République Algérienne Démocratique et Populaire
Ministère De L'enseignement Supérieur Et De La Recherche Scientifique
Université Amar Télidji Laghouat

Faculté Des Sciences Et De L'ingénierie
Département D'informatique

Mémoire en vue de l'obtention du diplôme d'ingénieur d'état en informatique option intelligence artificielle

Thème :

La prédiction des structures secondaires des protéines à partir de leurs cartes d'acides aminés

Présenté par :
Lakhdari Abdellah
Ben Bahaz Mohamed Lamine

Encadré par :
M.Guellouma Younes

N° d'ordre :..../2010 - PFE / DGI

Dédicaces

A mes chers parents,

A mes frères et sœurs,

A ma fiancée,

A tous les membres de ma famille,

A tous mes amis,

A tous ceux que j'aime,

Mohamed,

A ceux qui ont attendu avec patience les fruits de leur bonne éducation...,

Mes parents ;

A tous ceux que j'aime,

A tous ceux qui m'aiment,

Abdellah,

On dédie ce modeste travail.

Remerciement

En premier lieu au ***DIEU tout-puissant*** *qui nous a donné la patience et la santé afin d'accomplir ce travail.*

On tient à exprimer aussi nos remerciements aux membres du jury, qui ont accepté d'évaluer notre travail.

Un grand merci aux enseignant qu'on avait l'honneur d'étudier chez eux toute au long les ans de notre formation et en particulier à ***M. YAAGOUBI*** ***M^{ed} Bachir*** *et* ***M.OUINTEN Youcef****. et également au biologiste* ***M.SIFI Brahim*** *qui nous a familier avec son domaine.*

On voudrait exprimer nos reconnaissances envers les amis et collègues qui nous ont apporté leurs supports moraux et intellectuels tout au long de notre démarche.

Merci a tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Finalement on adresse un grand merci à nos familles qui ont toujours été présentes lorsqu'on en a eu besoin, en particulier nos parents.

Résumé

Les protéines sont des acteurs majeurs du monde vivant. Elles sont un constituant essentiel des cellules et jouent un rôle crucial dans les processus biologiques nécessaires au fonctionnement de celles-ci. Prédire les structures protéiques est l'objectif de plusieurs chercheurs dû au secret porté par la miraculeuse molécule "la Protéine", sa structure, son mode de fonctionnement, son comportement...etc.

Connaitre et maîtriser cette structure est un souci majeur dans le but de répondre à d'importantes questions qui persistent depuis sa découverte jusqu'à l'heure actuelle. Les méthodes d'apprentissage utilisant les arbres de décision, plus proche voisin et d'autres méthodes statistiques ont pu donner des bons scores de prédiction. Nous allons développer une méthode fondée sur l'apprentissage automatique pour la prédiction de la structure 2D des protéines.

Mots clés : *Bioinformatique, protéines, prédiction, structure 2D, Knn, ID3, acides aminés.*

Abstract

The proteins are major actors in our world. They are a major constituent of cells and play a crucial role in biological processes needed to operate them. Predicting protein structures is the goal of many researchers due to the secret carried by the miraculous molecule "protein", its structure, its mode of operating, its behavior...

Know and control protein structures are major concerns in order to answer an important questions that remain from its discovery until now. The methods of machine-learning using decision trees, nearest neighbors and other statistical methods are able to give good prediction scores. We will develop a method based on machine learning to predict the 2D structure of proteins.

Keywords : *Bioinformatics, protein, prediction, 2D structure, Knn, ID3, amino acids.*

Table des matières

Table des figures

Introduction générale

Une protéine est une macromolécule constituée de longues chaînes d'acides aminés (les éléments de base) reliées par des liaisons peptidiques. C'est à partir de la séquence d'acides aminés que l'on détermine sa structure. Jusqu'à 1985, les seules méthodes permettant de déterminer la structure protéique étaient la diffraction des rayons X ou cristallographie et la spectroscopie par résonance magnétique nucléaire RMN, devenues alors très coûteuses et trop lentes. Le recours à des méthodes de la bioinformatique semblait être une bonne solution.

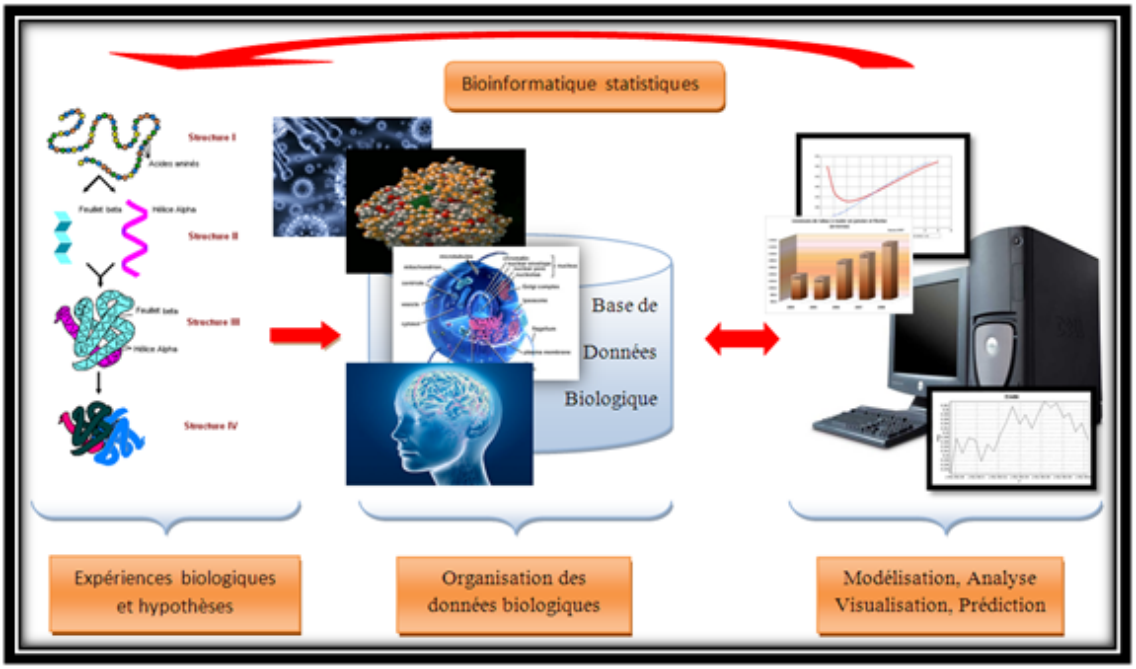

FIGURE 1 – Bioinformatique

Lorsque les hélices α et les feuillets β sont les structures 2D les plus fréquemment rencontrées, Néanmoins, d'autres structures régulières moins répandues existent : l'hélice 310, les coudes, les b coudes.

Alors, prédire la structure 2D revient à prédire les éléments conformationnels locaux hélices ALPHA, feuillets BETA.

Ce projet défie pour aider les biologistes, en remplaçant des méthodes de prédiction si complexe et moins fiable par un outil informatique en se basant sur les méthodes d'apprentissage supervisé ce qui fait faciliter les tâches et minimiser les durées et les couts dépensés.

Notre mémoire de fin d'études est structuré en quatre chapitres :

- Dans le premier chapitre on a présenté quelques définitions sur les protéines et la génétique inverse, en suite on a posé notre problématique.

- Le deuxième chapitre définit la bioinformatique, et les outils informatiques utilisés dans cette nouvelle discipline tel que les graphes, les dessins 3D et L'apprentissage supervisé.

- Le troisième chapitre est dédié à la conception, partant de la définition des besoins jusqu'à la définition et l'intégration des modules, passant bien sur par la conception.

- Le quatrième chapitre est consacré pour la présentation de notre outil **combio**.

Enfin, ce mémoire se termine par une conclusion générale mentionnant les connaissances acquises lors d'une telle étude et les perspectives envisageables.

Chapitre 1

Etat de connaissance

1.1 Introduction

Les protéines sont parmi les principaux types de molécules du vivant (les autres étant les acides nucléiques, les lipides et les glucides). Dans la cellule, elles assurent la grande majorité des fonctions enzymatiques, et une bonne part des fonctions de maintien de la structure et de transport, leurs étude fait une branche très importante dans la biologie. [1].

Dans ce qui suit on va définir les protéines, leurs structures et puis on montre une nouvelle stratégie d'études des protéines qui est la génétique inverse et on clôture ce chapitre par les problèmes et les contraintes rencontrées dans le domaine, ce qui fait appel à une intervention de l'informatique.

1.2 Les Protéines

1.2.1 Définition

Une protéine est une macromolécule biologique composée par une ou plusieurs chaîne(s) d'acides aminés liés entre eux par des liaisons peptidiques.

En général, on parle de protéine lorsque la chaîne contient plus de 50 acides aminés, pour des tailles plus petites, on parle de peptide et de polypeptide, mais plus souvent simplement de "petite protéine"[2].

1.2.2 Composition Des Protéines - Les Acides Aminés

Les protéines sont des polymères composés de 20 unités de base, les acides aminés (ou résidus). Ces derniers sont composés d'un atome carbone central (C_α) lié à un groupe aminé (NH2), à un groupe carboxylique (COOH), à un atome d'hydrogène (H_α), et à un des 20 groupements chimiques différents appelés chaînes latérales.

Les résidus et leurs fonctions chimiques différentes permettent la diversité fonctionnelle des protéines. Dans la chaîne polypeptidique, le groupe α-carboxylique d'un acide aminé est lié au groupe α-aminé de l'amino-acide suivant par une liaison amide (liaison peptidique -CO-NH-).

Les protéines sont des chaînes polypeptidiques fonctionnelles, et la plupart des protéines naturelles contiennent entre 50 et 2000 résidus d'acide aminé. La chaîne non ramifiée des résidus a une direction, elle commence à l'extrémité aminée (extrémité NH2 terminale) et se termine à l'extrémité carboxy terminale.

La chaîne d'atomes répétant régulièrement les liaisons peptidiques se nomme le squelette peptidique (ou carboné si seuls les C_α sont pris en compte).

La liaison peptidique est rigide et plane à cause du caractère partiel de double liaison de la liaison -CO-NH-, mais des rotations sont possibles autour des autres liaisons (C_α-CO et NH-C_α).

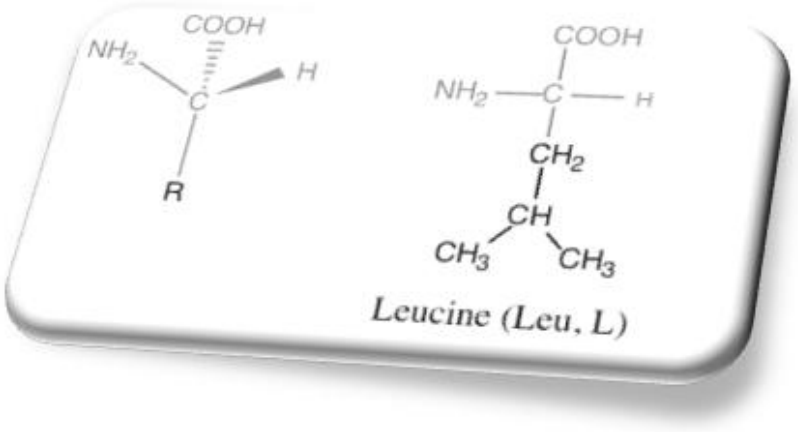

FIGURE 1.1 – *Gauche* : structure générale d'un acide aminé. *Droite* : formule chimique de la leucine

Les interactions entre atomes des chaînes latérales et celles entre atomes du squelette polypeptidique sont responsables de la structure des protéines.

Trois types de liaisons non covalentes dans les protéines sont observées : les liaisons hydrogènes, les liaisons de Van derWaals, et les liaisons électrostatiques.

La liaison hydrogène est formée lorsqu'un atome d'hydrogène est partagé entre un donneur d'hydrogène (l'atome lié covalemment à l'hydrogène) et un accepteur d'hydrogène. Les liaisons électrostatiques interviennent entre atomes chargés positivement et négativement.

Les liaisons de Van der Waals sont des liaisons de type dipôle induit-dipôle induit : une asymétrie passagère de charge autour d'un atome (possible parce que la distribution électronique de charge autour d'un atome évolue avec le temps) induit une asymétrie opposée dans un atome adjacent ; ils s'attirent alors mutuellement.

Les liaisons de Van der Waals sont plus faibles que les liaisons hydrogène ou électrostatiques, mais elles sont efficaces en grand nombre.

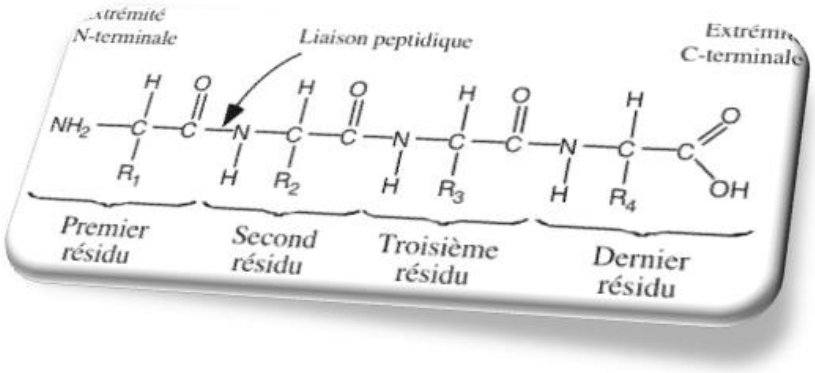

FIGURE 1.2 – Un polypeptide formé de 4 acides aminés (tétrapeptide)

D'autre part, lorsqu'une molécule non polaire est introduite dans un milieu aqueux, le réseau de liaisons hydrogène des molécules d'eau est perturbé : les molécules d'eau forment une "cage" autour de la molécule non polaire et moins de liaisons hydrogène sont donc possibles, d'où une diminution de l'entropie.

Si plusieurs molécules non polaires sont introduites, elles auront tendance à s'agréger car la surface de la cage de l'agrégat est inférieure à la somme des surfaces des cages des molécules isolées. La diminution de l'entropie est alors moindre.

Ces interactions hydrophobes sont donc en fait des forces indirectes qui favorisent le regroupement de molécules non polaires suite à des interactions entre des molécules polaires. Avec les liaisons hydrogène de la chaîne polypeptidique, les liaisons de Van der Waals et les interactions hydrophobes sont les principales responsables du repliement des protéines.

L'encombrement stérique joue également un rôle dans la structure des protéines car lorsque la chaîne protéique s'enroule, une chaîne latérale encombrante peut empêcher une courbure forte. La répulsion des atomes est provoquée par le recouvrement des nuages électroniques lorsque les atomes se rapprochent.

Les liaisons covalentes inter-résidus - sous forme de ponts disulfure entre deux résidus cystéine - permettent une meilleure stabilité pour les protéines qui sortent de l'environnement cellulaire [1].

Nom de l'acide aminé	**Code à 3 lettres**	**Code à 1 lettre**
Leucine	Leu	L
Glycine	Gly	G
Alanine	Ala	A
Isoleucine	Ile	I
Valine	Val	V
Sérine	Ser	S
Thréonine	Thr	T
Phénylalanine	Phe	F
Tryptophane	Trp	W
Tyrosine	Tyr	Y
Proline	Pro	P
Cystéine	Cys	C
Méthionine	Met	M
Asparagine	Asn	N
Glutamine	Gln	Q
Arginine	Arg	R
Lysine	Lys	K
Histidine	His	H
Acide Aspartique	Asp	D
Acide Glutamique	Glu	E

FIGURE 1.3 – Liste des 20 acides aminés

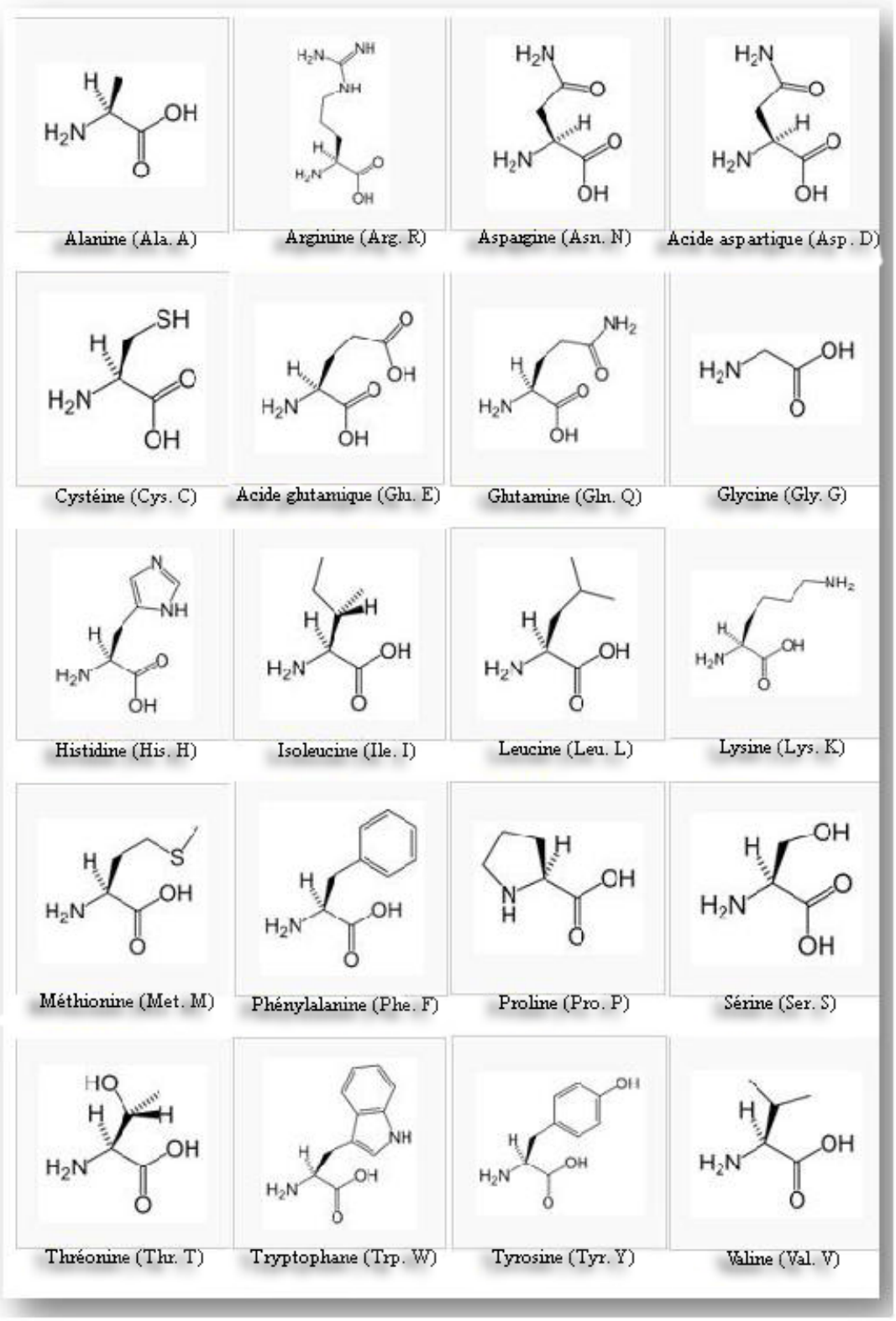

FIGURE 1.4 – Structure des 20 acides aminés standards représentés dans le code génétique [3]

1.2.3 Différentes Structures

La structure des protéines, illustrée en Figure 1.5, peut être définie selon quatre niveaux :

- **La structure primaire (I)** qui correspond à la séquence proprement dite de la protéine, c'est-à-dire à l'enchaînement des acides aminés.

- **La structure secondaire (II)** qui décrit le premier niveau de repliement de l'enchaînement peptidique. Elle est assurée par des interactions de type liaisons hydrogène entre les acides aminés voisins et se caractérise par une conformation répétitive localisée. Il existe deux types de structure secondaire : l'hélice α et le feuillet β.

- **La structure tertiaire (III)** correspond au repliement et à l'assemblage en domaine de différents éléments de la structure secondaire. Il s'agit de la structure tridimensionnelle de la protéine. Elle est stabilisée par des interactions hydrophobes, des liaisons hydrogène, des liaisons ioniques ou encore des ponts disulfures (S-S), véritable liaison covalente formée par oxydation entre les fonctions thiols de deux cystéines.

- **La structure quaternaire (IV)** concerne les protéines constituées de plusieurs chaînes polypeptidiques et correspond à l'association de celles-ci par des interactions semblables à celles établies à l'intérieur d'une chaîne polypeptidique, c'est-à-dire des interactions hydrophobes, des liaisons hydrogène et dans certains cas, des ponts disulfures [4].

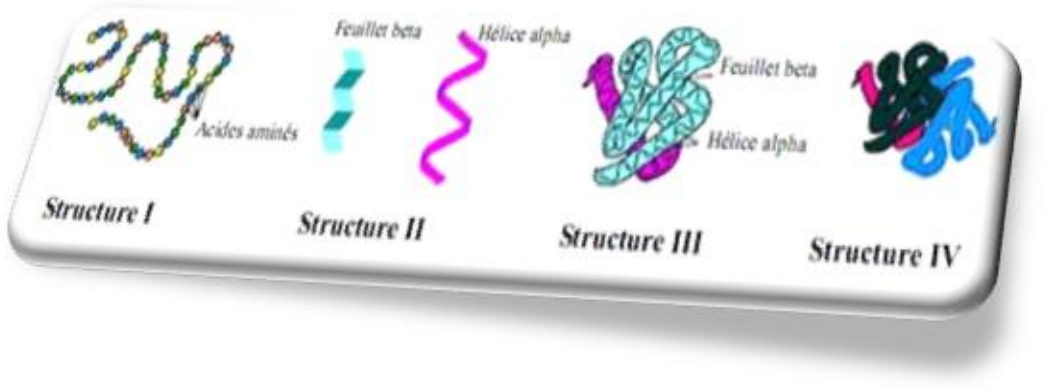

FIGURE 1.5 – Les quarte niveaux de structures des protéines

1.3 La génétique inverse

La génétique inverse définit les techniques qui permettent, à partir d'un gène ou fragment d'ADN, l'étude des fonctions de ce gène et de ses produits, par opposition à la génétique classique dont le but est de localiser le gène responsable de l'altération d'une fonction ou d'un caractère connus.

Appliquée aux maladies héréditaires, elle désigne l'identification directe de gènes impliqués dans des maladies héréditaires dont la protéine défectueuse est inconnue.

La stratégie consiste à isoler des séquences clonées de DNA génomique impliqué dans une pathologie héréditaire.

L'information y est ensuite recherchée sous forme de séquences nucléotidiques codantes d'où l'on déduit la séquence protéique [19].

Apres avoir généré les protéines ; des études comparatives sont faites alors, pour aboutir à la protéine défectueuse.

Les études qu'on a parlées au dessus, se basent sur des banques protéiques élaborées expérimentalement, elles contiennent des jeux de protéines bien décrites où on connait leurs structures et leurs fonctions. On les consulte, on compare avec la séquence obtenue et on prend la décision concernant cette nouvelle protéine.

1.4 Problématique

De ce qu'on a vu précédemment ; l'importance du domaine des protéines et la stratégie de la génétique inverse et plus précisément en se focalisant sur les études comparatives et les banques protéique, on peut toucher bien les difficultés rencontrées dans ces tâches.

Les expériences faites pour l'élaboration de la banque protéique et la reconnaissance des fonctions et structures de chaque protéine sont parfois très longues et couteuses, on peut citer quelques unes telles que *la Radiocristallographie aux rayons X* et *la RMN*....

Combien d'expériences ont été faites ? Comment faire la comparaison inter protéines ?....

Ces limites et ces contraintes offrent une bonne occasion pour l'introduction de l'informatique en biologie ce qui mène a la naissance d'une nouvelle discipline hybride appelée 'la bioinformatique'.

1.5 Conclusion

L'étude des protéines en biologie et l'appel des outils informatiques pour l'assistance fait apparaitre la bioinformatique, la nouvelle science qu'on va la détailler dans le prochain chapitre.

Chapitre 2

Bioinformatique

2.1 Introduction

L'informatique offre une panoplie toujours plus riche et complexe d'outils pour organiser, gérer et traiter la masse et la diversité des connaissances biologiques est aussi à l'origine de l'explosion des données disponibles. C'est de cette dualité qu'un domaine nouveau a vu le jour sous l'appellation de bioinformatique dont la maîtrise requier une double compétence Biologie-Informatique.

2.2 La Bioinformatique

2.2.1 Définition

Depuis une dizaine d'années, le rythme d'acquisition des données dans tous les domaines de la biologie s'accroît de manière spectaculaire. C'est particulièrement le cas en biologie moléculaire avec les grands programmes de séquençage, ainsi que dans les domaines où la modélisation et l'imagerie prennent une place essentielle. Parallèlement, l'informatique a fait une entrée définitive dans les laboratoires de recherche, les unités de production et de distribution où elle joue maintenant un rôle clé dans la vie de l'entreprise [5].

La bioinformation est l'information liée aux molécules biologiques : leurs structures, leurs fonctions, leurs liens de "parenté", leurs interactions et leur intégration dans la cellule.

Divers domaines d'études permettent d'obtenir cette bioinformation : la génomique structurale, la génomique fonctionnelle, la protéomique, la détermination de la structure spatiale des molécules biologiques, la modélisation moléculaire ...

La bioinformatique est l'analyse de la bioinformation.

C'est une discipline récente (quelques dizaines d'années), "hybride" (au même titre que la biochimie ou la biophysique) : elle est fondée sur des concepts et des formalismes issus de la biologie, de l'informatique, des mathématiques et de la physique. C'est une discipline qui utilise toutes les potentialités de traitement de l'informatique : modèles théoriques, algorithmes et programmes, ordinateurs, réseau Internet, bases de données ...

2.2.2 Les axes privilégies de la bioinformatique

- La formalisation de l'information génétique.
- L'analyse des séquences (biomolécules) et de leurs structures.
- L'interprétation biologique de l'information génétique.
- L'intégration des données (établissement de cartes et de réseaux d'interactions géniques, d'interactions protéiques ...).
- La prédiction fonctionnelle.

2.2.3 Les Méthodes De La Bioinformatique

- **Méthode comparative :** comparaison des séquences ou structures inconnues avec les bases de données (séquences et structures) de gènes et de protéines connues pour établir des rapprochements (similarités, homologies ou identités).
- **Méthode statistique :** Des logiciels appliquent des analyses statistiques aux données (sur la syntaxe des séquences) pour tenter de dégager et de repérer des règles et des contraintes présentant un caractère systématique, régulier ou général.
- **Approche par modélisation :** c'est une approche probabiliste, elle consiste à étudier les objets (ex. : séquences, structures, motifs,...) à

travers la construction d'un modèle qui tente d'en extraire les propriétés communes. La relation entre les objets d'étude (et/ou leur reconnaissance) est alors exprimée en référence à ce modèle optimal commun.

2.2.4 La Mission Du Bioinformaticien

Leur formation première peut être la biologie, les mathématiques ou l'informatique et ils ont suivi une formation complémentaire dans "l'autre domaine". Leur fonction est :

- mettre en œuvre les méthodes et en automatiser le traitement.
- nourrir la recherche en biologie des résultats de ces analyses.
- réduire les problèmes posés par les biologistes en termes accessibles aux mathématiciens et informaticiens.
- participer à l'élaboration de nouvelles méthodes.

2.3 L'informatique

Dû à la potentialité des outils informatiques, plusieurs techniques peuvent être utilisées dans la bioinformatique pour assister et aider à résoudre ses problèmes, les plus répondues sont :

2.3.1 Les graphes

La théorie des graphes est une matière aujourd'hui étudiée et développée tant du point de vue de l'informatique (algorithmique, Base de données, réseau...) que des mathématiques (optimisation combinatoire, probabilité, algèbre...). Les méthodes développées pour étudier les objets de cette théorie et leurs interactions ont de nombreuses applications dans tous les domaines liés à la notion de réseau (réseau social, réseau informatique, Télécom...) et dans bien d'autres domaines ; telles que la chimie, la biologie, les sciences sociales.

De manière générale, un graphe permet de représenter la structure, les connexions d'un ensemble complexe en exprimant les relations entre ses éléments. Les graphes constituent donc une méthode de pensée qui permet de

modéliser une grande variété de problèmes en se ramenant à l'étude de sommets et d'arcs, les deux composants de graphe.

2.3.2 Les Dessins trois dimensions

Trois dimensions ou tridimensionnel ou 3D (prononcer "trois D") sont des expressions qui caractérisent l'espace qui nous entoure, tel que perçu par notre vision, en termes de largeur, hauteur et profondeur.

En informatique, les modèles tridimensionnels (figures ou images de synthèse) nécessitent des calculs sans complexité particulière, mais extrêmement nombreux. Ils peuvent être représentés, soit par des perspectives de diverses directions sur un écran en deux dimensions.

L'évolution stupéfiante qu'a vu l'informatique en ses deux grandes aspects matériel (les cartes graphiques, les GPU ...), logiciel (traitement d'images, synthèse d'images ...) a démocratisé l'utilisation de 3D et la rendu pluridisciplinaire ; a nos jours on trouve 3D dans l'informatique ,la mécanique , la chimie , l'architecture, la médecine et la biologie ...etc. comme un outil pour modéliser ,simuler et représenter toutes les formes complexes a réaliser tels que les bâtis dans l'architecture et les moteur dans la mécanique, et pour la compréhension inductive des molécules dans la chimie, et les micro corps et organes dans la médecine et la biologie.

De ce fait, et en particulier la projection sur la biologie va nous mettre bien placé dans la bioinformatique ; elle utilise alors ces outils pour modéliser des phénomènes biologiques, des structures ou des systèmes Assi complexe, en les rendant si simple, bien lisible et bien compréhensible ce qui définit le but de cette discipline hybride.

2.3.3 L'apprentissage supervisé et ses techniques

Dés que l'outil proposé s'appuie sur les méthodes de l'apprentissage automatique, cette technique a été bien prise en compte dans sa description.

Définition

L'apprentissage automatique (machine learning en anglais) est l'un des sous-domaines de l'intelligence artificielle. Il a pour objectif d'extraire et d'ex-

ploiter automatiquement l'information présente dans un jeu de données. En cela il couvre un vaste champ d'objectifs comme la fouille de données, la classification non-supervisée, la sélection de variables, la discrimination, la régression, la sélection de modèle, la génération et l'inférence de règles, etc.

Il s'avère également être fortement pluridisciplinaire puisque selon les données et les objectifs, l'apprentissage automatique fait appel à l'informatique, aux neurosciences, au traitement du signal, aux sciences cognitives, à la théorie de l'information, à la biologie, aux statistiques, ...

L'apprentissage automatique, dans une définition très générale, consiste en l'élaboration de programmes qui s'améliorent avec l'expérience. Les applications sont nombreuses et concernent des domaines très variés. On peut citer, par exemple, la reconnaissance de formes avec, en particulier, la reconnaissance de la parole et du texte écrit, le contrôle de processus et le diagnostic de pannes, les programmes de jeu.

Les méthodes d'apprentissage à partir d'exemples sont très utilisées dans la recherche d'informations dans de grands ensembles de données. En effet, l'évolution de l'informatique permet de nos jours de manipuler des ensembles de données de très grande taille (datawarehouse) ou (entrepôt de données). Par exemple, les chaînes de magasin peuvent mémoriser de grandes quantités de données concernant les consommateurs et ce qu'ils achètent.

Le développement des technologies Internet et Intranet font que de nombreuses données issues de sources diverses et dans des formats variés deviennent accessibles.

Le processus de recherche d'informations dans de grands ensembles de données (KDD : Knowledge Discovery in Databases) comporte différentes étapes : la sélection des données (extraction des informations de l'entrepôt) ; la préparation des données (suppression des doublons, élimination des données aberrantes, ...) ; le codage des données (normalisation des données, choix de codage, ...) ; la phase d'extraction proprement dite appelée fouille de données (Data mining) ; la sortie des résultats.

La phase d'extraction d'information utilise les outils usuels d'interrogation tels que les requêtes SQL standard et les requêtes multidimensionnelles, mais aussi, pour l'extraction d'informations cachées, les algorithmes d'apprentissage à partir d'exemples.

Les algorithmes les plus utilisés sont : les k-plus proches voisins, les arbres de décision, les systèmes de règles (programmation logique inductive), les réseaux de neurones et les algorithmes génétiques. Citons, parmi d'autres, quelques applications et leur domaine [6] :

- Analyse financière : prévision d'évolution de marchés ;
- Marketing : établir un profil client, mailing (augmenter les taux de retour) ;
- Banque : attribution de prêts ;
- Médecine : aide au diagnostic ;
- Telecom : détection de fraudes.

L'apprentissage supervisé est une technique d'apprentissage automatique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage contenant des exemples de cas déjà traités (on connait la classe dont ils appartiennent) [6].

L'apprentissage supervisé consiste à extraire une fonction d'apprentissage depuis une *base d'apprentissage*. Cette base d'apprentissage est composée d'objets (typiquement des vecteurs) formant l'entrée du processus d'apprentissage, et réclament ainsi une sortie. Cette sortie peut être une valeur continue (on parle donc d'une *régression*, ou peut prédire la classe d'un nouvel objet en entrée de la fonction formée (on parle donc de la classification) [7].

Afin de résoudre un problème donné d'apprentissage supervisé, plusieurs étapes sont effectuées (voir Figure 2.1) :

1. Déterminer le type d'objets de la base d'apprentissage : avant de procéder à n'importe quelle étape, la décision du type de données utilisées doit être prise ;
2. Définir la base d'apprentissage : chaque objet de la base d'apprentissage doit être étiqueté par un expert ou par une mesure et chaque étiquette doit décrire une seule classe ;
3. Trouver une bonne représentation des éléments de la base d'apprentissage : la représentation la plus utilisée est la représentation vectorielle. Néanmoins, il existe d'autres formes de représentations ;
4. Déterminer la structure de la fonction de classification ainsi que le choix adéquat de l'algorithme de classification ;
5. Finaliser l'algorithme de classification : l'algorithme doit être testé par une *base de test* ou une *base de validation*.

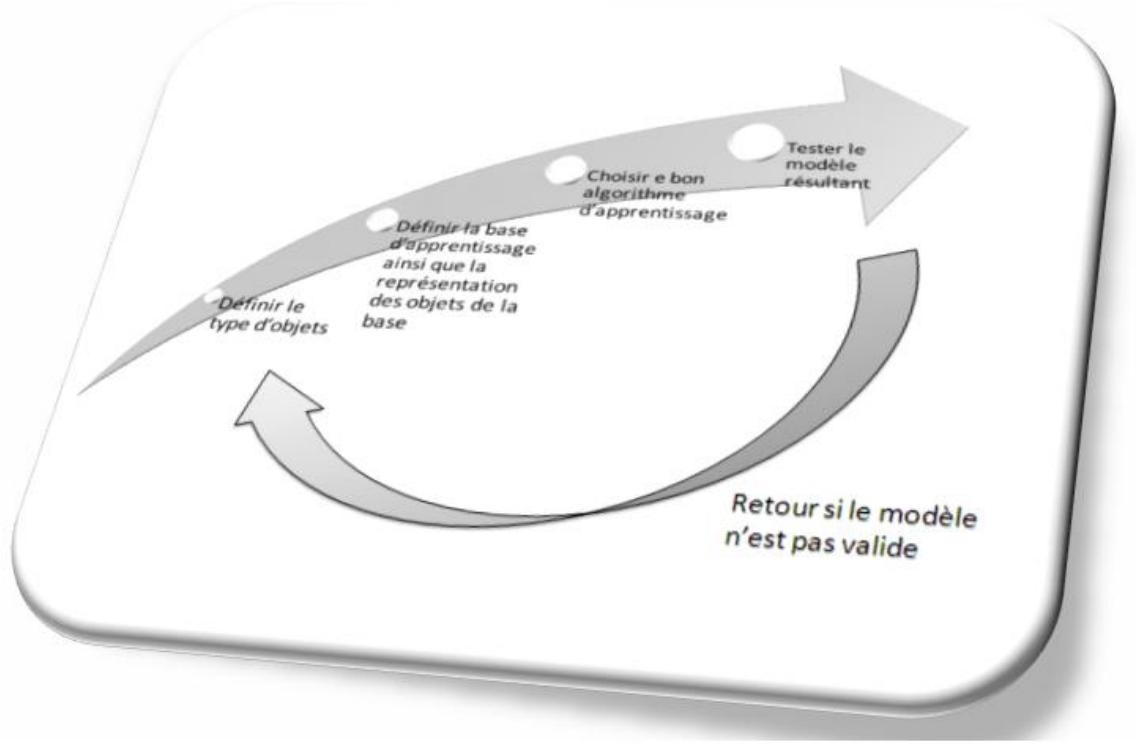

FIGURE 2.1 – Processus de l'apprentissage supervisé

Si on prend ϵ la base d'apprentissage composée d'un ensemble d'objets X et d'un ensemble d'étiquettes Y définissant les classes, dont chaque $x_i \in X$ est associé à une classe $y_i \in Y$:

$$\epsilon = \{(x, y), x \in X, y \in Y)\}$$

La fonction d'apprentissage est définie comme suit :

$$\begin{aligned} h : X &\rightarrow Y \\ x &\mapsto h(x) \end{aligned}$$

La fonction d'apprentissage h est telle que $h(x_i) = y_i$ si pour tout $j \neq i$ on a :

$$P((h(x) = y_i)/x) > P((h(x_i) = y_j)/x)$$

D'où

$$h(x) = \arg\max(Y)/P((Y = y))/x)$$

a - Estimation d'erreur et validation de l'apprentissage

L'apprentissage supervisé comporte deux partie, la première consiste en la construction d'un modèle d'apprentissage, c'est en faite la définition de la fonction h à partir d'un ensemble d'informations dites "base d'apprentissage", qui se compose d'exemples déjà traités (leurs classes sont connues au préalable).

La deuxième étape consiste à utiliser le modèle afin de définir leur nature après avoir testé la précision du modèle à travers des tests sur des individus déjà classés.

Afin de tester le modèle, une base de validation

$$V = \{(x', y') \mid x' \in X, y' \in Y)\}$$

Est définie. Ensuite, à chaque x_i on associe une étiquette $y_i \in Y$ grâce à la fonction h. la validation du modèle est faite donc moyennant une fonction d'estimation d'erreur E. Pour cette raison, on défini une fonction de perte élémentaire définie comme suit [8] :

$$\begin{array}{rcl} l : Y \times Y & \rightarrow & \{0, 1\} \\ (y, y') & \mapsto & \begin{cases} 0 \text{ si } y' = y \\ 1 \text{ sinon.} \end{cases} \end{array}$$

Alors, la fonction d'estimation de taux d'erreur E peut être calculée comme suit :

$$E(h) = \frac{1}{|V|} \sum_{(x', y') \in V} l(h(x'), y')$$

Soit H l'ensemble des fonctions d'apprentissage. Un modèle basé sur une fonction d'apprentissage $\widetilde{h}$ est valide s'il satisfait :

$$E(\widetilde{h}) = \arg\min_{h \in H}(E(h))$$

La phase de validation qui représente le critère d'acceptation de la fonction h peut être faite selon plusieurs techniques [9, 10] :

- Le partitionnement : deux ensembles indépendants sont utilisés dans la phase d'apprentissage et de validation ;

- La validation croisée : diviser l'ensemble d'apprentissage en k sous ensembles appelés *folds*, utiliser $k-1$ sous ensemble comme données d'apprentissage, et un sous ensemble comme données tests ;

- Le "Bootstrapping" : utiliser n éléments de la base d'apprentissage comme des éléments tests.

  La variante la plus utilisée est le *Bootstrap 0.632*, dans laquelle une nouvelle base est créée en sélectionnant n instances de la base originale, toute en permettant la répétition de la sélection d'instances.

  La base de tests sera donc constituée des éléments non choisis. La taille de la base de tests est obtenue en calculant la probabilité suivante :

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

b - Méthodes d'apprentissage supervisé

Dans cette partie, nous présentons le principe de fonctionnement de certaines méthodes d'apprentissage supervisé.

1. **Les arbres de décison**
   La construction des arbres de décision à partir de données est une discipline déjà ancienne. Les statisticiens en attribuent la paternité à Morgan et Sonquist [12] qui, les premiers, ont utilisé les arbres de regression dans un processus de prédiction et d'explication (AID - Automatic Interaction Detection).

   Il s'en est suivi toute une famille de méthodes, étendues jusqu'aux problèmes de discrimination et classement, qui s'appuyaient sur le même paradigme de la représentation par arbres. On considère généralement que cette approche a connu son apogée avec la méthode CART (Classification and Regression Tree) de Breiman et al. décrite en détail dans une monographie qui fait encore référence aujourd'hui.

   En apprentissage automatique, la plupart des travaux s'appuient sur la théorie de l'information. Il est d'usage de citer la méthode ID3 de Quinlan (Induction of Decision Tree - Quinlan 1979) qui, lui même, rattache ses travaux à ceux de Hunt (1962). Quinlan a été un acteur très actif dans la deuxième moitié des années 80 avec un grand nombre de publications où il propose un ensemble d'heuristiques pour améliorer

le comportement de son système.

Son approche a pris un tournant important dans les années 90 lorsqu'il présenta la méthode C4.5 qui est l'autre référence incontournable dès lors que l'on veut citer les arbres de décision. Il existe bien une autre version de cet algorithme, baptisée C5.0 (1993). Malheureusement, cette version est implémentée dans un logiciel commercial et il n'est pas possible d'en avoir les détails.

Un arbre de décision est une structure qui permet de déduire un résultat à partir de décisions successives. Pour parcourir un arbre de décision et trouver une solution il faut partir de la racine. Chaque nœud est une décision atomique.

Chaque réponse possible est prise en compte et permet de se diriger vers un des fils du nœud. De proche en proche, on descend dans l'arbre jusqu'à arriver sur une feuille.

Cette dernière représente la réponse qu'apporte l'arbre au cas que l'on vient de tester. Pour extraire une décision à partir d'un arbre il suffit de :
– Débuter à partir la racine de l'arbre.
– Descendre dans l'arbre en passant par les nœuds de test.
– La feuille atteinte permet de classer l'instance testée.

Très souvent on considère qu'un nœud pose une question sur une variable. La valeur de cette variable permet de savoir sur quels fils descendre. Pour les variables énumérées il est parfois possible d'avoir un fils par valeur.

On peut aussi décider que plusieurs variables différentes mènent au même sous arbre. Pour les variables continues il n'est pas imaginable de créer un nœud qui aurait potentiellement un nombre de fils infini. Il est nécessaire de discrétiser le domaine continu (arrondis, approximation), donc décider de segmenter le domaine en sous ensembles.

L'algorithme ID3 L'algorithme ID3 fut propose par Quinlan en 1979 afin de générer des arbres de décisions à partir de données. Imaginons que nous ayons à notre disposition un ensemble d'enregistrements.

Tous les enregistrements ont la même structure, à savoir un certain nombre de paires (attribut, valeur). L'un de ses attributs représente la catégorie (ou la classe) de l'enregistrement.

Le problème consiste à construire un arbre de décision qui sur la base de réponses a des questions posées sur des attributs non ciblés peut prédire correctement la valeur de l'attribut cible. Souvent l'attribut cible prend seulement les valeurs {vrai, faux} ou {échec, succès}. L'algorithme ID3 se base sur le principe de "gains d'informations ". C'est à dire, l'enregistrement qui rapporte un gain d'informations plus important est choisi en premier.

Si on prend deux clases N et P et un ensemble d'enregistrements avec p enregistrements qui appartiennent à P et n enregistrements qui appartiennent à N. L'entropie est définie comme suit [13] :

$$I(p,n) = -\frac{p}{p+n}log_2\frac{p}{p+n} - \frac{n}{p+n}log_2\frac{n}{p+n}$$

L'information nécessaire pour poser un enregistrement dans un sous arbre S_i est définie comme suit :

$$E(A) = \sum_{i=1}^{v}\frac{p_i+ni}{p+n}I(p_i+n_i)$$

Le gain d'information est alors défini comme suit :

$$Gain(A) = I(p,n) - E(A)$$

Si on note R l'ensemble d'attributs non-cibles, C l'attribut cible et S l'ensemble d'apprentissage, l'algorithme ID3 s'écrit comme suit :

(a) Si S est vide alors : retourner un simple nœud de valeur échec ;

(b) Si S est constituée uniquement de valeurs identiques pour la cible alors : retourner un simple nœud de cette valeur ;

(c) Si R est vide alors : retourner un simple nœud avec comme valeur la plus fréquente des valeurs de l'attribut cible trouvé dans S ;

(d) D reçoit l'élément avec le plus grand gain d'informations ;

(e) La racine de l'arbre sera donc D, les sous arbres de D sont construits à partir d'un appel récursif de cet algorithme en changeant l'ensemble R en $R - D$.

Procedure ID3 1 L'algorithme ID3

Données$R$: ensemble de variables, X : la variable de classement, S : la population
Début
1. **Si** S est vide retourner une feuille vide.
2. **Sinon Si** S est un ensemble d'individus prenant la méme valeur pour X
3. retourner une feuille avec cette valeur.
4. **Sinon Si** R est vide
5. retourner une feuille contenant le mode le plus courant de X pour les individus de S.
6. **Sinon**
7. Soit D la variable avec le plus grand gain d'information
8. Soient $d_1, \cdots, d_m$ les modalités de D
9. Soient $S_1, \cdots, S_m$ les sous-ensembles des individus prenant la valeur d_j pour D
10. **Retourner** un arbre ayant D pour racine, des arêtes partant de D et etiquetees $d_1, \cdots, d_m$ ayant respectivement comme sous-arbre ID3$(R/D, X, S_1), \cdots$,ID3$(R/D, X, S_m)$

Fin

2. La régression linéaire : la méthode des moindres carrées

Une situation courante en sciences biologiques est d'avoir à sa disposition deux ensembles de données de taille n, $y_1, y_2, ..., y_n$ et $x_1, x_2, ..., x_n$, obtenus expérimentalement ou mesures sur une population. Le problème de la régression consiste à rechercher une relation pouvant éventuellement exister entre les x et les y, par exemple de la forme $y = f(x)$. Lorsque la relation recherchée est affinée, c'est-a-dire de la forme $y = ax + b$, on parle de régression linéaire.

Mais même si une telle relation est effectivement présente, les données mesurées ne vérifient pas en général cette relation exactement. Pour tenir compte dans le modèle mathématique des erreurs observées, on considère les données $y_1, y_2, ..., y_n$ comme autant de réalisations d'une variable aléatoire Y et parfois aussi les données $x_1, x_2, ..., x_n$ comme autant de réalisations d'une variable aléatoire X. On dit que la variable Y est la variable dépendante ou variable expliquée et que la variable X est la variable explicative [14].

La droite des moindres carrées Les données $(x_i, y_i), i = 1, ..., n$ peuvent être représentées par un nuage de n points dans le plan. Le centre de gravite de ce nuage $(\overline{x}, \overline{y})$ peut se calculer comme suit :

$$(\overline{x}, \overline{y}) = \frac{1}{n}(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i).$$

Rechercher une relation affine entre les variables X et Y revient à déterminer une droite qui s'ajuste le mieux possible à ce nuage de points. Parmi toutes les droites possibles, on retient celle qui jouit d'une propriété remarquable : c'est celle qui rend minimale la somme des carres des écarts des valeurs observées y_i à la droite $\widehat{y}_i = ax_i + b$.

Si ϵ_i représente cet écart, appelé aussi résidu, le principe des moindres carrés ordinaire (MCO) consiste à choisir les valeurs de a et de b qui minimisent

$$E = \sum_{i=0}^{n} \epsilon_i^2 = \sum_{i=0}^{n} (y_i - (ax_i + b))^2$$

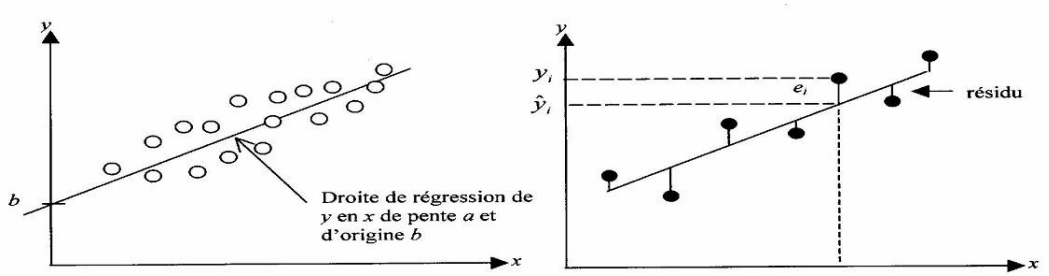

FIGURE 2.2 – Droite des moindres carrées

Il est facile de montrer que les valeurs notées $\widehat{a}$ et $\widehat{b}$ sont respectivement égales à

$$\begin{aligned} \widehat{a} &= \frac{\sum_{i=1}^{n}(x - \overline{x})(y - \overline{y})}{\sum_{i=1}^{n}(x - \overline{x})^2} \\ \widehat{b} &= \overline{y} - \widehat{a}(\overline{x}) \end{aligned}$$

et . On exprime souvent $\widehat{a}$ au moyen de la variance et de la covariance des variables aléatoires X et Y par :

$$\widehat{a} = cov_{xy}/s_x^2$$

En effet on a :

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x-\overline{x})^2$$

$$cov_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x-\overline{x})(y-\overline{y})$$

Pour mesurer la qualité de l'approximation d'un nuage (x_i, y_i) par sa droite des moindres carrées, on calcule le coefficient de corrélation linéaire défini par :

$$r_{xy} = \frac{cov_{xy}}{s_x s_y}$$

3. **Les séparateurs à vaste marge**

Les séparateurs à vaste marge SVMs (support vector machine) constituent une technique de classification très puissante. L'idée de base revient à chercher un hyperplan dont la distance minimale aux exemples d'apprentissage linéairement séparables est maximale.

On appelle cette distance "marge" entre l'hyperplan et les exemples. Comme on cherche à maximiser cette marge, on parlera de méthode des séparateurs à vaste marge (Figure 2.3). L'hyperplan optimal séparant les points de deux classes est celui qui passe "au milieu" de ces classes, c'est-à-dire dont la distance aux points les plus proches est maximale.

Ces points (exemples) qui suffisent à déterminer cet hyperplan sont appelés vecteurs de support, ou encore exemples critiques. La distance séparant l'hyperplan de ces points est appelée "marge" [15]. Si on considère h comme l'hypothèse d'apprentissage supervisé :

$$h(x_i) = w^T x + w_0 \begin{cases} \geq 0 & h(x) = +1 \\ \leq 0 & h(x) = -1 \end{cases}$$

Lorsqu'il existe une séparation linéaire entre les points d'apprentissage, il en existe une infinité, on peut alors chercher parmi ces séparatrices celle qui est la meilleure. Cet hyperplan optimal est défini par la formule :

$$\max_{w,w_0}(\min(\|x - x_i\| : x \in R^d, w^T x + w_0 = 0; i = 1, \cdots, m))$$

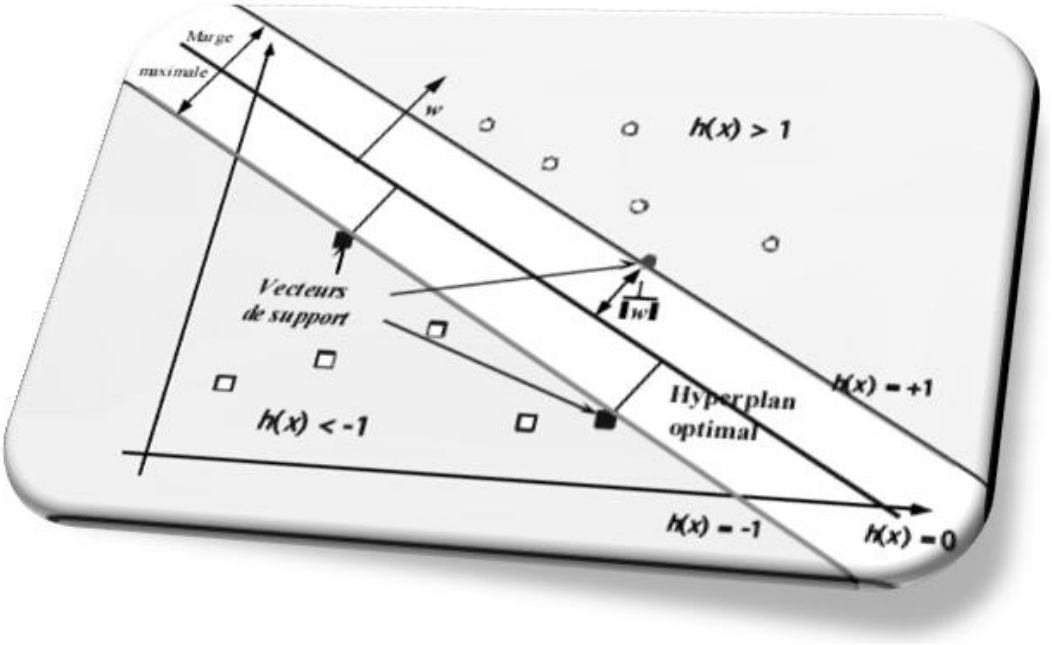

FIGURE 2.3 – La séparation des classes avec SVM

Dans le cas des SVMs, la fonction noyau est appliquée avant la séparation linéaire, elle consiste d'abord à transformer le problème de classification d'un problème non linéaire en un problème linéaire.

Le grand problème des SVMs dans la classification d'images des dégradations est du à la taille de représentations de ces images -généralement- de grande résolution. L'application de la fonction noyau n'est pas adaptée du fait qu'elle augmentera cette taille ce qui engendre un problème de complexité en temps de calcul et en mémoire.

4. **La méthode des K plus proches voisins KNN**

La méthode KNN ne contient pas une étape d'apprentissage, mais utilise un échantillon d'apprentissage constitué d'exemples résolus. La décision est prise en recherchant un ou plusieurs cas similaires.

La méthode des k-plus proche voisin a été décrite pour la première fois au début des années 1950, mais la méthode ne gagna sa popularité qu'au bout des années 60 avec l'augmentation de la puissance du calcul des grands processeurs.

Depuis, elle a été largement utilisée dans le domaine de la reconnaissance de formes. La classification par la méthode des k plus proches voisins est fondée sur l'apprentissage par analogie qui consiste à comparer un ensemble de tuples donnés avec un autre ensemble de tuples similaires.

L'ensemble de tuples d'apprentissage est décrit par le nombre n dont chaque tuple représente un point dans un espace de dimension n, alors tous les tuples sont stockés dans un espace de dimension n. quand un tuple inconnu est donné, le classifieur KNN cherche les k tuples les plus proches du tuple à classer, la décision se fait a partir de la classe la plus figurante dans cet ensemble.

Similarité et distance Soient X et Y deux objets décrits par p attributs. Les objets X et Y sont dit "similaires", s'ils possèdent q attributs identiques tel que : $p - q < \alpha$. Le nombre α étant fixé.

Bien que le calcul du degré de similarité entre deux objets ne semble pas une tâche facile, le moyen le plus efficace de le déterminer est de définir une mesure de distance entre les objets. Ceci passe par une représentation quantitative des attributs.

$$\begin{aligned} s(i,j) &= s(j,i) \\ s(i,j) &\leq S \\ s(i,i) &= S \end{aligned}$$

S étant l'indice de similarité maximal.

On peut définir l'indice s^* qui est une application de $\Omega \times \Omega$ vers $[0,1]$ par :

$$s^*(i,j) = \frac{1}{S} s(i,j)$$

Pour deux objets X et Y on a :

$$Distance(X,Y) = 1 - s^*(X,Y)$$

Il existe de multiples façons de calculer la distance entre deux objets. Soient $X = (x_1, x_2, x_3, ..., x_n)$ et $Y = (y_1, y_2, y_3, ..., y_n)$ deux objets avec n attributs quantitatifs. Voici quelques exemples de distances qu'on peut définir :

Distance de Manhattan

$$Distance(X,Y) = \sum_{i=1}^{n} |x_i - y_i|$$

Distance euclidienne

$$Distance(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Distance de Minkowski

$$Distance(X,Y) = \sqrt[n]{\sum_{i=1}^{n} (x_i - y_i)^n}$$

Distance de Tchebyschev

$$Distance(X,Y) = \max_{i=1,..n} |x_i - y_i|$$

Distance de Canberra

$$Distance(X,Y) = \sum_{1}^{n} \frac{|x_i - y_i|}{x_i + yi} \text{ avec } x_i \text{ et } y_i \text{ positif}$$

Séparation angulaire

$$Distance(X,Y) = \frac{\sum_1^n x_i y_i}{\sqrt{\sum_1^n x_i^2 \sum_1^n y_i^2}}$$

L'algorithme KNN Entrée : un tuple à classer
- Calculer la distance entre le tuple et l'ensemble des tuples d'apprentissage ;
- Choisir les K plus proches tuples ;
- Chaque tuple vote pour la classe à laquelle il appartient en incrémentant la variable associée à la classe ;
- Choisir la classe qui a eu plus de votes.

Un degré de "confiance" est affecté à la décision. La confiance est égale à la différence entre le nombre de votes pour la classe majoritaire et le nombre total des votes.

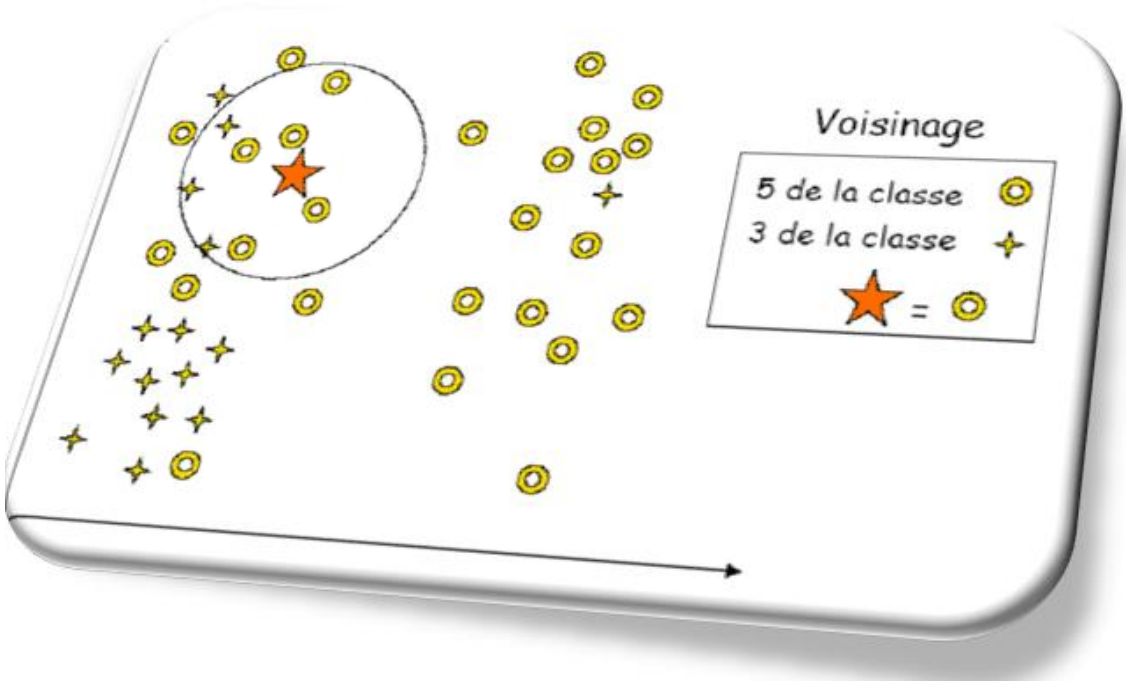

FIGURE 2.4 – La décision de la classe dans l'algorithme du KNN

Choix du voisinage Malgré qu'il existe plusieurs algorithmes pour définir une heuristique qui estime le meilleur nombre k (par exemple donner une règle de type k =*Le nombre d'attributs d'un tuple*).

Cependant, le meilleur moyen reste la définition de ce nombre empiriquement. Partant de $k = 1$ on utilise un test pour estimer le taux d'erreur de classification. Ce processus peut être répété chaque fois en

incrémentant le nombre k d'un voisin. La valeur qui retourne le taux d'erreur de classification minimal est choisie.

En général, le nombre k augmente en fonction de la taille d'echantillon d'apprentissage [17].

2.4 Conclusion

Nous avons vu dans ce chapitre les différentes notions nécessaires pour la compréhension du domaine BIOINFORMATIQUE ainsi que les techniques informatiques et une bonne part a été mise pour l'apprentissage supervisé. Le chapitre suivant a pour but de présenter les différentes étapes de conception de notre outil.

Chapitre 3

Conception et Développement

3.1 Introduction

La lecture du chapitre précédent donne une bonne connaissance sur les différents aspects de la bioinformatique et les outils informatiques qu'on puisse les utiliser.

Dans ce chapitre, on montre l'utilisation de quelques méthodes de l'apprentissage supervisé dans la conception de notre application, qu'on va montrer par la suite, toutes les étapes parcourues pour la réaliser.

3.2 Analyse et définition des besoins

3.2.1 Les besoins fonctionnels et non fonctionnels

Besoins fonctionnels

L'outil proposé reçoit une séquence protéique anonyme et son rôle est d'identifier sa structure secondaire soit elle est de la classe α ou β.

- L'utilisateur doit :
    1. Introduire une séquence de 50 acides aminés sous forme d'une chaîne de caractère où chaque lettre représente "le code à une lettre" de l'acide aminé.
    2. Choisir la méthode d'analyse knn ou ID3.
    3. Et il peut consulter la liste des acides aminés et les séquences protéiques.

– Le logiciel doit :

1. Reconnaitre la séquence saisie.
2. Stocker toutes les protéines connues dans la base de connaissance.

Besoins non fonctionnels

1 - À cause de l'utilisation des algorithmes de l'apprentissage supervisé, le système impose une configuration matérielles pas nécessairement performante mais au minimum (pentium III. 1,6 GHz. 128 Mo de RAM) pour supporter la complexité temporelle et la grande structure de données ; on recommande alors cette configuration (Pentium VI. 2,4 GHz. 512 Mo de RAM).

2 - Pourquoi java ?

– Un langage complètement objet facilite l'intégration des classes.
– Java est "portable", cela signifie que Java fonctionne tout aussi bien sur Windows, que sur Macintosh, Linux...
– Plus répondu par apport aux autres langages.
– Java est gratuit [18] .

3 - Besoin en matière de bases de données

Malgré que les données utilisées ne sont pas trop complexes, et bien structurées, il est plus bien meilleur de les organiser sous forme de base de données de séquences protéiques afin de :

– Séparer les données du programme, pour permettre plus de portabilité aux données.
– Permettre de bien utiliser, modifier et ajouter d'autres séquences. Pour cela, on propose de créer une base de données des séquences en utilisant le format (.ARFF) connu dans la communauté des spécialistes de datamining.

3.2.2 Base de connaissance

1 - Fichiers externes

Dés que l'outil vise à classifier, il lui faut alors une base d'apprentissage. Tant que la qualité de la base de connaissance influe directement sur la fiabilité des résultats de la classification, le choix était vraiment un dilemme entre plusieurs banques de données telles que : PDB, GENBANK, UNIPROT, SWISS-PROT, REFSEQ,...etc.

Celle qu'on a conçue a été prise depuis la base de données construite par **Mr. Hong-Bin Shen** de l'université **Shanghai Jiao Tong**.

2 - WEKA et ARFF

Weka est un logiciel libre qui propose un ensemble d'algorithmes d'apprentissage automatique. Il possède également toute une palette d'outils pour le traitement de données, la sélection d'attributs, la visualisation de distributions, de modèles et de résultats.

Il permet de faire de la classification, de la régression, du clustering et des règles d'associations. Son code java est ouvert à tous et disponible via le site web de l'université Waikato [1] .

Fonctionnant sur n'importe quelle plate forme (Windows, Linux/Unix, Max Os), Weka offre la possibilité d'utiliser une interface graphique complète, ou d'une ligne de commande spécifique.

Weka possède un format de fichier ouvert (*.ARFF), qui est un format texte, avec des spécifications ad hoc pour documenter les variables. Importer un fichier ARFF ne pose donc pas de problèmes particuliers, dès lors que l'on sait appréhender un fichier texte.

1. http ://www.cs.waikato.ac.nz/ml/weka/

Un fichier (*.ARFF) est formaté comme suit :

- **@Relation** nom de la table pour donner un nom a la table.
- Chaque attribue est défini a l'aide de la commande **@Attribute** nom de l'attribue suivi de son type.
- En fin les données sont introduit a l'aide de la commande **@Data**, les valeurs des attribues sont séparées par des virgules ", ", le nombre de valeurs doit exactement correspondre au nombre d'attributs définis.

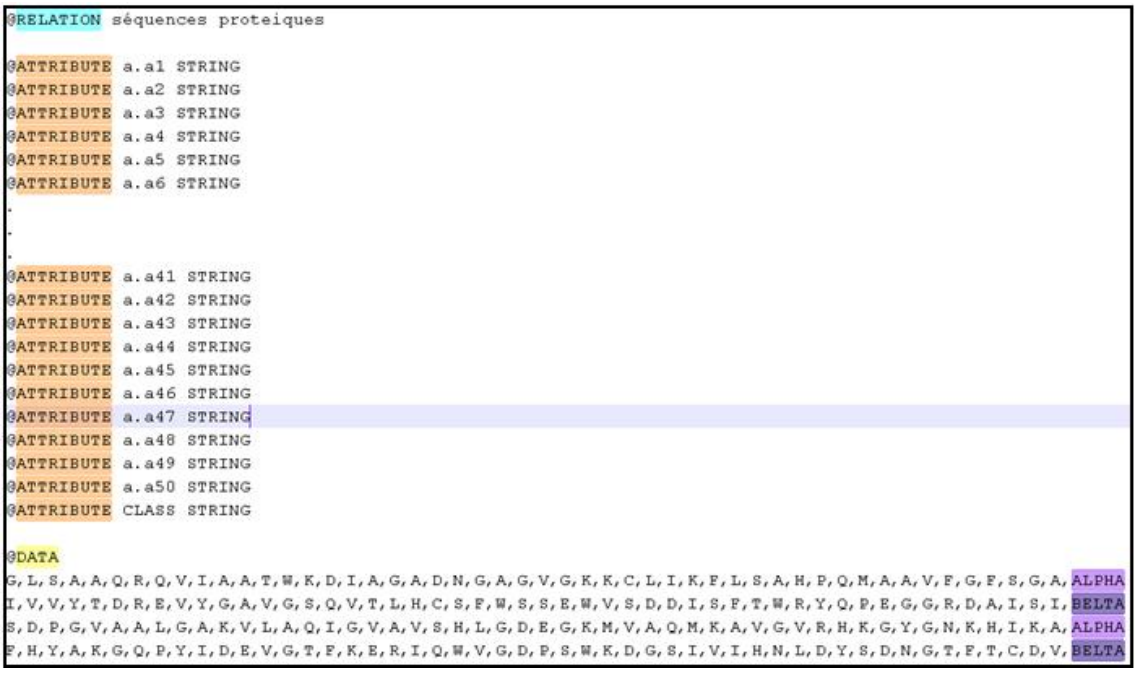

FIGURE 3.1 – le fichier ARFF

3 - Tests choisis

- validité croisée :

  On devise notre base de connaissance en 10 fenêtre de 20 séquences, la base de test sera donc composée de 20 séquences et la base d'apprentissage de 180 séquences, on réitère 5 fois ce processus.

- base de test :

  On génère une base de test aléatoire de 50 séquences, et on prend l'intégralité de séquences collectées comme base d'apprentissage.

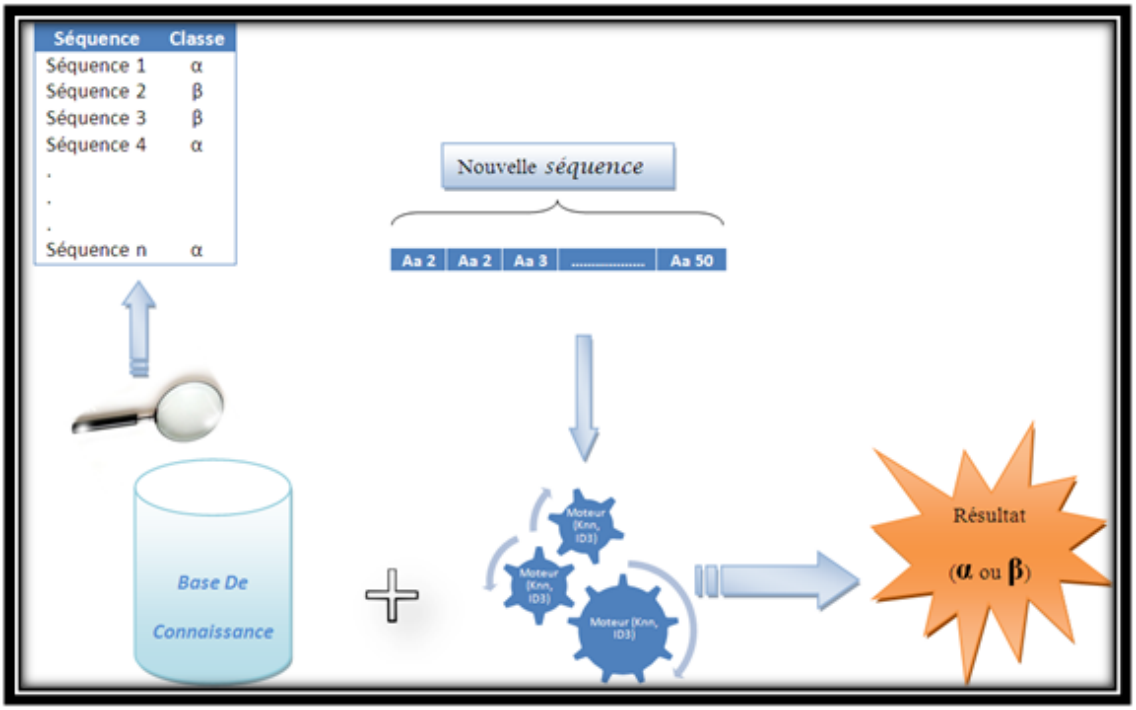

FIGURE 3.2 – Les fonctionnalités du système

3.2.3 Pourquoi faire une fenêtre fixe de 50 acides aminés ?

Comme la longueur d'une protéine est généralement entre 50 et 2000 acides aminés contribuant dans la séquence protéique, on a fixer alors la taille qu'on a choisi par 50 acides amines en prenant la taille minimale possible pour pouvoir prendre toutes les petites protéines de taille 50, et en plus on peut prendre des fragments des autres protéines, ce qui donne une variété a la base de connaissance qu'on va concevoir.

D'une autre part la fixation de la taille qu'on a choisie nous aidera bien lors du codage et l'implémentation de la base de connaissance ainsi que les traitements faits sur la séquence saisie qui est aussi fixe (50 acides amines).

3.3 Algorithmes choisis

3.3.1 Codage

1 - le codage le plus naturel est de prendre les séquences telles qu'elles sont alignées dans la base d'apprentissage et de test, pour cela, chaque acide aminé est représenté par une lettre unique.

Chaque séquence sera donc représentée par une chaine de caractère composée de 50 caractères. L'avantage de ce codage est de garder la même longueur pour toutes les séquences.

2 - pour des besoins qu'on va les détailler un peu plus loin, un deuxième codage est proposé : Une protéine ne sera plus présentée par les acides amines qu'ils composent mais par leurs fréquences.

Une séquence sera donc composée de 22 nombres entiers décrivant les fréquences des 22 acides aminés. Ce codage nous permet toujours de garder la même longueur de représentation.

3.3.2 Knn

Dans cette partie, on défini le paramétrage adéquat de l'algorithme Knn

Distance

- Pour le premier codage, on définit la fonction $D1$ de distance entre deux séquence de protéines :
Soit $S1(a_1, a_2, \cdots, a_{50})$ et $S2(b_1, b_2, \cdots, b_{50})$ deux séquence, la fonction $D1$ est définie comme suit :

$$D1 : S \times S \to N$$

$$D1(S_1, S_2) = \sum_{i=1}^{50} d_1(a_i, b_i)$$

$$d_1(a, b) = \begin{cases} 0 & \text{si } a = b \\ 1 & sinon \end{cases}$$

- pour le deuxième codage on définit la fonction de distance $D2$ comme suit :

$$D2 : S \times S \rightarrow R$$

$$D2(S_1, S_2) = \sqrt{\sum_{i=1}^{22}(a_i - b_i)^2}$$

Choix de voisinage

Afin d'améliorer la complexité, on utilise un rayon de voisinage r au lieu du nombre K.

Le voisinage est défini comme suit :

Soit r le rayon de voisinage, le voisinage ϑ d'une séquence δ est défini par :

$$\vartheta_\delta \subseteq S = \{s \in S / D(s, \delta) \leq r\}$$

3.3.3 ID3

Construction de l'arbre de décision

Pour ID3, on a choisi le deuxième codage celui qui liste les fréquences de chaque acide aminé dans la fenêtre.

Les deux classes définies sont la classe α et la classe β, pour cela le calcul de l'entropie sera défini comme suit :

$$I(a, b) = -\frac{a}{a+b}\log_2\frac{a}{a+b} - \frac{b}{a+b}\log_2\frac{b}{a+b}$$

Avec a le nombre d'individus qui ont α comme classe et b le nombre de ceux qui ont β comme classe.

Concernant les 3 paramètres d'ID3, l'algorithme qui va construire l'arbre de décision (S, R, X) :

S : Désigne l'ensemble des séquences protéiques de la base de connaissance numérotées de 1 à 200.

R : Les attributs non cibles, désigne l'ensemble de fréquences de chaque acide aminé dans les séquences.

X : L'attribut cible qui représente la classe de la séquence (α,β).

En premier lieu on va définir l'acide aminé (colonne) qui a le plus grand gain d'information qui est défini comme suit :

$$Gain(C) = I(a,b) - \sum_{i=1}^{v} \frac{a_i + b_i}{a+b} I(a_i, b_i) \quad o \quad C \in R$$

V : nombre de valeurs prises de l'attribut C.

a_i : Désigne le nombre d'individus qui ont la valeur V_i pour C et qu'ils sont de la classe α.

b_i : Désigne le nombre d'individus qui ont la valeur V_i pour C et qu'ils sont de la classe β.

$V_1, V_2, \ldots, V_m$: représentent les valeurs prises par C.

Après avoir choisi la colonne C_{ch} [2] qui a le plus grand gain, on construit un arbre qui a pour racine l'acide aminé correspondant à cette colonne choisie, des arrêtes partants étiquetées par les valeurs prises par les fréquences de cet acide $(V_1, V_2, ..., V_m)$ ayant respectivement des arbres construits à partir des appels à l'algorithme ID3 :

$$ID3(S_1, R - C_{ch}, X), ID3(S_2, R - C_{ch}, X), ..., ID3(S_m, R - C_{ch}, X)$$

Où S_i représente l'ensemble des enregistrements ayant V_i comme valeur de C_{ch}.

Parcours de l'arbre

Pour une séquence donnée, il faut parcourir l'arbre pour savoir sa classe, et lorsqu'elle est représentée par la liste des fréquences des acides aminées, ainsi que les arrêtes de l'arbre sont aussi étiquetées par les fréquences de leurs acides aminées sources, alors le parcours de l'arbre sera comme suit :

- On prend la fréquence de l'acide aminé racine dans la séquence saisie.
- On utilise la distance Euclidienne pour mesurer la distance entre chaque étiquette et la fréquence prise au dessus et on prend l'étiquette la plus proche.
- On suit l'arc de cette étiquette choisie pour savoir le prochain acide aminé à consulter dans la séquence.
- On cherche l'étiquette la plus proche comme précédemment.

2. Colonne choisie

– On réitère ce processus jusqu'a ce qu'on abouti a l'une des feuilles qui est soit α soit β, et on obtient la classe de cette séquence.

3.4 Conception

3.4.1 Le diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).

Voila une modélisation des cas d'utilisation de notre outil :

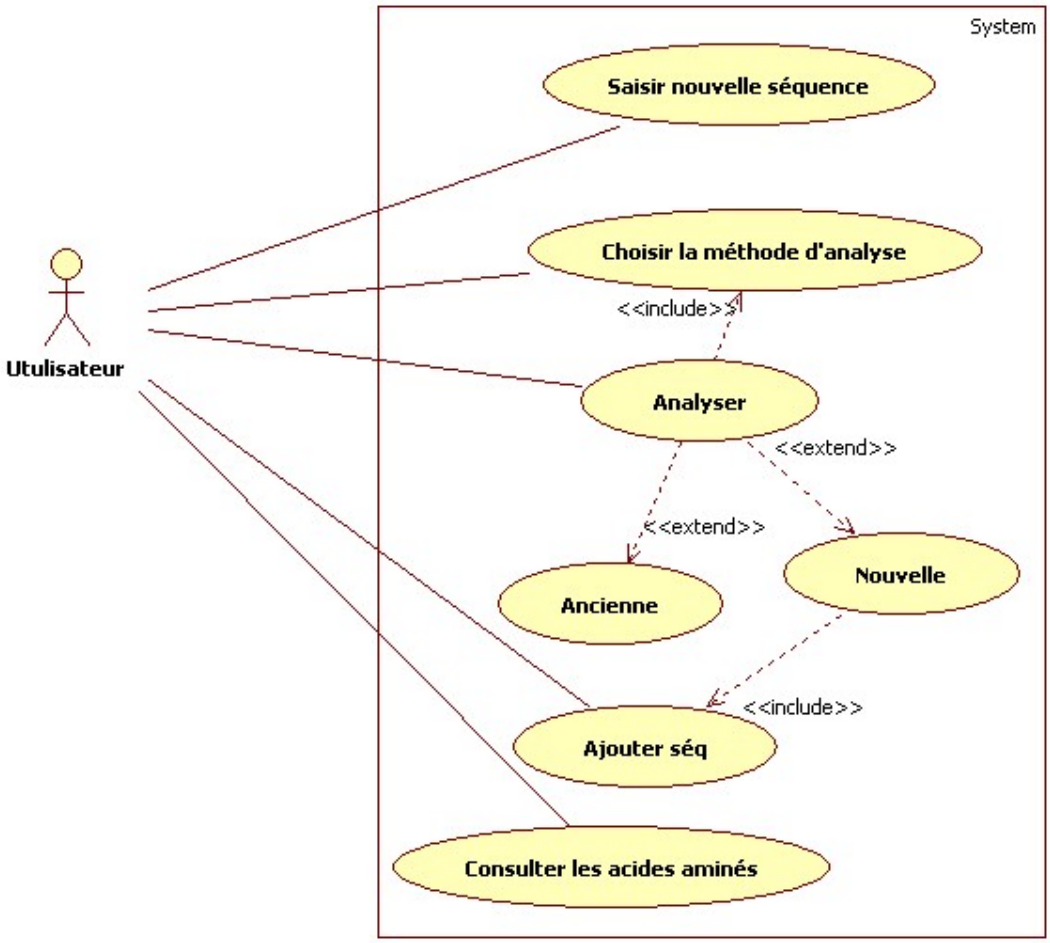

FIGURE 3.3 – Le diagramme de cas d'utilisation

3.4.2 Le diagramme de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language. On montre ces interactions dans le cadre d'un scénario d'un Diagramme des cas d'utilisation.

L'enchainement des tâches faites par notre application est représenté par ce diagramme comme suit :

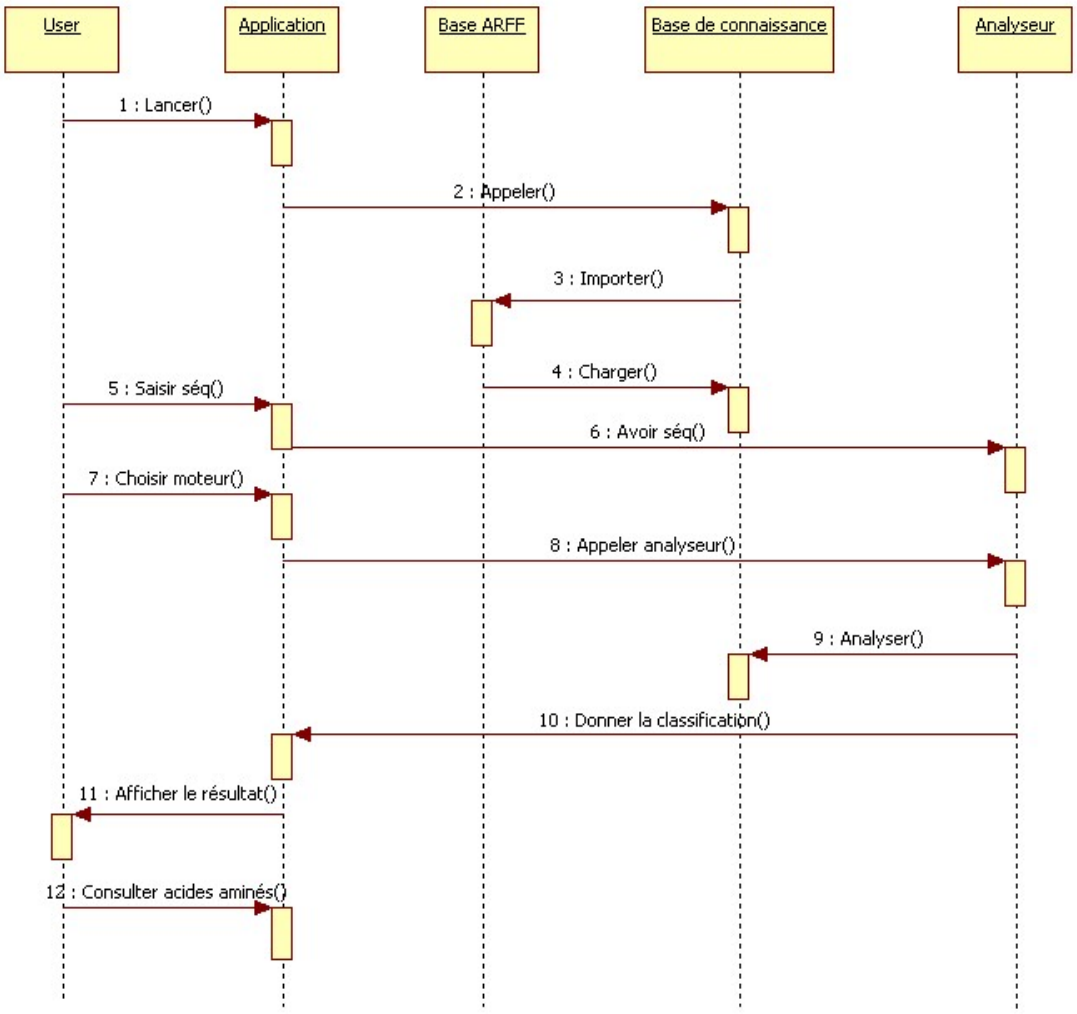

FIGURE 3.4 – Le diagramme de séquence

3.5 Modules et tests unitaires

3.5.1 Appel à la base de connaissance

Pour la base de connaissance on a consacré un ensemble de fonctions et procédures qui vont la charger depuis le fichier ***arff*** dans un tableau de chaine de caractère "*tableauintermediaire*" où chaque ligne de ce tableau correspond à une ligne du fichier ***arff*** :

1 - la procédure utilisée pour le tableau *BC* à 2 dimensions qui correspond au premier codage :

Procedure 2 Chargement BC

Début
1. **Pour** (i = 1 à 200) **faire**
2. bc [i , 0] = enlevervirguiletclass2d (tableauintermediaire[55+i]) ;
3. bc [i , 1] = getclasse2d (tableauintermediaire[55+i]) ;
4. **Fait**

Fin

Où :

- Le saut de 55 ([55+i]) a été fait pour se mettre dans la partie **@Data** dans le fichier *arff*, où se trouvent les séquences.

- **enlevervirguiletclass2d (tableauintermediaire[55+i]) :** retourne la séquence d'acides aminés depuis le tableau intermédiaire.

- **getclasse2d (tableauintermediaire[55+i]) :** retourne la classe d'une séquence qui se trouve dans le paramètre d'entrée.

2 - Un autre tableau *Rebc* à 2 dimensions qui correspond au deuxième codage, sa procédure est :

Procedure 3 Chargement rebc

```
Début
1.    Pour (i = 1 à 200) faire
2.          rebc [i , 0] = freq ('A' , bc [i , 0]) ;
3.          rebc [i , 1] = freq ('R' , bc [i , 0]) ;
4.          rebc [i , 2] = freq ('N' , bc [i , 0]) ;
5.          rebc [i , 3] = freq ('D' , bc [i , 0]) ;
6.          rebc [i , 4] = freq ('C' , bc [i , 0]) ;
7.          rebc [i , 5] = freq ('E' , bc [i , 0]) ;
8.          rebc [i , 6] = freq ('Q' , bc [i , 0]) ;
9.          rebc [i , 7] = freq ('G' , bc [i , 0]) ;
10.         rebc [i , 8] = freq ('H' , bc [i , 0]) ;
11.         rebc [i , 9] = freq ('I' , bc [i , 0]) ;
12.         rebc [i , 10] = freq ('L' , bc [i , 0]) ;
13.         rebc [i , 11] = freq ('K' , bc [i , 0]) ;
14.         rebc [i , 12] = freq ('M' , bc [i , 0]) ;
15.         rebc [i , 13] = freq ('F' , bc [i , 0]) ;
16.         rebc [i , 14] = freq ('P' , bc [i , 0]) ;
17.         rebc [i , 15] = freq ('S' , bc [i , 0]) ;
18.         rebc [i , 16] = freq ('T' , bc [i , 0]) ;
19.         rebc [i , 17] = freq ('W' , bc [i , 0]) ;
20.         rebc [i , 18] = freq ('Y' , bc [i , 0]) ;
21.         rebc [i , 19] = freq ('V' , bc [i , 0]) ;
22.               si getclasse2d (tableauintermediaire[55+i]) = "Alpha" alors
23.                     rebc [i , 20] = 1 ;
24.               sinon
25.                     rebc [i , 20] = 0 ;
26.               finsi
27.   Fait
Fin
```

Où :

- Pour la $20^{\text{ième}}$ colonne, la valeur 1 désigne la classe α et 0 désigne la classe β.

– **freq (Char C, Séquence Seq) :** retourne le nombre d'occurrences de ce caractère dans la séquence.

3.5.2 Saisie de la séquence

L'interface de l'application doit contenir un champ de texte par lequel la séquence protéique est introduite sous forme d'une chaine de 50 caractères. Pour les deux codages qu'on a proposés, la séquence reste telle qu'elle est pour le premier, et pour le deuxième codage on a utilisé une fonction de conversion :

Fonction 4 Conversion (Seq : chaine de caractére) : tableau d'entier [20]

Début

1. conversion [1] = freq ('A', Seq) ;
2. conversion [2] = freq ('R', Seq) ;
3. conversion [3] = freq ('N', Seq) ;
4. conversion [4] = freq ('D', Seq) ;
5. conversion [5] = freq ('C', Seq) ;
6. conversion [6] = freq ('E', Seq) ;
7. conversion [7] = freq ('Q', Seq) ;
8. conversion [8] = freq ('G', Seq) ;
9. conversion [9] = freq ('H', Seq) ;
10. conversion [10] = freq ('I', Seq) ;
11. conversion [11] = freq ('L', Seq) ;
12. conversion [12] = freq ('K', Seq) ;
13. conversion [13] = freq ('M', Seq) ;
14. conversion [14] = freq ('F', Seq) ;
15. conversion [15] = freq ('P', Seq) ;
16. conversion [16] = freq ('S', Seq) ;
17. conversion [17] = freq ('T', Seq) ;
18. conversion [18] = freq ('W', Seq) ;
19. conversion [19] = freq ('Y', Seq) ;
20. conversion [20] = freq ('V', Seq) ;
20. **retourner conversion** ;

Fin

Remarque L'ordre des acides aminés lors de la conversion doit être identique à celui de la procédure *chargement rebc*.

3.5.3 Choix d'analyseur

Aprés avoir introduit la séquence a classifier, l'utilisateur doit choisir l'un des deux analyseurs $(Knn, ID3)$.

1 - Knn

Pour cet algorithme on avait besoin des fonctions suivantes $Dis1, Dis2$:

Fonction 5 Dis1 (Seq_1 chaine de caractére, Seq_2 chaine de caractére) :entier

---

Début
1. $dis1 = 0$;
2. **Pour** (i = 1 à longueur Seq) **faire**
3. **si** iémechar(i, Seq_1)$\neq$ iémechar(i, Seq_2) **alors** ;
4. $dis1 = dis1 + 1$;
5. **finsi**
6. **Fait**
7. retourne $dis1$;

Fin

Où :
iémechar(i, Seq) : retourne le iéme caractére dans la séquence Seq.

Fonction 6 Dis2 ($CSeq_1$ tableau d'entier, $CSeq_2$ tableau d'entier) :entier

---

Début
1. $dis = 0$;
2. **Pour** (i = 1 à longueur $CSeq_1$) **faire**
3. $dis = dis + (CSeq_1[i] + CSeq_2[i])^2$;
4. **Fait**
5. $dis2 = \sqrt{dis}$
5. retourne (dis2) ;

Fin

Knn utilise les tableaux BC et $REBC$ et fait prédire la classe de la séquence saisie comme suit :

Fonction 7 Knn (K :entier, Sequence-Anonyme :chaine de car) : booléen

variable
retour : booléen ;
seuil : booléen initialisé à **faux** ;
votalpha : entier initialisé à **0** ;
votbeta : entier initialisé à **0** ;
tempon :tableau ;
Début
1. **Pour** (i = 1 à longueur bc) **faire**
2. **si** Dis1(Sequence-Anonyme, bc[i, 0])< K **alors**
3. **si** (bc[i,1] = "Alpha") **alors**
4. votalpha = votalpha + 1 ;
5. **sinon**
6. votbeta = votbeta + 1 ;
7. **finsi**
8. **finsi**
9. **fait**
10. **si** (votbeta + votalpha < seuillage) **alors**
11. votalpha = 0 ; votbeta = 0 ;
12. **pour** (i = 1 à longueur rebc) **faire**
13. **pour** (i = 1 à 20) **faire**
14. tempon [j] = rebc[i, j] ;
15. **fait**
16. **si** (Dis2(conversion(Seq-Anonyme), tempon)< K) **alors**
17. **si** (rebc [i, 2] = 1) **alors** votalpha ++ ;
18. **sinon** votbeta ++ ;
19. **finsi**
20. **finsi**
21. **fait**
22. **finsi**
23. **si** (votalpha > votbeta) **alors**
24. retour = vrai ;
25. **sinon**
26. retour = faux ;
27. **finsi**
28. retourner retour ;
Fin

Remarque :

- **Seuillage** : c'est le nombre de voisins nécessaire pour que Knn puisse prendre une décision.

- **retour** = vrai signifie que la séquence anonyme est de la classe α **retour** = faux veut dire qu'elle est de la classe β.

2 - ID3

La fonction ID3 utilise le $2^{\text{ième}}$ codage et s'appuie sur le tableau *rebc* pour générer l'arbre de décision de la manière suivante :

La structure de données pour implémenter l'arbre de décision est :

Enregistrement = arbre de décision

Début

Acide aminé : caractère ; // l'acide aminé de nœud courant

Fréquence : entier ; // la fréquence avenante a ce nœud

Feuille : chaine de caractère ; // alpha ou beta ou vide

Fils : tableau d'arbre de décision ; // les fils de ce nœud

Fin

Fonction 8 ID3(S :tableau d'entier R :tableau de caractére) : arbre de décision

variable **ID** : arbre de décision ;
Début
1. **si** (longueur(S) = 0) **alors**
2. retourner Null ;
3. **sinon**
4. meme = vrai ; memeval = rebc[S_1,21] ;
5. **pour**(i = 2 à longueur S) **faire**
6. **si** rebc [S[i], 21] $\neq$ memeval **alors**
7. meme = faux ;
8. break ;
9. **finsi**
10. **fait**
11. **si**(meme = vrai) **alors**
12. ID.feuille = memeval ;
13. **sinon**
14. **si**(longueur (R) = 0) **alors**
15. Calpha = 0 ;
16. Cbeta = 0 ;
17. **pour** (i = 1 à S.longueur) **faire**
18. **si** rebc (S[i ; 20] = 1) **alors**
19. Calpha = Calpha + 1 ;
20. **sinon**
21. Cbeta = Cbeta + 1 ;
22. **finsi**
23. **fait**
24. **si** (Calpha $\geq$ Cbeta) **alors**
25. ID.feuille = "Alpha" ;
26. **sinon**
27. ID.feuille = "Beta" ;
28. **finsi**
29. **sinon**
30. Calculer colonne gagnante (S, R) ;
31. Calculer valeur prise (colonnegagnante) ;
32. **pour** (chaque valeur prise) **faire**
33. S_1 = créer le tableau S correspond ;
34. R_1 = enlever la variable gagnante de R ;
35. ID.fils.ajouter (ID3 (S_1, R_1)) ;
36. **fait**
37. **finsi**
38. **finsi**
39. **finsi**
40. retourner ID ;
Fin

Remarque :
Dés que le code de la fonction ID3 est très long (plus de 200 ligne) on n'a fait qu'indiquer les jeux d'instructions de chaque tache par leurs buts tels que :

- Le calcul de la colonne gagnante à partir de S et R.
- Le calcul des valeurs prises par cette colonne gagnante.
- L'appel récursif pour chaque valeur prise après la construction des tableaux S et R correspondants.

Analyseur ID3 Après la construction de l'arbre de décision l'analyse de la séquence saisie sera faite en parcourant l'arbre généré au paravent par la fonction suivante :

Fonction 9 parcourID3 (ag : arbre decision ; sequenceanonyme : chainne de caractère) : structure 2D

Début
1. **si** (ag.feuille = Null) **alors**
2. D = | $freq(ag.acideamin, sequenceanonyme) - ag.fils[1].frequence$ | ;
3. choix = 1 ;
4. **pour**(i = 2 à longueur (ag.fils)) **faire**
5. **si**(D > (freq(ag.acideaminé, sequenceanonyme) - ag.fils [i].frequence))
6. D = ag.fils [i].frequence ;
7. choix = i ;
8. **finsi**
9. **fait**
10. retourner parcourID3 (ag.fils[choix],sequenceanonyme) ;
11. **sinon**
12. retourner ag.feuille ; // α ou β.
13. **finsi**

Fin

3.6 Intégration des modules

L'intégration consiste à rassembler tous les modules de programme pour réaliser l'application complète, comme on a vu les modules ; l'un utilise l'autre

par exemple *Knn* et *ID3* utilise le module *Appel BC*.

Le développement par incrément est un processus évolutif qui vise à développer en premier lieu une version initiale appelée " noyau " et puis elle porte des modifications successives jusqu'à l'obtention de la version finale.

Les modules qu'on a parlé au dessus ont été développé séquentiellement en commençant par le module *Appel BC* et puis les moteurs *Knn* et *ID3* qui nécessitent la terminaison de ce dernier, Par contre, ils peuvent êtres développés séparément l'un de l'autre.

En fin le module de l'interfaçage qui inclut l'introduction de la séquence à classifier et le choix de l'analyseur pour faire l'analyse.

3.7 Conclusion

Passant par toutes les étapes décrites au dessus, on est arrivé à réaliser notre application, qu'on a consacré le dernier chapitre pour sa description.

Chapitre 4

Evaluation du logiciel

4.1 Introduction

En dernier lieu, cette partie montrera le produit bioinformatique achevé par le suivi de processus de développement décrit dans le chapitre précédent.

4.2 Description du logiciel

Cette démonstration a été faite par le biais de la description des captures d'écran de l'interfaçage de l'application où on a fait des explications plus au moins détaillées depuis la première fenêtre qui s'affiche a l'utilisateur jusqu'à la visualisation des résultats, ainsi que des fenêtres optionnelles ont été décrites telles que la consultation des acides aminés et les séquences protéiques.

4.2.1 Combio

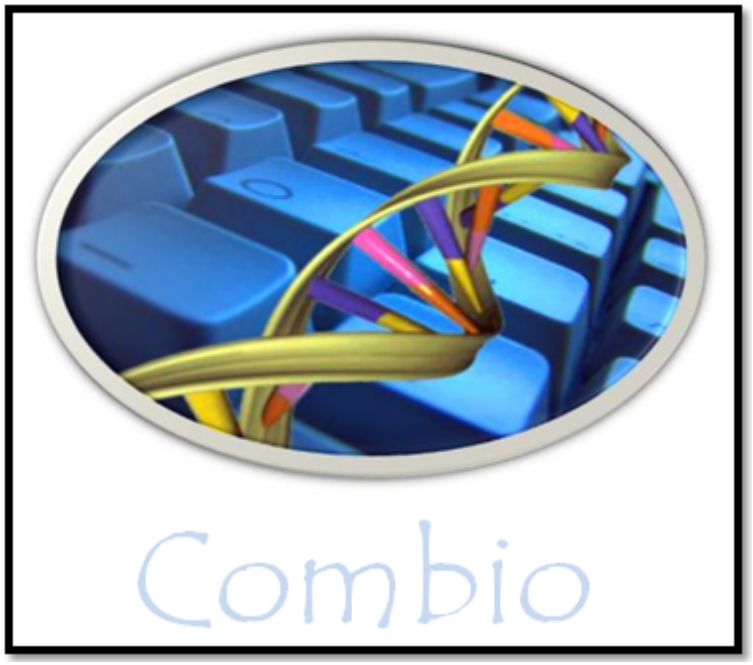

FIGURE 4.1 – Combio

Combio est le nom de l'application qu'on a réalisé, cette nomination composée reflète l'aspect hybride de l'outil où la première partie "*com*" est venue de **com**puting et la deuxième "*bio*" veut dire **bio**logy bien sur.

4.2.2 L'interface principale

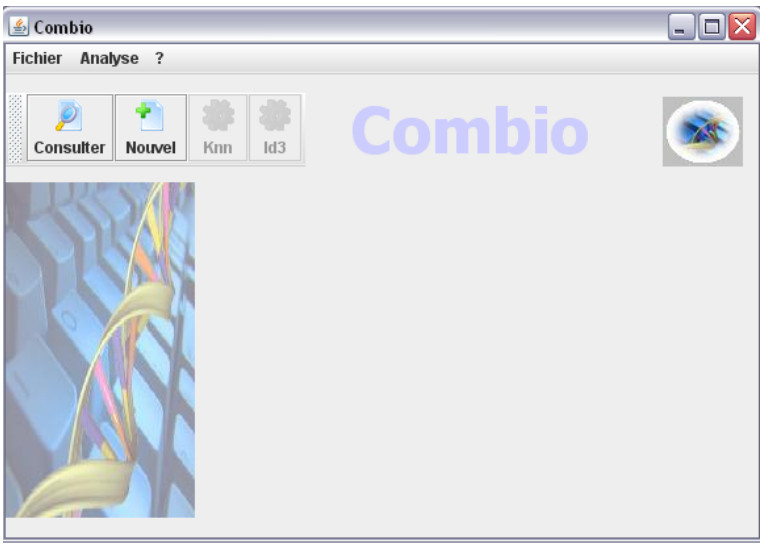

FIGURE 4.2 – L'interface principale

L'interface s'avère simple, inspirée du système de fenêtrage de Windows XP, c'est une fenêtre munie :

- d'une barre des menus pour l'accès direct aux fonctionnalités du programme, ainsi qu'une barre d'outils contenant les boutons principaux des opérations de l'application.
- le logo qui est représenté dans une image dans l'extrême gauche de la fenêtre et dans un petit icône dans le coin supérieur à droite.
- un champ vide pour le panneau de l'expérience qu'on va le détaillé plus loin.

4.2.3 les menus

On va présenter les menus de la barre des menus :

1. **Le menu fichier**

FIGURE 4.3 – Le menu fichier

On trouve dans ce menu :

- **Nouvelle séquence :** permet de visualiser le champ de l'expérience.
- **Consulter :** affiche une forme contient tous les acides aminés avec une petite explication pour chacun d'eux.
- **Quitter :** pour quitter le programme.

2. **Le menu analyse**

FIGURE 4.4 – Le menu analyse désactivé

Les deux éléments qu'il contient ne fonctionnent pas qu'après la satisfaction des entrées de programme :

FIGURE 4.5 – Le menu analyse activé

- **Knn :** lance le moteur Knn pour analyser la séquence saisie.
- **ID3 :** lance le moteur ID3 pour analyser la séquence saisie.

3. **Le menu aide**

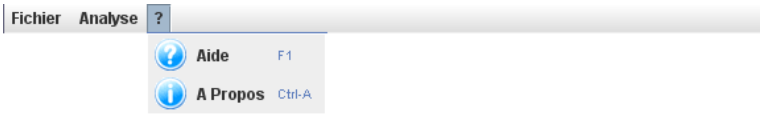

FIGURE 4.6 – Le menu aide

Son but est la familiarisation de l'utilisateur avec l'application.

4.2.4 la barre d'outils

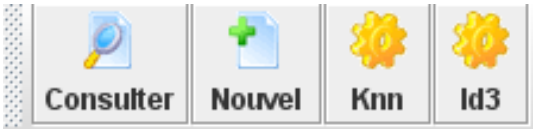

FIGURE 4.7 – la barre d'outils

Son rôle est de rapprocher et accélérer les appels aux fonctionnalités de programme telles que Consulter, Nouvel, Knn, ID3, Aide, qu'on a parlé au dessus, ils sont accessibles par cette barre.

4.2.5 le panneau d'expérience

FIGURE 4.8 – le panneau d'expérience

1. le champ de saisi pour saisir la séquence a analyser.
2. un bouton "*vérifier*" pour vérifier la syntaxe de la séquence saisie.
3. un bouton "*reset*" pour la réinitialisation du champ d'expérience.
4. une zone d'affichage pour montrer le type de l'erreur commise ou la validation de la séquence saisie après sa vérification.
5. une zone d'affichage montre le moteur choisi pour l'analyse et le résultat après avoir terminé l'inférence.

4.2.6 la fenêtre de calcul

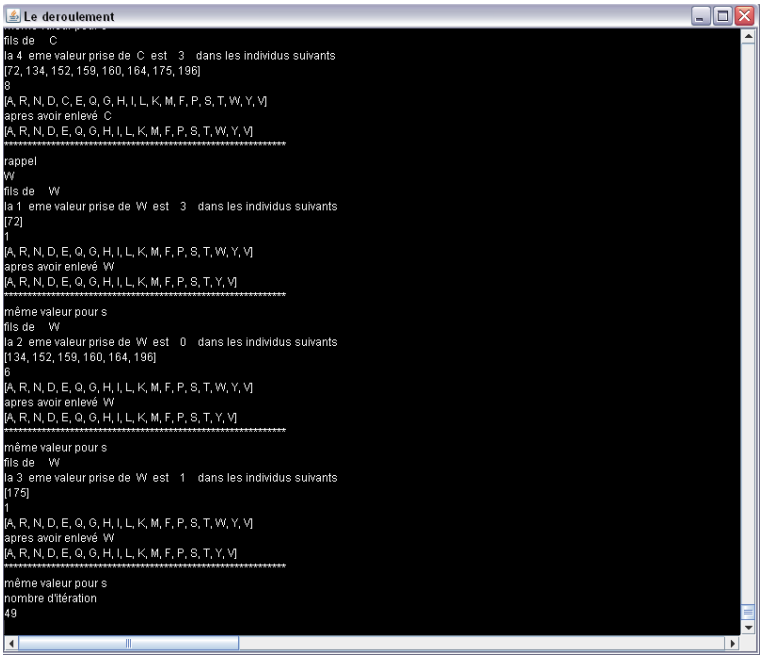

FIGURE 4.9 – la fenêtre de calcul

Son rôle est d'apercevoir les calculs faits lors de l'analyse que ce soit *Knn* ou *ID3* :

- pour *Knn* elle affiche les distances mesurées entre la séquence saisie et les individus de la base de connaissance et au fur et à mesure elle affiche le vote de chaque individu ou chacun vote pour sa classe et en fin la classe élue sera montrée.
- pour *ID3* la fenêtre affiche toute les itérations faites pour la génération de l'arbre de décision puis elle montre le parcours de cet arbre par la séquence saisie étape par étape jusqu'à l'affichage de la feuille achevée α ou β .

4.3 Etude de qualité é du logiciel

Une évaluation de l'application réalisée nous poussait à montrer les points suivants :

4.3.1 Avantages

Le produit comporte plusieurs point fort tels que :

- il offre une interface conviviale facile à utiliser.
- la façon d'entrer les données au programme est très facile, l'utilisateur n'a qu'amener une séquence protéique de 50 caractères.
- le fait de séparer les données du programme au traitement lors de l'implémentation, facilite énormément l'amélioration de la base de connaissance que le programme s'appuie sur elle sans la moindre modification dans les codes de programme.
- le logiciel est libre.

4.3.2 Manques

- une seule séquence peut être analysée à la fois.
- la phase de maintenance du logiciel était très courte, ce qui influe directement sur la détection des anomalies de l'implémentation.
- l'outil n'est pas destiné a tout public et son utilisation est retreinte sur les biologistes seulement.
- l'apprentissage supervisé qu'on a utilisé a des résultats toujours probables.

4.3.3 Qualité du logiciel

Pour qualifier l'outil on a suivi les métriques ISO [22] :

1. **la fiabilité**

   Quelques défaillances ont été trouvées lors des essais du programme ça revient a l'utilisation des algorithmes d'apprentissage car leurs résultats s'améliorent empiriquement ce qui nécessite une durée importante pour l'expérimentation ce qui ne convient pas à notre plan de travail.

2. **le rendement**

   Théoriquement le logiciel assure bien sa tâche mais n'empêche pas que ses résultats sont probables ce qui ressemble au rendement des expériences biologiques.

3. **la portabilité**

   Le logiciel a été testé dans toutes les plateformes disponibles, il est exécuté dans les versions récentes de Windows (xp, vista, seven 7) ainsi

que LINUX, ce bienfait revient au fameux java qu'on a utilisé pour l'implémentation.

4. **la maintenabilité**

   La programmation orientée objet permet une certaine structuration du code source qui permet aisément de le comprendre et le modifier si nécessaire a long terme.

4.4 Conclusion

L'application ressemble beaucoup les expériences biologiques dont ses résultats sont probables et s'améliorent par les tests dans le temps, elle s'appuie sur les méthodes de la bioinformatique et utilise les algorithmes de classification, son interfaçage est simple surtout après la lecture de ce chapitre.

Conclusion générale

La bioinformatique est un domaine de recherche très actif actuellement. L'abondance d'articles, de thèses, de conférence la concernant en est une preuve, on l'utilisait pour argumenter le choix de notre thème.

Les recherches faites et les efforts dépensés dans ce travail nous ont familiarisé avec l'étude des protéines dans la biologie, ainsi qu'une bonne formation on l'a acquis sur l'apprentissage automatique et ses techniques, où on a découvert qu'il peut s'appliquer dans beaucoup de domaines, d'ou on l'a utilisé pour concevoir une application fait prédire la structure secondaire d'une séquence protéique ; un petit projet on l'utilisait pour s'inscrire dans la communauté des bioinformaticiens.

Heureusement, car il reste beaucoup à découvrir dans ce domaine. C'est un domaine scientifique très jeune et qui n'a dévoilé pour l'instant qu'une partie infime de ses possibilités. L'avenir nous réserve de grandes surprises ; nous les attendons avec impatience.

Bibliographie

[1] Thése de doctorat de l'Université Pierre et Marie Curie par *MathildeCarpentier*, Méthodes de détection des similarités structurales : caractérisation des motifs conservés dans les familles de structures pour l'annotation des génomes. 9 décembre 2005

[2] Lubert Stryer, Jeremy Mark Berg, John L. Tymoczko (trad. Serge Weinman), *Biochimie*, Flammarion, « Médecine-Sciences », Paris, 2003, 5ième éd.

[3] Wikipédia http ://fr.wikipedia.org/wiki/Acide amin
Consulter le 07.06.2010 à 23 :50

[4] Thése de doctorat de l'Université Pierre et Marie Curie par *Annabelle CINGOZ*, Analyse d'une protéine ciblée par immunoaffinité et digestion sur microréacteur enzymatique couplés en ligne à une analyse par chromatographie liquide et spectrométrie de masse : synthèse, caractérisation et miniaturisation des outils bioanalytiques. 21 septembre 2009

[5] DESS Bioinformatique, Institut de Pharmacologie et Biologie Structurale du C.N.R.S., Formation par *M. Georges CZAPLICKI et Bernard MICHOT*

[6] *François Denis and Rémi Gilleron.* Apprentissage à base d'exemples. Support de cours, Université de Lille1, 2001.

[7] *Michèle Sebag.* Apprentissage supervisé et satisfaction des contraintes. Support de cours, INRIA, 2005.

[8] *François Olivier.* De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes. PHD thesis, Université de rouen, 2006.

[9] *Phillipe Choquette.* Nouveaux algorithmes d'apprentissage pour les classifieurs de type SCM. PHD thesis, Université Laval, Quebec, Canada, 10 2007.

[10] *Ian H Witten and Eibe Frank.* Data mining, practical machine learning tools and techniques. Morgan Kaufman, 2ième edition, 2005.

[11] *Eduardo SanchezSoto.* Réseaux bayésiens dynamiques pour la vérification du locuteur. PHD thesis, Ecole nationale supérieure des telecommunications ; Paris, 2005.

[12] *Ricco Racotomalala.* Arbres de décision. Support de cours, Laboratoires ERIC, 2005.

[13] *Benjamin DEVEZE and Matthieu FOUQUIN.* Data mining. Technical report, EPITA, Mai 2005.

[14] *Louis Esch.* Mathématique pour économistes et gestionnaires, volume 1. De Boeck Université, 3ième edition, 2007

[15] *Antoine Cornuéjols.* Une nouvelle méthode d'apprentissge, les SVMs. Technical Report 51, Université de Paris-Sud, Orsay, Juin 2002.

[16] *Wangxin Xiao, Xinping Yan, and Xue Zhang.* Pavement Distress Image Automatic Classification Based on DENSITY-Based Neural Network, volume 4062. Springer, Berlin, Allemagne, 1ière edition, Septembre 2006.

[17] *Jiawei Han and Micheline Kamber.* Datamining, concepts and technics. Morgan Kaufman, 2ième edition, 2006.

[18] http ://jnesis.com/pourquoi-java.html
Consulter le 08.10.2010 à 00 :01.

[19] http ://futura-sciences.com/fr/definition/t/genetique-2/d/genetique-inverse 4420/
Consulter le 14.06.2010 à 13 :11