% à l'une des deux séquences A ou B (pas nécessairement les deux à la fois), alors elle est ajoutée dans le regroupement. On poursuit le regroupement jusqu'à ce que l'on ait analysé toutes les séquences. On obtient alors plusieurs groupes de séquences similaires. A partir de chacun de ces groupes nous construisons une séquence qui issue est des votes majoritaires de chacune des séquences constituant le groupe. Seule cette séquence est ajouté à la base d'apprentissage.

De cette manière, nous avons bien un représentant de chaque groupe de séquences similaires dans l'alignement et l'on évite le fait d'être confronté aux problèmes posés par un nombre important de séquences très proches.

2.4 Utilisation des profils

Une autre manière d'utiliser les informations apportées par les alignements est de travailler sur des profils.

Un profil est une séquence non plus d'acides aminés, mais de fréquences d'acides aminés. De cette manière, un profile peut contenir autant d'informations qu'un alignement, mais en une seule séquence :

Alignement
AAA
AAC
ACC
CCC

$\Bigg\}$

profil

$$\begin{bmatrix} \frac{3}{4}\text{A} & \frac{1}{2}\text{A} & \frac{1}{4}\text{A} \\ \frac{1}{4}\text{C} & \frac{1}{2}\text{C} & \frac{3}{4}\text{C} \end{bmatrix}$$

L'utilisation des profils permet également de tirer profit de la pondération calculée. En effet, les fréquences des acides aminés constituant le profil peuvent tout à fait être pondérée par les poids associés à l'alignement.

Cependant les profils ne sont pas directement utilisables par la M-SVM. En effet, sonn noyau ne sait pas manipuler des séquences de fréquences d'acides aminés. Il est dédié au traitement de séquences d'acides aminés "entiers" et elle ne sait pas calculer de distance entre deux fréquences d'acides aminés. Pour palier ce problème il faut modifier le noyau, afin que des produits scalaires entre séquences de fréquences aient un sens.

Une solution consiste à effectuer un produit cartésien des deux séquences de fréquences.

Exemple : prenons le profil P suivant $\begin{array}{cc} P_1 & P_2 \\ \left[\begin{array}{cc} \frac{3}{4}\mathrm{A} & \frac{1}{2}\mathrm{A} \\ \frac{1}{4}\mathrm{C} & \frac{1}{2}\mathrm{C} \end{array}\right. \end{array}$

Le produit scalaire de ce profil avec un autre (ici lui même), peut s'exprimer de la sorte :

$$\langle P, P\rangle = \langle P_1, P_1\rangle + \langle P_2, P_2\rangle$$

Jusqu'ici rien ne change par rapport aux séquences protéiques ordinaires, on effectue la somme des produits scalaires des deux acides aminés de même indice. Cependant, ici, les composantes P_1 et P_2 du profil ne sont pas des acides aminés mais des fréquences. Ainsi pour $\langle P_1, P_1\rangle$ par exemple, le calcul devient :

$$\langle P_1, P_1\rangle = \frac{3}{4}\times\frac{3}{4}\langle A, A\rangle + \frac{3}{4}\times\frac{1}{4}\langle A, C\rangle + \frac{1}{4}\times\frac{3}{4}\langle C, A\rangle + \frac{1}{4}\times\frac{1}{4}\langle C, C\rangle$$

Les produits scalaires $\langle A, A\rangle$, $\langle C, A\rangle$ et $\langle C, C\rangle$ sont définis grâce à une matrice de distance produite grâce à l'expertise de biologistes (voir section 2.2.2).

Il est important de noter que le noyau obtenu vérifie toujours les propriétés de Mercer. Le temps de calcul croît de manière quadratique en fonction du nombre d'acides aminés différents possibles.

2.5 Résultats expérimentaux

2.5.1 Protocole expérimental

Les tests ont été effectués sur un ensemble de 150 protéines découpé en 10 sous ensembles de 15 afin de réaliser une validation croisée. L'apprentissage s'est donc déroulé sur des ensembles de 135 protéines avec une taille de fenêtre glissante de 13.

Il faut distinguer deux cas d'apprentissage, sans alignements et avec les alignements en entrée.

Dans le cas de l'apprentissage sans utiliser les alignements, chaque base contenait environ 26000 exemples (fenêtres glissantes). Cependant, avec les alignements (mais sans la méthode de regroupement vue section 2.3.3) le nombre d'exemple est multiplié par 10, ce qui est tout à fait normal car ous avons limité arbitrairement le nombre de séquences alignées que pouvait contenir un alignement à 10.

Nous avons fait 4 séries de tests pour évaluer les performances des différentes améliorations et les gains associés. Un test pour évaluer la solution de base (sans alignements), un test avec alignement dans la phase de prédiction et pondération uniforme, un autre en améliorant la pondération (Henikoff) et enfin un dernier en ajoutant les alignements également dans la phase d'apprentissage.

Les taux de confiance associés aux gains entre chaque méthode présentée par la suite sont obtenus par la formule de test statistique suivante :

Soit u la variable telle que :

$$u = \frac{|f_1 - f_2|}{\sqrt{\frac{f_1(1-f_1)}{n_1} + \frac{f_2(1-f_2)}{n_2}}}$$

où f_1 et f_2 représentent les taux de reconnaissance respectivement du premier et du deuxième test, n_1 et n_2 la taille des échantillons.

Le taux de confiance s'obtient alors en consultant la table de la fonction de répartition de la loi normale réduite avec la valeur u. Soit n la valeur obtenue, alors le résultat est statistiquement significatif avec une confiance égale à : $100 - (100 - n) * 2$.

2.5.2 Résultats

Résultats de base : sans alignements

Le taux de reconnaissance sans alignement est de 65,04% (tab 2.4). Il se situe dans la moyenne des taux de reconnaissance sans alignement des réseaux de neurone.

Ensemble	Exemples apprentissage	Exemples test	taux reconnaissance
1	27143	1618	64.524%
2	26372	2389	68.606%
3	26002	2759	62.885%
4	26580	2181	63.320%
5	26345	2416	66.763%
6	25436	3325	64.090%
7	25554	3207	64.234%
8	24477	4284	66.597%
9	25363	3398	64.803%
10	25577	3184	64.447%
			$\mu = 65,04\%$ $\sigma = 3.10\%$

TAB. 2.4 – Performances de la M-SVM de base en validation croisée

Avec alignements dans la phase de prédiction

Pondération uniforme Le taux de reconnaissance avec les alignements et une pondération uniforme est de 67.30% (tab 2.5). Le gain est de 2,26% par rapport à la SVM de base et il est statistiquement significatif avec une confiance de plus de 99.99%. Les alignements apportent bien un supplément d'information permettant d'augmenter la précision. Cependant, l'absence de pondération limite le gain en augmentant de manière excessive l'impact des séquences protéiques similaires.

Ensemble	sans pondération	Henikoff	Henkoff & Alignements apprentissage
1	65.822%	66.316%	69.633%
2	71.411%	71.997%	72.290%
3	66.22%	67.198%	69.264%
4	66.392%	65.887%	68.776%
5	70.406%	69.826%	71.192%
6	63.489%	65.381%	66.496%
7	66.386%	66.856%	68.538%
8	67.507%	67.647%	68.184%
9	68.423%	68.685%	70.777%
10	67.242%	67.962%	70.834%
	$\mu = 67.3\%$ $\sigma = 5.26\%$	$\mu = 67.78\%$ $\sigma = 3.95\%$	$\mu = 69.56\%$ $\sigma = 2.91\%$

TAB. 2.5 – Exploiatation des alignements multiples avec une pondération uniforme, de Henikoff, et de Henikoff avec alignements dans l'apprentissage

Avec pondération de Henikoff Le taux de reconnaissance avec les alignements et la pondération de Henikoff est de 67,78% (tab 2.5). Le gain de cette première pondération est de 0,5% par rapport à des alignements non pondérés (statistiquement significatif avec une confiance de 78,14%). Ce premier résultat devrait être amélioré par la pondération maximisant l'entropie de l'alignement. De plus, pour des soucis de temps de calculs, nous avons limité le nombre de séquences d'un alignement à 10 ce qui diminue l'influence d'une bonne pondération par rapport à une pondération uniforme.

Avec alignements dans la phase de prédiction et dans la phase d'apprentissage

Le taux de reconnaissance avec les alignements dans la phase de reconnaissance pondérés avec Henikoff, le tout cumulé avec les alignements en apprentissage, dépasse les 69.55%. Le gain est de 1.78% (statistiquement significatif avec une confiance supérieure à 99,99%). L'ajout des alignements dans la phase d'apprentissage augmentant la taille de cette dernière et ajoutant des informations évolutives il est tout à fait normal qu'un tel gain soit réalisé.

Conclusion

Résumé

De nombreuses méthodes de l'apprentissage numérique ont été développées ces dernières décennies pour prédire la structure secondaire des protéines globulaires. Alors que les systèmes à base de réseaux de neurones se sont imposés en premier, d'autres méthodes ont vu le jour avec des résultats tout à faire prometteurs et parmi elles, notamment, les SVM multiclasses.

Ainsi, nous avons vu dans un premier temps ce qu'est une M-SVM et comment l'appliquer au problème du traitement des séquences protéiques.

Nous avons ensuite vu comment se décomposait l'architecture hybride proposée et notamment comment mettre en œuvre le premier *module* concernant l'utilisation des alignements, avec les problèmes de pondération associés. Nous avons pu tester cette nouvelle architecture avec les alignements et observer que l'augmentation globale du taux de reconnaissance dépassait les 4.5%.

Apports personnels

Ce stage m'a beaucoup apporté dans différents domaines à la fois. Déjà, au travers de l'étude de l'existant, il m'a permis de découvrir les SVM et de comprendre leur fonctionnement. De plus, le contexte appelant l'utilisation de connaissances mathématiques, j'ai pu découvrir sous un angle pratique les notions que j'avais vues lors de mes études. Ensuite, j'ai pu compléter un logiciel existant (la SVM de l'équipe) en développant des modules permettant d'en accroître l'efficacité comme les alignements. Enfin, ce stage m'a permis de toucher à de nombreux sujets afin de mettre en oeuvre ces modules : la théorie de l'information avec l'entropie, la programmation mathématique avec la minimisation d'une fonctionnelle sous contraintes, etc. Sans oublier bien sur la biologie qui est le contexte principal qui justifia l'architecture hybride sur laquelle j'ai travaillé.

Travaux en cours et perspectives de recherche

Ce travail se poursuit et appelle encore d'importants développements. Concernant les alignements, nous sommes actuellement en train de mettre en oeuvre le regroupement proposé dans la section 2.3.3. L'étape suivante sera d'utiliser des

profils à la place des alignements [3]. Cependant, cela demandera de modifier le noyau de la SVM pour qu'elle puisse travailler sur des fréquences d'acides aminés, nous interviendrons donc au coeur de la SVM. Enfin, une autre amélioration majeure est tout simplement le deuxième module, le HMM qui permettra comme nous l'avons expliqué au début de la section 2 d'ajouter plus de connaissances biologiques.

Bibliographie

[1] P.Y. Chou and G.D. Fasman. Empirical predictions of protein conformation. *Annu Rev Biochem*, 47 :251–276, 1978.

[2] N. Qian and T.J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202 :865–884, 1988.

[3] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, 232 :584–599, 1993.

[4] Y. Guermeur. *Combinaison de classifieurs statistiques, application à la prédiction de la structure secondaire des protéines*. PhD thesis, Université Paris 6, 1997. (in French).

[5] S. Riis and A. Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *J. Comput. Biol.*, 3 :163–183, 1996.

[6] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, June 1998.

[7] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25 :821–837, 1964.

[8] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London, Series A*, 209 :415–446, 1909.

[9] Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. A new multi-class SVM based on a uniform convergence result. In *IJCNN'00*, volume IV, pages 183–188, 2000.

[10] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

[11] Y. Guermeur, A. Lifchitz, and R. Vert. A kernel for protein secondary structure prediction. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, chapter 9, pages 193–206. The MIT Press, 2004.

[12] A. Krogh and G. Mitchison. Maximum Entropy Weighting of Aligned Sequences of Proteins of DNA. *Int. Sys. for Mol. Biol.*, pages 215–221, 1995.

[13] M. Minoux. *Programmation Mathématique, Théorie et algorithmes*. Dunod, 1983.

[14] S. Henikoff and J.G. Henikoff. Amino acid substitution from protein blocks. *Proc. Natl. Acad.*, 89 :10915–10919, 1992.