Dissertation for the degree of
Doctor of Philosophy in Computer Science

University of Liège

Protein Structural Annotation

Multi-task learning and feature selection

by Julien Becker

Advisor :
Prof. Louis Wehenkel

Co-advisor :
Dr. Francis Maes

Summer 2013

Table des matières

Summary

Experimentally determining the three-dimensional structure of a protein is a slow and expensive process. Nowadays, supervised machine learning techniques are widely used to predict protein structures, and in particular to predict surrogate annotations, which are much less complex than 3D structures.

This dissertation presents, on the one hand, methodological contributions for learning multiple tasks simultaneously and for selecting relevant feature representations, and on the other hand, biological contributions issued from the application of these techniques on several protein annotation problems.

Our first methodological contribution introduces a multi-task formulation for learning various protein structural annotation tasks. Unlike the traditional methods proposed in the bioinformatics literature, which mostly treated these tasks independently, our framework exploits the natural idea that multiple related prediction tasks should be designed simultaneously. Our empirical experiments on a set of five sequence labeling tasks clearly highlight the benefit of our multi-task approach against single-task approaches in terms of correctly predicted labels. Our second methodological contribution focuses on the best way to identify a minimal subset of feature functions, *i.e.*, functions that encode properties of complex objects, such as sequences or graphs, into appropriate forms (typically, vectors of features) for learning algorithms. Our empirical experiments on disulfide connectivity pattern prediction and disordered regions prediction show that using carefully selected feature functions combined with ensembles of extremely randomized trees lead to very accurate models.

Our biological contributions are mainly issued from the results obtained by the application of our feature function selection algorithm on the problems of predicting disulfide connectivity patterns and of predicting disordered regions. In both cases, our approach identified a relevant representation of the data that should play a role in the prediction of disulfide bonds (respectively, disordered regions) and, consequently, in protein structure-function relationships. For example, the major biological contribution made by our method is the discovery of a novel feature function, which has - to our best knowledge - never been highlighted in the context of predicting disordered regions. These representations were carefully assessed against several baselines such as the 10th *Critical Assessment of Techniques for Protein Structure Prediction* (CASP) competition.

Résumé

Déterminer la structure tridimensionnelle des protéines de manière expérimentale est un processus lent et coûteux. De nos jours, pour palier ces difficultés, les techniques d'apprentissage automatique supervisé se sont fortement développées dans le domaine de la prédiction des structures de protéines, et plus particulièrement pour prédire des annotations partielles sous-jacentes à la structure 3D, quoique moins complexes.

Cette thèse présente, d'une part, des contributions méthodologiques dans le domaine de l'apprentissage multitâches (*multi-task learning*), où l'objectif est d'apprendre plusieurs tâches simultanément, ainsi que dans le domaine de la sélection de représentations pertinentes (*feature selection*), c'est-à-dire, des représentations capables d'extraire l'essence de l'information utile pour le problème de prédiction considéré. D'autre part, cette thèse présente également des contributions biologiques directement issues de l'application de nos méthodes à divers problèmes d'annotation de protéines.

Notre première contribution méthodologique réside dans la formulation multitâche de problèmes d'annotation liés à la prédiction de la structure des protéines. Dans la littérature bioinformatique, les problèmes d'annotation des protéines sont habituellement traités de manière indépendante. Or, au vu du lien fort qui unit ces différentes tâches, nous avons développé une méthodologie qui exploite l'idée que des tâches qui semblent liées entre-elles devraient être traitées simultanément. Nos expériences, menées sur un ensemble de cinq problèmes d'annotation de séquences liés à la prédiction de la structure des protéines, nous ont permis de clairement mettre en évidence l'intérêt de notre approche multitâche comparée à une approche qui ne traite qu'une seule tâche à la fois. Notre seconde contribution sur le plan méthodologique se focalise sur la manière d'identifier une représentation pertinente pour un problème particulier, où une représentation "pertinente" est définie comme un sous-ensemble minimal de jeux de variables (*feature functions*) permettant de réaliser la tâche de prédiction considérée. Dans ce contexte, un jeu de variables permet d'encoder une information particulière d'intérêt d'une séquence ou d'un graphe, sous un format exploitable par des algorithmes d'apprentissage automatique (généralement, un vecteur de nombres). Nos expériences pour déterminer le meilleur moyen de représenter les protéines pour la prédiction du motif formé par les ponts disulfures, ainsi que sur le meilleur moyen de représenter les protéines pour la prédiction de régions dites désordonnées, nous ont permis de montrer qu'une représentation soigneusement choisie combinée avec des ensembles d'arbres extrêmement aléatoires permettait d'obtenir des modèles prédictifs très performants par rapport à l'état de l'art.

Notre contribution biologique résulte de l'application de notre méthodologie de sélection de jeux de variables. Dans les deux études que nous avons menées (à savoir, la prédiction du motif formé par les ponts disulfures et la prédiction de régions désordon-

nées), notre approche nous a permis d'identifier des représentations pertinentes pour l'apprentissage de modèles prédictifs et ainsi d'aider à la compréhension de la relation structure-fonction des protéines. Par exemple, notre contribution principale au niveau biologique est la découverte d'un nouveau jeu de caractéristiques qui, à notre connaissance, n'ont encore jamais été mis en évidence dans le cadre de la prédiction de régions désordonnées. La pertinence de ce nouveau jeu caractéristiques a été rigoureusement évaluée par rapport à plusieurs points de comparaison dont la 10ème édition de la compétition CASP (*Critical Assessment of Techniques for Protein Structure Prediction*).

CHAPITRE 1

Introduction

Contents

1.1 Introduction

Proteins play a considerable role in life. They are among the most fundamental molecules of biology that are responsible for most processes that make life possible. The functions of proteins are extremely diverse. Among them, there are *structural proteins* that provide the structure and shape of cells ; *contractile proteins* that are involved in muscle contraction and movement ; *enzymes* that catalyze chemical reactions, *i.e.*, speed up a reaction to a sufficient rates for life ; *regulatory proteins* that control the transcription of genes by binding themselves to segments of the DNA ; *transport proteins* that move molecules from one place to another such as hemoglobin, which transports oxygen through the blood ; *receptor proteins* that cover the surfaces of cells and transfer information from the environment into the cell ; *antibodies* allow the immune system to recognize other proteins ; and this list is far from exhaustive.

Basically, a protein is a one-dimensional sequence of amino acids assembled just like a pearl necklace. Such a linear representation of a protein is called its *primary structure*. It is admitted that the primary structure of a protein determines its *tertiary structure*, *i.e.*, the position of each amino acid in the three-dimensional space when the primary structure is folded. Under suitable environmental conditions (*e.g.*, temperature and pressure), the tertiary structure is generally unique, stable and spontaneously adopted. The process by which the primary structure assumes its shape is called the *protein folding* and is subject

to the laws of thermodynamics, which assert that a system attempts to find a three-dimensional conformation that minimizes the Gibbs free energy. Nevertheless, in biology, there are no rules without exception and sometime additional proteins are required to assist the folding process.

Moreover, it is admitted that the function of a protein is directly determined by its tertiary structure. Generally, a small part of the structure is essential to achieve its function. For example, in the case of enzymatic proteins, this part is called the *active site*. The rest of the structure creates and fixes the spatial relationship. An alteration in the active site, *e.g.*, a suppression of one of the amino acids forming the active site, is often destructive and may lead to the loss of the function. More generally, protein mutation can involve the misfolding of the protein and, consequently, its malfunction, *i.e.*, the protein no more correctly works in the cellular machinery. Malfunctioning proteins cause a broad range of severe diseases. For example, the *Alzheimer's disease* is due to a deposit of misfolding proteins that have aggregated together within the brain causing an atrophy [21] ; a mutation of the regulatory protein $p53$, which occupies the most important role in the cancer resistance network, can cause the uncontrolled proliferation of cells [137] that is the hallmark of tumor formation ; the *diabetes* is a group of diseases related to insulin [125], which is an hormone that regulates sugar concentration in the blood ; mutations of antibodies directly affect the efficiency of our immune system and may lead to *autoimmune disorders*, *i.e.*, the immune system attacks its own healthy tissue. Clearly, the knowledge of the structure of proteins is crucial in medicine and drug design, and more generally in biology and bioengineering.

The great variety of protein structures and functions is explained by the "Sequence → Structure → Function" paradigm and the very large range of possible amino acid combinations. There exist twenty amino acids in human beings. Therefore, the number of possible combinations of ten residues length proteins is $20^{10} = 1.024 \times 10^{13}$ and, in theory, as many structures and functions. In practice, since some amino acids are more or less similar or share a common chemical property, the number of distinct folds and functions is drastically reduced but is still very high. These similarities enable to reduce the effect of mutations on structures. Protein structures are therefore more robust and more preserved than protein sequences over the time. For instance, the median protein length in human cell is ≈ 375 [19] and the number of unique proteins sequences deposited in the public databases is just of 16 million [1].

1.1.1 Protein structure prediction

Experimentally determining the three-dimensional structure of a protein is a slow and expensive process that requires sophisticated experimental techniques. Most popular methods for deriving macromolecular structures are : the *x-ray crystallography*, the *nuclear magnetic resonance* (NMR) spectroscopy and the *electron microscopy*. Among the known protein structures, 88.6% of them are determined by x-ray crystallography and 10.7% by

1. We have used the statistics of the *NCBI Reference Sequences*[102] database release no. 54 (July, 2012) : `http://www.ncbi.nlm.nih.gov/RefSeq/`. The database provides a comprehensive and non-redundant set of sequences of which 16 393 342 are protein primary structures.

NMR spectroscopy. The remaining 0.7% of structures are determined by either electron microscopy or hybrid methods [2].

Despite the advances of experimental techniques, the determination of protein structures is still time- and labor-intensive. On the contrary, the rapid progress of gene sequencing technology enables researchers to determine thousands of novel protein sequences per year. In order to fully understand their role in the organism, each of them should be associated to a three-dimensional structure and to one or several functions. Unfortunately, only a very limited number of them have an experimentally determined structure. For example, among the 16 million of unique proteins sequences mentioned above, just 70 000 protein structures have so far been experimentally determined. This enormous gap between the number of protein sequences and structures has driven research towards computational methods for predicting protein structures from sequences.

Protein structure prediction is thus one of the most venerable of *holy grails* in bioinformatics. The methods developed for predicting protein structures from amino acid sequences can be classified into four categories :

Homology modeling. Two sequences are homologous if they share a common ancestry, *i.e.*, the two sequences were initially the same gene but evolved differently through generations. The homology modeling is based on the observation that, during evolution, the structure is more stable than its sequence, so that homologous sequences still fold into similar structures. All homology modeling approaches consist of a multistep process : a search of homologous fragments among the known structures ; an alignment of those fragments w.r.t the query and an optimization of the model. An overview of current trends in homology modeling in drug discovery is available in [25].

Fold recognition methods. These methods attempt to determine which of known structures share significant similarities with a query primary structure. The advantage of these methods is that they take into account the available three-dimensional structures to predict how well a fold will fit the query. Fold recognition approaches can be particularly useful in the case of divergent evolution, *i.e.*, the sequences similarities are very distant. [121] proposes an overview on good-practice benchmarking for fold recognition methods.

***Ab initio* methods.** These modeling methods attempt to build the native conformation of proteins from scratch by using the principles of thermodynamics. They are based on the assumption that native state of a protein is the one that minimizes the free energy function. Since the application of physical force fields on all atoms of a macromolecules is impossible for today's computational capacity, *ab initio* methods have to deal with two major problems : (i) the protein energy function design, *e.g.*, by restricting the protein representation to its backbone ; and (ii) the vast conformational search space. We refer the reader to [76] for more information about *ab initio* methods.

Structural annotations. The core idea of these methods is based on the assumption that it seems easier to solve a one- or even a two-dimensional problem derived from

2. The percentages are derived from the statistics of the repository of biological macromolecular structures : *Protein Data Bank* [12] : `http://www.rcsb.org/pdb/statistics/` (release of September, 2012).

three-dimensional structure than the three-dimensional structure itself. The goal is to provide accurate indications about the 3D structure without attempting to explicitly assemble them into a 3D structure. A typical example of such a surrogate problem is the secondary structure prediction, which attempts to detect local regular regions in the sequence such as *helixes* or *sheets*.

Our research focuses on structural annotation for several reasons :

- *Limitation of comparative modeling* : homology modeling and fold recognition methods are the most accurate when there exist structure templates that share a (high) sequence identity with the query. Consequently, comparative modeling approaches are limited to known structures.
- *Thermodynamic versus kinetic hypothesis* : there is still a heavy debate [112, 37, 59] whether the native state of a protein is the one that minimizes the Gibbs free energy (thermodynamic hypothesis) or the one that is kinetically accessible (kinetic hypothesis). Regardless of this, *ab initio* approaches focus on thermodynamic stability, probably because it remains easier to formulate the problem as an optimization one.
- *Unreliable ab initio approaches* : in spite of recent progress, *ab initio* approaches are still far to be accurate enough. Moreover, all methods seem to fail on sequences longer than 100 residues. This can be explained by the unavoidable errors introduced by the simplification of the energy function and the fact that, in practice, not all native protein structures reach their global minimum free energy [8].
- *The sequence determines the structure* : according to Levinthal [77], "If the final folded state turned out to be the one of lowest configurational energy, it would be a consequence of biological evolution and not of physical chemistry".

Nowadays, both comparative modeling and *ab initio* methods are still limited. The current known structures limits the former and the latter is limited to proteins that do not show large kinetic barriers in the free energy landscape.

In this work, we believe that there exists a function that given an amino acid sequence determines a unique and stable structure w.r.t. their environmental conditions. However, this function is extremely complex and cannot be solely resumed to a search of the lowest energy. Therefore, we prefer to adopt a *bottom-up* viewpoint that consists in (i) *solving* structural annotation problems, (ii) *gathering* as much structural information as possible and then (iii) *assembling* them to form a three-dimensional structure. In this thesis, we focused on (i) and (ii).

1.1.2 Protein structural annotation

To progress towards the prediction of the tertiary structure of proteins, many research efforts have already been devoted to address surrogate annotation problems. According to the dimensionality of these structural annotations, they can be categorized into three groups :

Global property prediction. The input is a sequence of amino acids and the output is a single value that describes a biochemical property of the whole protein. Well-known examples are the protein function and the localization of the protein within the cell.

Sequence labeling. To each element of the amino acid sequence, the sequence labeling problem consists in assigning a label (numerical or categorical) that describes a residue property. Well-known examples of sequence labeling problems are : *secondary structure prediction*, where labels correspond to local 3D structures such as alpha helices, beta strands or turns ; *solvent accessibility prediction*, where labels are the level of exposition of protein residues to the solvent ; and *disordered regions prediction*, that aims at identifying amino acids belonging to a disordered region, *i.e.*, a region that does not adopt a stable 3D structure.

Contact map prediction. The input is an amino acid sequence of length N and the output is a binary two-dimensional matrix $N \times N$. For each pair of residues (i, j) of the sequence, the corresponding element of the matrix is 1 if the three-dimensional Euclidean distance between the i-th and the j-th amino acid residues is less than a user-defined threshold, and 0 otherwise. It has been shown [43] that the simplification of three-dimensional structures into contact maps is a valid approximation. Various distance definitions have been proposed such as the distance between $C_\alpha - C_\alpha$ atoms, *i.e.*, the carbon atoms that compose the main-chain of the protein ; the distance between $C_\beta - C_\beta$ atoms, *i.e.*, the carbon atoms of residues that are bonded to C_α ; or the distance between the centers of mass of the residues. There also exist simpler maps such as contact maps between β-sheets or contact maps between cysteine residues, which are known to form covalent bonds.

1.2 Contributions

In the bioinformatics literature, these problems have mostly been treated independently, *e.g.*, one designs and uses a predictor for inferring secondary structure and separately designs and uses another predictor for inferring solvent accessibility. On the other hand, the field of machine learning has investigated in the recent years so-called *multi-task* approaches, which aim at treating multiple related prediction tasks simultaneously with the hope to get an improvement on each one of the addressed tasks with respect to predictors designed and used in a single task fashion. Since the various protein structural annotation tasks are closely related, it is a natural idea to explore such multi-task approaches in this context.

In order to deal with the different kinds of data, the first part of our research focuses on the development of a multi-task framework called *iterative multi-task sequence labeling*. Chapter 4 presents this approach. We successfully applied this strategy on the top of a machine learning algorithm to jointly solve a set of protein sequence labeling tasks : secondary structure prediction, solvent accessibility prediction, disordered regions prediction and a rich *structural alphabet*, where labels correspond to local three-dimensional conformations.

However, the wide majority of available machine learning algorithms cannot process complex objects such as secondary structures, disordered regions, etc. They usually require the user to encode such objects into vectors of (categorical or numerical) *features*. Since, the way to perform this encoding typically has a major impact on the predictions, the second part of our research focuses on the best way to identify a minimal subset of feature representations. Chapter 5 introduces a tractable and interpretable *feature se-*

lection methodology, called *forward feature function selection*. We successfully applied this approach on top of ensembles of extremely randomized trees to predict disulfide connectivity patterns and disordered regions. In the former case, the aim was to identify the best way to represent pairs of cysteine residues from an input sequence in order to accurately determine those that form disulfide bridges, *i.e.*, a covalent link between two sulfides atoms. In the latter case, we focus on determining in a similar way a feature representation for identifying residues that do not adopt a stable 3D structure.

1.3 Thesis layout

This thesis is divided into seven chapters. The first chapters are intended to give the reader some background on supervised learning and on structural biology, particularly on protein structures and its structural annotations. The next chapters, comprising Chapter 4 though Chapter 6, describe our contributions in both the field of machine learning algorithms and the field of structural bioinformatics. The last chapter concludes and gives an overview of the perspectives raised by this thesis.

The chapters of this thesis are organized as follows :

Chapter 2 introduces relevant background on supervised learning. In particular, it provides precise descriptions of the machine learning algorithms that we used in our studies : *k-nearest-neighbors*, *support vector machines* and *extremely randomized trees*.

Chapter 3 provides a general background to understand what a protein is made of and how their tertiary structure is characterized. It describes the central dogma of biology : from the DNA to the protein and especially focuses on the paradigm of proteins : the sequence-structure-function relationship.

Chapter 4 introduces a framework for jointly solving several closely related sequence labeling tasks. We named this approach *iterative multi-task sequence labeling*. It then compares the results obtained on a set of five protein annotations to models designed in a single task fashion. It finally shows that the multi-task approach systematically outperforms the single-task approach on all tasks and even leads to a secondary structure predictor that outperforms the state-of-the-art.

Chapter 5 describes experiments on the best way to identify a minimal subset of relevant features for disulfide connectivity pattern prediction. This chapter first introduces a *forward feature function selection* algorithm and then applies this algorithm on various structural annotations, of which the five tasks studied in the previous chapter. The observations indicate that a very limited number of feature representations are sufficient to reach and outperform the state of the art for disulfide pattern prediction.

Chapter 6 applies the feature function selection developed in the previous chapter to the problem of predicting disordered regions. Among the selected feature functions, one is particularly novel in the field of disordered regions prediction. It then assesses a model using these feature functions against the proteins of the 10th CASP competition and shows that this model is very competitive with respect to the state-of-the-art.

Chapter 7 concludes this manuscript and gives some directions for future research.

1.4 Publications

International journals

[1] *Becker (Julien)*, Maes (Francis), Wehenkel (Louis)
On the relevance of sophisticated structural annotations for disulfide connectivity pattern prediction
PLoS One, 2013

[2] *Becker (Julien)*, Maes (Francis), Wehenkel (Louis)
On the encoding of proteins for disordered regions prediction
Accepted for publication in PLoS One, 2013

International conferences

[3] Maes (Francis), *Becker (Julien)*, Wehenkel (Louis)
Iterative multi-task sequence labeling for predicting structural properties of proteins
19th European Symposium on Artificial Neural Networks (ESANN'11)

National conferences

[4] Maes (Francis), *Becker (Julien)*, Wehenkel (Louis)
Prédiction structurée multitâche itérative de propriétés structurelles de proteins
7th Plateforme de l'Association Francaise pour l'Intelligence Artificielle (AFIA'11)

The publications [3,4] describe our work on iterative multi-task sequence labeling approach described in Chapter 4. The publications [1,2] are about our forward feature function selection algorithm. This work is described in Chapters 5 and 6.

CHAPITRE 2

Background in machine learning

Contents

Machine learning is the area of artificial intelligence that studies the ability of computer programs to automatically learn decision rules from examples. This field is also called *statistical learning* or *automatic learning*. One of the fundamental ideas is to extract knowledge from observations in order to recognize complex patterns and make decisions. Over the past sixty years, many techniques and methodologies have been developed characterized by the type of data and by the properties of the decision rules, such as (i) the predictive performance that measures the ability to make correct decision, (ii) the computational scalability, that is, the ability to handle large amounts of data and/or to exploit high-dimensional observation spaces in a reasonable manner in terms of computational resource consumption and (iii) the interpretability, that is, the ability to be confronted to human knowledge domains.

Since the remainder of this manuscript exclusively treats problems where the aim is to learn decision rules given a set of labeled examples, this chapter is intended to provide

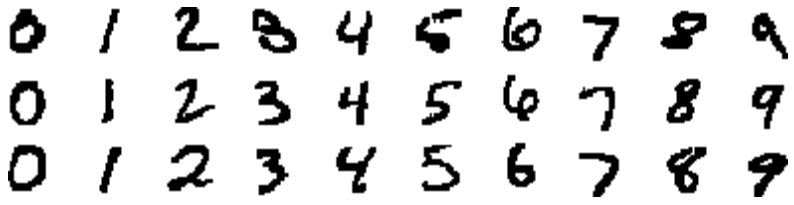

FIGURE 2.1: **Recognition of handwritten digits.** It consists in classifying 28×28 pixels images into a digit. This task can be expressed as a supervised learning problems.

the sufficient understanding of the general family of supervised learning problems. This chapter has not the purpose to give an exhaustive overview of machine learning.

After an introduction to supervised learning in Section 2.1, the four following Sections 2.2, 2.3, 2.4 and 2.5 describe four different families of supervised learning algorithms, which are further used in our experiments. Section 2.2 defines the notion of stochastic gradient descent and focuses on linear models. Section 2.3 shows a way to construct a hyperplane in a linearly separable space and how to deal with non-linear separators. Section 2.4 describes approaches based on decision trees. In particular, it introduces the notion of extremely randomized tree and ensemble-based methods. As a baseline, Section 2.5 presents a very simple method that relies on a distance metric. Finally, Section 2.6 gives an introduction on the importance of selecting relevant descriptions of the observations in order to make better decisions.

2.1 An introduction to supervised learning

Supervised learning aims at inferring a function $f : \mathcal{X} \to \mathcal{Y}$ that predicts the value of an output object $y \in \mathcal{Y}$ based on an input object $x \in \mathcal{X}$. The function f is induced from a collection D of n training examples $D = \{(x^{(i)}, y^{(i)})\}_{i \in [1,n]}$, sometimes called observations, sample or dataset.

A classical example to depict supervised learning is the problem of recognizing handwritten digits, illustrated on Figure 2.1. Each digit corresponds to a 28×28 pixels image. The goal is to induce a function f that is able to recognize the digit : f : $\mathsf{image} \to \{0, 1, \ldots, 9\}$.

Usually, supervised learning makes sense on problems where the observation of $\mathcal{X}$ is easy or cheap, while the observation of $\mathcal{Y}$ often requires human expertise and is costly. It is particularly the case in many fields of biological research such as in *genome annotation*, where high-throughput sequencing methods can rapidly sequence a whole genome while the annotation process requires human expertise ; or in *protein structure prediction*, where the three-dimensional determination of the structure requires very sophisticated and time-consuming experiments.

Moreover, supervised learning commonly assumes that the distribution of the training set D is representative of the actual joint distribution $P(\mathcal{X}, \mathcal{Y})$ of input-output pairs. In other words, if we would like to learn an function f, the learning examples must be independently and identically drawn from the joint distribution of the variables $(\mathcal{X}, \mathcal{Y})$, which is usually unknown.

2.1.1 Examples of supervised learning tasks

Supervised learning embraces various different learning tasks, depending on the nature of the output space $\mathcal{Y}$. In the example of handwritten digits, the output space is a finite set of ten labels $\{0, 1, \ldots, 9\}$. Such problems can be expressed as a multiclass classification tasks.

Binary classification. Binary classification is the task of classifying objects into two possible classes, by convention : $\mathcal{Y} = \{-1, +1\}$. Examples of binary classification problems include i.) *medical diagnosis*, that determines if a patient has a certain disease on the basis of several measurements (age, height, weight, smoker, etc.) and tests (rest heart rate, blood pressure, blood type, etc.) while $\mathcal{Y} = \{\mathsf{unhealthy}, \mathsf{healthy}\}$; ii.) *quality control*, that is the process by which a factory tests the quality of their products to uncover defects and discard them ($\mathcal{Y} = \{\mathsf{defect}, \mathsf{not\ defect}\}$) ; iii.) *spam filtering*, that is the processing that filters unsolicited email messages such as advertising ($\mathcal{Y} = \{\mathsf{spam}, \mathsf{not\ spam}\}$).

Multi-class classification. Multi-class classification is the problem of classifying objects into more than two possible classes. The large body of multi-class classification problems can be expressed as *document classification* problems. Document classification consists to assign a document to one or several genres. The documents to be classified may be texts, where the goal is to predict the topic of the document according to predetermined categories (*e.g.*, $\mathcal{Y} = \{\mathsf{fantastic}, \mathsf{romantic}, \mathsf{fiction}, \mathsf{horror}\}$) ; music, where the goal is to classify pieces of music with respect to their genre ; images, as illustrated in the handwritten digit recognition problem ; movies, etc.

Regression. Regression focuses on problem with scalar outputs $\mathcal{Y} \subseteq \mathbb{R}$. Examples of regression problems include *dating objects*, which aims to estimate the date of an object based on observable evidences such as tree rings or carbon-14. The problem of dating objects belongs to a larger family of problems known as *calibration*. It also includes *stock market prediction*, where the goal is to determine the future value of a company stock, *earthquake forecasting*, which attempts to predict the magnitude or time of occurrence of a future earthquake.

Structured prediction. Structured prediction concerns tasks with arbitrary complex outputs such as sequences, trees or graphs. This stands in contrast to previous approaches, where an input data is mapped to a single value (discrete or continuous). Structured prediction corresponds to a wide range of real-world problems. Examples where structured output object arise include *sequence labeling*, which assigns a label to each member of an input sequence ; *parsing*, which assigns a concrete syntax tree (*i.e.*, a syntactic structure according to some formal grammar) to an input sentence. A well-known example of parsing is *machine translation*, which aims at automatically translating a sentence from a language to another. Sequence labeling is extensively used in *computational biology*, where the wide majority of problems consist to map an input sequence of DNA, RNA or amino acids into a labeled sequence of the same length, where a label describes a physico-chemical property. Figure 2.2 gives an application of sequence labeling : the secondary structure prediction. In this example, the input is a sequence of amino acids $(x_1, x_2, \ldots, x_T)$ where each element $x_t \in \{\mathsf{alanine}, \mathsf{arginine}, \ldots, \mathsf{valine}\}$ is mapped into a label $y_t \in \{\mathsf{sheet}, \mathsf{helix}, \mathsf{coil}\}$ in order to form an output sequence $(y_1, y_2, \ldots, y_T)$.

Multi-task learning. Multi-task approaches aim at treating multiple related prediction tasks simultaneously. This stands in contrast to previous approaches, where re-

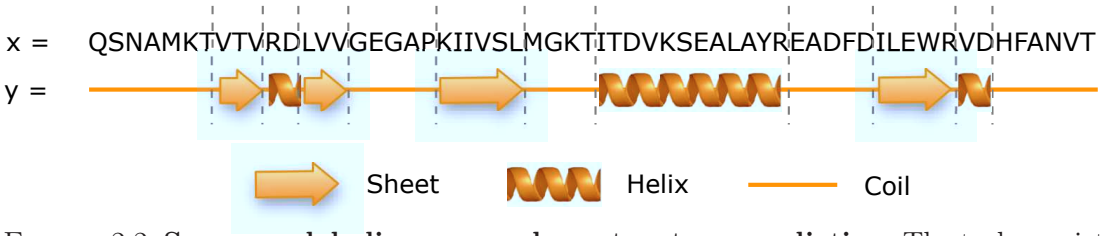

FIGURE 2.2: **Sequence labeling : secondary structure prediction.** The task consists in classifying each member of the primary structure into one of three possible classes : sheet, helix or coil. This example is a part of the protein 4GUG.

FIGURE 2.3: **Multi-task learning.** The goal is to jointly classify images (x) into human or animal (y_1) and into gender of human subjects (y_2).

lated problems are treated independently, *i.e.*, one designs and uses a predictor for inferring a task and designs and use another predictor for inferring another task related to the first one. The purpose is then to learn a mapping from input $x \in \mathcal{X}$ to targets $y_1, y_2, \ldots, y_T \in \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_T$ for each task $t \in [1, T]$. The training set is therefore composed of pairs composed of an input associated with some or all of the target tasks $D = \{(x^{(i)}, y_1^{(i)}, y_2^{(i)}, \ldots, y_t^{(i)})\}_{i \in [1,n]}$.

Figure 2.3 illustrates an example of multi-task learning. Here, inputs are images and the goal is to distinguish humans from animals and girls from boys $\mathcal{Y} = \{\text{human}, \text{animal}\} \times \{\text{girl}, \text{boy}, -\}$.

Many real-world problems can be formulated as multiple related classification or regression tasks. For example, in *therapy determination*, the tasks are the estimation of the effectiveness of several drugs for a given patient. In *disease progression*, the task of predicting the evolution of a disease at different time points can be considered as a multi-task problem where the tasks are temporally related. In the field of *natural language processing*, which studies the interactions between computers and human languages, there exist several related tasks. These tasks include *part of speech tagging*, which aims at labeling each word of a sentence with a tag that indicates its syntactic role as plural noun, verb, adjective, etc. *chunking*, which aims at labeling segments of a sentence with a syntactic tag ; *named entity recognition*, which labels each element in a sentence into a category.

In some sense, multi-class classification can be expressed as a multi-task problem using a *one-versus-all* strategy. The idea is to consider each class $t \in \{1, 2, \ldots, T\}$ as a binary classification problem, where the goal is to distinguish data that belong to the t-th

class from the others. Especially under the constraint that if the t-th predictor classifies an example in the t-th class, all the other r predictors, $r \neq t$, should not classify such an example in their class. Without this constraint, the approach can assign multiple labels to inputs. Such problems are called *multi-label* classification problems.

2.1.2 The global architecture of supervised learning algorithms

Machine learning algorithms are rarely able to process complex input objects directly (*e.g.*, images, music, protein, etc.), hence it is necessary to define a mapping from these input objects to a standardized representation. The wide majority of available supervised learning algorithms require the user to encode such inputs into vectorial representations $\phi : \mathcal{X} \to \mathbb{R}^d$. The elements of $\phi(\cdot)$ are called *features*. For example, when dealing with images a possible choice is to have one feature per pixel and per channel (*e.g.*, red, green and blue channels). In the case of recognizing handwritten digits (Figure 2.1), an input 28×28 pixel grayscale image x can be represented by a vector $\phi(x)$ of 784 real numbers (one per pixel). The goal is then to build a function $f : \phi(\mathsf{image}) \to \mathsf{digit} = \mathbb{R}^{784} \to \{0, 1, \ldots, 9\}$. *Feature*

The way to perform this encoding typically has a major impact on the performance of predictors. In many cases, extracting the appropriate features is an art rather than a science as it often relies on domain knowledge, heuristics and intuition.

The global architecture of a supervised learning approach consists of a *training phase* and an *inference phase*. Given a feature representation $\phi : \mathcal{O} \to \mathcal{X}$, where $\mathcal{O}$ is the space of observations, and given a data set $D = \{(\phi(o^{(i)}), y^{(i)})\}_{i \in [1,n]}$ of input-output pairs, called in this context the *training set*, the *learning algorithm* learns a *model* of the function $f : \mathcal{X} \to \mathcal{Y}$. This model is further exploited by the *inference algorithm* that performs predictions of new input objects. Figure 2.4 summarizes the process of supervised learning approaches. *Learning* *Inference*

The training phase can be deterministic or stochastic. In the former case, two calls of the training algorithm always produce the same model on an identical learning set. It is not always the case in stochastic algorithms, where the model produced by the learning algorithm may vary from one call to another, even if applied to a same learning set.

2.1.3 Supervised learning and the theory of risk minimization

From a statistical learning point of view, all supervised learning problems can be formalized as expected risk minimization problems. The statistical machine learning framework stems on two fundamental concepts.

The first fundamental concept is the hypothesis that the learning examples $D = \{(x^{(i)}, y^{(i)})\}_{i \in [1,n]}$ are independently and identically drawn from the joint distribution over the input and output spaces $P(\mathcal{X}, \mathcal{Y})$, that is, $D \sim P^n(\mathcal{X}, \mathcal{Y})$.

The second fundamental concept is the notion of *loss function* $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. The loss function measures how bad is the error between the prediction $\hat{y} = f(x)$ and the actual output y. Higher $\Delta(\hat{y}, y)$ is, worst the prediction $\hat{y}$ will be. Depending on the learning tasks being solved (*e.g.*, classification, regression, etc.), the choice of the loss *Loss function*

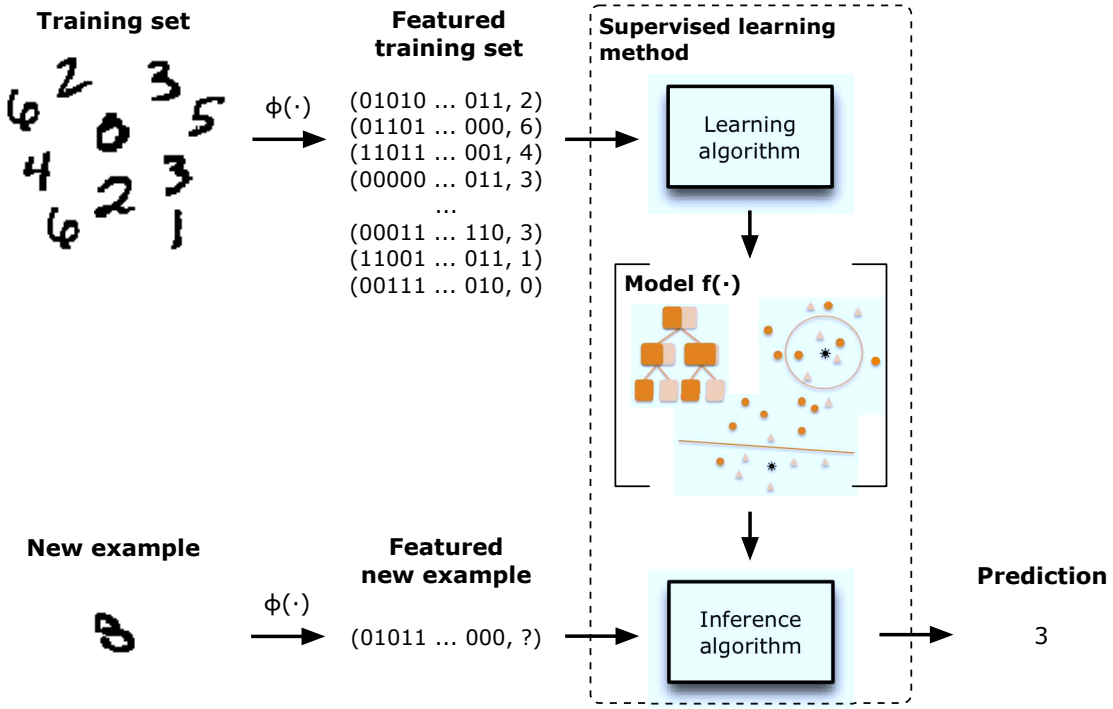

FIGURE 2.4: **Global architecture of a supervised learning.** The raw training set is converted into vectorial representations according to the feature function ϕ. Based on these feature vectors, the learning algorithm builds a model f. According to this model, the inference algorithm is able to make predictions for new input examples.

function may vary. A typical loss function for classification problems is the *zero-one loss*, which equals zero if the prediction is correct and one otherwise :

$$\Delta^{0/1} = \begin{cases} 0 & \text{if } \hat{y} = y \\ 1 & \text{if } \hat{y} \neq y \end{cases},$$

whereas a typical loss function for regression problems is the *squared error* given by :

$$\Delta^{squared} = (\hat{y} - y)^2.$$

Expected risk

Supervised learning aims at seeking the function f that minimizes the *expected risk* $R(f)$, *i.e.*, the expectation of loss over the joint distribution $P(\mathcal{X}, \mathcal{Y})$ and defined as follows :

$$R(f) = E_{P(\mathcal{X},\mathcal{Y})}\{\Delta(f(x), y)\}.$$

Empirical risk

Unfortunately, it is generally not possible to solve this problem exactly, because the distribution $P(\mathcal{X}, \mathcal{Y})$ is unknown. However, thanks to the first fundamental concept, supervised learning can approximate the expected risk by calculating the *empirical risk* $\hat{R}(f)$, which is the average loss over the instances of the training set :

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \Delta(\hat{y}^{(i)}, y^{(i)}),$$

where each $\hat{y}^{(i)} = f(x^{(i)})$. Therefore, since the empirical risk converges to the expected risk as n goes to infinity (according to the law of large numbers), supervised learning becomes the problem of selecting the function $f^* \in \mathcal{F}$, where $\mathcal{F}$ denotes a set of candidate functions, that minimizes the empirical risk :

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}(f) \simeq \underset{f \in \mathcal{F}}{\operatorname{argmin}} R(f).$$

However, the minimal empirical risk does not necessarily converge to the minimal expected risk for an arbitrary function space $\mathcal{F}$ as n goes to infinity (i.e. the learning algorithm is not necessarily consistent). Also, for finite on even infinite samples, the loss induced by the inability of f^* to reach the minimal expected risk on average is called the *bias* of the learning algorithm. Moreover, the function f^* may be sensitive on peculiarities of the training set used to learn it. Namely, several instances of the learning algorithm on different data sets of the same size n will produce different models. The variation of these models may lead to different predictions for an input x. The measure of how strongly depends a function f^* on the training set is called *variance*. Both bias and variance contribute to suboptimal expected risk values of a learnt model.

Consistency

Bias

Variance

When learning a model, the empirical risk decreases, in a way that has a joint effect on the bias, that decreases, and the variance, that increases. In particular, when the model perfectly fits the training data, the bias is often very small whereas the variance may be rather large. Due to the fact that the empirical risk is calculated on training examples, which are used to optimize the model, the empirical risk normally underestimates the actual risk (generalization risk), *i.e.*, the expected value of the loss function on new examples that are independent from those that are in the training set. This phenomenon is known as *overfitting*. Generally, there is a tradeoff between bias and variance, *i.e.*, a tradeoff between the complexity of the function f and the generalization error. Figure 2.5 illustrates this phenomenon.

Overfitting

In order to avoid the overfitting phenomenon, where the model is too strongly tailored to the particularities of the training set and generalizes poorly to new data, it is common to insert a *regularization term* $\Omega(\cdot)$. The term $\Omega(f)$ aims at measuring the complexity of f. Higher the complexity of f is, higher $\Omega(f)$ will be. Therefore, supervised learning becomes a problem of selecting a function f, which is a good compromise between low empirical risk and simplicity (and consequently, generalizability) :

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}(f) + \lambda \Omega(f),$$

where λ is a user-defined parameter that weights the regularizer so as to control the tradeoff between the minimization of the empirical risk and the minimization of the regularizer. This function is known as the *regularized empirical risk* or *structural risk minimization.*

Regularized empirical risk

The direct estimation of the generalization error on an independent testing set is not always possible. For example, in many fields, the amount of data available is limited and a large body of studies prefers to use as many as possible observations in order to obtain better predictors. In such a situation, a widely used technique is the *k-fold cross-validation*. It consists in running the learning algorithm once for each of k different non-overlapping train/test splits of the data. In this setting, the learning algorithm is

k-fold cross-validation

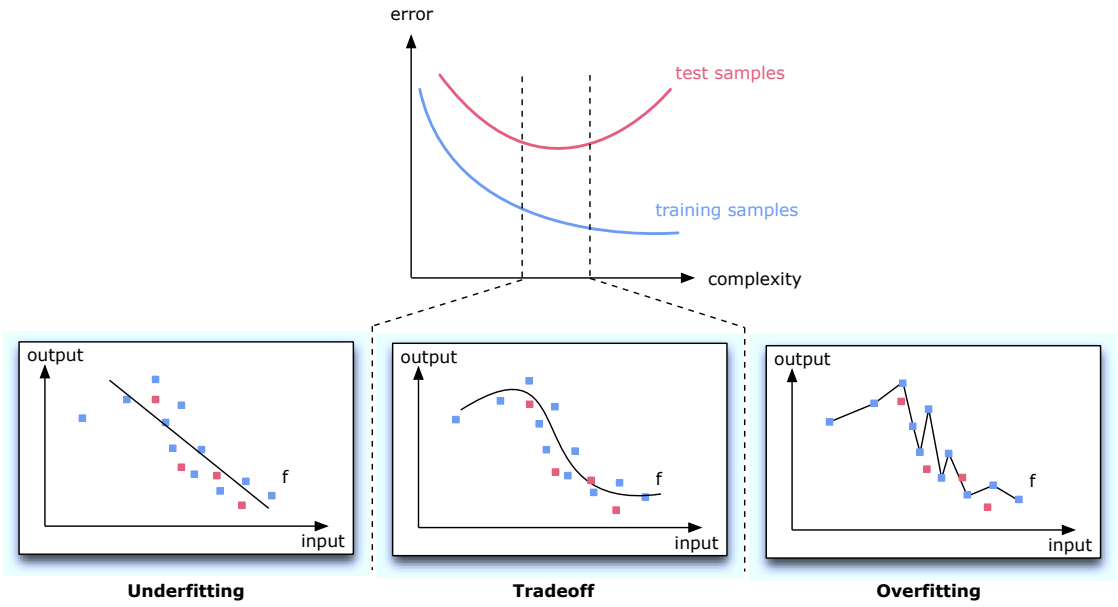

FIGURE 2.5: **Bias/variance tradeoff.** Curves on the top represent a typical behavior of a loss function obtained when the complexity of the model varies. Bottom plots represent three possible states of the model on a one-dimensional regression problem. Bottom-left : the model is too simple and both training and testing errors are high. Bottom-middle : the model is a good compromise between complexity and generalization error. Bottom-right : the model is too complex, it perfectly fits training samples at the price of generalization error.

executed k times on a training dataset composed of the union of $k-1$ splits of the data and the learnt function is evaluated using the remaining k-th split of the dataset. The generalization error is then calculated by averaging the error computed over the k models thus learnt.

For more details, one can refer to one of the multiple existing books introducing supervised learning [14, 129, 60]. We describe below four supervised learning techniques used in this thesis : stochastic gradient descent methods (Section 2.2), support vector machines (Section 2.3), extremely randomized trees (Section 2.4) and k-nearest neighbors (Section 2.5).

2.2 Stochastic gradient descent for linear models

Given a family $\mathcal{F}$ of functions f_θ parameterized by a weight vector $\theta \in \mathbb{R}^d$, gradient-based learning aims at minimizing the regularized empirical risk $\hat{R}(f_\theta) + \lambda\Omega(f_\theta)$ using a *gradient descent*. Namely, such approach aims at finding the best parameter θ^* such as :

Gradient descent

$$\theta^* = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}}\, \hat{R}(f_\theta) + \lambda\Omega(f_\theta),$$

using an iterative procedure that updates the weight vector θ on the basis of the gradient of the regularized empirical risk at each iteration :

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\partial(\hat{R}(f_\theta) + \lambda\Omega(f_\theta))}{\partial\theta},$$

where α_t, called the *learning rate*, is a parameter that weights the gradient so as to control the convergence speed of the descent.

Nevertheless, this algorithm requires computing the gradient of the regularized empirical risk exactly. This involves calculating the average loss over the instances of the training set at each iteration. A drastic simplification of the gradient descent is the *stochastic gradient descent*, which estimates this gradient on the basis of a single example (x, y) : *Stochastic gradient descent*

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\partial(\Delta(f_\theta(x), y) + \lambda\Omega(f_\theta))}{\partial\theta_j}.$$

In this manuscript, we focus on stochastic gradient descent method for linear models with a *hinge loss* function :

$$\Delta(f_\theta(x), y) = \begin{cases} 1 - y.F_\theta(x) & \text{if } yF_\theta(x) \leq 0 \\ 0 & \text{otherwise} \end{cases},$$

where $F_\theta(x) = \langle\theta, x\rangle$ is the *scoring function*. In our case, the scoring function is defined as the dot product between the parameters and the input features. Moreover, we only consider a widely used regularizer in *gradient descent* : the l_2-norm of the parameters $\|\theta\| = \sqrt{\theta_1^2 + \theta_2^2 + \cdots + \theta_d^2}$.

For prediction purposes, and in order to deal with the nature of the outputs, different function spaces $\mathcal{F}$ are proposed. The simplest one is the case of regression, where candidate functions f_θ are identical to the scoring functions F_θ :

$$f_\theta^{regression}(x) = F_\theta(x).$$

In the case of binary classification, the output space is $\mathcal{Y} = \{-1, +1\}$ and the candidate functions are defined as :

$$\begin{aligned} f_\theta^{binary}(x) &= sign(F_\theta(x)) \\ &= \begin{cases} -1 & \text{if } F_\theta(x) < 0 \\ +1 & \text{if } F_\theta(x) > 0 \end{cases} \end{aligned}$$

The extension to multiclass problems requires defining one weight vector θ_y per class $y \in \mathcal{Y}$. The function space is the set of functions defined by :

$$f_\theta^{multiclass}(x) = \underset{y\in\mathcal{Y}}{\text{argmax}}(F_{\theta_y}(x)).$$

That is, given the parameters θ_y, the prediction for an input x is the class that has the highest score.

The biggest advantage of learning a linear model learned with a stochastic gradient descent is the scalability of the method, that is, the ability to handle large amounts of data in a reasonable manner. Indeed, the learning complexity is linear with respect to the number of training examples n and the dimensionality d of the input feature space. The second advantage is the efficiency of making predictions of a new input, which solely consists in performing a dot product.

2.3 Kernel methods and support vector machines

The present section provides a description of support vector machines and kernel-based support vector machines, which are able to deal to with non-linear data. As the mathematics behind support vector machines is somewhat intimidating, the explanation is divided into two parts. The first part (Section 2.3.1) gives a general introduction with few mathematical considerations, whereas the second part (Section 2.3.3) gives a more detailed description on the related convex optimization problem and may not be accessible to readers without a mathematical background.

2.3.1 Introduction

Support vector machines (SVM). SVM is a supervised learning method used for binary classification problems $\mathcal{Y} = \{-1, +1\}$. SVM algorithm was originally invented by Vladimir N. Vapnik in the nineties [35]. In this section, we focus on the classical binary classification SVM. However, it can easily be extended to multiclass classification problems by reducing single multiclass problems into multiple binary classification problems (*e.g.*, via *one-against-all* or *one-against-one* strategies). There also exists a version for regression problems, named *support vector regression*.

Support vector machine

In simple words, the basic idea of SVM is to construct a hyperplane in $\mathcal{X}$ that separates all training examples $(x^{(i)}, y^{(i)})$ into two groups $\{-1, +1\}$ by a clear gap that is as wide as possible. Therefore, a new example $x \in \mathcal{X}$ is classified based on which side of the hyperplane it is. More formally, the family of candidate functions $f \in \mathcal{F}$ is given by the weighted sum :

$$f(x) = sign\left(\sum_{i=1}^{n} y^{(i)} \alpha_i \langle x^{(i)}, x\rangle + b\right),$$

where each training example $(x^{(i)}, y^{(i)})$ is weighted by a coefficient $\alpha_i \geq 0$.

Figure 2.6 illustrates the intuitive idea of SVM on a two-dimensional problem. There exist an infinite number of hyperplanes that separate the two classes. Among them, the SVM selects the one that maximize the margin, *i.e.*, the distance between the hyperplane and the closest samples. Samples on the margin are called the *support vectors*.

Kernel support vector machines. However, it is often the case that input spaces $\mathcal{X}$ are not linearly separable. In order to tackle non-linear problem, Boser *et al.*[15] have proposed to replace every dot product $\langle \cdot, \cdot \rangle$ by a non-linear kernel function $K(\cdot, \cdot)$ (as

Kernel

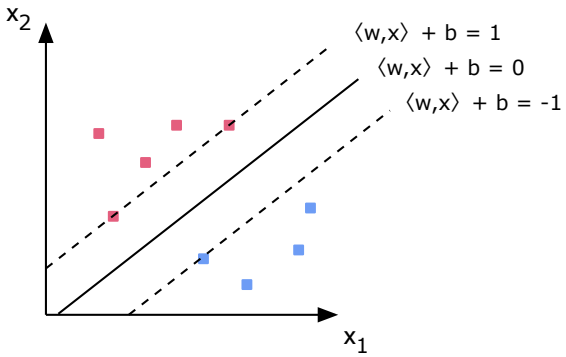

FIGURE 2.6: **Idea of support vector machine on a two-dimensional problem.** The plain line is the hyperplane that separate examples into two classes with the largest margin. Dashed lines represent the bounds of the margin. Samples on the margin are the support vectors.

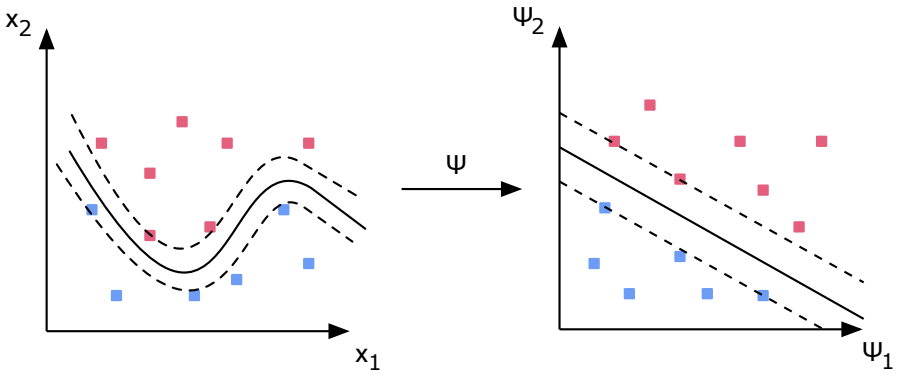

FIGURE 2.7: **Idea of kernel-based support vector machines on a two-dimensional problem.** Left : the original data distribution is not linearly separable. Right : according to the mapping function ψ, the data distribution in the new space is linearly separable.

illustrated in Figure 2.7) :

$$f(x) = sign\left(\sum_{i=1}^{n} y^{(i)}\alpha_i K(x^{(i)}, x) + b\right).$$

The goal of the kernel $K(x^{(i)}, x)$ is to measure how similar x is to example $x^{(i)}$. Nevertheless, in order to correctly use this *kernel trick*, K must satisfies a mathematical constraint : K must be positive definite. According to the constraint, it was shown that K can be represented as an inner product $K(x^{(i)}, x) = \langle\psi(x^{(i)}), \psi(x)\rangle$, where ψ is a mapping from a general observation space into an inner product space that can have a higher and even infinite dimension. The goal of such mappings is to obtain a new space that is linearly separable. Fortunately, based on Mercer's theorem [90], it is not necessary to know an explicit representation of ψ due to the fact that the computation only involves the kernel and the training examples.

2.3.2 Common kernel functions

Among the kernel functions that are known to satisfy the conditions of Mercer's theorem, some of them are common :

Linear kernel. The linear kernel is the simplest kernel function as it is given by the dot product of its inputs.

$$K(z, x) = \langle z, x \rangle.$$

Polynomial kernel. The polynomial kernel is a non-stationary kernel and seems to be well suited for problems where all the training data are normalized.

$$K(z, x) = (\langle z, x \rangle)^m,$$

where m is a constant hyper-parameter that correspond to the degree of the polynomial.

Gaussian radial basis kernel. The Gaussian kernel or normal distribution function is particularly widely used in pattern recognition. These kernels are rotationally and translationally invariant, *i.e.*, their level curves are circles.

$$K(z, x) = exp(-\gamma \|z - x\|^2),$$

where $\gamma > 0$ is a shape (or scale) hyper-parameter that tune the shape of level circles.

Hyperbolic tangent. The hyperbolic tangent kernel comes from the *neural networks* field, where the sigmoid function is often used as an activation function for neurons.

$$K(z, x) = \tanh(\gamma \langle z, x \rangle + c),$$

where the slope α is an adjustable hyper-parameter and the intercept c is a constant hyper-parameter.

2.3.3 Beyond support vector machines

As presented in the introduction, the goal of SVM is to construct a hyperplane $F(x)$ in the space $\mathcal{X} \in \mathbb{R}^d$ to separate two classes.

$$F(x) = \langle w, x \rangle + b = 0,$$

where the $w \in \mathbb{R}^d$ is the normal vector of the hyperplane and $b \in \mathbb{R}$ determines the offset from the origin. A new example x is classified based on the sign of $F(x)$:

$$f(x) = \text{sign}\left(F(x)\right).$$

Hard margin. In *hard margin* point of view, the model assumes that the space is linearly separable, *i.e.*, the hyperplane perfectly separate the two classes. As multiple hyperplanes may be possible, a heuristic is to select the one that maximizes margins from both classes and will lead to the smallest generalization error. This problem can be viewed as a problem of maximizing the distance between two parallel hyperplanes

$\langle w, x\rangle + b = +1$ and $\langle w, x\rangle + b = -1$ that separate the data. Figure 2.6 gives an illustration of these hyperplanes.

Therefore, we wants to maximize the distance $\frac{2}{\|w\|}$, so, minimize $\|w\|$. As the computation of the norm $\|w\|$ involves a square root, which makes the minimization difficult, the use of a mathematical trick modified the equation to :

$$\min_w \frac{1}{2}\|w\|^2,$$

subject to the constraint that each sample $(x^{(i)}, y^{(i)})$ must satisfies the condition :

$$y^{(i)}(\langle w, x^{(i)}\rangle + b) \geq 1.$$

This non-linear quadratic optimization problem can be solved by considering Lagrangian duality. The problem can be expressed as :

$$\min_{w,b} \max_{\alpha_i, \forall i} \left\{ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y^{(i)}(\langle w, x^{(i)}\rangle + b) - 1] \right\},$$

where $\alpha_i \geq 0, \forall i$, are the Lagrange multipliers and must satisfy $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$. As the equation is subject to the *Karush-Kuhn-Tucker condition* :

$$w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)},$$

it is possible to substitute this condition in the Lagrangian function that gives the following *dual* optimization problem :

$$\max_{\alpha_i, \forall i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)}\rangle,$$

subject to $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$ with $\alpha_i \geq 0, \forall i$.

Once the dual problem is solved, the hyperplane $F(x)$ can be expressed as :

$$F(x) = \langle w, x\rangle + b = \sum_{i=1}^{n} y^{(i)} \alpha_i \langle x^{(i)}, x\rangle + b,$$

by substituting w with the Karush-Kuhn-Tucker condition. The solution of the dual problem is usually sparse and has a significant number of α_i coefficients equal to zero. Samples associated to a non-zero α_i are the *support vectors*.

Soft margin. However, in practice, a perfect separation of classes is often not possible. This may be due to an overlapping in the distribution of data or the presence of noises in the observations. For dataset that are not linearly separable, Cortes and Vapnik[35] proposed the *soft margin* method that allows the margin to make few mistakes, *i.e*, some samples are inside or on the wrong side of the margin.

In order to construct a hyperplane that splits the data as cleanly as possible, the soft margin methods introduces slack variables $\xi_i \geq 0$, which measure how misclassified the training example $x^{(i)}$ is. The optimization problem becomes :

$$\min_{w,\xi_i,b} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \right\},$$

where $C > 0$ is a user-defined parameter that weight the importance of slack variables so as to control the tradeoff between the maximization of the margin and the minimization of the slack variables. Note that when $C \to +\infty$, the problem is equivalent to the hard margin one. The equation is subject to the constraint that each example $(x^{(i)}, y^{(i)})$ must satisfy the condition :

$$y^{(i)}(\langle w, x^{(i)}\rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

As done in hard margin, the problem of minimizing $\|w\|$ can be solved by considering Lagrangian duality :

$$\min_{w,\xi_i,b} \max_{\alpha_i,\beta_i,\forall i} \left\{ \frac{1}{2}\|w\|^2 + \ C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[y^{(i)}(\langle w, x^{(i)}\rangle + b) - 1 + \xi_i] - \sum_{i=1}^{n} \beta_i\xi_i \right\},$$

with $\alpha_i \geq 0$ and $\beta_i \geq 0$.

Similarly to hard margin, it is possible to formulate the dual problem. The maximization problem is identical to the one of hard margin except for the constraints on parameters α_i :

$$\max_{\alpha_i,\forall i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y^{(i)}y^{(j)}\langle x^{(i)}, x^{(j)}\rangle,$$

subject to $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$ with $0 \leq \alpha_i \leq C, \forall i$.

Finally, once the dual problem is solved, we obtain the same hyperplane formulation $F(x)$ as previously :

$$F(x) = \langle w^T, x\rangle + b = \sum_{i=1}^{n} y^{(i)}\alpha_i\langle x^{(i)}, x\rangle + b.$$

For more details, the reader is invited to refer to the books of Cortes *et al.*[35] and Cristianini *et al.*[36].

2.4 Tree-based ensemble methods for classification

Decision trees are machine learning methods based on a hierarchical tree structure. Decision trees learn to organize data in a hierarchical way based on questions on the input variables $x \in \mathcal{X}$. Decision trees can be used to solve classification problems or regression problems. Since the organization depends on the answers (and not directly on x), the input space $\mathcal{X}$ can potentially be any kind of object under the condition to

provide questions that can deal with such objects. Commonly, $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d$, where each $\mathcal{X}_i$ can be either continuous or (finite) discrete. The elements of x are called *attributes* with respect to *features* that are restricted to be in $\mathbb{R}$.

Tree

Tree. A *tree* is defined recursively as a node, called the *root node*, which consists of a value and a list of nodes, called the *children nodes*. Note that all the nodes are different. Symbolically, this definition can be written as :

$$
\begin{aligned}
&\textit{tree} ::= \textit{node} \\
&\textit{node} ::= \textit{value} \\
&\textit{node} ::= \textit{value}, \textit{list-of-node} \\
&\textit{list-of-node} ::= \textit{node} \\
&\textit{list-of-node} ::= \textit{node}, \textit{list-of-node}
\end{aligned}
$$

This definition recursively describes the structure of a tree in the sense that a tree (line 1) is simply a node (the *root node*) and that a node is either composed of a value (line 2) or composed of a value and a list of children nodes (line 3), which at their turn can be composed of several children nodes. Nodes with no children are called *leafs* or *terminal nodes* while nodes with at least one child are called *internal nodes* or *interior nodes*.

Decision tree

Decision tree. A *decision tree* [18] is a tree where internal nodes correspond to *test nodes* with a child node for each of the possible answers (*e.g.*, yes or no, true or false, pass or fail). A test node consists of a question about one of the input attribute $\mathcal{X}_i$. The question depends of the type of $\mathcal{X}_i$. In case of a continuous attribute $\mathcal{X}_i \in \mathbb{R}$, the question may be whether the value x_i is greater than a (learned) threshold. The question for a finite discrete attribute may be whether x_i belongs to a (learned) subset of values. In decision trees, a terminal node represents a predicted output value that corresponds to the majority class (for classification) or the mean of the output (for regression) taken over the training samples that belong to that node.

Figure 2.8 shows the structure of a decision tree. This example modelizes the decision process of a marathoner to run or not run ($\mathcal{Y} = \{\mathsf{run}, \mathsf{don't\ run}\}$) based on climate conditions $\mathcal{X} = \mathcal{X}_{outlook} \times \mathcal{X}_{humidity} \times \mathcal{X}_{windy}$, where $\mathcal{X}_{outlook} = \{\mathsf{sunny}, \mathsf{overcast}, \mathsf{rain}\}$, $\mathcal{X}_{humidity} \in [0, 100]$ and $\mathcal{X}_{windy} = \{\mathsf{true}, \mathsf{false}\}$.

In order to make one prediction, the basic idea is quite simple. The new example traverses the tree from the root node to a leaf. At each encountered internal node in the tree, the algorithm answers the associated question and follows the branch that corresponds to the outcome. In the case where the branching operation leads to a leaf, the algorithm returns the output value associated to this leaf. Otherwise, the algorithm recursively follows the branches until it reaches a leaf. Therefore, making predictions is fast and the calculation is easy as it just consists of an if-then-else cascade.

In the classification problem illustrated on Figure 2.8, the prediction of the example $x = \{\mathsf{sunny}, 83, \mathsf{false}\}$ is Don't run and the path of nodes form the root to the leaf is Outlook? $\rightarrow$ Humidity < 70? $\rightarrow$ Don't run.

In practice, decision trees are represented by binary trees, *i.e.*, internal nodes have at most two children. A node with more than two children can always be accommodated as

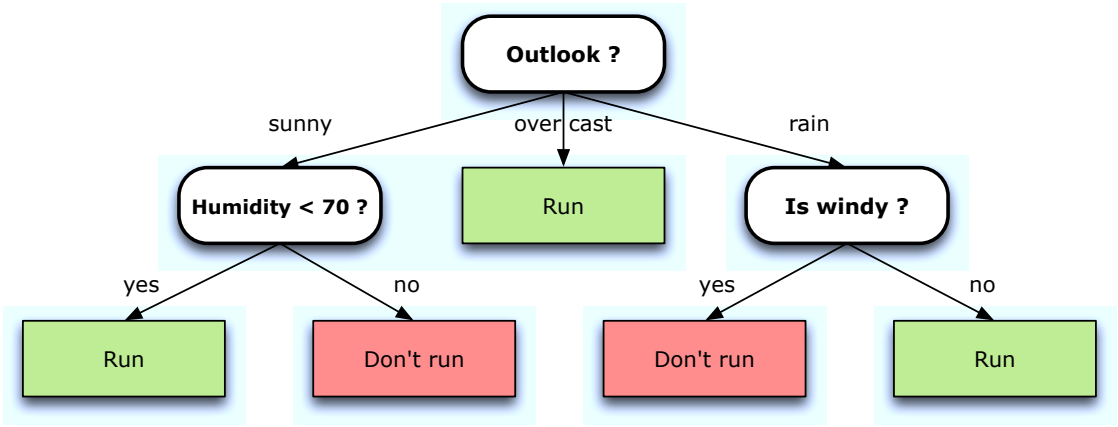

FIGURE 2.8: **Structure of a binary classification decision tree.** Internal nodes (in white) perform a test on one attribute. Each internal node has as many children nodes as possible outcomes of the test. Terminal nodes (in red or green) are labeled by the output value.

multiple binary nodes. In the same way, asking a question about multiple attributes at once is equivalent to asking multiple questions about single attribute.

Each internal node must carefully select its question in order to create the most efficient and, consequently, the smallest tree. Unfortunately, finding the smallest tree that minimizes the training error is known to be a NP-complete problem [62]. Namely, there is no known efficient way to construct such an optimal tree. However, tree-based learning algorithms are based on greedy algorithms that make locally optimal sub-trees (or nodes) with the hope of finding the global optimal tree.

The common way to construct decision tree is a to adopt a top-down approach, starting from the root node to leafs. The algorithm starts with all the training samples D and creates the most efficient internal node, *i.e.*, determines the question on one attribute that best splits the data : $D \rightarrow (D_{left}, D_{right})$, where $D_{left} \cup D_{right} = D$ and $D_{left} \cap D_{right} = \emptyset$. This process is then recursively applied on D_{left} and D_{right} until a stopping criterion is triggered (*e.g.*, all elements of D share the same output). The most common way to determine the "best" split is to select the attribute and the question that reduce the indecision of the output variable in D_{left} and D_{right}, as formulated in the following equation :

$$\max_{D_{left}, D_{right}} |D|.I(D) - |D_{left}|.I(D_{left}) - |D_{right}|.I(D_{right}),$$

where $I(\cdot)$ is the indecision of the output for a given set. There exist different manners of measuring the indecision. Among them, two are widely used for classification : the *Shannon entropy* and the *Gini impurity*.

The Shannon entropy is defined by :

$$I_{Shannon}(D) = -\sum_{y \in \mathcal{Y}} p_y \log_2 p_y,$$

where p_y is the proportion of elements in D labeled with the value y. In a different way, the Gini impurity measures how often a randomly chosen sample from D would be

incorrectly predicted if it were randomly predicted according to the distribution of $\mathcal{Y}$ in D_{left} and D_{right}. The Gini impurity equation is :

$$I_{gini}(D) = 1 - \sum_{y \in \mathcal{Y}} p_y^2.$$

Overfitting is a significant difficulty in decision tree induction. It is frequently the case that a fully developed tree, where each terminal node contains elements of the same class, perfectly classifies training samples but does not generalize well on unseen examples. In order to avoid overfitting, there exist two types of pruning mechanisms : *pre-pruning*, that stops growing the tree earlier, and *post-pruning*, that fully develops the tree and then post prunes it. Pre-pruning consists in defining a stopping criterion that determines whether a given node should be subdivided or should be a leaf. In post-pruning, the basic idea is to randomly divide the available data into two folds. The first one is used to build a fully developed tree and the second is used to assess the quality of more or less strongly pruned versions of the tree, *e.g.*, obtained by iteratively merging two leafs with a common parent.

In the experiments presented in this manuscript, we used $I_{Shannon}$ as measure of node uncertainty and the *size cutoff* pre-pruning method. Namely, if the number of samples reaching the node is greater or equal to a predetermined threshold $N_{\min}$, then the node is divided (when possible). However, an *a priori* estimation of $N_{\min}$ is difficult and often requires preliminary experiments form the user.

Ensemble methods

Ensemble methods. Over the past two decades, approaches based on the use of multiple models have emerged. The core idea of *ensemble methods* is to learn a set of T functions $f_i : \mathcal{X} \to \mathcal{Y}$ and to combine the predictions of each f_i in order to form a better predictor f. Since learning multiple functions f_i may lead to a more expressive function f, one of the advantages of ensemble methods could be the reduction of bias. In theory, this flexibility can however overfit the training data. Thus, in practice, ensemble methods mainly aim at reducing the variance by introducing randomization either by subsampling the training set, which makes predictions less dependent on the particularities of a single training set, or by making the algorithm stochastic. The aggregation of the predictions can be made by an average of each output $\hat{y}_i = f_i(x)$ (for a regression problem), by a majority vote (for a classification problem) or any more evolved calculation such as a weighted average or a weighted majority vote that give more importance to some f_i.

Tree-based ensemble methods. *Tree-based ensemble methods* are ensemble methods where each function f_i is a decision tree. Figure 2.9 depicts the general scheme of an inference procedure of a tree-based ensemble method for a binary classification problem. Basically, decision trees are deterministic. Consequently, tree-based ensemble methods differ in the way they introduce diversity among the different trees. As an example, Breiman proposed the *Bootstrap AGGregatING* [16] technique (often abbreviated as *Bagging*) that introduces randomization by building each tree using a subsampling of the training set. Five years later, he proposed the *Random Forests* [17], which combine Bagging and random attribute selection. Namely, for each internal node of the tree, the algorithm calculate the "best" split for K randomly chosen attributes among the d candidate attributes. When $K = d$, the method is equivalent to Bagging.

Bagging

Random Forests

Extremely randomized trees. The *extremely randomized trees* (ETs), proposed by

Extremely randomized trees

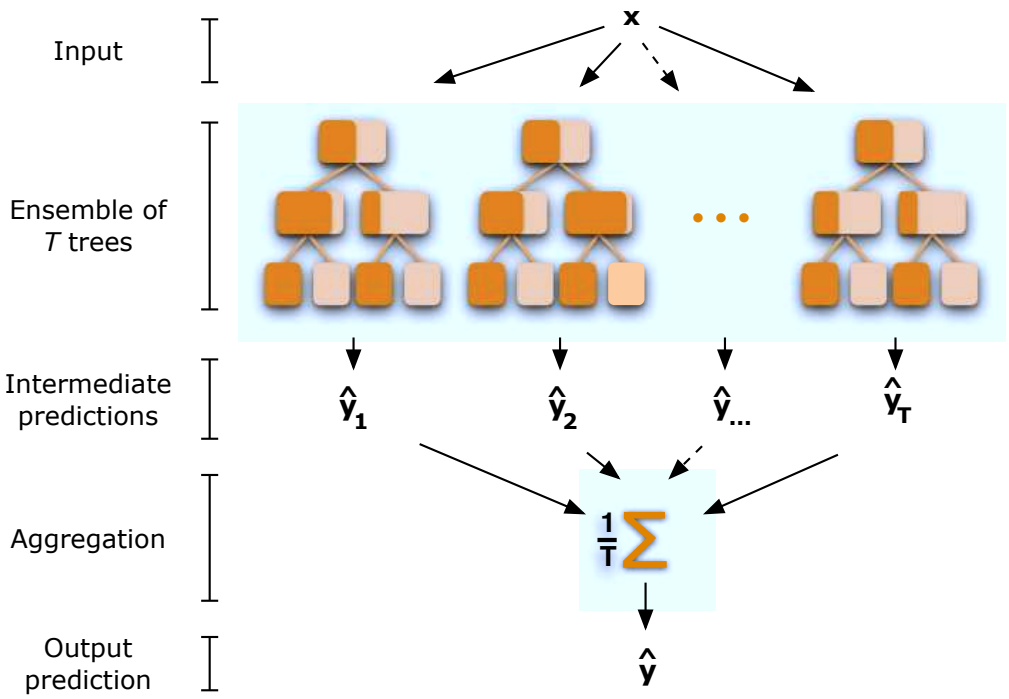

FIGURE 2.9: **Inference of a tree-based ensemble method.** A new example x traverses each of the T trees, which return T predictions $\hat{y}_i$ (one per tree). Then, the predictions are combined to form the output value $\hat{y}$. At each node, the two colors represent an example of the output distribution of samples reaching the node (for a binary classification problem).

Geurts *et al.* [58], add an extra level of randomization compared to the Random Forests. ETs do not attempt to determine the "best" cut-off value (for continuous attribute) or the "best" subset of values (for finite discrete attributes) for each of the K randomly chosen variables. Instead, ETs fix cut-off and subset values in a random fashion. The "best" split is then determined among K random splits. Unlike the Random Forests method, each tree is built using the original learning samples and does not rely on bootstrap replicates. The parameter K determines the randomness of trees. When $K = 1$, the trees are completely randomized as each internal node is defined by one randomly chosen attribute associated to a random cut-off (or subset). In opposite, when $K = d$, each node can select the "best" split among d random splits. In the empirical study of Geurts *et al.* [58], the authors showed that $K = \sqrt{d}$ seems to be near-optimal on classification problems.

2.5 k-Nearest neighbors

The *k-nearest neighbors* method (kNN) is perhaps the most simple machine learning technique. The training stage is trivial. It just stores all training samples. Such techniques are also called *memory-based learning*. In order to make a prediction of a new example, the algorithm identifies the k nearest training samples, according to a distance metric, and returns the mean of the output value (for regression) or the most frequent class

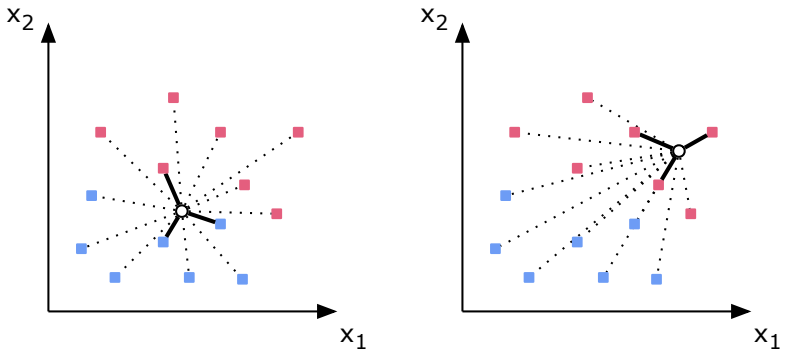

FIGURE 2.10: **Idea of a 3-nearest neighbors on a two-dimensional binary classification problem.** Circles are the query examples. Dashed lines represent the distance between the query to training samples. Solid lines represent the three smallest distances. Left : among the three nearest neighbors of the query, there are two blue and one red. Right : all the three nearest neighbors are red samples.

(for classification) among these neighbors. When $k = 1$, the method simply consists in returning the output value of the nearest neighbor. When $k = n$, the algorithm always returns the same value, namely, the mean (or the majority vote) over all samples. An example of binary classification problem using a 3-nearest neighbors is depicted on Figure 2.10.

In order to be a valid distance metric, the distance between two feature vectors a and b must conform to the following four criteria :

1. $dist(a, b) \geq 0$ non-negativity
2. $dist(a, b) = 0$ only if $a = b$ identity
3. $dist(a, b) = dist(b, a)$ symmetry
4. $dist(a, b) \geq dist(a, c) + dist(c, b)$ triangle inequality

Typical distance metrics are derived from the l_2-norm (also called *Euclidean distance*) and the *normalized Euclidean distance*. Given a training set $D = \{(x^{(i)}, y^{(i)})\}_{[1,n]}$ and an unknown example x, the l_2-norm distance is defined as :

$$dist_{l_2}(x, x^{(i)}) = \sqrt{\sum_{j=1}^{d} (x_j - x_j^{(i)})^2},$$

where x_j (resp. $x_j^{(i)}$) is the j-th feature of x (resp. $x^{(i)}$). However, the Euclidean distance is biased towards features that have a larger range of value. In order to reduce this effect, the normalized Euclidean distance proposes to normalize each feature x_j by its standard deviation σ_j calculated over the training set :

$$dist_{norm}(x, x^{(i)}) = \sqrt{\sum_{j=1}^{d} \left(\frac{x_j - x_j^{(i)}}{\sigma_j} \right)^2}.$$

Among the experiments presented in this thesis, some of them concern the application of kNNs on biological data. In this context, we have used a weighted version of normalized

Euclidean distance :

$$dist_{weighted}(x, x^{(i)}) = \sqrt{\sum_{j=1}^{d} w_j \left(\frac{x_j - x_j^{(i)}}{\sigma_j} \right)^2},$$

where the weight w_j enables to control the impact of the j-th feature in the calculation of the distance. This may be particularly interesting when several features are known to be correlated such as when they are involved in the description a common property of the input object. Correlated features produce the same kind of side effect than mixing features with large range and low range of values when using (not normalized) Euclidean distance. Note that setting each weight to 1 is equivalent to using the classical normalized Euclidean distance metric.

2.6 An introduction to feature selection

From a theoretical point-of-view, when the distribution $P(\mathcal{X}, \mathcal{Y})$ is known, it is easy to create a *Bayes rule* that classifies a new example as the most probable class :

$$\underset{y \in \mathcal{Y}}{\operatorname{argmax}} P(Y = y | X = x),$$

where Y denotes the output variable and $X = (X_1, X_2, \ldots, X_d)$ is the vector of input random variables. When the distribution $P(\mathcal{X}, \mathcal{Y})$ is assumed to be known, adding new features (random variables) cannot decrease the predictive performance of the classifier.

Feature relevancy

A feature can be *strongly relevant*, *weakly relevant* or *irrelevant*. A feature is strongly relevant if its removal will result in degradation of the predictive performance. By contrast, a feature X_i is weakly relevant if it is not strongly relevant and there exists a subset of features X' that do not include X_i such that X' is worse than $X' \cup X_i$ in terms of predictive performance. A feature is irrelevant if it is not strongly or weakly relevant, *i.e.*, features that are not necessary for the prediction problem.

In theory, it is never advised to restrict learning to a subset of features. However, in practical learning scenarios, we typically observe a degradation of performances of learning algorithms when faced with irrelevant features. This phenomenon is explained by the fact that the exact distribution $P(\mathcal{X}, \mathcal{Y})$ is rarely known, but instead we have at our disposal a dataset $D \sim P(\mathcal{X}, \mathcal{Y})$. As a consequence, among the training samples, an input variable can be fortuitously correlated with the output that may lead the model to overfit and then increase the generalization error.

Since the goal of supervised learning is to minimize the generalization error, a fundamental field of research in machine learning is devoted to the development of algorithms that identify subsets of features that are relevant for the predicting task. This field of

Feature selection

research is known as *feature selection*, *variable subset selection* or *feature reduction*. Furthermore, feature selection may be interesting for additional reasons :

1. To speed-up an algorithm or reduce its memory consumption. It is particularly the case for large-scale datasets such as *DNA microarrays*, which contain tens of thousands of microscopic fragments of DNA used to measure the expressions levels of large numbers of genes simultaneously ;

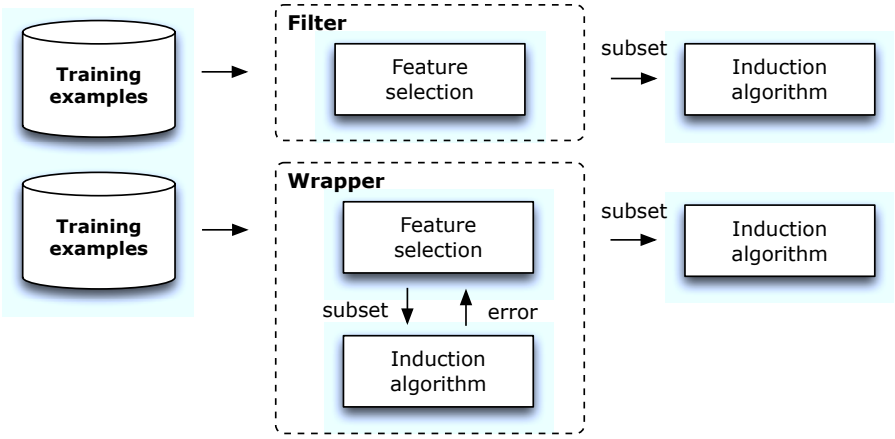

FIGURE 2.11: **Scheme of filter and wrapper approaches.** Filter methods discard features prior to the induction algorithm whereas wrapper selection search is driven by the evaluation of the induction algorithm.

2. To improve the model interpretability. By removing most irrelevant and redundant features, feature selection helps to draw more general scientific conclusions by a better understanding of which features are important and how they are related with each other ;

3. To obtain a simpler model according to *Ockham's razor principle* that states : "Simpler explanations are, other things being equal, generally better than more complex ones". Namely, we should prefer the simple feature subset, *i.e.*, the subset with the smallest dimensionality, among different feature subsets leading to models that are not significantly suboptimal in terms of generalization error.

Note that feature selection aims at finding the subset of relevant features from the original features set such that the generalization error is the lowest, but does not aim at constructing or extracting new features from the data. However, because different learning algorithms may use different heuristics, biases and tradeoffs, the subset of features that works optimally for an algorithm may be different for another algorithm. Therefore, the problem of finding the relevant subset of features is reduced to the problem of finding an optimal subset with respect to a particular machine learning algorithm.

Optimal feature subset

Formally, given an learning algorithm A and a dataset $D \sim P(\mathcal{X}, \mathcal{Y})$ with the random variables $X = (X_1, X_2, \ldots, X_d)$, an *optimal feature subset* $X_{opt} = (X_1^{opt}, X_2^{opt}, \ldots, X_t^{opt})$, $t \leq d$, with respect to A is a subset of features that minimizes the generalization error of A. Since two features may be perfectly correlated, the optimal feature subset is not necessary unique. Consequently, a feature that is in an optimal feature subset of a learning algorithm does not imply that it is relevant and a relevant feature does not imply that it is in an optimal feature subset of a given learning algorithm.

Subset selection algorithms are essentially divided into *wrappers*, *filters* and *embedded methods*. Figure 2.11 summarizes the process of filter and wrapper approaches.

2.6.1 Filters

Filter approaches attempt to discard irrelevant features prior to the application of the learning algorithm. They rely on the characteristics of the training data. Therefore, filter approaches do not take into account the effects of the selected features on the performance of the machine learning algorithm. The independence of filter approaches with respect to the predictor are particularly interesting when the number of features is very large because they are computationally less expensive.

Basically, filter approaches rank the features using a score function S and select the top-ranked ones. There also exist more evolved filters that directly rank subsets of features. Below are few examples of scoring functions S.

Variance score. The variance score is probably the simplest score function. The variance is an unsupervised filter that prefers features with a large distribution of values. Therefore, the score of the i-th feature is $S^{var}(X_i) = \sigma_i^2$.

Fisher score. The Fisher score is a supervised measure that assigns higher score to features that require that examples of different classes have a large value difference.

$$S^{fisher}(X_i) = \frac{\sum_{y \in \mathcal{Y}} |D_y|(\overline{X_i^y} - \overline{X_i})^2}{\sum_{y \in \mathcal{Y}} |D_y|(\sigma_i^y)^2},$$

where D_y is the set of samples of D of class y, $\overline{X_i}$ is the mean value of the i-th feature in D, $\overline{X_i^y}$ and σ_i^y are the mean value and variance of the i-th feature in D_y.

Correlation-based feature selection. The Correlation-based Feature Selection (CFS) is a filter that ranks feature subsets $X' \in X$ based on the feature-class correlation $corr(X_i, Y)$ and the feature-feature intercorrelation $corr(X_i, X_j)$. An example of CFS is the heuristic "merit" :

$$S^{cfs}(X') = \frac{|X'| \ \overline{corr(X', Y)}}{\sqrt{|X'| + |X'| \ (|X'| - 1) \ \overline{corr(X', X')}}},$$

where $\overline{corr(X', Y)}$ is the mean value of all feature-class correlations and $\overline{corr(X', X')}$ is the mean value of all feature-feature correlations.

For more information about filter approaches, the reader is invited to refer to [44].

2.6.2 Wrappers

Wrapper methods make use of the learning algorithm to choose a set of relevant features. The search for a good subset of features is driven by the evaluation of the machine learning algorithm on different candidate feature subsets.

Brute-force. The simplest way to determine the optimal subset of features would be to evaluate all possible combinations of feature subsets. Obviously, since the number of possible subsets $(2^d - 1)$ growth exponentially with respect to the number of features d, an exhaustive search is computationally prohibitive and usually impractical.

Greedy forward selection. The greedy forward selection is a iterative algorithm, starting from an empty set of selected features and adding one element at each iteration according to an heuristic. The goal of the heuristic is to locally select the optimal feature with the hope of finding a global optimal subset of features. Typically, the heuristic selects the feature that minimizes the evaluation error when coupled with the current set of selected features. The greedy approach starting from the full set of features and discarding one element at each iteration is called *greedy backward selection*. Note that due to the greedy nature, the feature selection may fall into local minima.

Genetic algorithm. In a genetic algorithm, a population of many randomly sampled subsets of features is first generated. Among the subsets, the most promising ones in terms of generalization error are used to generate a new population. Each element of the new population is obtained by applying a *genetic operator* on the promising subsets. The operators are inspired from the process of natural evolution in life such as *mutations* (add or remove one feature) and *crossover* (combine two subsets). The process is repeated until a stopping criterion is reached (*e.g.*, fixed number of generations).

However, by proceeding in this way, the user of wrapper methods must be aware that the selected features may lead to a model that overfits the samples, if no special care is taken. Indeed, if we consider the simple case in which training samples are used to select the subset of features, the evaluation of the model on an independent set of samples may be strongly lack of generalization since the selected features can perfectly fit the training samples. To tackle this phenomenon, a widely used approach is to cross-validate the selection, *i.e.*, to run the feature selection once for different train/test splits. Although this method is usually well adapted to evaluate a model with a fixed set of features, this approach is not suitable for feature selection. Indeed, by proceeding in this way, the same data would be used for both selecting the set of features and assessing the quality of this selected set. Ambroise *et al.* [6] have been shown that this approach is biased due to using the same data for selecting and for evaluating and that it could lead to highly over-estimated the model performance. To avoid this risk of overfitting, we recommend a more evolved approach based on a second loop of cross-validation, in which the training stages perform a cross-validated feature selection on a part of the data and the testing stages perform the evaluation of the selected features on the remaining data. This approach is further detailed in Chapter 5 and 6.

For more information about wrapper approaches, the reader is invited to refer to [71].

2.6.3 Embedded methods

Embedded methods are learning algorithms that have a built-in mechanism to perform variable selection. For example, in decision trees, the variable that composes an internal node is chosen according to its importance for the classification task. Therefore, it is possible to estimate the importance of each feature and then to perform a selection. For linear models, the importance of a variable can be estimated using the weight associated to all variables.

For an introduction to embedded methods, we refer the reader to one of the multiple existing articles devoted to this subject, for example Lal *et al.*[75].

CHAPITRE 3

Background on structural biology

Contents

Structural biology is an area of molecular biology that studies the structure of macromolecules, especially proteins. In molecular biology, there exists a fundamental principle that describes the genetic information flow within biological systems. This principle, known as the *central dogma of molecular biology*, states that DNA sequences are transcribed into *messenger*-RNA sequences and that these mRNA sequences are translated into protein sequences. In addition, in structural biology, it is widely assumed that the protein sequence determines the three-dimensional structure and that the three-dimensional structure confers the function to the protein. However, the process by which the amino acid sequence assumes its conformation is extremely complex because the protein can be

subject to a variety of posttranslational modifications (*e.g.*, phosphorylation, glycosylation, sulfation, etc.).

Biology is "a science of exceptions". Namely, there is an exception to every rule. As an example, some RNA fragments may play important functions without being transcribed into proteins, *e.g.*, *transfer-RNAs* are molecules of RNA actively used in peptide synthesis, *micro-RNAs* are small non-coding RNA fragments, which may bind to mRNAs and thus prevent their translation into proteins [28]. There also exist proteins that are known to be intrinsically disordered, *i.e.*, that do not fold into stable structures.

This chapter introduces relevant background on proteins to understand the experimental parts of the thesis. After an introduction to the biochemistry of proteins in Section 3.1, Section 3.2 reviews the different ways to experimentally derive the three-dimensional structure of macromolecules. Section 3.3 presents the Protein Data Bank, in which experimentally determined structures are recorded. Finally, Section 3.4 introduces the structure-based annotations that will be addressed in the next chapters.

3.1 Fundamentals of protein structure

Proteins are among the most fundamental molecules of biology that are responsible for most activities that make life possible. The functions of proteins are extremely diverse. Among them, we roughly distinguish three kinds of proteins : *globular proteins* that are soluble within the cell and speed-up specific metabolic reactions ; *fibrous proteins* that play structural roles (*e.g.*, skin, muscle fibers, hair, etc.) ; and *membrane proteins* that transfer information from the environment into the cell.

Proteins are usually regarded at four levels of structure : *primary structure*, *secondary structure*, *tertiary structure* and *quaternary structure*. Section 3.1.1 briefly describes the common chemical interactions involved in protein structures. Section 3.1.2 introduces primary structures as sequences of amino acids in polypeptide chains. Local regions of primary structures adopt ordered arrangements. Section 3.1.3 describes these regular conformations, called secondary structures. Section 3.1.4 discusses around the global shape (tertiary structure or three-dimensional structure) of proteins. Finally, Section 3.1.5 briefly discusses the affinity that some protein structures may have with other ones and then form complexes of proteins.

3.1.1 Chemical bonds

There are four commonly mentioned types of interactions responsible for the structure and function of proteins : *covalent bonds*, *hydrogen bonds*, *van der Waals forces*, and *hydrophobic effect*.

Covalent bonds are the most common chemical interactions. A covalent bond involves the sharing of electrons between atoms, which are very close to each other. This kind of interaction is considered as "strong", *i.e.*, difficult to break.

Hydrogen bonds are strong interactions dipole-dipole attraction between a polar molecule in which the hydrogen interacts with a highly electronegative atom (typically,

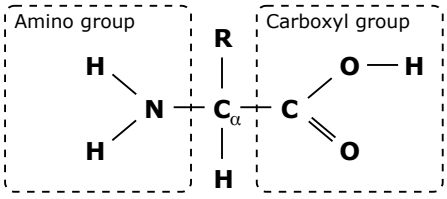

FIGURE 3.1: **Structure of an amino acid.** R is the side chain of the amino acid and vary from one amino acid to another. C_α is the central carbon atom.

nitrogen or oxygen in the context of protein structure). Since a hydrogen bond does not rely on the sharing of electrons but on an attraction phenomena, hydrogen bonds are "weaker" than covalent bonds, *e.i.*, it is easier to break a hydrogen bond than a covalent bond.

Van der Waals forces is the sum of the attractive or repulsive forces between atoms and molecules caused by the polarizations of nearby particles. These interactions are weaker than hydrogen bonds and can easily be broken.

Hydrophobic effect, meaning "water-fearing", is the tendency of nonpolar molecule or part of molecule to aggregate together in aqueous solution, such as a mixture of oil and water. This effect is a consequence of Van der Waals force and plays an important role in the formation of protein structure.

3.1.2 Primary structure

The primary structure can be pictured as a pearl necklace of different kinds of pearl shapes. The primary structure is a linear sequence of amino acids. *Amino acids* are molecules composed of an amino group, a carboxyl group and a radical R, often called *side chain*, which varies from one amino acid to another. The three groups and an additional hydrogen atom are attached to a carbon atom C_α. Figure 3.1 depicts the structure of an amino acid. *Amino acid*

There exist 20 common side chains and, consequently, 20 standard amino acids. Among the amino acids, 9 of them are *essential* for human being, *i.e.*, our organism is not able to synthesize them. Essential amino acids are then supplied in the diet (typically, in meats). Essential amino acids may vary upon the species and sometimes upon the stage of development of the organism. Table 3.1 lists the 20 standard amino acids and their usual three-letter and single-letter codes [1]. In addition, they are grouped according to the pK_a value of their side chains, *i.e.*, the acid dissociation constant that measures of the strength of the acid in solution. The following example (in single-letter code) is a subsequence of the human hemoglobin primary structure :

RLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN

Among the standard amino acids, three of them have special properties : *glycine*, *proline* and *cysteine*. *Glycine* is the smallest with a side chain that solely consists of a

1. The origin of the single-letter code is discussed at
`http://www.biology.arizona.edu/biochemistry/problem_sets/aa/Dayhoff.html`.

	Name	3-letter	1-letter	Group	Surface (Å^2)
	Alanine	Ala	A	Hydrophobic	118.1
	Arginine	Arg	R	Electrically charged (+)	256.0
	Asparagine	Asn	N	Polar uncharged	165.5
	Aspartic acid	Asp	D	Electrically charged (−)	158.7
	Cysteine	Cys	C	Special case	146.1
	Glutamic acid	Glu	E	Electrically charged (−)	186.2
	Glutamine	Gln	Q	Polar uncharged	193.2
	Glycine	Gly	G	Special case	88.1
•	Histidine	His	H	Electrically charged (+)	202.5
•	Isoleucine	Ile	I	Hydrophobic	181.0
•	Leucine	Leu	L	Hydrophobic	193.1
•	Lysine	Lys	K	Electrically charged (+)	225.8
•	Methionine	Met	M	Hydrophobic	203.4
•	Phenylalanine	Phe	F	Hydrophobic	222.8
	Proline	Pro	P	Special case	146.8
	Serine	Ser	S	Polar uncharged	129.8
•	Threonine	Thr	T	Polar uncharged	152.5
•	Tryptophan	Trp	W	Hydrophobic	266.3
	Tyrosine	Tyr	Y	Hydrophobic	236.8
•	Valine	Val	V	Hydrophobic	164.5

TABLE 3.1: **The twenty standard amino acids.** Dots mark the essential amino acids in human being. The surfaces are defined by Rose *et al.* [107] and expressed in unit of square ångstroms.

hydrogen atom. This particularity enables the amino acid to fit into hydrophobic as well as into hydrophilic environments. Usually, hydrophobic residues avoid being in contact with the solvent. They attempt to be buried into the core of the protein. *Proline* is one of the exceptions of the biology. In fact, proline is not a “standard" amino acid because its side chain forms a loop with the amino group, *i.e.*, its radical R is bound to the nitrogen N. This particularity introduces a conformational constraint that can play an important role in the folding process. *Cysteine* is the unique residue able to form a covalent link with another cysteine. Therefore, cysteines play a key role in the stabilization of three-dimensional structures. Cysteines will be discussed more in that context, later on in this manuscript.

Peptide bond

Each amino acid is connected to the next one in the sequence. Indeed, the carboxyl group of one amino acid and the amino group of another amino acid react together to form a covalent chemical bond : a *peptide bond*. This reaction releases a molecule of water. Figure 3.2 draws the chemical equation of a peptide bond between two amino acids.

3.1.3 Secondary structure

A *secondary structure* is a local segment of the primary structure that forms a regular pattern in the three-dimensional space. Secondary structures rely on hydrogen bonds

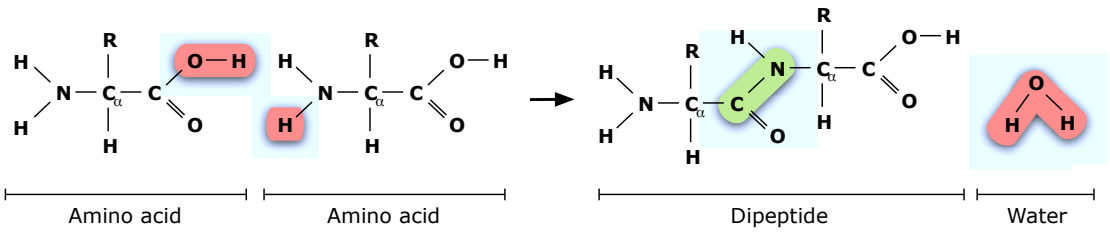

FIGURE 3.2: **The peptide bond of two amino acids.** R represents the side chains of the amino acids and C_α represents the central carbon atoms. Left : two non-bonded amino acid residues. Right : formation of a dipeptide through a peptide bond and a molecule of water. In red : atoms that form the molecule of water. In green : the peptide bond.

between amino groups and carboxyl groups. A *hydrogen bond* is an electromagnetic interaction of a hydrogen atom and an electronegative atom (typically, an oxygen). This kind of attraction is weaker than covalent bonds and can be easily broken alone. However, the regularity and the large number of hydrogen bonds make secondary structures stable. *hydrogen bond*

There exist two common secondary structures : *α-helices* and *β-sheets*. An α-helix is a segment of amino acids in which the conformation of the main chain forms a helix. In this conformation, the side chains are located on the outside of the helix. Each residue that composes a helix is stabilized by two hydrogen bonds. In an α-helix, there are 3.6 residues per turn. There also exist two additional forms of helices : 3_{10}*-helices*, which are more compact with only 3 residues per turn, and π*-helices*, which are larger with 4.4 residues per turn. A β-sheet is a stretch of amino acids connected laterally to another stretch of amino acids by hydrogen bonds. By contrast to helices, β-sheets require partners to exist. Two β-sheets can be parallel, *i.e.*, amino and carboxyl groups share the same orientation, or antiparallel. In sheets, the side chains point perpendicularly to the plane and successive residues usually point to opposite directions. *α-helices* *β-sheets*

Figure 3.3 shows the structures of an α-helix and three β-sheets of crystalized protein fragments. The two proposed views reveal the orientation of the side chains (outside of the helix or perpendicular to the plane of the sheets).

3.1.4 Tertiary structure

Whereas the secondary structure is the configuration of local regions, the *tertiary structure* is the three-dimensional arrangement of the whole protein within the space. Tertiary structures rarely form long linear spaghettis. Indeed, since the environment of proteins is aqueous (*i.e.*, mainly composed of polar molecules of water), the tertiary structure is mainly driven by hydrophobic interactions between its hydrophobic amino acids (Table 3.1) and the solvent. As a consequence, proteins often have a hydrophobic core, in which side chains are buried from the solvent. This is typically the case of globular proteins. In addition to these weak interactions (hydrogen and hydrophobic interactions), cysteines can sometimes form stronger interactions by forming covalent bonds.

Among the wide variety of ways to fold proteins, it is usually assumed that the native tertiary structure of a protein is the one that minimizes the Gibbs free energy [59], *i.e.*,

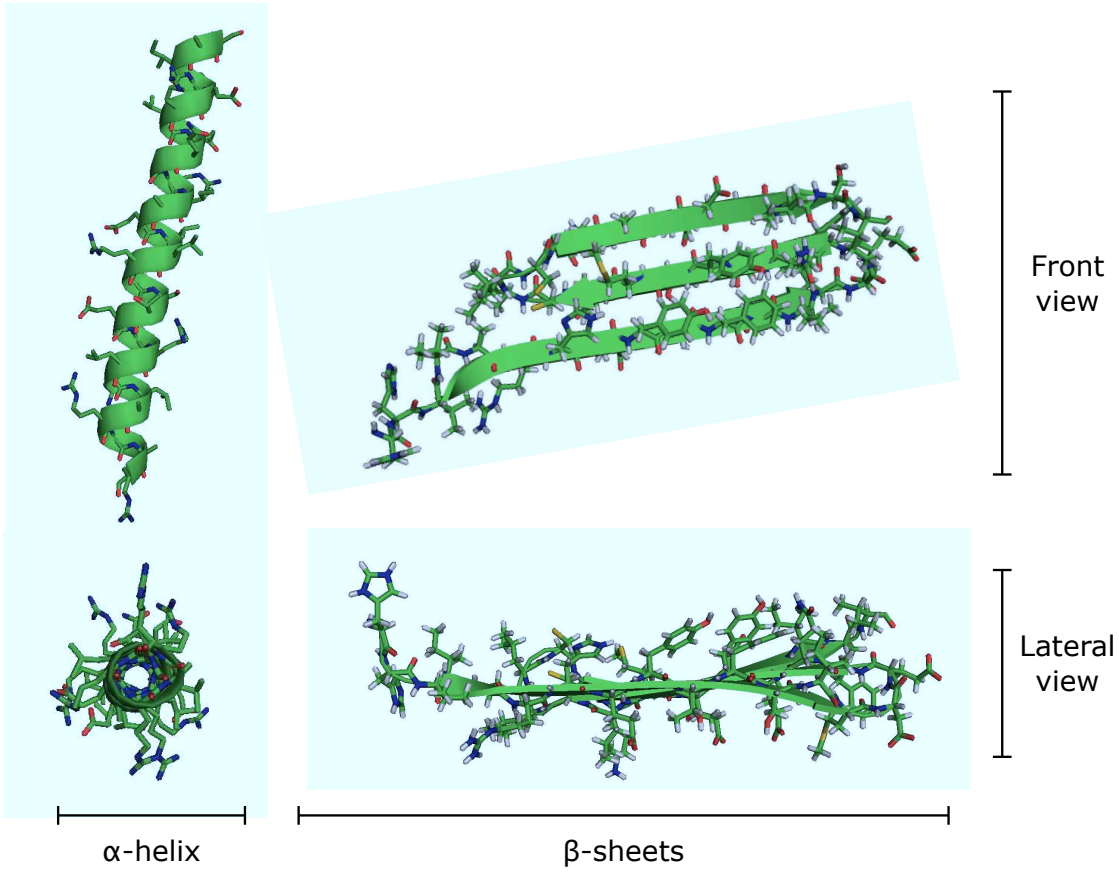

FIGURE 3.3: **An example of an α-helix and a β-sheet.** The α-helix and β-sheets structures are fragments from crystal structures : PDB file 3NOU and 2JOX, respectively. The structures are rendered by the PyMOL Molecular Graphics System. The cartoon shapes (large green lines) highlight the secondary structures. Sticks represent covalent bonds.

the energy of a system that can be used to do work at a constant temperature and pressure. However, there exist studies [112, 37] that claim that the native structure is not necessarily the minimum one but the one that is kinetically accessible, *i.e.*, the one with a small energy and with a path in the free energy surface that is reasonably smooth.

3.1.5 Quaternary structure

In order to acquire their function, many proteins aggregate with one or more independently folded proteins. The *quaternary structure* is the arrangement of single polypeptides into an oligomeric complex. Proteins involved in an oligomer are called *subunits*. It is often the case that subunits of an oligomer are replicates of the same protein. Quaternary structures are mainly stabilized by hydrophobic interactions among the subunits. Namely, segments of non-polar side chains on the surface (*i.e.* exposed to the solvent) gather to form a hydrophobic core. Figure 3.4 shows the example of the human hemoglobin. This is a tetramer (*i.e*, a oligomer composed of four subunits) of two distinct proteins (replicated

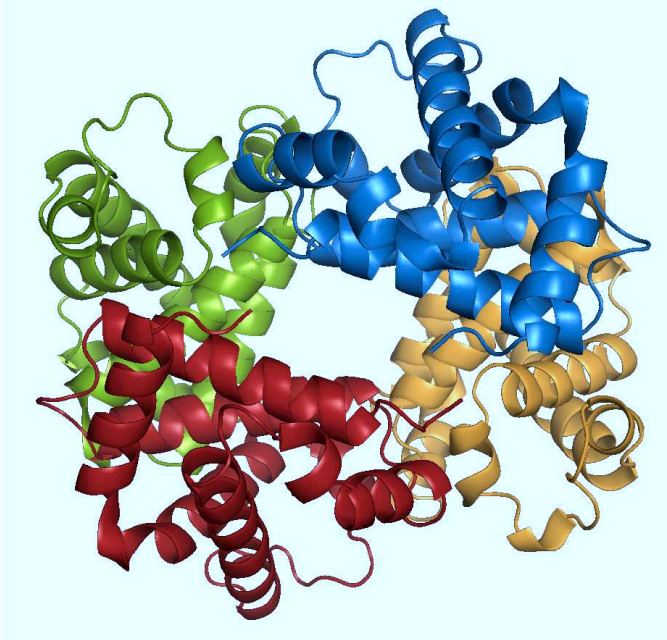

FIGURE 3.4: **The structure of human hemoglobin.** The structure is depicted using a cartoon representation. Each subunit of the complex has a distinct color. The PDB file of the crystal structure is 1A3N. The structure is rendered by the PyMOL Molecular Graphics System.

twice).

3.2 Structure determination of biological macromolecules

Experimentally determining the three-dimensional structure of a macromolecule is a slow and expensive process that requires sophisticated instruments. The most popular methods for experimentally deriving protein structures are briefly described in the following sub-sections.

3.2.1 X-ray crystallography

X-ray crystallography

The *X-ray crystallography* is based on the principle of diffraction caused by a beam of X-ray radiation onto a pure crystal of the protein. Since a crystal is a solid with a very precise and periodic arrangement of its molecules in all three dimensions, the diffraction pattern resulting from X-rays that strike it is specific to the molecules that compose the crystal. From the observation of the diffraction pattern, it is possible to reconstruct the three-dimensional position of each atom of the molecules.

The main limitation is that only proteins able to form crystals are examinable. A typical example of proteins unable to form crystals are those that have many hydrophobic amino acids such as proteins located in the membrane of cells. A second limitation is that proteins cannot be examined in its cell environment. Moreover, as the formation of the

crystal needs specific conditions (*e.g.*, solvent, temperature or pressure) to grow, the reconstructed map of atoms may not represent the native structure of the protein but a conformation forced by the crystallization conditions.

3.2.2 Nuclear magnetic resonance spectroscopy

NMR spectroscopy

The *nuclear magnetic resonance* (NMR) spectroscopy is based on the principle that the spin-moments of nucleons in the nucleus of atoms resonate when a molecule is placed into a magnetic field. The NMR observes the transitions of the spin-moments by varying the wavelength of the magnetic field around the average transition energy of the atoms of interest, such as hydrogen.

The NMR produces a spectrum that describes the chemical shifts of the atoms, *i.e.*, the strength of the electromagnetic field needed for ensuring that the spin-moments are transiting. As chemical shifts directly dependent on nearby atoms and on their distances from each other, the signals that NMR produces are sets of distances between specific pairs of atoms. By repeating the experiment on different pairs of atoms, the NMR generates several sets of constraints. Mathematically, these constraints can be solved in different ways leading to multiple possible structures, rather than a single structure.

The major limitation of the approach is the molecular mass of the protein. The higher the molecular mass is, the less accurate the determination of its structure will be.

3.2.3 Electron microscopy

Electron microscopy

An *electron microscope* is basically a microscope that uses a beam of electrons to illuminate and image the molecule. Two main techniques are used in molecular electron microscopy : *electron crystallography* and *single particle analysis*. The former crystalizes the protein in two-dimensions in opposition to X-ray crystallography that need a three-dimensional crystal. The structure is determined using techniques based on the diffraction of electrons. The latter technique is an image processing technique used to determine the structure by averaging several images of the molecules.

The resolution of the method is quite weak but it can yield structural information of macromolecular complexes.

3.3 The Protein Data Bank

Experimentally determined 3D structures of biological macromolecules are recorded in a single worldwide repository : the *Protein Data Bank* (PDB). The PDB archive is freely and publicly available. The data are maintained by the *Worldwide Protein Data Bank* (`http://www.wwpdb.org/`) and distributed by four centers over the world : *RCSB PDB* (`http://www.rcsb.org/`, USA), *PDBe* (`http://pdbe.org/`, Europe), *PDBj* (`http://www.pdbj.org/`, Japan) and *BMRB* (`http://www.bmrb.wisc.edu/`, USA).

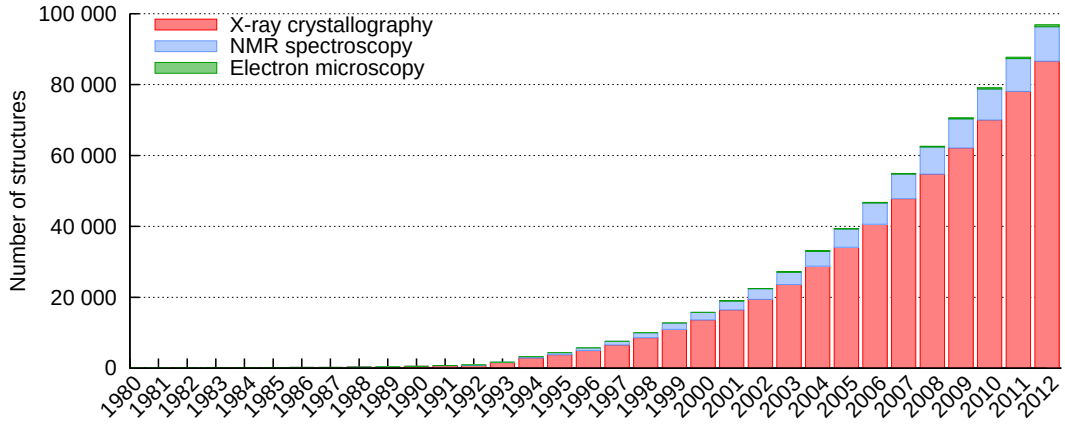

FIGURE 3.5: **Yearly growth of structures experimentally solved.** The lowest (in 1980) and highest (in 2012) number of solved structures are 69 and 86 611, respectively. The histogram derived from the RCSB PDB repository at December 4, 2012.

Method	Proteins	Others	Total
X-ray crystallography	71 175	5 020	76 195
NMR spectroscopy	8 508	1 212	9 720
Electron microscopy	327	152	479
Hybrid & others	144	22	166
Total	80 199	6 406	86 611

TABLE 3.2: **Distribution of structures in PDB.** (December, 2012).

3.3.1 Statistics

This section reports some statistics of the PDB repository at the beginning of December, 2012. Although the repository began on 1972, we have limited our charts to 1980 for clarity reason.

Figure 3.5 shows the growth of the number of PDB structures per experimental techniques over years. There were 69 experimentally determined macromolecular structures in 1980. From 1992 to 2007, the number of determined structures per year rapidly increased. Since 2007, $\sim$ 7500 structures are deposited each year. What we also observe is that the proportion between the three experimental techniques is roughly identical from year to year. $88-90\%$ of structures are determined by X-ray crystallography, $\sim 10\%$ by NMR spectroscopy, while electron microscopy and hybrid methods represent less than 1%.

Table 3.2 reports the distribution of protein and other macromolecule structures in PDB at the beginning of December, 2012. The remaining macromolecules are RNA fragments, Protein-RNA complexes and other carbohydrates macromolecules. Protein structures represent 93% of the repository.

Figure 3.6 shows the yearly growth of the number of distinct folds according to SCOP : a Structure Classification Of Proteins database [93]. The SCOP database classifies protein based on the relationship between structural similarities and evolutionary origins that

SCOP classification

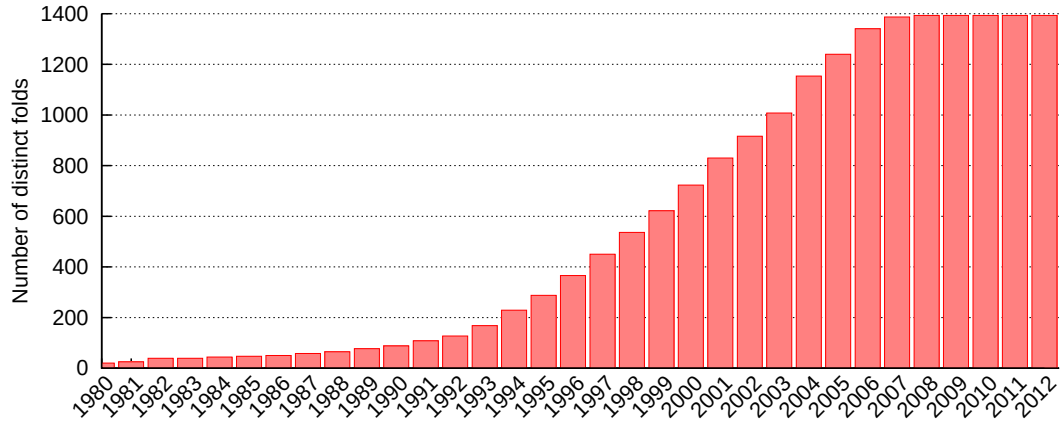

FIGURE 3.6: **Yearly growth of distinct folds (as defined by SCOP).** The lowest (in 1980) and highest (in 2009–2012) number of folds are 20 and 1 393, respectively. The histogram derived from the RCSB PDB repository at December 4, 2012.

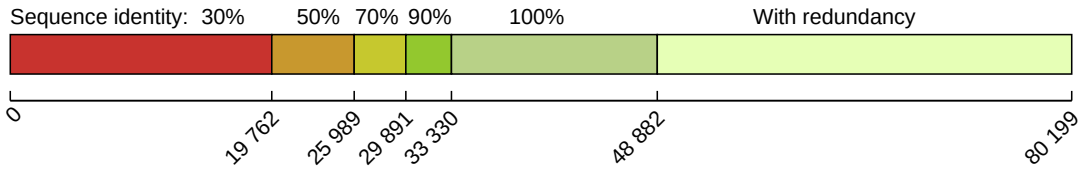

FIGURE 3.7: **Number of non-redundant sequences.** The whole box represents the set of data available in PDB. Each subset, starting from 0, represents a subset of non-redundant data in which sequence similarities are below a determined threshold.

a protein shares with others. We observe that the number of new folds per year mostly increased between 1990 and 2007. More intriguing, we remark that no new fold has been identified since 2008. In other words, most new protein structures fall into a few common folds.

Figure 3.7 illustrates the number of proteins available in PDB according to determined sequence similarity thresholds. The sequence similarity measure defines how identical two polypeptides are. The complete dataset is composed of 80 199 proteins. By removing all identical sequences (threshold : 100%), the number 'unique' structures falls to 48 882. That means that 39% of the sequences in PDB are identical to others. By varying the similarity threshold from 100% to 30%, we observe that PDB is indeed highly redundant. At 30% of sequence identity, considered as an acceptable redundancy threshold in the literature, there only remain 25% of the original structures.

3.4 Protein structure-based annotations

Tertiary structures of proteins can be derived into surrogate and simpler representations, called *protein structure-based annotations* or *structural annotations*. In this section, we introduce the six structure-based annotations treated as supervised learning tasks further in this thesis. The section therefore aims at describing the way to determine each annotation from a three-dimensional structure. These structure-based annotations will further form the supervision values for machine learning algorithms.

Structural annotation

Each annotation is presented in a dedicated section. The first five annotations are sequence-wise. Namely, to each element of a primary structure, they assign a label describing a particular structural property of amino acids. The last annotation is a particular case of contact map description, and focuses on the problem of assigning a label to each pair of cysteine residues of a protein.

3.4.1 DSSP secondary structure

The secondary structure, previously discussed in Section 3.1.3, is commonly assigned by the *Dictionary of Secondary Structures for Proteins* (DSSP). The DSSP program was designed by Kabsch and Sander [67] to standardize secondary structure assignment given the three-dimensional structure of proteins. The method relies on eight hydrogen bonding patterns :

3_{10}-helix : at least two consecutive residues forming a 3 turn helix ;
α-helix : at least two consecutive residues forming a 4 turn helix ;
π-helix : at least two consecutive residues forming a 5 turn helix ;
Turn : single residue forming a 3, 4 or 5 turn helix ;
Strand : at least two consecutive residues bridged in parallel or antiparallel ;
Isolated β-bridge : single bridged residues that do not form a turn ;
Bend : the angle formed by residues i, $i-2$ and $i+2$ is greater than 70 degrees ;
None : no secondary structure.

3.4.2 Secondary structure

However, the eight conformational states defined by DSSP are commonly reduced to three states : helix, sheet and coil. We consider the following reduction method :

Helix : 3_{10}-helix, α-helix and π-helix ;
Sheet : strand and isolated β-bridge ;
Coil : turn, bend and none.

Namely, helices with a minimum of two residues length form a more general group of helices. β-sheets, even composed of a single residue, from the group of sheets. The rest are reduced to coil.

3.4.3 Solvent accessibility

The *solvent accessible surface area* (SAS) is the surface area that is accessible to a

Solvent accessible surface area

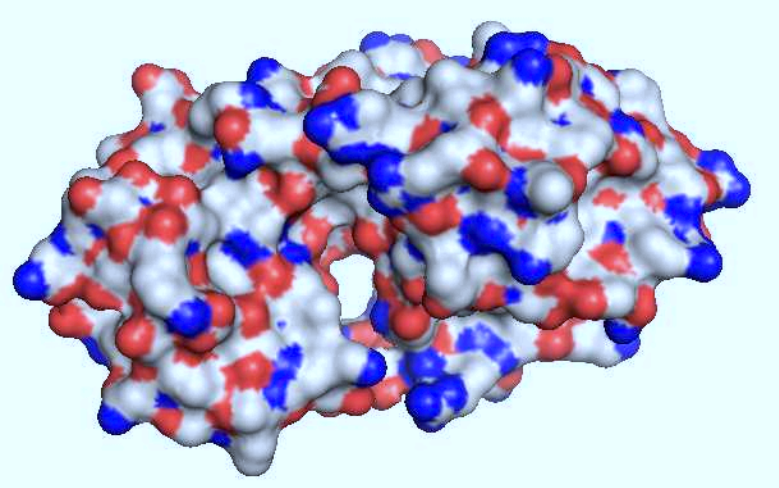

FIGURE 3.8: **Solvent accessibility of the lysosome structure.** The solvent-accessible surface area is calculated by rolling water molecules on the protein at all possible positions. The PDB file of the crystal structure is 148L. The structure is rendered by the PyMOL Molecular Graphics System.

solvent (typically, water). It is measured in unit of square ångstroms ($Å^2$). In addition to derive the secondary structure, the DSSP program also calculates the SAS. From the SAS, it is possible to apply a threshold as criterion to determine whether a residue is *buried* (inaccessible to the solvent) or not (exposed to the solvent). Figure 3.8 shows the solvent accessible area of a lysosome.

Relative solvent accessible area

However, the total surface area varies from one amino acid to another. Depending on the context, it is recommended to use a normalized value of SAS such as the *relative solvent accessible area* (RSA). RSA consists in dividing the SAS by the total surface area of the amino acid. In Table 3.1, we reported the surface area estimated by Rose *et al.* [107] for each amino acid X in the tripeptide Glycine-X-Glycine.

In our experiment, we use a threshold of 20% on the RSA to define the two states ("buried" and "exposed") of the solvent accessibility :

Buried : RSA $\leq 20\%$;
Exposed : RSA $> 20\%$.

3.4.4 Disordered regions

Disordered regions

Disordered regions refer to regions in proteins that do not adopt a stable three-dimensional structure when they are not in presence of their partner molecules. Over the last decade, several experimental studies have shown that proteins with disordered regions play various and critical functions in many biological processes. The flexibility of these regions makes it possible for a protein to interact, recognize and bind to many partners. For example, disordered regions are often involved in regulatory and signaling interactions [128] such as the regulation of cell division, the transcription of DNA or the translation of mRNAs. They also play a role in the self-assembly of protein complexes,

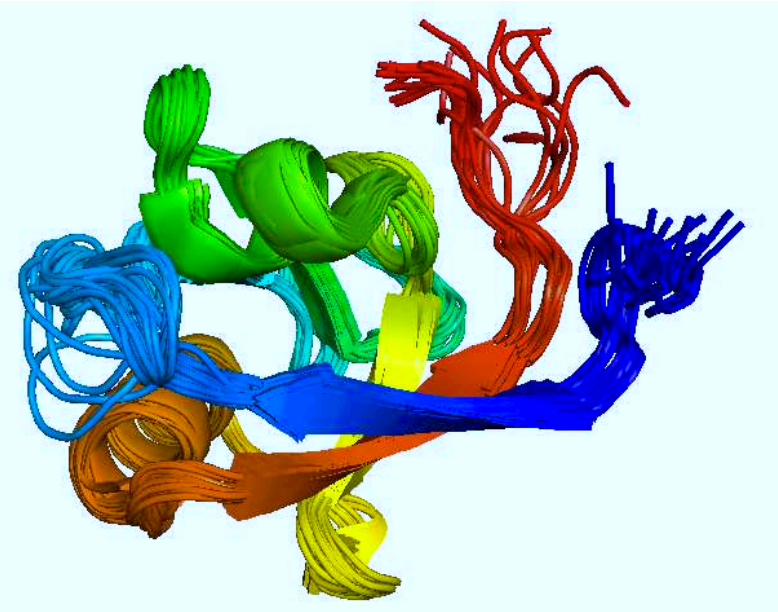

FIGURE 3.9: **NMR structure of disordered regions.** This protein has three disordered regions : the two red and dark-blue terminal parts in the upper-right part, and the light-blue loop on the left of the figure. The PDB file of the NMR structure is 2ERM. The structure is rendered by the PyMOL Molecular Graphics System.

and in the storage and/or transportation of small molecules [127, 46]. Figure 3.9 shows a protein with three disordered regions.

The length of disordered regions ranges from a few consecutive residues to the full length of the protein. While regions longer than 30 residues clearly differ from well-structured regions, short regions (usually less than 10 residues length) are sometimes just considered as loops in well-structured proteins. Many long disordered regions are not robust against *in silico* mutation [115]. This observation suggests that disordered regions are far from meaning random regions and seem to be evolutionarily conserved [116, 46], in particular, long loopy regions [82] appear to be more conserved than their flanking regions.

There are many reasons for automatically characterizing disordered regions in proteins. For example, in X-ray crystallography, determining macromolecular structures may take months and disordered regions may render protein analysis very difficult and often impossible when long disordered regions exist. An automatic annotation tool can therefore save valuable time. Such tools can also be useful to increase the accuracy of sequence similarity analysis by avoiding aligning unstructured regions with ordered regions. Recently, in disorder-based rational drug design, bioinformatics tools helped researchers to design some peptides that block interactions between structured and unstructured partners [32, 45].

The notion of disorder is not uniquely defined. In our studies, we use the definition of the CASP [94] competition (*Critical Assessment of Techniques for Protein Structure Prediction*). Namely, segments longer than three residues but lacking atomic coordinates in the crystal structure were labeled as "*disordered*" whereas all other residues were

labeled as "*ordered*".

3.4.5 Structural alphabet

The structural alphabet is a discretization of the protein backbone conformation as a series of overlapping fragments of four residues length. This annotation is less common in the literature than the previous structural annotations.

Our structural alphabet relies on the study of Camproux *et al.* [22], which determined an optimal decomposition of the conformational variability of four residues. According to their results, they proposed 27 states, denoted as $[a, A, B, \ldots, Z]$. The 27 states are described based upon four distances :

1. $d_1 = |C_\alpha^1 - C_\alpha^3|$;
2. $d_2 = |C_\alpha^1 - C_\alpha^4|$;
3. $d_3 = |C_\alpha^2 - C_\alpha^4|$;
4. P_4, the oriented projection of C_α^4 to the plane $(C_\alpha^1, C_\alpha^2, C_\alpha^3)$,

where C_α^i are the coordinates of the C_α of the i-th residue in the four length segment. The first three descriptors are the three distances between the non-consecutive C_α atoms.

Table 3.3 reports the average values of the four descriptors of each state. In order to determine the state that best corresponds to each residue i of a primary structure, we first calculate (d_1, d_2, d_3, P_4) on the segment $(i-1, i, i+1, i+2)$ and then select the state s that minimizes the Euclidean distance. In order words,

$$\underset{s \in [a,A,B,\ldots Z]}{\operatorname{argmin}} \sqrt{(d_1 - d_1^s)^2 + (d_2 - d_2^s) + (d_3 - d_3^s)^2 + (P_4 - P_4^s)^2},$$

where $(d_1^s, d_2^s, d_3^s, P_4^s)$ are the expected descriptor values of the state s. Note that, as the alphabet is based on segments of four residues, the first residue, as well as the last two residues, of a given protein sequence is not annotated.

3.4.6 Disulfide pattern connectivity

Disulfide bridge

A *Disulfide bridge* is a covalent link resulting from an oxidation-reduction process of the thiol group of two cysteine residues. Both experimental studies in protein engineering [7, 89, 70] and theoretical studies [136, 13] showed that disulfide bridges play a key role in protein folding and in tertiary structure stabilization. The knowledge of the location of these bridges adds strong structural constraints to the protein, which enable to drastically reduce the conformational search space in the context of protein structure prediction.

Disulfide pattern connectivity

The *disulfide pattern connectivity* of a protein is the set of disulfide bridges present in its tertiary structure.

There exist two methods to determine whether two cysteines form a bridge from the tertiary structure. The first one assigns disulfide bonds using the SSBOND records of the PDB file. These SSBOND records are generated automatically by a PDB processing program and compared with those of the depositor (if supplied). The second method is to use the DSSP program, which (i) checks bond distances between cysteines referred by SSBOND records and (ii) seeks additional bonds. (i) and (ii) are based on a bond distance of 3Å but they do not take into account the bonding state of cysteines, *e.g.*,

State	d_1	d_2	d_4	P_4
a	5.39	5.09	5.38	2.92
A	5.43	5.09	5.42	2.94
B	5.41	5.23	5.61	2.86
C	5.62	5.25	5.42	2.87
D	5.59	5.49	5.78	2.58
E	5.40	5.58	5.42	3.39
F	5.78	5.68	6.07	1.46
G	5.55	7.74	5.60	-3.31
H	5.60	6.71	5.50	3.69
I	5.69	8.09	5.67	3.09
J	5.66	8.95	6.54	2.09
K	5.66	8.91	6.66	-1.46
L	5.66	8.07	6.70	2.96
M	5.70	7.26	7.02	0.88
N	6.03	6.85	5.64	-0.63
O	6.47	5.92	5.56	0.53
P	6.57	8.96	5.58	-2.19
Q	6.71	8.27	5.47	-3.56
R	6.21	9.21	5.77	0.27
S	6.87	8.28	6.03	-3.44
T	6.89	8.94	6.76	-0.48
U	6.72	9.12	6.41	-3.31
V	6.71	9.64	6.50	-2.60
W	6.39	9.93	6.75	-1.07
X	6.87	10.06	6.51	-1.41
Y	6.48	10.17	7.09	0.66
Z	6.80	10.35	6.85	-0.25

TABLE 3.3: **The 27 states of the structural alphabet.** The values derive from the study of Camproux *et al.* [22].

DSSP sometimes detects several disulfide bridges that share a common cysteine. In our implementation, we discard all cysteine-pairs that have a bond distance greater than 3Å. For conflicting pairs (*i.e.*, sharing a common cysteine), we select the one with the smallest distance.

In practice, several researchers have focused on two intermediate representations, which are detailed below.

Chain bonding state. This simple representation consists in discriminating chains that contain some disulfide bridges from those that do not contain any disulfide bridge. It exploits the key fact that free cysteines (not involved in any bond) and oxidized cysteines (involved in a bond but not necessarily an intra-chain disulfide bridge) rarely co-occur and that theirs sequential environments are different.

Cysteine bonding state. This second commonly used representation consists in identifying cysteines that are involved in a disulfide bridge and those that are not.

CHAPITRE 4

Iterative multi-task sequence labeling

Contents

In the two previous chapters, we introduced relevant background to understand the thesis, according to two domains : supervised learning and structural biology. We showed that experimentally determining tertiary structures is time-consuming and cost-expensive and that *ab initio* prediction of the protein structures (*i.e.,* computing 3D positions of all their atoms, from their amino-acid sequences) remains an extremely difficult and largely unsolved problem.

To progress towards this goal, many research efforts have already been devoted to address surrogate (and simpler) problems. In this chapter, we focus on a set of five protein annotation tasks : *secondary structure* prediction, *DSSP secondary structure* prediction, *solvent accessibility* prediction, *disordered regions* prediction and *structural alphabet* prediction. Since these problems are closely related, we propose to jointly solve them in a multi-task machine learning based approach.

The following section (Section 4.1) gives an overall view of related works. In Section 4.2, we introduce notation and express the addressed problems as sequence labeling tasks.

We then present our generic framework for iteratively learning several sequence labeling tasks in a multi-task context. Section 4.3 describes our experimental protocols and presents the results of the application of our multi-task algorithm on the set five protein annotation tasks. Section 4.4 concludes and highlights further research directions.

This chapter is reporting the work published in [84, 85].

Publications related to this chapter

[1] Maes (Francis), *Becker (Julien)*, Wehenkel (Louis)
Iterative multi-task sequence labeling for predicting structural properties of proteins
19th European Symposium on Artificial Neural Networks (ESANN'11)

[2] Maes (Francis), *Becker (Julien)*, Wehenkel (Louis)
Prédiction structurée multitâche itérative de propriétés structurelles de proteins
7th Plateforme de l'Association Francaise pour l'Intelligence Artificielle (AFIA'11)

4.1 Introduction

In the bioinformatics literature, the structural annotation problems have mostly been treated independently : *i.e.* one designs (*e.g.* by machine learning) and uses a predictor for inferring secondary structure and separately designs and uses another predictor for inferring solvent accessibility. On the other hand, the field of machine learning has investigated in the recent years so-called *multi-task* approaches [23], which aim at treating multiple related prediction tasks simultaneously (both at the learning stage and at the prediction stage) with the hope to get an improvement on each one of the addressed tasks with respect to predictors designed and used in a single task fashion.

Since the various protein structure prediction tasks are closely related, it is a natural idea to explore such multi-task approaches in this context. Although not formulated explicitly in these terms, one example of such a multi-task approach for protein structure prediction has already been proposed by [2], by combining solvent accessibility prediction and secondary structure prediction within a unified system. Recently, after the publication of our research [84], Qi *et al.* [103] proposed a unified multi-task architecture for predicting local protein properties using a deep neural network. Their work was motivated by a previous application of their architecture in the field of natural language processing [34]. As our results, their study demonstrates the power of multi-task learning by achieving a state of art performance on almost all of the protein annotation tasks they considered : predicting the two versions of secondary structure, the solvent accessibility and the transmembrane topology, and identifying DNA-binding residues, protein-binding residues, signal peptides and coiled-coil regions.

However, most multi-task learning approaches rely on the use of an internal representation shared over all considered tasks, such a shared representation being likely to better capture the essence of the input data by exploiting commonalities among the different

tasks. We adopt here another approach to multi-task learning, namely *black-box multi-task learning*[1] : we combine the learning of single-task sequence labeling base-models, where base-models are considered as black boxes and may be of any kind, from simple classification-based approaches to modern structured prediction approaches [74, 126].

4.1.1 Related works

This section presents a brief review of computational methods for predicting secondary structure, solvent accessibility and disordered regions.

Secondary structure prediction. In 1988, Qian *et al.* [104] are the first to propose an automated secondary structure predictor. They used a simple feed-forward neural network on the top of a sliding window of amino acids. A decade later, Rost and Sander [109], and David T. Jones [64] achieved a second breakthrough. The former introduced the notation of evolutionary profiles, which result from sequence alignments against a database. The latter used two feed-forward neural networks based on a sliding window on Position-Specific Scoring Matrices (PSSM). His study makes popular the use of evolutionary profiles. Nowadays, this source of information is very common among predictors that rely on primary structures.

Among the large number of methods devoted to secondary structure prediction in the scientific literature, the majority of them are based on neural networks [104, 109, 64, 9, 101, 100]. The other approaches include nearest neighbors [105], hidden Markov models [20], Bayesian probabilistic models [117], support vector machines [134, 81, 69] and hybrid methods [96]. According to the critical assessment made by Zhang *et al.* [144], the *ab initio* methods PsiPred and Spinex outperform the other secondary structure standalone predictors. PsiPred [64] is the simple method based on PSSM introduced earlier. Rather than directly predicting secondary structures, Spine X [50] prefers to predict the backbone torsion angles (dihedral angles between each consecutive pair of residues) and then to discretize them to obtain secondary structure predictions.

For more details, one can refer to one of the multiple existing reviews in secondary structure prediction [108, 98, 144].

Solvent accessibility prediction. This task consists in distinguishing amino acids of a given protein that are accessible to the solvent from those that are buried inside the protein. In the literature, there also exists more complex definitions such as three- or ten-states of exposure [110], but also regression formulations aiming at the real value of surface exposure prediction [4, 57].

Several approaches were developed to predict solvent accessibility. Some of them are similar to those used to predict secondary structure. These methods include neural networks [110, 1, 3], support vector machines [142, 68, 143], nearest neighbors [66, 120] and Bayesian analysis. Usually, the best predictive performances are achieved by artificial neural networks or support vector machines that use evolutionary profiles. However, the current state of the art is Real-Spine [49], which predicts the real value surface exposure based on predicted backbone torsion angles of proteins.

1. See discussion at `http://hunch.net/?p=160`

Disordered regions prediction. Several automatic methodologies have been proposed to predict disordered regions from primary sequences. They range from simple methods based on the sequence complexity [138] to more sophisticated machine learning approaches often relying on neural networks or SVMs [65, 38, 97, 141, 146]. For example, the POODLE tool is based on three adjacent classifiers, which are specialized in making short [118] or long [61] disordered regions predictions, or unfolded protein predictions [119], while the SPRITZ tool [130] uses two specialized SVMs for either short or long disordered regions. Recently, meta-predictors have also appeared in the literature. These approaches consist in combining predictions of a large number of existing disordered regions predictors [63, 91], *e.g.*, GSMETADISORDER gathers no less than 12 different predictors.

Nowadays, there exist more than 50 disordered region predictors. Fortunately, since 2004, a part of the biannual competition "*Critical Assessment of Techniques for Protein Structure Prediction*" (CASP) is devoted to the comparison of the participant disordered regions predictors. For more information about disordered regions predictors, one can refer to the reports of these assessments [92] or to the recent comprehensive overview of computational protein disorder prediction methods made by Deng *et al.* [39].

4.2 Materials and methods

The first section introduces notations and formalizes the multi-task sequence labeling problem. It then expresses the five structural annotations we consider as sequence labeling tasks. The second section describes our *iterative multi-task black-box* algorithms. The last section details the two datasets further used in our experiments.

4.2.1 Notations and problem statement

The multi-task sequence labeling problem aims to learn a mapping $f(\cdot)$ from input sequences $x \in \mathcal{X}$ to target sequences $y_1, \ldots, y_T \in \mathcal{Y}_1, \ldots, \mathcal{Y}_T$ for each task $t \in [1, \ldots, T]$. We adopt a supervised-learning formulation of the problem, where we assume to have access to a dataset composed of pairs of input sequences associated with some or all of the target sequences. We denote this dataset $D = \{(x^{(i)}, y_1^{(i)}, \ldots, y_T^{(i)})\}_{i \in [1,n]}$, where n is the number of training examples.

More concretely, the input space $\mathcal{X}$ is the space $\mathcal{P}$ of all proteins described by their primary structure and the output spaces $\mathcal{Y}_1, \ldots, \mathcal{Y}_T$ are the five structural-related annotations we consider $\mathcal{A} = \{SS3, SS8, SA, DR, StAl\}$: secondary structure ($SS3$), DSSP secondary structure ($SS8$), solvent accessibility (SA), disordered regions (DR) and structural alphabet ($StAl$). Section 3.4 gives a detailed description of each annotation. Note that the two versions of secondary structure give two different levels of granularity and seem to be redundant but, in our experiments, we have noted an improvement of both tasks when both are present. As well, we used the structural alphabet as a third level of granularity for local 3D structures and this appears to also improve the predictions made for the other tasks.

We denote by $\mathcal{L}_A$ the set of labels corresponding to the annotation $A \in \mathcal{A}$ and by L_A the size of this set. We therefore have : $L_{SS3} = 3$, $L_{SS8} = 8$, $L_{SA} = 2$, $L_{DR} = 2$ and

$L_{StAl} = 27$. For a given primary structure of length $|P|$, where $P \in \mathcal{P}$ is one particular protein, a structural-related annotation A is represented as a set y_A of labels $y_{A,p} \in \mathcal{L}_A$ where $p \in [1, |P|]$ denotes the residue index.

For the sake of clarity, we arbitrary ordered the set of annotations $\{SS3, SS8, SA, DR, StAl\}$ to $[SS3, SS8, SA, DR, StAl]$. We then denote the dataset $D = \{(P^{(i)}, y^{(i)}_{SS3}, y^{(i)}_{SS8}, y^{(i)}_{SA}, y^{(i)}_{DR}, y^{(i)}_{StAl}\}_{i\in[1,n]}$, where $P^{(i)} \in \mathcal{P}$ is the i-th protein and $y^{(i)}_A$ is the structural-related annotation A of $P^{(i)}$. Given the dataset D, the aim is to learn a predictor $f(\cdot)$ that maps proteins $P \in \mathcal{P}$ to sets of predicted annotations $\hat{y}_{SS3}$, $\hat{y}_{SS8}$, $\hat{y}_{SA}$, $\hat{y}_{DR}$, $\hat{y}_{StAl}$ $= f(P)$.

We consider two performance measures to evaluate the quality of predictions : the label accuracy (Q) and the Matthews Correlation Coefficient (MCC) [2]. The label accuracy Q corresponds to the proportion of correctly predicted labels :

$$Q = \frac{1}{\sum_{i=1}^{n} |P^{(i)}|} \sum_{i=1}^{n} \sum_{p=1}^{|P^{(i)}|} \mathbb{1}\left\{ y^{(i)}_{\mathcal{A},p} = \hat{y}^{(i)}_{\mathcal{A},p} \right\},$$

where $\mathbb{1}\{Pr\}$ is the indicator function whose value is 1 if Pr is true or 0 otherwise. The label accuracy is very common in classification. It is often considered as the default scoring measure. However, when the problem is strongly unbalanced, *i.e.*, one class is strongly underrepresented in the evaluated set, using label accuracy is not appropriate. This is typically the case for disordered regions labeling, in which only a few number of examples are disordered. Instead, we have used a classical evaluation measure for disordered regions prediction : the Matthews Correlation Coefficient [2]. The MCC can be calculated by the following formula :

$$MCC_{DR} = \frac{TP \,.\, TN - FP \,.\, FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}},$$

where TP is the number of *true positives*, *i.e.*, the number of correctly predicted positive examples, TN is the number of *true negatives*, FP is the number of *false positives* and FN the number of *false negatives*.

The MCC varies from -1 to $+1$. A coefficient of $+1$ represents prefect predictions, 0 represents predictions that are not better than random and -1 represents the worse predictions. Hence, when the MMC is less than 0, inverting predictions is enough to obtain a better classifier.

4.2.2 Iterative black-box multi-task learning

This section describes in a general manner our multi-task approach, which aims at simultaneously learning several tasks. Our algorithm is based on two unique aspects :

– *Black-box multi-task learning* : we want to combine the learning of single-task base-models, where *base-models* are considered as black boxes and may be of any kind, from simple classification-based approaches to modern structured prediction approaches.

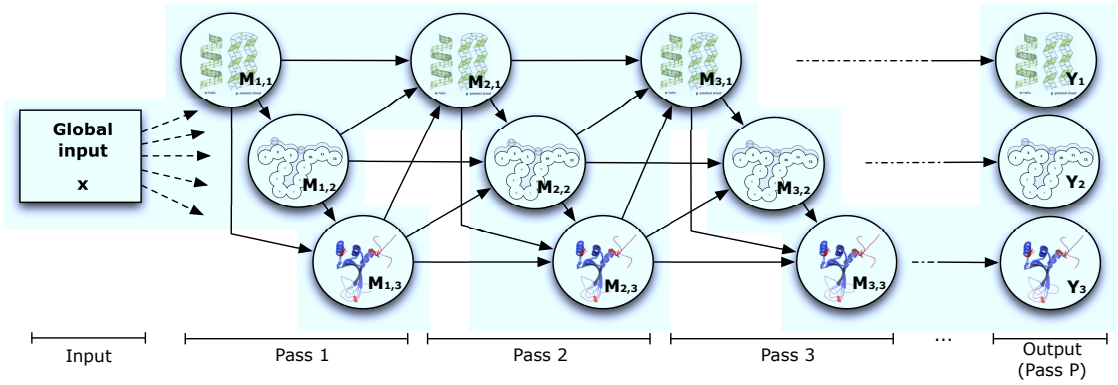

FIGURE 4.1: **Sequential iterative multi-task learning.** Each step (circle) is a distinct base-model, whose inputs are the global input x and the current state s, which is composed of the last predicted target of each task (or ϵ_t if the target is not specified). Each pass is composed of three distinct steps, one per task. The output is directly derived from the state at the end of the last step.

– *Re-estimation of the targets* : we want to iteratively re-estimate the targets using the predictions of the last learned base-models. This is motivated by the fact that the input-output distribution of the underlying learning problems may slightly change from one iteration to another.

In order to treat these requirements, we have developed an iterative multi-task learning approach, in which the core idea is to iteratively re-estimate the targets $y_1, \ldots, y_T$, using at each step the global input x and the last predicted targets of each task as input of the base-model. We denote the "state space" $\mathcal{S}$ of the multi-task problem by $\mathcal{S} = (\mathcal{Y}_1 \cup \{\epsilon_1\}) \times \cdots \times (\mathcal{Y}_T \cup \{\epsilon_T\})$, where ϵ_t denotes a special output label used to represent the fact that the target y_t is not specified. The process is initialized with empty predictions for all targets, *i.e.*, $s = (\epsilon_1, \ldots, \epsilon_T)$. At the first step, the first target y_1 is predicted with a first base-model. A second base-model is then used to predict y_2 given x and the predicted y_1. The third model predicts y_3 given x, and the predictions of y_1 and y_2, and so on. Once all the targets have been estimated once, we have performed one *pass* (iteration). The complete model is composed of $P \times T$ models used in this way by performing P passes sequentially (P is a meta-parameter of the algorithm). Predictions are obtained from the state s at the end of the last pass. Figure 4.1 illustrates a complete model of P passes and three tasks.

The iterative multi-task learning is a chain of base-models denoted $(M_{1,1}, \ldots, M_{1,T}, M_{2,1}, \ldots, M_{P,T})$, where $M_{p,t}$ is the model of the p-th pass and the t-th task. Since targets are re-estimated at each pass, distinct models are learned at each pass ; this is motivated by the fact that, for example, estimating a target for the first time (given the input only) is not the same problem as estimating it for the second time (given the input and the t initial predictions).

Algorithm 1 and Algorithm 2 respectively describe inference and training in our iterative multi-task learning approach. Given the model chain, inference simply chains the base-inferences iteratively, by maintaining $s \in \mathcal{S}$, the current state of all target, *i.e.* $s = (\hat{y}_1, \ldots, \hat{y}_T)$. It is initialized with *unspecified* targets $(\epsilon_1, \ldots, \epsilon_T)$ (line 1) and each

step consists in predicting a target sequence $\hat{y}_t$ and replacing it in the current state (lines 4–5). The final predictions are given by s at the end of inference (line 8).

Algorithm 1 Iterative black-box multi-task inference

Given an input $x \in \mathcal{X}$ and a model chain $(M_{1,1}, \ldots, M_{1,T}, M_{2,1}, \ldots, M_{P,T})$

1: $s \leftarrow (\epsilon_1, \ldots, \epsilon_T)$ ▷ initial state
2: **for** $p = 1$ to P **do** ▷ for each pass
3: **for** $t = 1$ to T **do** ▷ for each task
4: $\hat{y}_t \leftarrow M_{p,t}(x, s)$ ▷ estimate target t
5: $s \leftarrow (s_1, \ldots, s_{t-1}, \hat{y}_t, s_{t+1}, \ldots, s_T)$ ▷ update targets state
6: **end for**
7: **end for**
8: **return** s ▷ return current state of all targets

Training consists in creating the model chain given the training set. Similarly to inference, this is performed iteratively and relies on a set of current states $\{s^{(1)}, \ldots, s^{(n)}\}$, one per training sample. These current states are first initialized to unspecified targets (line 1). Each learning step then adds an element to the model chain. This involves creating a (single-task) training set (line 4), training a base-model (line 5) and updating the current state of each example (line 6). Base-model training inputs contain both the global input x and the current state $s = (\hat{y}_1, \ldots, \hat{y}_T)$.

Algorithm 2 Iterative black-box multi-task training

Given a training set $D = \{(x^{(i)}, y_1^{(i)}, \ldots, y_T^{(i)})\}_{i \in [1,n]}$,
Given a learning base-model algorithm $\mathcal{M}$,
Given a number of passes P,

1: $S \leftarrow \{s^{(i)} = (\epsilon_1, \ldots, \epsilon_T)\}_{i \in [1,n]}$ ▷ initial state
2: **for** $p = 1$ to P **do** ▷ for each pass
3: **for** $t = 1$ to T **do** ▷ for each task
4: $D_t \leftarrow \{((x^{(i)}, s^{(i)}), y_t^{(i)})\}_{i \in [1,n]}$ ▷ create training set
5: $M_{p,t} \leftarrow \mathcal{M}(D_t)$ ▷ train a model for task t
6: $S \leftarrow$ update S given D_t and $M_{p,t}$ ▷ update current state
7: **end for**
8: **end for**
9: **return** $(M_{1,1}, \ldots, M_{1,T}, M_{2,1}, \ldots, M_{P,T})$ ▷ return model chain

It is important to note that, since the chain of models may potentially be long (up to 40 models in our experiments), particular care must be taken to avoid over-fitting. Indeed, training examples may quickly be perfectly learned by the first models in the model chain, hence dangerously biasing the training data for all remaining models of the chain. Since we used very large training sets and simple linear classifiers in our experiments, we did not encounter this problem. However, if necessary, such over-fitting problems could be avoided in at least two ways : either by generating intermediate predictions through the use of cross-validation as exposed for the stacked learning approach in [33], or by introducing noise into intermediate predictions as proposed in [86].

4.2.3 Datasets and annotations

In order to assess our methods, we used two datasets extracted from the Protein Data Bank (PDB) [12]. We built the first one, PDB30, by randomly selecting 500 proteins. To ensure significant differences between training and testing proteins, we based our selection on a subset of the PDB, which is composed of proteins that contain at least 20 amino acids and a maximum pairwise identity of 30%. To compare our method with the state of the art in the field of secondary structure prediction, we used a second dataset, PSIPRED, that has been built by David T. Jones [64]. The proteins that compose PSIPRED are organized into two subsets : a training set and a test set. The training set is a collection of 1385 proteins while the test set is composed of 187 highly resolved structures (resolution < 1.8Å). In order to reduce over-representation of particular protein folds, PSIPRED was filtered using a structural similarity criterion. Namely, rather than removing proteins that share a significant sequence similarity to any member of the testing set, David T. Jones preferred a more stringent criterion based on fold similarity as defined by the CATH protein structure classification [95].

We enrich the primary structure by using evolutionary information in the form of a position-specific scoring matrix (PSSM). We computed the PSSMs by running three iterations of the PSI-BLAST tool [5] on the non-redundant NCBI database [102]. The primary structure of a protein and its associated PSSM form the input x of our method whereas the supervision information $y_{SS3}, y_{SS8}, y_{SA}, y_{DR}, y_{StAl}$ are derived from the structure as described in Section 3.4.

4.3 Multi-task sequence labeling for protein annotations

This section describes our empirical experiments on a set of five protein annotation tasks. We first present the base-model and detail the way to adapt our iterative black-box multi-task method into an iterative multi-task sequence labeling method. We then introduce the feature encoding scheme used in our experiments to make the input data usable by the base-model. We finally apply our method on the PDB30 and PSIPRED datasets and propose a comparison between “single-task" learned models and “multi-task" learned models.

4.3.1 Base sequence-labeling model

In order to deal with the large number of base-models to learn, we adopted a simple classification-based approach : a linear SVM model trained with stochastic gradient descent (Section 2.2). This kind of classifier has the main advantage to handle large amounts of data in a reasonable manner, *e.g.*, its learning time complexity is linear with respect to the number of training examples and its evaluation solely consists in performing a dot product $F_\theta(x) = \langle \theta, x \rangle$.

In our experiments, we used the hinge loss as the loss function Δ. For binary classification (SA and DR), we used the sign function of $F_\theta(x)$ to determine the output class whereas, for multiclass problems ($SS3$, $SS8$ and $StAl$), we defined one weight vector θ_y per class $y \in \mathcal{Y}$. The outcome is determined as the class with the highest score $F_{\theta_y}(x)$.

In order to express the uncertainties around predictions made by a model and further used as inputs of other models, we normalized the predicted values into probabilities. To achieve this conversion, we relied on the following strict monotonic function $\sigma : \mathbb{R} \rightarrow [0, 1]$:

$$\sigma(r) = \frac{1}{1 + exp(-r)}.$$

The aim of σ is to scale any real value into the range [0,1]. In the case of binary classification, the probability of classifying an example x as positive is $\mathbb{P}(x) = \sigma(F_\theta(x))$ and the probability of classifying x as negative is hence obtained by $1 - \mathbb{P}(x)$. In multiclass problems, we estimated the probabilities in a similar way :

$$\mathbb{P}_l(x) = \frac{\sigma(F_{\theta_l}(x))}{\sum_{\forall y \in \mathcal{Y}} \sigma(F_{\theta_y}(x))}, \forall l \in \mathcal{Y}$$

Namely, we scale each score $F_{\theta_y}(x)$ as for the binary case, and then we normalize the values in order to sum to 1. These probabilities are further used to compute features used at subsequent passes.

According to preliminary experiments on the training set, we tuned the learning rates to 5 for $SS3$, 10 for $SS8$ and 1 for all other targets, and we limited the number of learning iterations to 100, which seemed to yield a good bias/variance tradeoff.

Note that, for the moment, the proposed base-model is unable to directly deal with sequences. To circumvent this difficulty, we slightly adapted the method : each label is predicted independently of the others. Consequently, the aim of linear SVM is to learn a function $f_\theta(\cdot)$ that maps pairs (P, P_i), where P_i is the i-th residue of the protein P, to predicted labels $\hat{y}_i^{\mathcal{A}}(P) = f_\theta((P, P_i))$.

4.3.2 Features encoding

The pairs (P, P_i) are encoded in an appropriate form for the classification algorithm. This encoding is performed through a feature function $\phi : \mathcal{P} \times \mathcal{R} \rightarrow \mathbb{R}^d$ that, given a protein $P \in \mathcal{P}$ and one of its residues $P_i \in \mathcal{R}$, where $\mathcal{R}$ is the space of residues, computes a vector of d real-valued features describing local and global properties of the protein.

Some of our features are annotation-related features and are defined for each type of annotation $\mathcal{A} \in \{AA, PSSM, SS3, SS8, SA, DR, StAl\}$, where AA is the primary structure. A predicted annotation $\mathcal{A}$ is represented as a set of probabilities $\alpha_{p,l}^{\mathcal{A}} \in [0, 1]$ where $p \in [1, |P|]$ denotes the residue index and $l \in \mathcal{L}_{\mathcal{A}}$ is a label. The $\alpha_{p,l}^{\mathcal{A}}$ probabilities reflect uncertainties about predictions. Note that the primary structure (AA) is always known perfectly. Therefore, we have $\alpha_{p,l}^{AA} = 1$ if l is the residue at position p or 0 otherwise.

Our feature set is similar to those proposed by [64, 145] and is as follows :

- *Number of residues* : computes one feature which is the number of residues in the primary structure.
- *Labels global histogram* : returns one feature per label $l \in \mathcal{L}_{\mathcal{A}}$, equal to $\frac{1}{|P|} \sum_{p=1}^{|P|} \alpha_{p,l}^{\mathcal{A}}$. These features describe the distribution of labels $\mathcal{L}_{\mathcal{A}}$.

- *Labels local histogram* : returns one feature per label $l \in \mathcal{L}_{\mathcal{A}}$, per residue P_i, equal to $\frac{1}{W}\sum_{i-W/2}^{i+W/2} \alpha_{p,l}^{\mathcal{A}}$ and one special feature equal to the percentage of out-of-bounds positions, *i.e.*, position p such that $p \notin [1, |P|]$. These features the distribution of labels $\mathcal{L}_{\mathcal{A}}$ among the W residues around P_i.
- *Labels local window* : returns one feature per label $l \in \mathcal{L}_{\mathcal{A}}$, per residue P_i and per relative position $\delta \in [-\frac{W}{2}, \frac{W}{2}]$, equal to $\alpha_{i+\delta,l}$. When the position is out-of-bounds, *i.e.*, $i + \delta \notin [1, |P|]$, the feature is set to 0.

According to the literature [64] and some preliminary experiments, we fixed the window size W to 15 for both local histograms and local windows.

Note that in the first stage, in which the state s is initialized with empty predictions for all targets, the annotations $SS3$, $SS8$, SA, DR and $StAl$ are not necessary available. In such a case, the features corresponding to missing annotations are set to 0.

Feature discretization. In order to deal with the non-linear behavior of features, we enlarged the feature space by independently discretizing each feature described above into partitions of k equal width. This method, called *equal width interval binning* [42], consists in dividing the range of observed values of a variable $x \in [x_{\min}, x_{\max}]$ into k new binary variables $[bin_1(x), \ldots, bin_k(x)]$ defined as :

$$bin_i(x) = \begin{cases} 1 & \text{if } x \text{ is inside the } i\text{-th interval} \\ 0 & \text{otherwise.} \end{cases}$$

Formally, the i-th interval is delimited by $[x_{\min} + (i-1).w;\ x_{\min} + i.w[$.

In our experiments, the number of residues was discretize into $k = 20$ intervals whereas probability features were discretized into $k = 5$ partitions.

4.3.3 Results

We have trained iterative multi-task sequence labeling with up to $P = 8$ passes, which gives model chains of length $P \times T = 40$. To observe the effect of the iterative re-estimation of targets, we have evaluated each task by "cutting" the model chain after a given number of passes $P_{\max} \in [1, 8]$. Figure 4.2 gives the test scores for each task as a function of the number of passes on the PsiPred dataset. It is clear that all the tasks benefit from iterative re-estimation of targets, especially during the first passes. During the last passes, some scores occasionally degrade, but we do not observe strong over-fitting in these experiments. Importantly, in all cases, the re-estimated targets after several passes are significantly better than the initially estimated targets.

To measure to what extend our positive results are due to multi-tasking, we have performed one baseline experiment per task by using iterative sequence labeling in a single-task setup. These baselines rely on iterative re-estimation of targets, but do not use predictions from the other tasks. The comparison between our multi-task model and its single-task counterparts is given in Table 4.1, for models inferred after 5 passes. We observe from these results that on both datasets, the multi-task approach systematically outperforms the single-task approach, *e.g.* : +2.31% for secondary structure prediction and +0.114 MCC for disordered regions prediction on the PsiPred testing set. We also

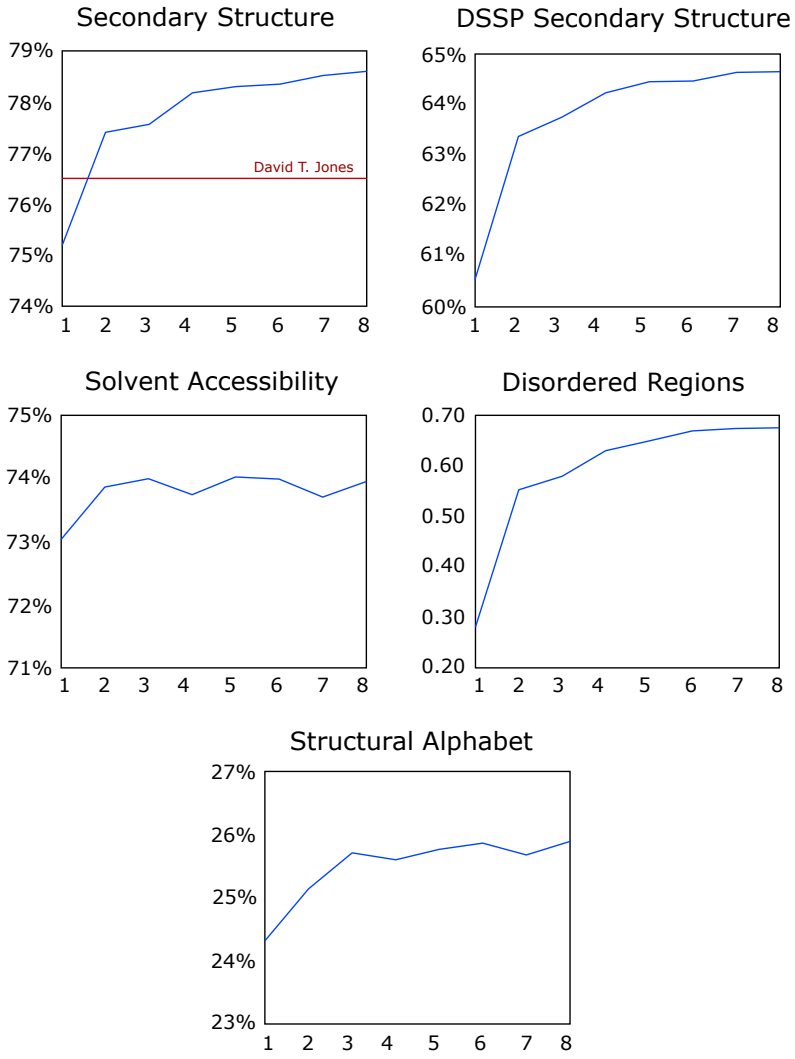

FIGURE 4.2: **Test scores after growing numbers of passes on the PSIPRED dataset.** Scores are expressed in label accuracy, except for disordered regions, which is expressed in MCC. The red line shows the state of the art results [64] on the secondary structure prediction task.

observe that only the multi-task approach outperforms the state-of-the-art results on PSIPRED with +2.1% improvement.

4.4 Conclusion

In this chapter, we have introduced a conceptually simple framework for *iterative multi-task learning*, a new multi-task machine learning approach to jointly solve multiple related tasks, and which can take advantage of any sequence labeling algorithm. We have made experiments with a set of five protein sequence labeling tasks and by using a linear SVM base-learner trained by stochastic gradient descent. In this setting, we have shown that our approach systematically outperforms single-task learning on all tasks and on two

		PDB30		PSIPRED	
Task	**Labels**	**Single-task**	**Multi-task**	**Single-task**	**Multi-task**
Secondary structure	3	75.45%	76.35%	76.29%	78.60%
DSSP Sec. structure	8	60.38%	62.69%	62.25%	64.64%
Solvent accessibility	2	71.56%	73.52%	73.51%	73.95%
Disordered regions	2	0.4212	0.4983	0.5611	0.6749
Structural alphabet	27	16.81%	18.14%	24.88%	25.89%

TABLE 4.1: **Single-task vs multi-task**. Performance scores are evaluated at the 5-th pass on the test sets. Scores are expressed in label accuracy, except for disordered regions, which is expressed in MCC.

datasets of medium and large scale. We have also shown that our approach significantly outperforms the state-of-the-art (+2.1% improvement) results for secondary structure prediction.

Since our iterative multi-task approach is - as a matter of fact - not restricted to predicting sequence labels, we believe that the iterative multi-task framework proposed in this study may be applied in many other complex application domains (text processing, image analysis, network monitoring and control, robotics), where data is available about several related tasks and where synergies could similarly be exploited to enhance machine learning solutions.

CHAPITRE 5

Feature selection for disulfide connectivity prediction

Contents

In the previous chapter, we introduced an *iterative multi-task sequence labeling* framework and applied it on a set of five related structural annotation tasks. We now move on a much more challenging task : prediction of disulfide connectivity patterns. Here, the outputs are graphs and the aim is to identify disulfide bridges, which can strongly constrain the native structure of many proteins. Predicting their formation is therefore a key sub-problem of protein structure and function inference. However, the comparison of the conclusions of works in the literature is difficult because they often slightly differ in their experimental protocol.

In this chapter, we propose an extensive study of the relevance of various structural annotations and feature encodings. Section 5.1 motivates and explains our main contribution with respect to the most recent successful methods developed to solve this problem. Section 5.2 gives an overall view of related work and its multiple sub-problems. Section 5.3 introduces the problem statement and formalizes it as a three step pipeline. It then details the different steps : the datasets and their annotations, the set of candidate feature functions, which aim at describing cysteine pairs in an appropriate form for three supervised learning algorithms we consider, and a post-processing step based on a maximum weight graph matching. Finally, the last part of Section 5.3 introduces our *forward feature function selection* algorithm, which aims at identifying a set of relevant features functions among those proposed. Section 5.4 determines the most promising supervised learning algorithms in the context of disulfide pattern prediction and reports the set of relevant feature functions obtained by applying our feature selection algorithm. In Section 5.5, we perform our feature selection approach on two sub-problems : chain classification and cysteine bonding state prediction. Section 5.6 concludes the chapter.

This chapter is reporting the work published in [11].

Publication related to this chapter

Becker (Julien), Maes (Francis), Wehenkel (Louis)
On the relevance of sophisticated structural annotations for disulfide connectivity pattern prediction
PLoS One, 2013

Software related to this chapter

From this study, we made available a web-application at

`http://m24.giga.ulg.ac.be:81/x3CysBridges`.

x3CysBridges is a tool designed for biologists that attempt to determine the three-dimensional structure of protein molecules. Based on the fact that disulfide bridges add strong constraints to the native structure, the main function of *x3CysBridges* is to predict the disulfide bonding probability of each cysteine-pair of a protein and to propose a disulfide connectivity pattern that maximizes the sum of these probabilities.

5.1 Introduction

Given an input primary structure, the disulfide pattern prediction problem consists in predicting the set of disulfide bridges appearing in the tertiary structure of the corresponding protein. This problem can be formalized as an edge prediction problem in a graph whose nodes are cysteine residues, under the constraint that a given cysteine is linked to at most to a single other one. Most recent successful methods to solve this problem are pipelines composed of three steps, which are illustrated in Figure 5.1. First, they enrich the primary structure using evolutionary information and sometimes structural-related

predictions. Second, they apply a binary classifier to each pair of cysteines to estimate disulfide bonding probabilities. Finally, they use a maximum weight graph matching algorithm to extract a valid disulfide pattern maximizing the sum of these probabilities.

The central component of this three step pipeline is the binary classifier that predicts bonding probabilities for all cysteine pairs. The wide majority of available binary classification algorithms cannot process complex objects such as cysteine pairs natively, hence they require the user to encode such objects into vectors of (categorical or numerical) features. Since the way to perform this encoding typically has a major impact on the classification accuracy, a large body of work has been devoted to studying different feature representations for cysteines and cysteine-pairs. However, it is often the case that these different studies rely on different kinds of binary classifiers and slightly differ in their experimental protocol. Therefore, the comparison of the conclusions of these works is difficult. In consequence, the relevance of some features is still a subject under heavy debate. It is for example not clear whether the use of (predicted) secondary structure or (predicted) solvent accessibility can significantly improve disulfide pattern predictors [30, 78, 53].

The main contribution of our work reported in this chapter is an extensive study which aims at establishing the relevance of various structural-related annotations and of various feature encodings in the context of a disulfide pattern predictor such as the one presented in Figure 5.1. We consider various structural annotations, some which were already studied in the context of disulfide pattern prediction – position-specific scoring matrix, secondary structure and solvent accessibility – and some others which are more original in this context : 8-class secondary structure, disordered regions and structural alphabet. For each such annotation, we consider four different procedures in order to encode it as a feature vector. The combination of annotations with feature encodings leads to a large set of possible feature functions. In order to identify a minimal subset of feature functions that are relevant to disulfide pattern prediction, we introduce a tractable and interpretable feature selection methodology, based on forward selection of feature functions. We adopt a computational protocol that avoids any risk of overfitting and apply our approach in combination with two usual classifiers : k-nearest neighbors (kNN) and support vector machines (SVMs), as well as with one classifier, which was not yet considered for disulfide pattern prediction : extremely randomized trees (ETs)[58].

As a result of this study, we show that only a very limited number of feature functions are sufficient to construct a high performance disulfide pattern predictor and that, when using these features, extremely randomized trees reach a disulfide pattern accuracy of 58.2% on the benchmark dataset SPX+, which corresponds to +3.2% improvement over the state of the art. However, since SPX+ only contains proteins with at least one intrachain disulfide bridge, we further consider the more heterogeneous and less redundant benchmark dataset SPX− which also contains a significant number of proteins without any intrachain bridge. We then investigate the behavior of our disulfide pattern predictor on both datasets by coupling it with filters predicting the presence of intrachain bridges and the bonding states of individual cysteines. We consider both the case where bonding states are known a priori and the case where bonding states are estimated thanks to another predictor. We show that predicting the bonding states significantly improves our disulfide pattern predictor on SPX−, but slightly degrades it on SPX+. When the bonding states are known a priori, we reach very high accuracies : 89.9% on SPX− and

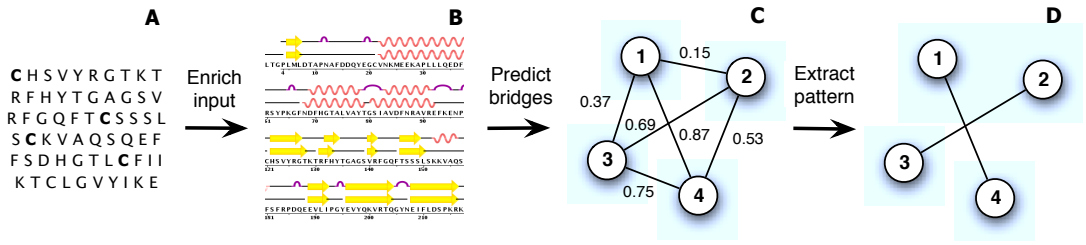

FIGURE 5.1: **Three-step approach for disulfide pattern prediction.** (A) an input primary structure, which contains four cysteine residues. (B) The sequence is first enriched using evolutionary information and sometimes structural-related predictions such as the secondary structure. (C) A bridge classifier, then, predicts disulfide bonding probabilities for each cysteine pair and finally (D) a graph matching algorithm extracts the disulfide pattern with maximal weight.

75.8% on SPX+.

5.2 Related works

The following two sub-sections give an overall view of related work by first discussing multiple sub-problems of disulfide pattern prediction and then presenting the kinds of features that have been proposed to describe cysteines and cysteine pairs in supervised learning approaches. We refer the reader to [48] for an extensive recent overview of the field.

5.2.1 Disulfide bridge related prediction problems

While the ultimate goal of disulfide bridge prediction is to infer correctly the whole connectivity pattern of any protein from its primary sequence, several researchers have focused on intermediate simpler sub-problems, which are detailed below.

Chain classification. This sub-problem aims at predicting for a given protein, whether (a) none of its cysteines participate to a disulfide bridge, (b) some of its cysteines are involved in disulfide bridges or (c) all of its cysteines are involved in disulfide bridges. Frasconi *et al.* [55] proposed a support vector machine classifier to solve this task. Fiser *et al.* [54] have exploited the key fact that free cysteines (not involved in any bond) and oxidized cysteines (involved in a bond but not necessarily in a an intra-chain disulfide bridge) rarely co-occur and they showed that their sequential environments are different. From those observations, subsequent studies have reduced this sub-problem to a binary classification task : (a) or (c).

Cysteine bonding state prediction. This second commonly studied sub-problem consists in classifying cysteines into those that are involved in a disulfide bridge and those that are not. To solve this binary classification problem, several machine-learning

algorithms were proposed such as multi-layer neural networks [52], two-stage support vector machines that exploit chain classification predictions [55] and hidden neural networks [88].

Disulfide bonding prediction. While chain classification works at the protein level and cysteine bonding state prediction works at the cysteine level, disulfide bonding prediction works at the level of cysteine pairs and aims at predicting the probability that a specific pair of cysteines will form a disulfide bridge during protein folding. Depending on the studies, some authors assume to have an *a priori* knowledge on the bonding state of isolated cysteines. This prior knowledge can be the actual state [131, 106, 83] or a prediction made by a cysteine bonding state predictor [26].

Disulfide pattern prediction. Once one or several of the previous tasks have been solved, the most challenging step is to predict the disulfide connectivity pattern. Fariselli *et al.* [51] were the first to relate the problem of predicting the disulfide pattern to a maximal weight graph matching problem. Several authors have since adopted this approach and proposed disulfide pattern predictors that fit into the three step pipeline of Figure 5.1. Baldi *et al.* [10, 30] have used two-dimensional recursive neural networks to predict bonding probabilities, which are exploited by a weighted graph matching algorithm. Lin *et al.* [78, 79] used the same graph matching approach while predicting bonding probabilities with support vector machines.

5.2.2 Features for cysteines and cysteine pairs

Machine learning algorithms are rarely able to process complex objects such as cysteine pairs directly, hence it is necessary to define a mapping from these objects to vectors of features. A large body of research on disulfide bridge prediction has been devoted to the analysis of such encodings into feature vectors.

In 2004, Vullo *et al.* [131] suggested to incorporate evolutionary information into features describing cysteines. For each primary sequence, they generate a position-specific scoring matrix (PSSM) from a multiple alignment against a huge non-redundant database of amino-acid sequences. This evolutionary information was shown to significantly improve the quality of the predicted disulfide bridges, which led the large majority of authors to use it in their subsequent studies. Generally, the PSI-BLAST program [5] is used to perform multiple alignments against the SWISS-PROT non-redundant database [124].

Zhao *et al.* [147] introduced cysteine separation profiles (CSPs) of proteins. Based on the assumption that similar disulfide bonding patterns lead to similar protein structures regardless of sequence identity, CSPs encode sequence separation distances among bonded cysteine residues. The CSP of a test protein is then compared with all CSPs of a reference dataset and the prediction is performed by returning the pattern of the protein with highest CSP similarity. This approach assumes to have an *a priori* knowledge on the bonding state of cysteines. In our work, we introduce a slightly different definition of CSPs based on separation distances among all cysteine residues (see *Candidate feature functions*).

From the earlier observation that there is a bias in the secondary structure preference of bonded cysteines and non-bonded cysteines, Ferrè *et al.* [53] have developed a neural network using predicted secondary structure in addition to evolutionary information. Cheng *et al.* [30] proposed to also include predictions about the solvent accessibility of residues. The predictions of secondary structure and/or solvent accessibility used in their experiments were however not accurate enough to obtain significant performance improvements. Nevertheless, they observed that using the *true values* of secondary structure and solvent accessibility can lead to a small improvement of 1%. More recently, Lin *et al.* [78] proposed an approach based on support vector machines with radial basis kernels combined with an advanced feature selection strategy. They observed a weak positive influence by using predicted secondary structure descriptors, but their experimental methodology could suffer from overfitting so that this result should be taken with a grain of salt. Indeed, in this study, the same data is used both for selecting features and for evaluating the prediction pipeline. As detailed in [6], proceeding in this way often leads to an overfitting effect and hence to over-optimistic scores. Notice that the three studies [53, 30, 78] were all based on the secondary structure predicted by the PSIPRED predictor [64].

More recently, Savojardo *et al.* [114] reported an improvement of their predictive performance by taking into consideration the relevance of protein subcellular localization since the formation of disulfide bonds depends on the ambient redox potential.

5.3 Materials and Methods

5.3.1 Notations and problem statement

This section introduces notations and formalizes the disulfide pattern prediction problem. Let $\mathcal{P}$ be the space of all proteins described by their primary structure and $P \in \mathcal{P}$ one particular protein. We denote $\mathbf{C}(P) = (C_1(P), \ldots, C_{n_C}(P))$ the sequence of $n_C = |\mathbf{C}(P)|$ cysteine residues belonging to protein P, arranged in the same order as they appear in the primary sequence. A disulfide bonding connectivity pattern (or disulfide pattern) is an undirected graph $G = (\mathbf{C}(P), B)$ whose nodes $\mathbf{C}(P)$ are cysteines and whose edges B are the pairs of cysteines $\{(C_i, C_j)\}$ that form a disulfide bridge.

Since a given cysteine can physically be bonded to at most one other cysteine, valid disulfide patterns are those that respect the constraint $degree(C_i) \leq 1, \forall i \in [1, n_C]$. This constraint enables to trivially derive an upper bound on the number b of disulfide bridges given the number of cysteines : $b \leq \lfloor \frac{n_C}{2} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. If we know in advance the number $b \geq 1$ of disulfide bridges, we can derive the number of valid disulfide patterns using the following closed form formula [122] :

$$C_{n_C}^{2b} \prod_{i=1}^{i \leq b} (2i - 1), \tag{5.1}$$

where $C_{n_C}^{2b} = \frac{n_C!}{(2b)!(n_C - 2b)!}$ denotes the number of possible subsets of size $2b$ of the set of n_C cysteines. As an example, a protein with $n_C = 6$ cysteines and $b = 3$ bridges has 15 possible disulfide patterns and a protein with $n_C = 11$ cysteines and $b = 5$ bridges has

$11 \times 945 = 10\ 395$ possible patterns. Figure 5.2 illustrates the three possible disulfide connectivity patterns of a protein with four cysteines and two disulfide bridges.

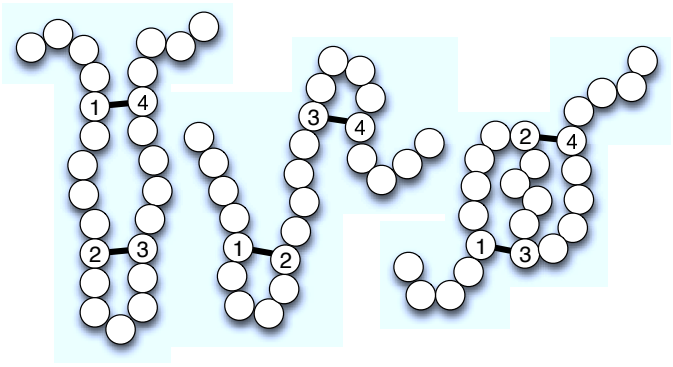

FIGURE 5.2: **Example of disulfide patterns.** A protein with two disulfide bridges and its three possible disulfide connectivity patterns.

When the number of bridges is unknown, the number of possible disulfide connectivity patterns for a protein P with n_C cysteines becomes

$$\sum_{b=1}^{\lfloor n_C/2 \rfloor} C_{n_C}^{2b} \prod_{i=1}^{i \leq b} (2i-1) + 1. \tag{5.2}$$

Note that the term $+1$ represents the case where no cysteine residue is bonded. As an example, a protein with $n_C = 10$ cysteines has $45 \times 1 + 210 \times 3 + 210 \times 15 + 45 \times 105 + 1 \times 945 + 1 = 9\ 496$ possible valid disulfide patterns.

We adopt a supervised-learning formulation of the problem, where we assume to have access to a dataset of proteins (represented by their primary structure) with associated disulfide patterns. We denote this dataset $D = \{(P^{(i)}, B^{(i)})\}_{i \in [1,N]}$, where $P^{(i)} \in \mathcal{P}$ is the i-th protein and $B^{(i)}$ is the set of disulfide bridges associated to that protein. We also denote $n_C^{(i)} = |\mathbf{C}(P^{(i)})|$ the number of cysteines belonging to the protein $P^{(i)}$. Given the dataset D, the aim is to learn a disulfide pattern predictor $f(\cdot)$: a function that maps proteins $P \in \mathcal{P}$ to sets of predicted bridges $\hat{B} = f(P)$. Given such a predicted set, we can define the predicted connectivity pattern as following : $\hat{G} = (\mathbf{C}(P), \hat{B})$.

We consider two performance measures to evaluate the quality of predicted disulfide patterns : Q_p and Q_2. Q_p is a protein-level performance measure that corresponds to the proportion of entirely correctly predicted patterns :

$$Q_p = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left\{B^{(i)} = \hat{B}^{(i)}\right\}, \tag{5.3}$$

where $\mathbb{1}\{Pr\}$ is the indicator function whose value is 1 if Pr is true or 0 otherwise. Q_2 is a cysteine-pair level performance measure that corresponds to the proportion of cysteine pairs that were correctly labeled as *bonded* or *non-bonded* :

$$Q_2 = \frac{1}{\sum_{i=1}^{N} n_C^{(i)}(n_C^{(i)} - 1)/2} \sum_{i=1}^{N} \sum_{j=1}^{n_C^{(i)}} \sum_{k=j+1}^{n_C^{(i)}} \mathbb{1}\left\{\left(\{C_j, C_k\} \in B^{(i)}\right) = \left(\{C_j, C_k\} \in \hat{B}^{(i)}\right)\right\}. \tag{5.4}$$

Note that both Q_p and Q_2 belong to the interval $[0, 1]$ and are equal to 1 in case of perfectly predicted disulfide patterns. While the ultimate goal of disulfide pattern prediction is to maximize Q_p, we will also often refer to Q_2 since, in the pipeline depicted in Figure 5.1, Q_2 is directly related to the quality of the cysteine pair classifier.

5.3.2 Disulfide pattern prediction pipeline

This section first presents the datasets and the five kinds of structural-related predictions we consider. It then details the different steps of our prediction pipeline : the dataset annotation, the pre-processing step that enriches the primary structure with evolutionary information and structural-related annotations, the classification step of cysteine pairs that predicts bridge bonding probabilities and the post-processing step that constructs a disulfide pattern from these probabilities using maximum weight graph matching.

Dataset and annotations

In order to assess our methods, we use two datasets that have been built by Cheng *et al.* [30] and extracted from the Protein Data Bank [12]. The first one, SPX+, is a collection of 1 018 proteins that contain at least 12 amino acids and at least one intrachain disulfide bridge. We use this dataset for the problem of pattern prediction. However, since it does not contain any protein without disulfide bridges it is not adapted to address chain classification and cysteine bonding state prediction. For these tasks, we use the other dataset, SPX−, which is made of 1 650 proteins that contain no disulfide bridge and 897 proteins that contain at least one bridge. In order to reduce the over-representation of particular protein families, both datasets were filtered by UniqueProt [123], a protein redundancy reduction tool based on the HSSP distance[113]. In SPX−, Cheng *et al.* used a HSSP cut-off distance of 0 for proteins without disulfide bridge and a cut-off distance of 5 for proteins with disulfide bridges. In SPX+, the cut-off distance was set to 10. To properly compare our experiments with those of Cheng *et al.*, we use the same train/test splits as they used in their paper. Statistics of the two datasets are given in Table 5.1.

	Proteins				**Cysteines**			**Bonds**
	All	None	Mix	Total	Positive	Negative	Total	**per protein**
SPX−	757	1 650	140	2 547	4 942	7 844	12 786	0.97 ± 1.78
SPX+	718	0	300	1 018	5 082	901	5 983	2.50 ± 2.14

TABLE 5.1: **Dataset statistics.** *All* : proteins in which all cysteines are bonded. *None* : proteins with no disulfide bridges. *Mix* : proteins with both bonded cysteines and non-bonded cysteines. *Positive* : number of bonded cysteines. *Negative* : number of non-bonded cysteines.

We enrich the primary structure (denoted as AA) by using two kinds of annotations : evolutionary information in the form of a position-specific scoring matrix (PSSM) and structural-related predictions, such as predicted secondary structure or predicted solvent accessibility. We computed the PSSMs by running three iterations of the PSI-BLAST

program [5] on the non-redundant NCBI database. To produce structural-related predictions, we use the iterative multi-task sequence labeling method detailed in the Chapter 4. To our best knowledge, predicted DSSP secondary structure, predicted disordered regions and structural alphabet annotations have never been investigated in the context of disulfide pattern prediction.

We use the cross-validation methodology proposed in [33] that works as follows. First, we randomly split the dataset into ten folds. Then, in order to generate "true" predictions for one fold, we train our system on all data except this fold. This procedure is repeated for all ten folds and all predictions are concatenated so as to cover to whole dataset.

Table 5.2 reports the cross-validation accuracies that we obtained with this procedure. The default scoring measure is label accuracy, *i.e.*, the percentage of correctly predicted labels on the test set. Since disordered regions labeling is a strongly unbalanced problem, label accuracy is not appropriate for this task. Instead, we used a classical evaluation measure for disordered regions prediction : the Matthews correlation coefficient [2].

Annotation	$\mathbf{L_A}$	**Measure**	**SPX+**	**SPX−**
Secondary structure	3	Accuracy	73.50% ± 0.68%	68.00% ± 2.61%
Secondary structure	8	Accuracy	55.60% ± 0.76%	57.83% ± 2.10%
Solvent accessibility	2	Accuracy	77.45% ± 0.54%	77.82% ± 0.30%
Disorder regions	2	MCC	0.892 ± 0.03	0.352 ± 0.05
Structural alphabet	27	Accuracy	19.01% ± 0.30%	21.32% ± 0.47%

TABLE 5.2: **Cross-validated accuracies of annotations.** The scoring measure is label accuracy, *i.e.*, the percentage of correctly predicted labels on the test set except for disordered regions that use the Matthews correlation coefficient (MCC).

Candidate feature functions

The feature generation step aims at describing cysteine pairs in an appropriate form for classification algorithms. This encoding is performed through *cysteine-pair feature functions* $\phi : \mathcal{P} \times \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}^d$ that, given a protein P and two of its cysteines (C_i, C_j), computes a vector of d real-valued features. In our experiments, we extracted cysteine-pairs (C_i, C_j) in such a way that $1 \leq i < j \leq n_C$, where n_C is the number of cysteine residues of P. Consequently, we extract $\frac{n_C \times (n_C - 1)}{2}$ cysteine-pairs from P. The purpose of the feature selection methodology described in the next section is to identify a subset of relevant ϕ functions among a large panel of candidate ones that we describe now.

Our set of candidate feature functions is composed of primary-structure related functions and annotation related functions. The former are directly computed from the primary structure alone and are the following ones :

- *Number of residues* : computes one feature which is the number of residues in the primary structure.
- *Number of cysteines* : computes one feature which is the number of cysteine residues in the primary structure.
- *Parity of the number of cysteines* : computes one feature which indicates whether the number of cysteines is odd or even.

- *Relative position of cysteines* : computes two features which are the residue indices of cysteines C_i and C_j, denoted $pos(C_i)$ and $pos(C_j)$, divided by the protein length.
- *Normalized position difference* : returns one feature which corresponds to the number of residues separating C_i from C_j in the primary structure, *i.e.*, $pos(C_j) - pos(C_i)$, divided by the protein length. Note that as $j > i$ and therefore $pos(C_j) > pos(C_i)$, this difference is always greater than zero.
- *Relative indices of cysteines* : computes two features which are the cysteine indices i and j divided by the number of cysteines.
- *Normalized index difference* : computes one feature which corresponds to the number of cysteines separating C_i from C_j divided by the number of cysteines.
- *Cysteine separation profile window* : computes one feature per cysteine $C_k \in \{C_i, C_j\}$ and per relative position $\delta \in [-\frac{W}{2}, \frac{W}{2}], \delta \neq 0$ whose value is the position difference $pos(C_{k+\delta}) - pos(C_k)$ divided by the protein length, where $W > 0$ is called the window size parameter.

Annotation-related feature functions are defined for each type of annotation $\mathcal{A} \in \{AA, PSSM, SS3, SS8, SA, DR, StAl\}$ of the residues of the protein P. We denote by $\mathcal{L}_\mathcal{A}$ the set of labels corresponding to annotation $\mathcal{A}$ and by $L_\mathcal{A} = |\mathcal{L}_A|$ the size of this set. For our annotations, we have : $L_{AA} = 20$, $L_{PSSM} = 21$ (the twenty amino acids and the gap), $L_{SS3} = 3$, $L_{SS8} = 8$, $L_{SA} = 2$, $L_{DR} = 2$ and $L_{StAl} = 27$. For a given primary structure of length $|P|$, an annotation $\mathcal{A}$ is represented as a set of probabilities $\alpha^\mathcal{A}_{p,l} \in [0, 1]$ where $p \in [1, |P|]$ denotes the residue index and $l \in \mathcal{L}_\mathcal{A}$ is a label.

Note that in the general case, $\alpha^\mathcal{A}_{p,l}$ probabilities may take any value in range $[0, 1]$ to reflect uncertainty about predictions. Since the primary structure (AA) is always known perfectly, we have :

$$\alpha^{AA}_{p,l} = \begin{cases} 1 & \text{if } l \text{ is the residue at position } p \\ 0 & \text{otherwise.} \end{cases}$$

For each annotation $\mathcal{A}$, we have four different feature functions :

- *Labels global histogram* : returns one feature per label $l \in \mathcal{L}_\mathcal{A}$, equal to $\frac{1}{|P|} \sum_{p=1}^{|P|} \alpha^\mathcal{A}_{p,l}$.
- *Labels interval histogram* : returns one feature per label $l \in \mathcal{L}_\mathcal{A}$ equal to $\frac{\sum_{p=pos(C_i)}^{pos(C_j)} \alpha^\mathcal{A}_{p,l}}{pos(C_j) - pos(C_i) + 1}$.
- *Labels local histogram* : returns one feature per label $l \in \mathcal{L}_\mathcal{A}$ and per cysteine $C_k \in \{C_i, C_j\}$, equal to $\frac{1}{W} \sum_{p=pos(C_k)-W/2}^{pos(C_k)+W/2} \alpha^\mathcal{A}_{p,l}$ and one special feature equal to the percentage of out-of-bounds positions, *i.e.*, positions p such that $p \notin [1, |P|]$.
- *Labels local window* : returns one feature per label $l \in \mathcal{L}_\mathcal{A}$, per cysteine $C_k \in \{C_i, C_j\}$ and per relative position $\delta \in [-\frac{W}{2}, \frac{W}{2}]$, equal to $\alpha_{pos(C_k)+\delta,l}$. When the position is out-of-bounds, *i.e.*, $pos(C_k) + \delta \notin [1, |P|]$, the feature is set to 0.

Our candidate feature functions are summarized in Table 5.3. Note that three of them are parameterized by window size parameters. Figure 5.3 shows an illustration of the three kinds of histograms. We will see how to tune window sizes and how to select a minimal subset of feature functions in the next section.

Cysteine pair classifiers

Let $\{\phi_1, \ldots, \phi_m\}$ be a subset of the candidate feature functions described above and let d_i denote the dimensionality of the i-th function of this set. A cysteine pair classifier

Symbol	Parameter	d	Description
$\|P\|$	-	1	Number of residues
n_C	-	1	Number of cysteines
$n_C \mod 2$	-	1	Parity of the number of cysteines
$pos(C_i)/\|P\|,\ pos(C_j)/\|P\|$	-	2	Relative position of cysteines
$(pos(C_j) - pos(C_i))/\|P\|$	-	1	Position difference
$i/n_C,\ j/n_C$	-	2	Indices of cysteines
$(i-j)/n_C$	-	1	Index difference
$csp(W)$	window size	$2(W-1)$	Cysteine separation profile window
$h^{global}(\mathcal{A})$	-	$L_{\mathcal{A}}+1$	Labels global histogram
$h^{interval}(\mathcal{A})$	-	$L_{\mathcal{A}}$	Labels interval histogram
$h^{local}(\mathcal{A}, W)$	window size	$2L_{\mathcal{A}}+2$	Labels local histogram
$w(\mathcal{A}, W)$	window size	$2WL_{\mathcal{A}}$	Labels local window

TABLE 5.3: **Feature functions used in our experiments to encode cysteine pairs.** Symbols, parameters, number of features (d) and description of our candidate feature functions. Top : feature functions that are directly computed from the primary structure. Bottom : feature functions defined for every kind of annotation $\mathcal{A} \in \{AA,\ PSSM,\ SS3,\ SS8,\ SA,\ DR,\ StAl\}$.

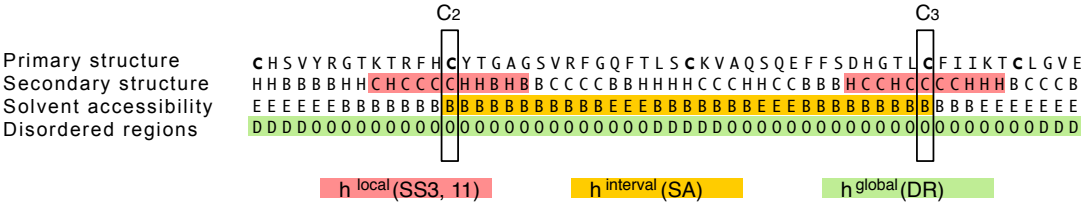

FIGURE 5.3: **Example of local, interval and global histograms.** C_2 and C_3 are the two cysteines of interest. In red, we show the labels local histograms of size 11 of the secondary structure $h^{local}(SS3, 11)$. In yellow, we show the labels interval histogram of the solvent accessibility annotation $h^{interval}(SA)$. In green, we show the global histogram of the disordered regions sequence $h^{global}(DR)$.

processes feature vectors of dimension $\sum_{i=1}^{m} d_i$, in order to predict disulfide bonding probabilities. In this study, we consider three such binary classifiers :

– *K-nearest neighbors* (kNN). In order to determine the disulfide bonding probability of a new example, the algorithm returns the frequency of bonded cysteines among these neighbors. The distance between two feature vectors A and B is computed using a normalized version of the l2-norm, which is defined as follows :

$$dist(A, B) = \sqrt{\sum_{i=1}^{m} \frac{1}{d_i} \sum_{j=1}^{d_i} \left(\frac{A_i^j - B_i^j}{\sigma_i^j} \right)^2},$$

where A_i^j and B_i^j denote the j-th components of the i-th feature generator ϕ_i, and where σ_i^j denotes the empirical standard deviation of this component, computed on

the training data. Since we are concataining feature functions with very different dimensionalities (d varies from 1 to $\mathcal{O}(10^3)$), the effect of the traditional l2-norm would be largely dominated by high-dimensional feature functions. The term $\frac{1}{d_i}$ enables to avoid this problem. Dividing by the standard deviations σ_i^j is a classical strategy to be less dependent on the domain of the different features.

– *Support vector machines* (SVM). Among the common kernel functions used to cope with non-linear feature interactions, we use the Gaussian radial basis function $exp(-\gamma.dist(A,B)^2)$, where $\gamma > 0$ is a bandwidth hyper-parameter and where $dist(\cdot,\cdot)$ is the same norm as previously. Note that previous studies on disulfide pattern prediction [78, 80] also relied on the Gaussian radial basis function. In our experiments, we used the well-known LibSVM implementation [27]. In order to convert SVM predictions into probabilities, we use the default probability estimation method of LibSVM, which was proposed by Platt [99] and Wu *et al.* [139].
– *Extremely randomized trees* (ETs). We use the probabilistic version of ETs, in which each leaf is associated to a bonding probability, which is the empirical proportion of bonded cysteine pairs among the training samples associated to that leaf. To our best knowledge, tree-based ensemble methods, and in particular ETs, were not yet applied to disulfide connectivity pattern prediction, despite the fact that several studies have shown that these methods very often outperform other methods such as support vector machines or neural network [24].

Maximum weight graph matching

Given bonding probabilities for every cysteine pair of a protein, the aim of this last step of the disulfide pattern prediction pipeline is to select a subset of disulfide bridges so as to respect the constraint $degree(C_i(P)) \leq 1, \forall i \in [1, n_C]$. As proposed previously, this problem is formalized as a maximum weight graph matching problem : the weight of a disulfide pattern is defined as the sum of probabilities attached to its edges and the aim is to find the valid pattern with maximal weight.

A naive solution to solve the maximum weight graph matching problem is to perform an exhaustive search over all valid disulfide patterns. The complexity of this procedure is however exponential in the number of cysteines, which is problematic for large proteins. This issue is often solved using the maximum weight matching algorithm of Gabow [56] whose time complexity is cubic w.r.t. the number of cysteines n_C and whose space complexity is linear w.r.t. n_C. In our experiments, we used Blossom V, which is a more recent and optimized implementation proposed by Kolmogorov [72].

Notice that, because this algorithm searches for a full matching, *i.e.*, where each cysteine is associated to another one, it cannot be directly applied on proteins that have an odd number n_C of cysteines. To deal with such proteins, we run the matching algorithm on each one of the n_C subsets of $n_C - 1$ cysteines and select the solution with maximal weight.

5.3.3 Forward feature function selection

This section describes our forward feature function selection algorithm, which aims at determining a subset of relevant feature functions among those described above. Fea-

ture selection is an old topic in machine learning and a common tool in bioinformatics [111]. Our feature selection problem departs from traditional feature selection w.r.t. three unique aspects :

- *Feature function selection* : we want to select feature functions rather than individual features. Given that feature functions can be parameterized by window sizes, our algorithm has to perform two tasks simultaneously : determining a subset of feature functions and determining the best setting for associated window sizes.
- *Insertion in a pipeline* : we want to optimize the performance Q_p of the whole pipeline rather than the accuracy Q_2 of the classifier for which we perform feature selection. Preliminary studies have shown that these two performance measures are not perfectly correlated : a binary classifier with higher accuracy can lead to worse disulfide pattern predictions when combined with the graph matching algorithm, and conversely.
- *Interpretability* : our approach not only aims at constructing a pipeline maximizing Q_p, but also at drawing more general scientific conclusions on the relevance of various annotations of the primary structure. We thus require the result of the feature selection process to be interpretable.

In order to fulfill these requirements, we adopt a *wrapper* approach that repeatedly evaluates feature function subsets by cross-validating the whole pipeline and that is directly driven by the cross-validated Q_p scores. In order to obtain interpretable results, we rely on a rather simple scheme, which consists in constructing the feature function set greedily in a forward way : starting from an empty set and adding one element to this set at each iteration.

In order to treat feature functions with parameters and those without parameters in an unified way, we express the feature functions as a set of parameterized feature functions $\mathbf{\Phi} = \{\Phi^{(1)}, \ldots, \Phi^{(M)}\}$ where each $\Phi^{(i)}$ contains a set of alternative feature functions $\Phi^{(i)} = \{\phi_1^{(i)}, \ldots, \phi_{a_i}^{(i)}\}$. In the case where the feature function has no parameters (*e.g.*, number of residues or labels global histogram), this set is a singleton $\Phi = \{\phi\}$. Otherwise, when the feature function is parameterized by a window size, there is one alternative per possible window size, *e.g.*, $\Phi_W^{csp} = \{csp(1), csp(3), \ldots, csp(19)\}$.

Our forward feature function selection approach is depicted in Algorithm 3. We denote by $S(\cdot, \cdot, \cdot) \in \mathbb{R}$ the objective function that evaluates the Q_p score associated to a given set of feature functions, based on a cysteine pair classifier $\mathcal{C}$ and a dataset of proteins D. In our experiments, this objective function is computed by performing a 10-fold cross-validation of the whole prediction pipeline and by returning the test Q_p scores averaged over the ten folds.

The feature function is first initialized to an empty set $\emptyset$ (line 1). Each iteration then consists in inserting a candidate feature functions $\phi_j^{(i)}$ taken in the set $\mathbf{\Phi}$ into Υ. For this, we try to add each candidate $\phi_j^{(i)}$ to the current feature function set and select the best feature function w.r.t. the obtained cross-validation Q_p scores (line 3). This feature function is then inserted into Υ (line 4) and the corresponding set of alternatives $\Phi^{(i^*)}$ is removed from $\mathbf{\Phi}$. After a given stopping criterion is fulfilled, the constructed function set Υ is returned (line 7). In our experiments, this stopping criterion is simply a fixed number of iterations. An alternative consists in stopping the algorithm when no additional feature functions enable to improve the S score.

Algorithm 3 Forward feature function selection algorithm.

Given a set of parameterized feature functions $\mathbf{\Phi} = \{\Phi^{(1)}, \ldots, \Phi^{(M)}\}$
Given an objective function $S(\cdot, \cdot, \cdot) \in \mathbb{R}$
Given a cysteine pair classifier $\mathcal{C}$
Given a dataset D

$\Upsilon \leftarrow \emptyset$ ▷ initial empty feature function set
repeat
  $(i^*, j^*) \leftarrow \underset{i,j}{\operatorname{argmax}}\ S(\Upsilon \cup \{\phi_j^{(i)}\}, \mathcal{C}, D)$ ▷ evaluate candidate $\phi_j^{(i)}$ functions
  $\Upsilon \leftarrow \Upsilon \cup \{\phi_{j^*}^{(i^*)}\}$ ▷ add the best feature function
  $\mathbf{\Phi} \leftarrow \mathbf{\Phi} \setminus \phi^{(i^*)}$ ▷ remove the best parameterized feature function
until some stopping criterion is fulfilled
return Υ ▷ return feature function set

Note that due to its greedy nature, our feature selection may fall into local minima. However, compared to traditional feature selection, it may be the case that selecting feature functions instead of individual features significantly reduces the importance of this problem (the dimensionality of our search problem is much smaller than in the case of individual feature selection). We show in the next section that this algorithm is a tractable feature function selection approach that provides interpretable results, from which we can draw some general conclusions about the relevance of primary structure annotations.

5.4 Disulfide pattern prediction

This section describes our experimental study on disulfide pattern prediction using the SPX+ benchmark dataset. We first make an overall comparison of the three binary classification algorithms described previously and show that extremely randomized trees lead to significantly better results than the two other algorithms. We then apply our forward feature function selection approach using this algorithm and show that only a few feature functions are sufficient to construct a high performance disulfide pattern predictor. We finally compare this predictor with the state of the art and propose an analysis of the sensitivity of extremely randomized trees w.r.t. their hyper-parameters. Note that, for the moment, our prediction pipeline always tries to construct fully connected disulfide patterns and that it does not enable predicting partially connected disulfide patterns. We address this issue in the next section, by coupling our predictor with filters based on the bonding state of individual cysteines.

5.4.1 Comparison of the cysteine pair classifiers

Comparing cysteine pair classifiers in our context is not trivial for two reasons. First, we are primarily interested in the Q_p score of the whole prediction pipeline rather than in the classification accuracy. Second, we do not have a fixed feature representation and different classification algorithms may require different feature function sets to work

optimally. To circumvent these difficulties, we compare cross-validated Q_p scores obtained with the three classifiers on a large number of randomly sampled feature function sets. To sample a feature function set of size $m \in [1, 18]$, we proceed as follows. First, we draw a subset $\{\Phi^{(1)}, \ldots, \Phi^{(m)}\}$ from $\mathbf{\Phi}$. Then, for each member $\Phi^{(i)}$ of this subset, we select a feature function $\phi_j^{(i)}$, using the following rules : (i) local window sizes are sampled according to the Gaussian distribution $\mathcal{N}(15, 15^2)$, (ii) local histogram sizes are sampled according to $\mathcal{N}(51, 50^2)$ and (iii) CSP window sizes are sampled from $\mathcal{N}(7, 11^2)$. These values were chosen according to preliminary studies using the three classifiers.

We set the hyper-parameters in the following way :

- *kNN.* By studying the effect of k, we found out that large values of k drastically decrease the performance of kNN and low values do not enable to distinguish patterns well since the set of possible predicted probabilities is limited to $k+1$ values. In the following, we use the default value $k = 5$, which we found to be a generally good compromise.
- *SVM.* It turns out that the best setting for γ and C is highly dependent on the chosen feature function set. For each tested set of feature functions, we thus tuned these two parameters by testing all combinations of $\gamma \in \{2^{-14}, 2^{-7}, 2^0, 2^7, 2^{14}\}$ and $C \in \{2^0, 2^5, 2^{10}, 2^{15}\}$ and by selecting the values of (γ, C) that led to the best Q_p scores.
- *ETs.* We use a default setting that corresponds to an ensemble of 1 000 fully developed trees ($T = 1\ 000, N_{\min} = 2$) and K is set to the square root of the total number of features $\sqrt{d}$, as proposed by Geurts *et al* [58].

The results of our comparison on SPX+ are given in Figure 5.4. As a first remark, note the large range in which the Q_p scores lie : from $\simeq 15\%$ to $\simeq 60\%$. This shows that all three classifiers are highly sensitive to the choice of the features used to describe cysteine pairs, which is a major motivation for our work on feature function selection. The experiments are color-encoded w.r.t the size m of their feature function set. This color-encoding enables us to notice that, in general, larger feature function sets lead to better classifiers.

The mean and standard deviations of these results are $34.23\% \pm 7.45\%$ for kNN classifiers, $43.96\% \pm 5.31\%$ for SVM classifiers and $47.85\% \pm 7.17\%$ for ETs classifiers. In 73.25% of the experiments, the best pattern accuracy is given by ETs and in 20.35% of them by SVMs. In the remaining 6.40% experiments, exactly the same number of disulfide patterns were correctly predicted by ETs and SVM. kNN was always outperformed by the other two classifiers. We have used the paired t-test to assess the significance of the out-performance of ETs. The p-value against kNN is $\mathcal{O}(10^{-128})$ and the p-value against SVM is $\mathcal{O}(10^{-38})$, which make it clear that ETs significantly outperform kNN and SVM. Moreover, ETs work well with a default setting contrarily to SVM that required advanced, highly time-consuming, hyper-parameters tuning.

Given those observations, we proceed in the remainder of this study by restricting to the ETs method.

5.4.2 Feature functions selection

We now apply our feature function selection approach on top of extremely randomized trees. We rely on the set of parameterized feature functions $\mathbf{\Phi}$ described in Table 5.3 and

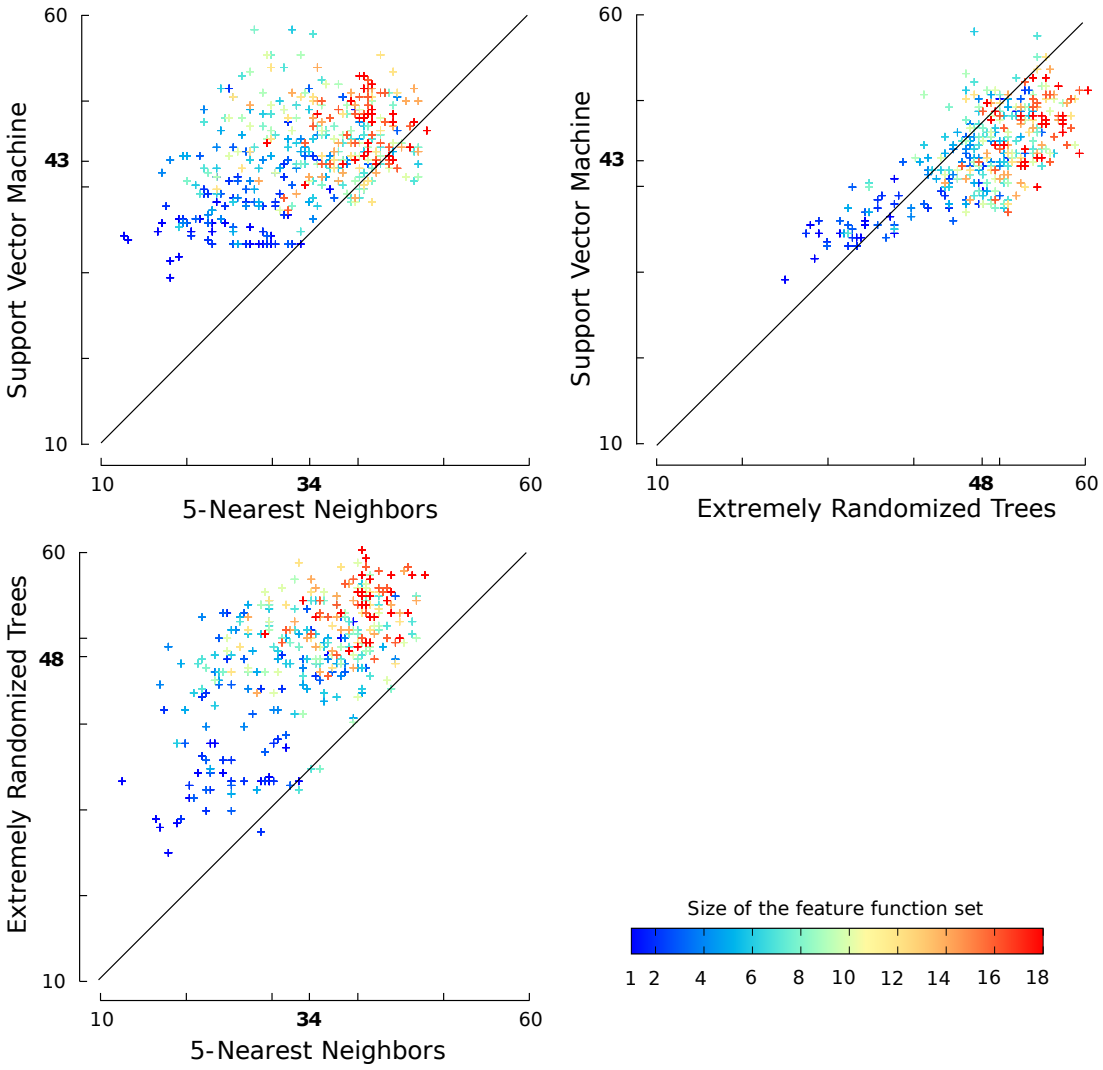

FIGURE 5.4: **$\mathbf{Q_p}$ scores for three binary classification algorithms and randomly sampled feature function sets.** The experiments are performed on the SPX+ dataset. In bold on the horizontal and vertical axes, the means of the classifiers. The diagonal lines indicate situations where both classifiers have the same score. The experiments are color-encoded w.r.t. the size of their feature function set.

consider the following window size values :

- *Cysteine separation profile window* : 1, 3, 5, 7, 9, 11, 13, 15, 17, 19.
- *Local histograms* : 10, 20, 30, 40, 50, 60, 70, 80, 90.
- *Local windows* : 1, 5, 9, 11, 15, 19, 21, 25.

This setting leads to a total of 150 candidate features functions. As cysteine pair classifier, we use ETs with the same default setting as previously ($T = 1\,000$, $K = \sqrt{d}$, $N_{min} = 2$).

The simplest way to apply our algorithm would be to apply it once on the whole SPX+ dataset. By proceeding in this way, the same data would be used for both selecting the set of feature functions and assessing the quality of this selected set. It has been shown

that this approach is biased due to using the same data for selecting and for evaluating and that it could lead to highly over-estimated performance scores [6].

To avoid this risk of overfitting, we adopted a more evolved approach, which consists in running the feature selection algorithm once for each of our 10 different train/test splits. In this setting, the whole feature selection algorithm is executed on a training dataset composed of 90% of the data and the generalization performance of the selected feature functions is evaluated using the remainder 10% of data. There are thus two different objective functions. We call *cross-validated score* the value returned by $S(\cdot,\cdot,\cdot)$, *i.e.*, the 10 cross-validated Q_p score using 90% of the data, and we call *verification score* the Q_p score computed over the remainder 10% of the data.

Figure 5.5 shows the evolution of the cross-validated score and the verification score for five iterations of the feature selection algorithm on each of the 10 train/test splits. Note that, since the cross-validated score is the score being optimized, its value increases at every iteration of each of the 10 runs. The evolution of the verification score, which represents the true generalization performance, is far from being so clear, as, in most cases, the optimum is not located after the fifth iteration.

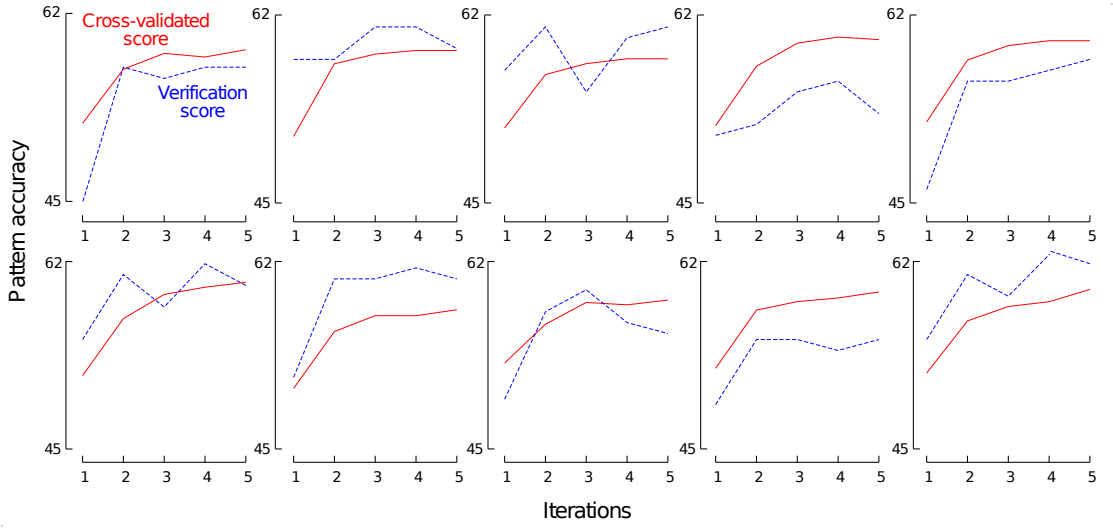

FIGURE 5.5: **Forward feature function selection with 10 train/test splits of the SPX+ dataset.** Each figure reports the results of five iterations of our forward feature function selection algorithm on one of ten train/test splits. Solid red lines are the cross-validated scores and dashed blue lines are the verification scores.

Table 5.4 reports the selected feature functions for each of the 10 runs. We observe that the first selected feature function is always $w(PSSM,\cdot)$ with a window size varying in $\{9, 11, 15, 19\}$. This means that, taken alone, the best individual feature function is always a window over the position-specific scoring matrix. The fact that this results was observed during each run is very strong, since the selection algorithm has to select between 150 different functions. Similarly, the second selected feature function is always $csp(\cdot)$ with a window size varying in $\{9, 13, 17, 19\}$.

After the two first iterations, the selected feature functions become more disparate and only lead to tiny improvements. This probably indicates that the system starts to

Fold	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
1	$\mathbf{w(PSSM, 15)}$	$\mathbf{csp(17)}$	$h^{inteval}(AA)$	$\mathbf{h^{local}(SS8, 90)}$	$w(SS3, 25)$
2	$\mathbf{w(PSSM, 15)}$	$\mathbf{csp(19)}$	$\mathbf{h^{local}(SS8, 80)}$	$w(AA, 1)$	$h^{local}(SA, 80)$
3	$\mathbf{w(PSSM, 19)}$	$\mathbf{csp(19)}$	$h^{global}(SA)$	$\mathbf{h^{local}(SS8, 60)}$	$h^{local}(AA, 30)$
4	$\mathbf{w(PSSM, 11)}$	$\mathbf{csp(9)}$	$h^{local}(SA, 90)$	$h^{local}(DR, 60)$	$pos(C_k)$
5	$\mathbf{w(PSSM, 9)}$	$\mathbf{csp(13)}$	$h^{local}(DR, 90)$	$w(SA, 19)$	$h^{local}(PSSM, 20)$
6	$\mathbf{w(PSSM, 9)}$	$\mathbf{csp(19)}$	$\mathbf{h^{local}(SS8, 90)}$	$h^{local}(SS3, 30)$	$w(SA, 21)$
7	$\mathbf{w(PSSM, 11)}$	$\mathbf{csp(13)}$	$\mathbf{h^{local}(SS8, 50)}$	$w(SS3, 5)$	$h^{local}(PSSM, 50)$
8	$\mathbf{w(PSSM, 19)}$	$\mathbf{csp(17)}$	$h^{local}(AA, 80)$	$h^{local}(DR, 50)$	$w(SA, 21)$
9	$\mathbf{w(PSSM, 15)}$	$\mathbf{csp(19)}$	$h^{local}(SS3, 90)$	$\mathbf{h^{local}(SS8, 90)}$	$pos(C_k)$
10	$\mathbf{w(PSSM, 19)}$	$\mathbf{csp(17)}$	$h^{local}(AA, 50)$	$h^{local}(SS3, 40)$	$h^{local}(SS8, 70)$
			Mean		
CV	51.8% ± 0.64%	56.9% ± 0.63%	58.3% ± 0.67%	58.6% ± 0.84%	58.9% ± 0.75%
Score	51.6% ± 4.19%	57.8% ± 2.83%	57.4% ± 2.22%	58.7% ± 2.83%	58.0% ± 2.72%

TABLE 5.4: **Forward feature functions selection with 10 train/test splits of the SPX+ dataset.** In bold, the most frequent feature function (without consideration of the window size parameters) of each iteration. *Mean* : averages over the ten cross-validated scores and the ten verification scores.

overfit, by selecting feature functions that are specifically tailored to a specific subset of the training proteins. In iterations 3–4, we note that $h^{local}(SS8, \cdot)$ occurs slightly more often than the other feature functions (6 times over 20). From the two last rows, which give the averaged cross-validated scores and the averaged verification scores, we observe that while the cross-validated score systematically increases, the verification score becomes unstable after the two first iterations. These observations reinforce the fact that the selected feature functions become more and more specific to training samples. From these results, it is clear that the feature functions $w(PSSM, \cdot)$ and $csp(\cdot)$ bring the major part of the predictive power that can be obtained by our feature functions.

According to these results, we focus in the following on the feature functions $w(PSSM, 15)$, $csp(17)$ and $h^{local}(SS8, 77)$, where we chose windows sizes by taking the average sizes reported in Table 5.4. Note that, contrarily to the observation of Figure 5.4 that suggested large feature function sets, our method carefully selected a very small number of relevant feature functions that led to a more simpler and still very accurate classifier.

5.4.3 Evaluation of the constructed prediction pipeline

We now compare our constructed prediction pipeline with the state of the art. We consider three baselines that were evaluated using the same experimental protocol as ours (10 cross-validated Q_p). The first baseline is the recursive neural network approach proposed by Cheng *et al.*[30]. These authors, who introduced the SPX+ dataset, reached a pattern accuracy of 51% using the true secondary structure and solvent accessibility information. Lin *et al.* [78] proposed to predict the bonding state probabilities using a fine tuned support vector machine. They obtained a pattern accuracy of 54.5% by using the same data for feature selection and for evaluation, making this results probably over-estimated. Vincent *et al.* [87] proposed a simple approach based on a multiclass one-nearest neighbor algorithm that relies on the fact that two proteins tend to have the

same disulfide connectivity pattern if they share a similar cysteine environment. This method reaches a pattern accuracy of 55%.

Table 5.5 reports the performance obtained by ETs with feature functions $w(PSSM, 15)$, $csp(17)$ and $h^{local}(SS8, 77)$. We observe that using only $w(PSSM, 15)$ already leads to a pattern accuracy of 51.6%, which is better than the baseline of Cheng *et al.*[30]. A significant improvement of +6.6% is achieved by adding the feature function $csp(17)$, which leads to a model that significantly outperforms the state of the art. The feature function $h^{local}(SS8, 77)$ leads to small further improvement of the Q_p score, but due to the large variance, this improvement cannot be shown to be significant.

Features	$\mathbf{Q_p}$
$\{w(PSSM, 15)\}$	51.6% ± 3.58%
$\{w(PSSM, 15), csp(17)\}$	58.2% ± 2.74%
$\{w(PSSM, 15), csp(17), h^{local}(SS8, 77)\}$	58.3% ± 3.04%
Baseline	
Vincent *et al.* [87]	55.0%
Lin *et al.* [78]	54.5%
Cheng *et al.* [30]	51.0%

TABLE 5.5: **Evaluation of the proposed prediction pipeline.** We report the mean and standard deviation of the Q_p scores obtained using 10-fold cross-validation on the SPX+ dataset.

From these results, we conclude that only the following two feature functions are sufficient for high-quality disulfide pattern prediction in combination with ETs : local PSSM windows and CSP windows. Note that it might be the case that, by using larger datasets, feature functions such as medium-size histograms on predicted DSSP secondary structure could slightly improve the quality of the system.

Table 5.6 reports the pattern accuracy as a function of the true number of disulfide bridges. By comparing the results with the three baselines, we observe that our method outperforms the baselines, except for proteins with 4 potential disulfide bonds where the approach proposed by Vincent *et al.* [87] obtains a better pattern accuracy.

Number of bridges	**Cheng *et al.* (2006)**	**Vincent *et al.* (2008)**	**Lin *et al.* (2009)**	**Becker *et al.* (2013)**
1	59%	59%	60.6%	61.8%
2	59%	63%	65.9%	66.6%
3	55%	64%	59.8%	67.6%
4	34%	48%	36.4%	41.8%
all	51%	55%	54.5%	58.3%

TABLE 5.6: **Comparison of $\mathbf{Q_p}$ scores on SPX+.** Q_p scores obtained using 10-fold cross-validation.

5.4.4 Sensitivity of ETs to its hyper-parameters

This series of experiments aims at studying the impact of the hyper-parameters (T, K and N_{min}) when using the feature functions $\{w(PSSM, 15), csp(17)\}$. With these two feature functions, the number of features is $d = 662$. The default setting is $T = 1\,000, K = \sqrt{d}, N_{\text{min}} = 2$ and we study the parameters one by one, by varying their values in ranges $T \in [10, 10^4]$, $K \in [1, d]$ and $N_{min} \in [2, 100]$.

Figure 5.6 reports the Q_p and Q_2 scores in function of the three hyper-parameters. As a matter of comparison, we also reported the Q_p scores of the three baseline described previously. We observe that the Q_p score grows (roughly) following a logarithmic law w.r.t. T. The value of $T = 1\,000$ occurs to be very good tradeoff between performance and model complexity. Concerning K, we observe that the value maximizing Q_p is $K \simeq 50$, which is a bit larger than the default setting $K = \sqrt{d}$. Note that the protein-level performance measure Q_p and the cysteine-pair level performance measure Q_2 do not correlate well in terms of the effect of parameter K, which confirms the interest of directly optimizing Q_p in our feature function selection algorithm. N_{min} controls the complexity of built trees and, hence, the bias-variance tradeoff by averaging output noise. It is usually expected that a small value of N_{min} improves performance. In our case, we observe that increasing N_{min} never improves the performance measures and that Q_p has a large variance.

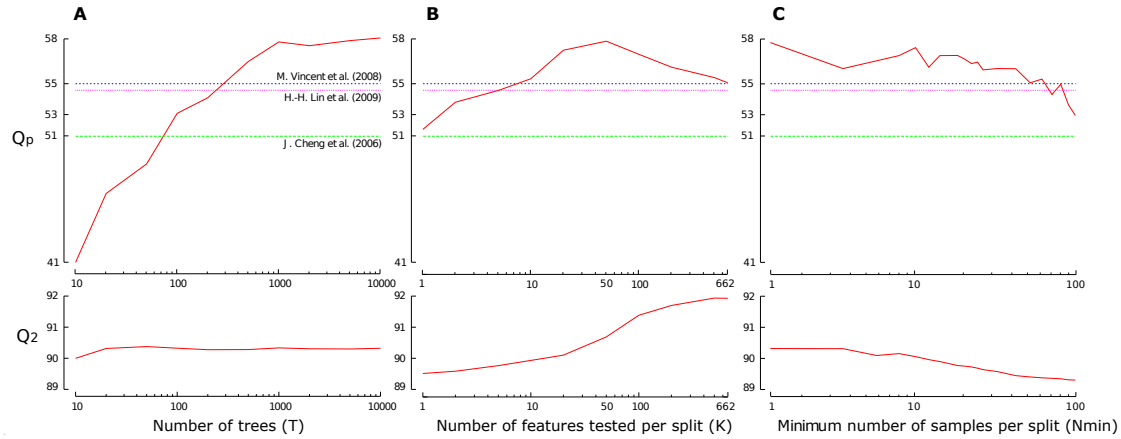

FIGURE 5.6: **Sensitivity of ETs w.r.t. hyper-parameters.** The experiments are performed on the SPX+ dataset. We used the two feature functions $w(PSSM, 15)$ and $csp(17)$. (A) Impact of the number of trees T (from 10 to 10 000) with $K = \sqrt{d}$ and $N_{min} = 1$, where $d = 662$ is the number of features. (B) Impact of K (from 1 to d) with $T = 1\,000$ and $N_{min} = 1$. (C) Impact of N_{min} (from 2 to 101).

5.5 Chain classification and cysteine bonding state prediction

Until now, our pipeline relies on a perfect graph matching algorithm that always attempts to predict patterns involving all cysteines. Due to this, our approach is, for the

moment, unable to deal with partially connected disulfide patterns (except for proteins with an odd number of cysteines having a single non-bonded cysteine). This can be harmful, especially on datasets containing many non-bonded cysteines. For example, if we apply our pipeline to the SPX− dataset, the pattern accuracy Q_p is only 22%, since most proteins of this dataset do not contain any disulfide bridges. We now focus on this issue by coupling our predictor with filters based on the output of a chain classifier and on the output of a cysteine bonding state predictor. We first construct a chain classifier and a cysteine bonding state predictor by applying our feature function selection approach. We then study combinations of these predictors with our disulfide pattern predictor.

5.5.1 Chain classification

We consider the *binary* chain classification problem, which consists in classifying proteins into those that have at least one disulfide bridge and those that have no disulfide bridge. In order to construct a chain classifier, we apply the same methodology as before : we perform feature function selection on top of extremely randomized trees. Since chain classification works at the level of proteins, the set of candidate feature functions is restricted to labels global histograms. We also include as candidates the simple feature functions returning the number of residues, the number of cysteines and the parity of the number of cysteines. We use the following default setting for ETs : $T = 1\ 000, K = d$ and $N_{min} = 2$. According to preliminary experiments, we found out $K = d$ to be a good default setting for this task. This is probably due to the fact that we have far less features than we had before.

We performed ten runs of the feature function selection algorithm on the SPX− dataset, which contains both proteins without disulfide bridge and proteins with disulfide bridges. The performance measure is the accuracy, i.e., the percentage of proteins that are correctly classified. In every feature function selection run, the first selected feature function was $h^{global}(PSSM)$ and the second one was $h^{global}(AA)$. Starting from the third iteration, the results are more diverse and the system starts to overfit. By keeping the two first feature functions, we reach a 10 fold cross-validation accuracy of 79.5% on SPX−, which is not very far from the 82% accuracy obtained by [87].

5.5.2 Cysteine bonding state prediction

Cysteine bonding state prediction consists in classifying cysteines into those that are involved in a disulfide bridge and those that are not. To address this task, we apply our feature function selection approach on top of extremely randomized trees ($T = 1\ 000, K = \sqrt{d}$ and $N_{min} = 2$). The set of candidate feature functions is composed of those depending only on the protein (number of residues, number of cysteines, parity of the number of cysteines, labels global histograms) and those depending on the protein and on a single cysteine (labels local histograms, labels local windows, cysteine separation profile window). We consider the same window size values as in previous section. The evaluation measure is binary accuracy, i.e., the percentage of cysteines that are correctly classified.

We ran the feature selection algorithm once for each of the ten different train/test splits of SPX−. We observed that the selected feature functions set $\{w(PSSM, 11),$

$h^{global}(PSSM)$, $n_C\}$ led to a binary accuracy of 87.4%, which outperforms the result of 87% obtained by Vincent *et al.* [87]. On SPX+, we obtain a similar accuracy of 87.8%.

Note that once we have a cysteine bonding state predictor, we can use it to also solve the chain classification task as follows. In order to predict whether a protein contains disulfide bridges or not, we run the cysteine bonding state predictor on each cysteine, and see if at least one cysteine is predicted as being bonded. By applying this strategy to SPX−, we obtain a chain classification accuracy of 81.4%, which is comparable to the score of [87].

Table 5.7 summarizes the feature functions that were selected for the three tasks that we consider in this study.

Task	**Features**
Chain classification	$\{h^{global}(PSSM), h^{global}(AA)\}$
Cysteine bonding state prediction	$\{w(PSSM, 11), h^{global}(PSSM), n_C\}$
Disulfide pattern prediction	$\{w(PSSM, 15), csp(17)\}$

TABLE 5.7: **Selected feature functions of the three tasks.** The feature functions were determined by the application of our selection algorithm on top of extremely randomized trees, using the SPX− dataset for chain classification and cysteine bonding state prediction and the SPX+ dataset for disulfide pattern prediction.

5.5.3 Impact on pattern prediction

Now that we have constructed a chain classifier and a disulfide bonding state predictor, we focus on the question of how to exploit the corresponding predictions in order to improve disulfide pattern prediction. Note that, in some cases, the user may have prior knowledge of either the chain class (whether the proteins contains any disulfide bridges or not) or of the cysteine bonding states (which are the cysteines that participate to disulfide bridges). To take the different possible scenarios into account, we study the following four settings :

- *Chain class known :* in this setting, we assume that the chain classes are known a priori and simply filter out all proteins that are known to not contain any disulfide bridge. For the proteins that contain disulfide bridges, we run our disulfide pattern predictor as in previous section.
- *Chain class predicted :* in this setting, we replace the knowledge of the chain class by a prediction. We therefore rely on the chain classifier derived from the cysteine bonding state predictor, which obtained a chain classification accuracy of 81.4%.
- *Cysteine states known :* we here assume that the bonding states of cysteines is known a priori. We modify the disulfide pattern predictor by setting a probability of zero to any cysteine pair containing at least one non-bonded cysteine.
- *Cysteine states predicted :* in this setting, we first run our cysteine state predictor and then perform disulfide pattern prediction by only considering cysteine pairs in which both cysteines where predicted as bonded.

Note that, since the SPX+ dataset is entirely composed of proteins with at least one bridge, our two first settings based on chain classification are irrelevant for this

dataset. In these experiments, we learnt models using a 10-fold cross-validation of ETs ($T = 1\ 000$,$N_{min} = 2$ and $\sqrt{d}$).

Table 5.8 summarizes the results of our experiments on chain classification, cysteine bonding state prediction and disulfide pattern prediction with our four different settings. When the chain classes are known, we observe a significant improvement of the Q_p score : from 22% to 82.5% on SPX−. When replacing the true chain classes with predicted chain classes, we still have a relatively high Q_p score : 70.9%. This result is detailed in Table 5.9 as a function of the true number of disulfide bridges. We observe that our method clearly outperforms the method of Vincent *et al.* [87] on proteins containing one or two disulfide bonds and performs slightly worst on proteins with three disulfide bonds. Given that a majority of proteins in SPX− contain less than two bonds, these results leads to an overall score that is significantly better than that of Vincent *et al.* When the cysteine bonding states are known, we obtain impressive disulfide pattern accuracies : more than 75% on SPX+ and almost 90% on SPX−. When using predicted cysteine bonding states, we still observe an impressive improvement on SPX− : from 22% to 71.4%. However, on SPX+, the score slightly degrades (−1.4%). This is probably related to the fact that, as soon as one cysteine is falsely predicted as being non-bonded, it becomes impossible to recover the correct disulfide pattern.

Filter	**SPX−**	**SPX+**
Chain classification		
–	79.5% ± 2.40%	–
Cysteine states predicted	81.4% ± 2.66%	–
Cysteine bonding state prediction		
–	87.4% ± 1.14%	87.8% ± 2.20%
Disulfide pattern prediction		**(Q_p)**
–	22.0% ± 2.00%	58.2% ± 2.74%
Chain class known	82.5% ± 2.24%	–
Chain class predicted	70.9% ± 3.10%	–
Cysteine states known	89.9% ± 1.57%	75.8% ± 2.09%
Cysteine states predicted	71.4% ± 2.76%	56.8% ± 2.52%

TABLE 5.8: **Evaluation of the three tasks.** We report the mean and standard deviation of the binary accuracy for chain classification and cysteine bonding state prediction while the Q_p score is used for disulfide pattern prediction. The symbol – indicates that all cysteines are used in the experiment.

5.6 Discussion

Disulfide connectivity pattern prediction is a problem of major importance in bioinformatics. Recent state of the art disulfide pattern predictors rely on a three step pipeline, in which the central component is a binary classifier that predicts bridge bonding probabilities given cysteine pair representations. However, the comparison of the conclusions of these works is difficult because it is often the case that these different studies rely on different kinds of binary classifiers and slightly differ in their experimental protocol.

Number of bridges	**Vincent *et al.* (2008)**	**Becker *et al.* (2013)**
1	30%	77.1%
2	49%	63.5%
3	61%	60.4%
4	37%	44.2%
all	39%	70.9%

TABLE 5.9: **Comparison of Q_p scores on SPX− when chain classes are predicted.** Q_p scores obtained using 10-fold cross-validation.

Therefore, the relevance of some features is still a subject under heavy debate. This work has proposed an extensive study on the best way to represent cysteine pairs in the form of features. We considered three classification algorithms : k-nearest neighbors, support vector machines and extremely randomized trees, and we proposed a forward feature function selection algorithm that we applied on the standard benchmark dataset SPX+.

Our experiments have shown that extremely randomized trees (ETs) are highly promising in terms of predicted disulfide pattern accuracy Q_p. ETs are easy to tune and thanks to their use of decision trees, they benefit from good scaling properties, making them applicable to large sets of training proteins and large sets of features. The result of our feature selection experiments with ETs is that the primary structure related features functions $w(PSSM, 15)$ (a local window of size 15 on the evolutionary information) and $csp(17)$ (a window of size 17 on the cysteine separation profile) are sufficient to reach a very high performing disulfide pattern predictor : ETs with these two kinds of features predict correct disulfide connectivity patterns in 58.2% of proteins, which outperforms the state of the art[87] with a +3.2% improvement. Furthermore, we showed that appending any other feature function does not lead to significant subsequent improvements or even decreases the accuracy.

We also investigated the question of how to exploit our disulfide pattern predictor with filters based on the output of either a chain classifier or of a cysteine bonding state predictor. Among the four scenarios that we considered, we observed an important potential for improvement when the cysteine bonding states are known, with scores reaching 75% on SPX+ and almost 90% on SPX−. When using predicted cysteine bonding states, we still observe an impressive improvement on SPX− (from 22% to 71.4%) but the score slightly degrades (−1.4%) on SPX+. This degradation is probably due to the fact that, as soon as one cysteine is falsely predicted as being non-bonded, it becomes impossible to construct the correct disulfide pattern. Therefore, one direction of future research is to develop more sophisticated methods to couple the cysteine bonding state prediction task with the pattern prediction task. One direction for such a better coupling is to apply the ideas developed in [84] on multi-stage and multi-task prediction, *e.g.*, by iteratively re-estimating the disulfide bond probabilities.

Note that despite the fact that several studies have shown that tree-based ensemble methods often reach state of the art results in supervised learning (see e.g. [24]), these methods were surprisingly few applied to structural bioinformatics problems yet. We believe that ETs in combination with feature function selection provide a general metho-

dology that can be applied to a wide range of protein related prediction problems and more generally to any kind of classification problems involving many different possible representations.

CHAPITRE 6

Feature selection for disordered regions prediction

Contents

In the previous chapter, we proposed a systematic methodology to study the relevance of various feature encodings in the context of disulfide connectivity pattern prediction. As a result, we have shown that a very limited number of carefully selected feature functions are sufficient to construct a high performance disulfide pattern predictor.

In this chapter, we adapt this methodology to the problem of predicting disordered regions and assess our models on proteins from the 10th CASP competition, as well as on a very large subset of proteins extracted from PDB. Section 6.1 introduces our work as the adaptation of the pipeline presented in Chapter 5. Section 6.2 formulates the problem of predicting disordered regions as a sequence labeling task. It then describes the four required components needed by the feature function selection algorithm, *i.e.*, the datasets (Section 6.2.1), the list of candidate feature functions (Section 6.2.3), the base learner and the criterion to optimize. In Section 6.3, we perform the feature selection process on the Disorder723 dataset and we analyze the performance obtained by ensembles of ETs using the selected feature functions on Casp10 and Pdb30 datasets. Finally, Section 6.4 concludes the chapter.

Publication related to this chapter

Becker (Julien), Maes (Francis), Wehenkel (Louis)
On the encoding of proteins for disordered regions prediction
Accepted for publication in PLoS One

Software related to this chapter

From this study, we made available a web-application at

`http://m24.giga.ulg.ac.be:81/x3Disorder`.

x3Disorder is a tool designed for biologists that attempt to determine the three-dimensional structure of protein molecules. The function of *x3Disorder* is to predict the disorder state of each residue of a protein.

6.1 Introduction

In the previous chapter, we motivated our work by the fact that the way to encode cysteine-pairs into vectors of features typically has a major impact on the performance of predictors. In the context of disordered regions prediction, the situation is similar. Namely, one key factor of the success of methods developed to automatically predict disordered regions of proteins is the way in which protein information is encoded into features.

The main contribution of the present chapter is the adaptation of the systematic feature function selection methodology presented in Chapter 5 to establish a relevant representation of residues in the context of disordered regions prediction. For this purpose, we consider various feature encodings and, in addition to the primary structure, three in-sillico annotations : position-specific scoring matrices (PSSM), predicted secondary structures and predicted solvent accessibilities. We apply the feature function selection pipeline in combination with ETs, a model which gave excellent results in the previous chapter. In order to avoid any risk of overfitting or over-estimation of our models, we use three distinct datasets : Disorder723 [31], Casp10 [1] and Pdb30. We first apply feature selection on Disorder723 and then assess the relevance of the selected feature functions on both Casp10 and on Pdb30.

The main result of our study is to highlight a novel feature function encoding the proximity along the primary sequence of residues predicted as being accessible (resp. inaccessible) to the solvent. This feature function is identified as the second most important for predicting the belonging of a residue to a disordered region, just after evolutionary information derived from the PSSM. To our best knowledge, these features encoding solvent accessibility have never been highlighted in previous studies of disordered regions prediction. The majority of the remaining relevant feature functions that we found (*e.g.*, evolutionary information and sequence complexity) were already suggested

1. `http://www.predictioncenter.org/casp10/`

by other studies of disordered regions [138], and we thus confirm in a fair way their relevance. Furthermore, even though our approach treats each residue independently, *i.e.*, without explicitly modelling global properties of disordered regions, our predictors are very competitive in terms of accuracy with respect to CASP10 assessments of competing methods, as well as on our very large independent test set extracted from PDB30.

6.2 Materials and Methods

In order to use the forward feature function selection algorithm developed in Chapter 5, the algorithm requires four components : a dataset, a list of candidate feature functions, a criterion to optimize and a base learner. This section describes how we have adapted each of the four components to the problem of predicting disordered regions of proteins.

The first part presents the three datasets DISORDER723, CASP10 and PDB30 and how we enrich the primary structures of each of these datasets with three annotations : position-specific scoring matrices ($PSSM$), predicted secondary structures (SS) and predicted solvent accessibilities (SA). The second part of this section formulates disordered regions prediction as a supervised-learning problem and, more specifically, as a binary classification problem, which aims at predicting the disorder state (`ordered` or `disordered`) of each protein residue. It also defines five measures to assess the quality of the predictions, of which the criterion to optimize during the selection process. The third part enumerates the candidate feature functions that we consider for our experiments. The fourth component (the base learner) is an ensembles of ETs. We refer the reader to Section 2.4 for a description of this tree-based ensemble method.

6.2.1 Datasets and annotations

This study relies on three datasets. The first one, DISORDER723 [2], has been built by Cheng *et al.* [31] and was extracted from the Protein Data Bank [12] in May 2004. The dataset is made of 723 non-redundant chains that contain at least 30 amino acids in length and that were solved by X-ray diffraction with a resolution of around 2.5 Å. In order to reduce the over-representation of particular protein families, the dataset has been filtered by UniqueProt [123], a protein redundancy reduction tool based on the HSSP distance [113], with a cut-off distance of 10.

The second dataset, CASP10, was the one used during the 10th CASP competition that took place in 2012. During the competition, the candidate predictors had to make blind predictions, *i.e*, they had to predict disordered regions of proteins close to being solved or close to being published and that have no detectable similarity to available structures. At the end, the candidate predictors were assessed on 94 experimentally determined proteins available for download on the official CASP website [3]. Note that unlike DISORDER723, the way to resolve protein structures is not restricted to X-ray diffraction and that CASP10 also contains protein structures determined by NMR.

2. `http://casp.rnet.missouri.edu/download/disorder.dataset`
3. `http://predictioncenter.org/download_area/CASP10/targets/casp10.DR_targets.tgz`

The last dataset, that we denote by PDB30, is far larger than the two previous ones. We created PDB30 on one of the clustered versions of the Protein Data Bank (as of August 31, 2013) [4]. The clustering is defined on a protein chain basis with a maximum pairwise sequence identity of 30%. The authors of this clustered version of PDB used BLASTClust [40] to perform the clustering and selected the representative structure of each cluster according to their quality factor. We then filtered out any proteins that were less than 30 amino acids in length, that had no X-ray structure or that had resolution coarser than 2.5Å. Next, we discarded the proteins that share a sequence identity of at least 30% with a protein of DISORDER723 (our training set). The final dataset is made of 12,090 proteins and 2,991,008 residues of which 193,874 (6.5%) are disordered. Figure 6.1 shows a histogram of the protein lengths. The average (± standard deviation) protein length is 247.4 ± 162.8. Figure 6.2 shows a histogram of the disordered region lengths of our dataset. The average disordered region length is 12.3 ± 15.6. Our dataset is available at : `http://m24.giga.ulg.ac.be:81/x3Disorder/pdb30.dataset`.

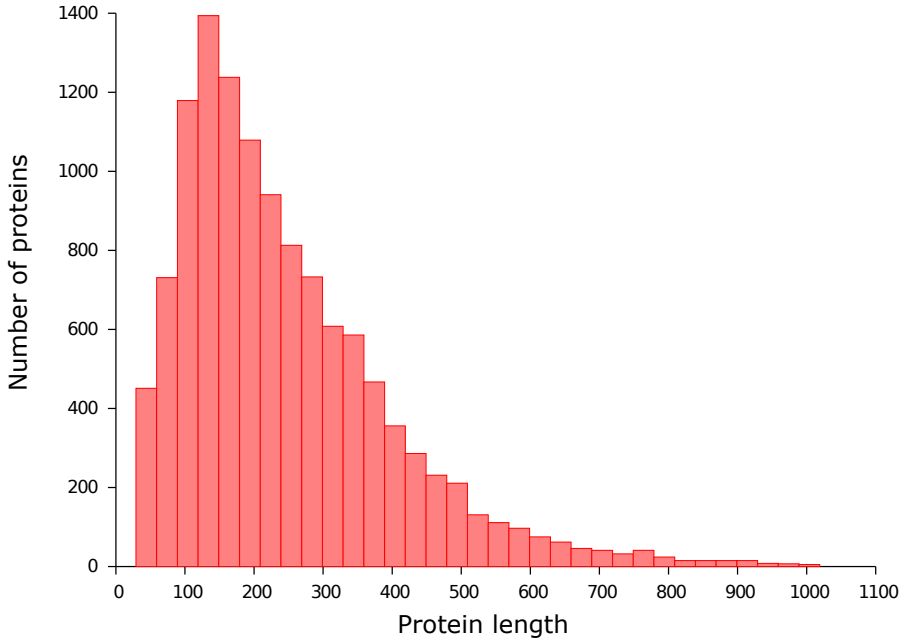

FIGURE 6.1: **Protein length distribution of PDB30.** There are 12,090 proteins. The average protein length is of 247.4 residues.

In our experiments, we mainly use DISORDER723 to identify a subset of relevant feature functions while CASP10 and PDB30 are used to assess the quality of the selected feature functions. It is important to note that no protein in the CASP10 or PDB30 sets share more than 30% sequence identity with one of those of DISORDER723. This therefore makes it possible to fairly evaluate and compare our results with those that have participated to the 10th CASP competition.

We used the same definition of disorder as Cheng *et al.* and as the CASP competition, *i.e.*, segments longer than three residues but lacking atomic coordinates in the crystal structure are labelled as "`disordered`" whereas all other residues are labelled as

4. available at `http://www.rcsb.org/pdb/statistics/clusterStatistics.do`

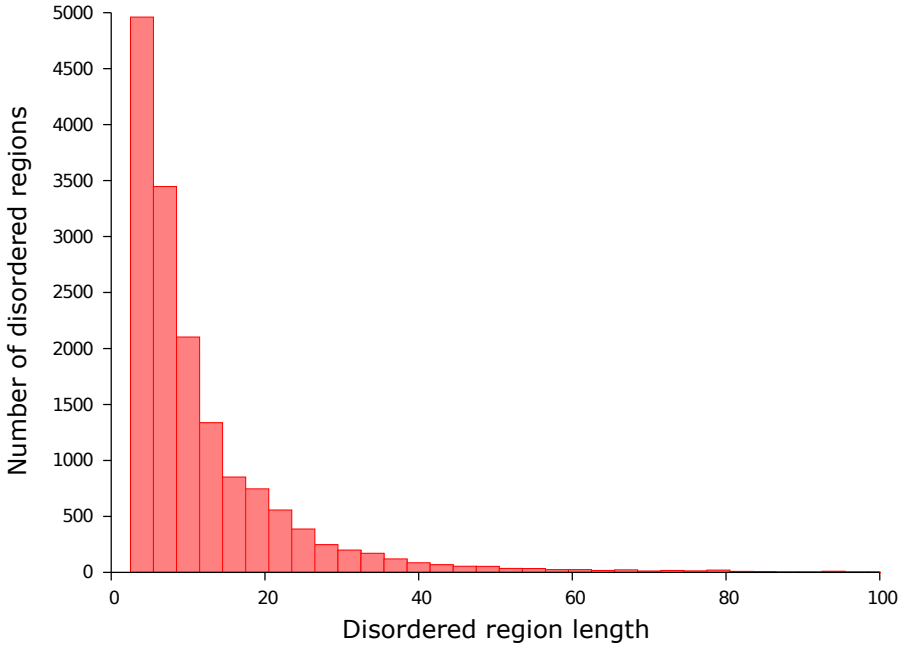

FIGURE 6.2: **Disordered region length distribution of PDB30.** There are 15,726 disordered regions. The average length of a disordered region is of 12.3 residues and the average number of disordered regions per protein is of 1.3.

"`ordered`". According to this definition, Table 6.1 shows that all three datasets contain $\sim$ 6% of disordered residues and $\sim$ 94% of ordered residues. Some residues in CASP10 were not classified by the CASP assessors. These residues were not taken into account in our experiments.

	Proteins	**Ordered residues**	**Disordered residues**	**Residues**
DISORDER723	723	201,703 (93.55%)	13,909 (6.45%)	215,612
CASP10	94	22,688 (93.79%)	1502 (6.20%)	24,190
PDB30	12,090	2,797,134 (93.52%)	193,874 (6.48%)	2,991,008

TABLE 6.1: **Composition of datasets.** Number of proteins, number (and portion) of ordered/disordered residues and number of residues in DISORDER723, CASP10 and PDB30 datasets. All datasets have roughly the same proportion of disordered residues ($\sim$6%). PDB30 contains $\sim$ 127 times more proteins and $\sim$ 124 times more residues than CASP10.

We enriched the primary structure (denoted as AA) by using three additional annotations : evolutionary information in the form of a position-specific scoring matrix ($PSSM$), predicted secondary structure (SS) and predicted solvent accessibility (SA). We computed the PSSMs by running three iterations of the PSI-BLAST program [5] on the non-redundant NCBI database [102]. To produce predicted annotations, we used the SSpro and ACCpro [29] programs for the predicted secondary structure ("`helix`", "`strand`" or "`coil`") and the predicted solvent accessibility (under or over 25% exposed), respectively.

6.2.2 Problem statement

Let $\mathcal{P}$ be the space of all proteins and $P = (AA, PSSM, SS, SA, Y) \in \mathcal{P}$ one particular protein described as the 5-tuple containing its primary structure AA, its $PSSM$, its two predicted annotations SS and SA, and its disordered regions Y. Each of these annotations is described as a sequence of n labels, where n is the number of residues composing P. For example, the primary structure is defined as $AA = (AA_1, AA_2, \ldots, AA_n)$, where AA_i is the label corresponding to the amino acid of the i-th residue of P, and the disordered regions annotation is defined as $Y = (y_1, y_2, \ldots, y_n)$, where $y_i \in \{$`ordered`, `disordered`$\}$. The disordered regions prediction task consists in assigning a label y_i to each residue of P.

In the supervised-learning formulation of the problem, we assume to have access to a dataset of proteins in which residues are labeled either `ordered` or `disordered`. We denote this dataset $D = \{P^{(i)}\}_{i \in [1,N]}$, where $P^{(i)} \in \mathcal{P}$ is the i-th protein. Given such a dataset D, the aim is to learn a disordered regions predictor $f : \mathcal{P} \setminus \mathcal{Y} \rightarrow \mathcal{Y}$ that maps a protein $P \in \mathcal{P}$ to a sequence $\hat{Y}$ of n predicted labels $\hat{y_i} \in \{$`ordered`, `disordered`$\}$, where n is the length of P.

It is important to note that disordered regions are segments, *i.e.*, consecutive residues tend to share the same label. More and more machine learning approaches such as conditional random fields [133], recursive neural networks [31], meta-predictors [140] or post-filtering steps [73] are able to exploit the structured aspect of the problem.

However, as the goal of this study is to determine a set of relevant feature functions in general, we do not focus on such advanced prediction approaches here. We instead simplify the general problem into a standard binary classification problem. The aim is to learn a predictor $f : (\mathcal{P} \setminus \mathcal{Y}) \times \mathbb{N} \rightarrow \{$`ordered`, `disordered`$\}$ that maps the i-th residue of a protein P to the predicted label y_i. This formulation is rather simple in the sense that it treats each residue independently, *i.e.*, regardless with respect to predictions made on neighboring residues of the same protein.

Evaluation measures

In order to evaluate the quality of the predictions made by our models, we considered five residue-level performance measures : the balanced accuracy (Acc), the sensitivity, the specificity, the area under the ROC curve (AUC) and the F-measure. Each of these measures can be formulated using a tuple of four values : the number of *true positives* (TP), *false positives* (FP), *true negatives* (TN), and *false negatives* (FN), where a positive example is a disordered residue and a negative example is an ordered residue. Therefore, a true positive is a correctly predicted disordered residue and a false negative is an ordered residue falsely predicted as a disordered one.

According to these notations, the sensitivity $[TP \div (TP + FN)]$ is the fraction of disordered residues that are successfully predicted as disordered, whereas the specificity $[TN \div (TN + FP)]$ is the fraction of ordered residues that are successfully predicted as ordered. As the problem of disordered regions prediction is strongly imbalanced (only $\sim$6% of residues are disordered), using the conventional accuracy may inflate performance estimates and is therefore not appropriate. However, the balanced accuracy, defined as

the arithmetic mean of sensitivity and specificity, is robust against imbalanced datasets as well as the F-measure, which is used in recent CASP assessments. The F-measure is defined as the harmonic mean of the precision – the fraction of predicted disordered residues that are truly disordered – and the sensitivity (also called recall).

Since, a large number of available binary classifiers produce probabilities rather than strict classes, these criteria rely on a user-defined *decision threshold* to discriminate positive from negative examples. Depending on how users fixed their threshold, a bias might be introduced, which might lead to an unfair comparison between distinct studies. To tackle this issue, one can compare the performance of distinct models by their ROC curve, which is obtained by plotting the sensitivity against the false positive rate $[FP \div (FP + TN)]$ when varying the decision threshold. However, the comparison is not easy, especially when the curves are similar. A common simplification is therefore to calculate the area under the ROC curve (AUC). An area of 1.00 corresponds to a perfect predictor while an area of 0.50 corresponds to a random predictor.

6.2.3 Candidate feature functions

The feature generation is performed through *residue feature functions* $\phi : (\mathcal{P} \setminus \mathcal{Y}) \times \mathbb{N} \rightarrow \mathbb{R}^d$ that, given the residue position i of a protein P, compute a vector of d real-valued features.

Among the panel of candidate functions ϕ already described in our previous work, we adopted i) the *number of residues* function, ii) the *number of cysteines* function, iii) the *labels global histogram* function, iv) the *labels local histogram* function and, v) the *labels local window* function. In addition to them, we defined three other feature functions directly computed from the primary sequence and four annotation-related feature functions. We now describe in detail all these feature functions. However, since only few of these features will effectively be selected, the reader can understand the rest of our study without considering the detailed descriptions of all candidate feature functions.

- *Number of residues* : returns the number of residues in the primary sequence.
- *Number of cysteines* : returns the number of cysteine residues in the primary sequence. This feature is made from the intuition that larger the number of cysteines is, larger the number of disulfide bonds will be, which usually lead to more stable structures.
- *Unnormalized global histogram* : computes twenty features, one per standard amino acid type, which are the numbers of residues of each type in the primary structure.
- *Position of residue* : returns the position i of the residue in the primary structure.
- *Relative position of residue* : computes one feature which is the residue position i divided by the protein length n. Although this feature may seem redundant with the previous one, the encoded information is different. The previous feature aims at encoding the absolute position of the residue with respect to the N-terminus. The intuition behind this feature is that the position of a residue might determine its disordered state (*e.g.*, the first four residues are prone to be disordered). Whereas, the relative position, which varies in $[0, 1]$, suggests a position regardless of the protein length.

We use the following notations to describe the annotation-related feature functions. For each type of annotation $\mathcal{A} \in \{AA, PSSM, SS, SA\}$, $\mathcal{L}_\mathcal{A}$ is the set of labels corresponding to $\mathcal{A}$ and $L_\mathcal{A} = |\mathcal{L}_A|$ is the size of this set. We thus have : $L_{AA} = 20$, $L_{PSSM} = 21$ (the twenty amino acids and the gap), $L_{SS} = 3$, $L_{SA} = 2$. For a given primary structure of length n, an annotation $\mathcal{A}$ is represented as a set of probabilities $\alpha^\mathcal{A}_{i,l} \in [0,1]$ where $i \in [1,n]$ denotes the residue position and $l \in \mathcal{L}_\mathcal{A}$ is a label. *E.g.*, $\alpha^{SS}_{3,\texttt{helix}}$ is the probability that the third residue of the protein is part of a helix.

In the general case, the $\alpha^\mathcal{A}_{i,l}$ probabilities may take any value in range $[0,1]$ to reflect uncertainty about annotations. However, since the predictions made by SSpro and ACCpro are classes and that primary structures (AA) are always known perfectly, we have :

$$\alpha^{\mathcal{A}\in\{AA,SS,SA\}}_{i,l} = \begin{cases} 1 & \text{if } l \text{ is the residue or predicted class of the } i\text{-th residue} \\ 0 & \text{otherwise.} \end{cases}$$

As PSSM elements typically range in $[-7,7]$, we scale them to $[0,1]$ by using the function proposed in [69] and defined as following :

$$\alpha^{PSSM}_{i,l} = \begin{cases} 0.0 & \text{if } x \leq -5 \\ 0.5 + 0.1x & \text{if } -5 < x < 5 \\ 1.0 & \text{if } x \geq 5 \end{cases},$$

where x is the value from the raw profile matrix.

For each annotation $\mathcal{A}$, we have defined seven different feature functions :

- *Labels global histogram* : computes one feature per label $l \in \mathcal{L}_\mathcal{A}$, equal to $\frac{1}{n}\sum_{p=1}^{n} \alpha^\mathcal{A}_{i,l}$.
- *Labels local histogram* : computes one feature per label $l \in \mathcal{L}_\mathcal{A}$ equal to $\frac{1}{W}\sum_{p=i-W/2}^{i+W/2} \alpha^\mathcal{A}_{p,l}$ and one special feature equal to the percentage of out-of-bounds positions, *i.e.*, positions p such that $p \notin [1,n]$.
- *Labels local window* : computes one feature per label $l \in \mathcal{L}_\mathcal{A}$ and per relative position $\delta \in [-\frac{W}{2}, \frac{W}{2}]$, equal to $\alpha^\mathcal{A}_{i+\delta,l}$. When the position is out-of-bounds, *i.e.*, $i+\delta \notin [1,n]$, the feature is set to 0.
- *Separation profile window* : this feature function is inspired from the *cysteine separation profile window* function, which focuses on the distances that separate consecutive cysteine residues and encodes the distances around the cysteine residue of interest into features. According to the results presented in our previous work, this feature function led to an impressive improvement of our disulfide connectivity pattern predictor. Here, we propose a generalization of this function in order to be able to tackle any kind of annotation $\mathcal{A}$. Figure 6.3 shows an illustration of a separation profile window of size 11 over exposed residues.
  Given a residue position i, our generalized feature function describes the proximity of the $\frac{W}{2}$ closest residues of the N-terminus side to the i-th residue (respectively, the $\frac{W}{2}$ closest residues of the T-terminus side) that share a common label $l, \forall l \in \mathcal{L}_\mathcal{A}$. The proximity of a residue at the j-th position is expressed as the distance, in terms of number of amino acids in the primary structure, that separates the j-th

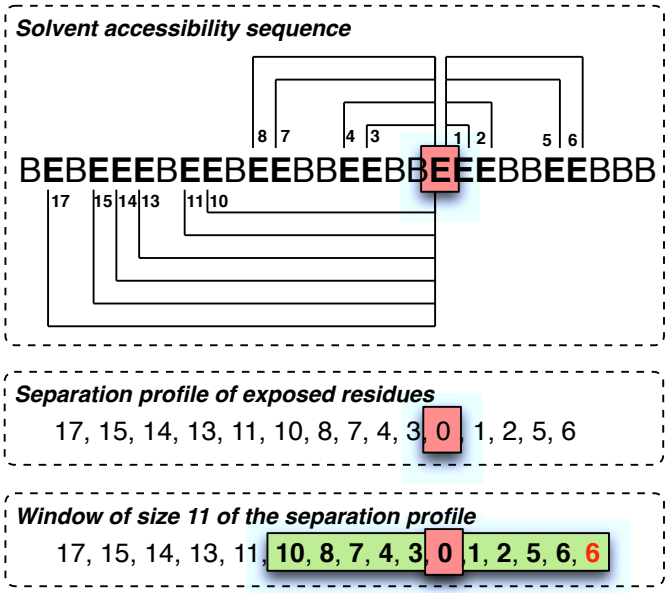

FIGURE 6.3: **Illustration of the *separation profile window* function on exposed residues.** Top : the functions first computes the amino acid distances that separate the residue of interest (highlighted by a red square). Middle : the separation profile of exposed residues. Bottom : the feature function returns the window (highlighted by a green rectangle) of size 11 centered around the residue of interest. In this example, the window slightly goes beyond the end of the sequence. As explained in the main text, in such cases we replace non available features by the maximal possible value, which is the 6 shown in red here.

from the i-th residue, *i.e.*, $|j - i|$. Note that, when using probabilistic predictors, the label of a residue is determined as the one with the highest probability $\alpha^{\mathcal{A}}_{i,\cdot}$. When the number of residues that share l at the N-terminus side (respectively, at the T-terminus side) is insufficient, the missing distances are set to the greatest distance, *i.e*, the distance with the farthest residue that share l within the same terminus side.

– *Labeled segments window* : this is similar to the *labels local window* function except that rather than describing neighboring residues at position $i + \delta$, it describes neighboring segments $s_{i+\delta}$. A segment consists in a sub-sequence of consecutive residues that share a common label l, in the sense of the highest probability $\alpha^{\mathcal{A}}_{\cdot,\cdot}$. Therefore, given a segment s_i, the function returns one description of this segment (in the form of feature vectors) per relative position $\delta \in [-\frac{W}{2}, \frac{W}{2}]$. A segment $s_{i+\delta}$ is described by $L_{\mathcal{A}}$ (one per label $l \in \mathcal{L}_{\mathcal{A}}$) plus one features. Among the first $L_{\mathcal{A}}$ features, the one corresponding to the label of $s_{i+\delta}$ is equal to 1 while the other ones are set to 0. The last feature is the length of $s_{i+\delta}$. When the position $s_{i+\delta}$ is out-of-bounds the features are all set to 0.
– *Dimeric global histogram* : this feature function is an extension of *labels global histogram* with the difference that instead of calculating the frequency of occurrence of each single label, it computes the frequency of occurrence of each pairs of labels. A pair of labels is formed by the labels of two consecutive residues (a word of size 2). The hope is that the distribution of some pairs of labels are significantly different

in the case of disordered residues with respect to ordered ones. For example, a larger proportion of consecutive exposed residues may intuitively involve a larger disposition to form disordered regions. More formally, it returns one feature per pair of labels $(l_i, l_j) \in \mathcal{L}_\mathcal{A} \times \mathcal{L}_\mathcal{A}$, equal to

$$\frac{1}{n-1} \sum_{p=1}^{n-1} \mathbb{1} \left\{ \underset{l \in \mathcal{L}_\mathcal{A}}{\operatorname{argmax}}\, \alpha_{p,l}^\mathcal{A} = l_i \wedge \underset{l \in \mathcal{L}_\mathcal{A}}{\operatorname{argmax}}\, \alpha_{p+1,l}^\mathcal{A} = l_j \right\}.$$

– *Dimeric local histogram* : this feature function is identical to the dimeric global histogram one except that it computes the frequency within a sliding window. More formally, given a residue position k, it returns one feature per pair of labels $(l_i, l_j) \in \mathcal{L}_\mathcal{A} \times \mathcal{L}_\mathcal{A}$ equal to

$$\frac{1}{W} \sum_{p=k-W/2}^{k+W/2} \mathbb{1} \left\{ \underset{l \in \mathcal{L}_\mathcal{A}}{\operatorname{argmax}}\, \alpha_{p,l}^\mathcal{A} = l_i \wedge \underset{l \in \mathcal{L}_\mathcal{A}}{\operatorname{argmax}}\, \alpha_{p+1,l}^\mathcal{A} = l_j \right\}.$$

Our candidate feature functions are summarized in Table 6.2. Note that five of them are parameterized by window size parameters. To apply the feature function selection algorithm, we consider the following discrete sets of window sizes :

– *Local windows, separation profile window, labeled segments window and dimeric local histogram* : 1, 5, 11, 15, 21.

– *Local histograms* : 10, 20, 30, 40, 50, 60, 70, 80, 90.

This setting leads to a total of 109 candidate features functions.

Symbol	Parameter	d	Description
n	-	1	Number of residues
n_C	-	1	Number of cysteines
n_{AA}	-	20	Unnormalized global histogram
i	-	1	Position of residue
i/n	-	1	Relative position of residue
$h^{global}(\mathcal{A})$	-	$L_\mathcal{A}$	Labels global histogram
$h^{local}(\mathcal{A}, W)$	window size	$L_\mathcal{A} + 1$	Labels local histogram
$w(\mathcal{A}, W)$	window size	$W.L_\mathcal{A}$	Labels local window
$sep(\mathcal{A}, W)$	window size	$W - 1$	Separation profile window
$seg(\mathcal{A}, W)$	window size	$W.(L_\mathcal{A} + 1)$	Labeled segments window
$di^{global}(\mathcal{A})$	-	$L_\mathcal{A}^2$	Dimeric global histogram
$di^{local}(\mathcal{A}, W)$	window size	$L_\mathcal{A}^2$	Dimeric local histogram

TABLE 6.2: **Feature functions used in our experiments to encode residues.** Symbols, parameters, number of features (d) and description of our candidate feature functions. Top : feature functions that are directly computed from the primary structure. Bottom : feature functions defined for every kind of annotation $\mathcal{A} \in \{AA, PSSM, SS, SA\}$.

6.3 Results

This section describes our experimental study on disordered regions prediction. The first part presents the results of the main contribution of our work, which aims at determi-

ning a relevant representation on DISORDER723. The second part aims at constructing a model based on this relevant representation and ETs, and assessing this model on CASP10 and PDB30. In the third part, we investigate the novel feature function and attempt to interpret its role in the prediction of disordered regions.

6.3.1 Identification of a set of relevant feature functions

We now apply the feature function selection approach on top of ETs with the candidate feature functions of Table 6.2. We use a default setting of hyper-parameters of ETs that corresponds to an ensemble of 1 000 fully developed trees ($T = 1\ 000$, $N_{\min} = 2$) and K is set to the square root of the total number of features $\sqrt{d}$, as proposed by Geurts *et al* [58].

To avoid any risk of overfitting, we performed the selection on 10 different train/test splits of DISORDER723 and the performance measure being maximized by each run is the cross-validated AUC score of the training set. Table 6.3 reports the selected feature functions for each of the 10 independent runs. For the five iterations we consider, we observe that the selected feature functions on each of the 10 train/test splits are always $w(PSSM, 21)$, $sep(SA, 21)$, $h^{local}(AA, \cdot)$ with a window size varying in $\{50, 60, 70\}$, $w(SS, \cdot)$ with a sliding window size in $\{11, 15\}$ and $w(AA, \cdot)$ with a window size in $\{1, 5, 11, 15\}$. Regardless to window size parameters, the fact that we observed these feature functions during each run is very strong, since the selection algorithm has to select between 109 different candidate feature functions.

Fold	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**
1	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 60)$	$w(SS, 11)$	$w(AA, 1)$
2	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 60)$	$w(SS, 11)$	$w(AA, 11)$
3	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 60)$	$w(SS, 11)$	$w(AA, 5)$
4	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 50)$	$w(SS, 11)$	$w(AA, 1)$
5	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 50)$	$w(SS, 15)$	$w(AA, 15)$
6	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 60)$	$w(SS, 15)$	$w(AA, 5)$
7	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 70)$	$w(SS, 15)$	$w(AA, 15)$
8	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 50)$	$w(SS, 11)$	$w(AA, 5)$
9	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 50)$	$w(SS, 11)$	$w(AA, 1)$
10	$w(PSSM, 21)$	$sep(SA, 21)$	$h^{local}(AA, 60)$	$w(SS, 11)$	$w(AA, 1)$
Mean					
CV Score	0.852 ± 0.003	0.876 ± 0.003	0.884 ± 0.003	0.890 ± 0.003	0.894 ± 0.003
V Score	0.850 ± 0.029	0.874 ± 0.021	0.883 ± 0.022	0.888 ± 0.022	0.892 ± 0.22

TABLE 6.3: **Forward feature functions selection with 10 train/test splits.** *Mean* : averages over the ten *cross-validated scores* and the ten *validation scores*. The cross-validated score is calculated as the mean of the AUC score of each fold. The AUC score of a fold is calculated by sub-dividing the fold into 10 train/test splits and by averaging the AUC score of each of the 10 train/test splits. The validation score is the AUC score obtained when evaluating on the test set.

Note that, among the selected feature functions, two of them (the second and the fourth) rely on predicted structural annotations (the predicted solvent accessibility and

the predicted secondary structure, respectively), which tend to show that predicted structural annotations contribute to make better disordered regions predictors.

Not surprisingly, the most important feature function detected by the selection is a sliding window of evolutionary information, which confirms that disordered regions differ from ordered regions in terms of their conservation profile. This feature function is also important for many other protein structure prediction tasks (*e.g.*, [11]).

On the other hand, the second most important feature function highlighted by our algorithm, namely $sep(SA, \cdot)$, has - to our best knowledge - never been proposed in previous studies. Its discovery at a very early iteration was unexpected. It suggests that the proximities of a residue r (in terms of amino acid positions in the primary sequence) to its nearest exposed or to its nearest buried residues are correlated with the fact that r belongs to a disordered region. It is important to note the difference with $w(SA, \cdot)$. Indeed, $w(SA, \cdot)$ describes the solvent accessibility label of the flanking residues of r. The proximity is fixed and limited by the number of flanking residues to take into consideration. Whereas, $sep(SA, \cdot)$ describes the inverse. Namely, it describes the proximity of the nearest residues to r that correspond to fixed labels.

One way to explain the usefulness of this feature function is to look at the distributions of the distances that separate disordered (resp. ordered) residues to their nearest buried residue. Figure 6.4 shows the probability of a residue of being disordered (resp. ordered) according to the distance to its nearest buried residue, over the PDB30 dataset. We remark that the probability of a residue being disordered increases quickly when its distance to the next buried residue increases, and is above 0.5 as soon as the closest buried residue is at least 5 residues away.

Another important aspect of this discovery is that the $sep(SA, 21)$ feature function is systematically detected just before the local amino acid composition $h^{local}(AA, \cdot)$ and far before $w(AA, \cdot)$. Indeed, these other two feature functions describe in different ways the sequence complexity, which is well-known to be low within disordered regions [138]. This therefore reinforces the fact that $sep(SA, 21)$ may be a key-aspect in our understanding of protein disordered regions and, consequently, protein structure-function relationships.

The fourth selected feature function is a short sliding window over predicted secondary structures $w(SS, \cdot)$. The usefulness of these features may be related to the strong difference between the distributions of predicted secondary structures within disordered regions with respect to ordered ones. For example, Table 6.4 shows that 70.98% of disordered residues are predicted as coils against 40.57% as it is the case with ordered residues and that solely 5.76% are predicted as sheets against 20.76% for ordered regions.

According to these results, we focus in the following on assessing the relevance of the feature functions $w(PSSM, 21)$, $sep(SA, 21)$, $h^{local}(AA, 60)$, $w(SS, 11)$ and $w(AA, 1)$, where we chose windows sizes by taking the most frequent sizes reported in Table 6.3. Indeed, contrarily to the observation made in [11] that suggested a very small number of relevant feature functions in the context of disulfide bridge prediction, the selection algorithm identified here a larger set of interesting feature functions.

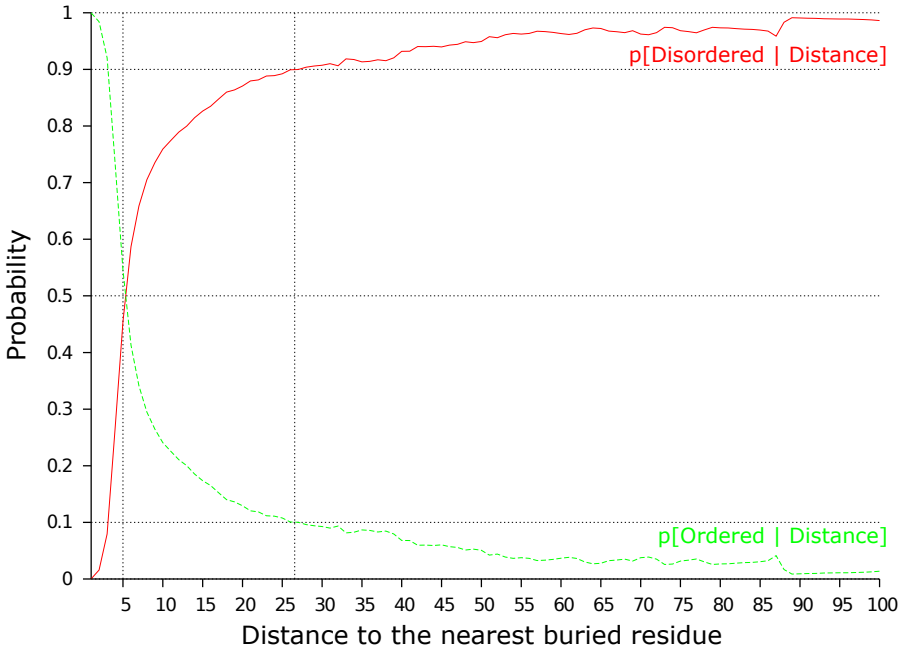

FIGURE 6.4: **Probability of being (dis)ordered w.r.t. the distance to the nearest buried residue.** For a given distance d, the probability $p[Disorder|d]$ of being disordered is calculated as the portion of disordered residues among the residues that have their nearest buried residue located at a distance d. We computed these curves on the actual values of the solvent accessibility of PDB30.

	Ordered		Disordered		Total	
Predicted helices	77,989	38.67%	3,235	23.26%	81,224	37.67%
Predicted sheets	41,874	20.76%	801	5.76%	42,675	19.79%
Predicted coils	81,840	40.57%	9,873	70.98%	91,713	42.54%
Total	201,703		13,909		215,612	

TABLE 6.4: **Distribution of predicted secondary structure.** Distribution of the number of ordered/disordered residues and the total number of residues for each secondary structure class on DISORDER723.

6.3.2 Evaluation of the selected feature functions

We now compare our models in terms of accuracy against a number of state-of-the-art methods on DISORDER723, CASP10 and PDB30. As previously, we use ETs with a default setting of its hyper-parameters. For each run, we use 80% of the training set to build an ensemble of trees predicting the probability to belong to a disordered region for a residue, and the remaining 20% to fix an 'optimal' decision threshold on this probability.

For DISORDER723, we consider two baselines. Both evaluated their predictive performance using a 10-fold cross-validation on DISORDER723. The first baseline is Cheng *et al.* [31], the authors of the DISORDER723 dataset. They proposed an ensemble of 1D-recursive neural networks that reached an area under the ROC curve of 0.878. The second

baseline is Eickholt *et al.* [47], who used boosted ensembles of deep networks to make predictions. They obtained a very high balanced accuracy (82.2%) and AUC (0.899).

The top of Table 6.5 reports our predictive performances when including successively the feature functions $w(PSSM, 21)$, $sep(SA, 21)$, $h^{local}(AA, 60)$, $w(SS, 11)$ and $w(AA, 1)$, while the bottom of the Table 6.5 reports the scores of the two baselines from the literature. We observe that using only $w(PSSM, 21)$ leads to a balanced accuracy (Acc) of 77.5%, an AUC of 0.853 and a F-measure of 49.6, which already outperforms the state of the art (46.3).

Features	**Balanced Acc**	**Sensitivity**	**Specificity**	**AUC**	**F-measure**
10-fold cross validation of our algorithm over DISORDER723					
$\{w(PSSM, 21)\}$	77.5 ± 2.43	74.1 ± 5.95	80.8 ± 3.13	0.853 ± 0.028	49.6 ± 3.38
$\{w(PSSM, 21), sep(SA, 21)\}$	79.0 ± 1.95	76.5 ± 4.14	81.6 ± 2.59	0.875 ± 0.019	51.7 ± 4.20
$\{w(PSSM, 21), sep(SA, 21), h^{local}(AA, 60)\}$	80.3 ± 2.17	78.2 ± 4.90	82.4 ± 2.47	0.884 ± 0.019	52.7 ± 3.85
$\{w(PSSM, 21), sep(SA, 21), h^{local}(AA, 60), w(SS, 11)\}$	80.6 ± 1.69	**79.0 ± 4.64**	82.2 ± 2.11	0.891 ± 0.020	53.4 ± 3.55
$\{w(PSSM, 21), sep(SA, 21), h^{local}(AA, 60), w(SS, 11), w(AA, 1)\}$	**81.1 ± 1.83**	78.6 ± 4.69	**83.5 ± 2.08**	**0.894 ± 0.021**	**55.3 ± 3.27**
$\{w(PSSM, 21), h^{local}(AA, 60), w(SS, 11), w(AA, 1)\}$	80.4 ± 1.94	76.8 ± 5.30	83.9 ± 2.37	0.883 ± 0.026	54.5 ± 2.70
Baselines tested on DISORDER723					
Cheng *et al.* (2005) [31]	-	-	-	0.878	-
Eickholt *et al.* (2013) [47]	**82.21 ± 0.49**	**74.60 ± 1.10**	**89.84 ± 0.18**	**0.899 ± 0.002**	**46.34 ± 4.5**

TABLE 6.5: **Accuracy evaluation on the DISORDER723 dataset.** Top : the mean and standard deviation of the scores obtained when 10-folds cross-validating DISORDER723 through the relevant feature functions. Bottom : baselines using DISORDER723 to assess their model.

Moreover, we remark that by incrementally adding the remaining selected feature functions to the set systematically leads to significant improvements on Acc, AUC and F-measure. We have used the paired t-test on the AUC scores to statistically assess the significance of each increment. We noted that the corresponding p-values ($2.6e^{-3}$, $5.4e^{-4}$, $2.2e^{-4}$ and $4.6e^{-3}$) are well below the classical null hypothesis threshold (0.05). This observation reinforces the fact that the selected feature functions are relevant. When comparing our model based on all five selected feature functions to the state-of-the-art, we obtain a disordered regions predictor, which is very competitive in term of Acc (81.1%), equivalent in term of AUC (0.894) and clearly better in term of F-measure of 55.3. The middle of Table 6.5 shows the impact on the predictive performance of our model when we do not consider $sep(SA, 21)$ among the input feature functions. As expected, the scores significantly deteriorate with a p-value of $1.9e^{-2}$ with respect to the model that comprise $sep(SA, 21)$. This observation reinforces the fact that this kind of feature function should be taken into account when predicting disordered regions.

To assess our models on CASP10, we compare our results against several baselines such as DNdisorder and PreDNdisroder, which were developed by Eickholt *et al.* [47].

Among the baselines, a number of them participated in the 10th CASP experiment. In order to make the comparison in a fair way, we construct our models on DISORDER723 using feature functions that were selected according to DISORDER723. Moreover, since DISORDER723 does not contain any overlapping sequences with CAPS10 and that DISORDER723 was formed well before CASP10, we are in the same blind prediction setting than the participants of the competition.

The top part of Table 6.6 reports our results with the different sets of relevant feature functions while the bottom part of Table 6.6 reports the scores obtained by the baselines considered in [47]. Once again, we observe that enlarging the feature functions set systematically leads to significant improvements except for $w(AA, 1)$. Two reasons may explain this phenomena, either the CASP10 dataset is too small and, consequently, prone to larger variances than big datasets, or the fifth iteration of the selection procedure starts to overfit DISORDER723, which means that $w(AA, 1)$ is not portable to other datasets. We believe that the second reason is more likely to be the true explanation, because the function $w(AA, 1)$ consists in discriminating disordered residues from ordered ones based on their amino acid type, which may be too dataset specific. As mentioned, the p-value of $4.6e^{-3}$ determined when including this feature set was indeed quite larger than those resulting from the inclusion of the other feature sets.

According to Table 6.6, we remark that our model based on $\{w(PSSM, 21)$, $sep(SA, 21)$, $h^{local}(AA, 60)$, $w(SS, 11)\}$ achieves excellent performances with respect to the state-of-the-art. We even slightly improve the state-of-the-art with a balanced accuracy of 77.29% against 77.06%, however, according to the variations, this improvement is not significant. We nevertheless outperformed the method of Eickholt *et al.* [47] (DNdisorder), which presented similar performances than our model on DISOPRED723.

Although CASP10 is an entirely independent test set that had no detectable similarity to available structures at this time, its very limited size does not enable it to capture the universe of protein disorder. This is why we also evaluated our model on the far larger dataset PDB30. Table 6.7 compares the predictive performances obtained by three freely and easily downloadable methods(DISOPRED2[135], IUPred[41] and ESpritz[132]) with respect to our model. We observe that our approach outperforms the three baselines with a balanced accuracy of 80.3% and presents a comparable area under the ROC curve (0.883) to ESpritz, even though our approach treats each residue independently, *i.e.*, without explicitly exploiting the key-fact that disordered regions are made of contiguous residues. Figure 6.5 shows the ROC curves for DISORDER2, ESpritz and IUpred on PDB30. We observe that our method and ESpritz are very close to each other and that ESpritz is slightly better in the low false positive rate regime.

Note that since PDB30 and DISORDER723 are independent, the evaluation of our model is fair. However, we do not have access to the learning stage of the compared methods, which has possibly used sequences similar to the ones present in PDB30. This may lead to an over-estimation of the predictive performance of those methods.

6.4 Discussion

Predicting and understanding the nature of disordered regions is a key sub-problem of protein structure and function inference. In this chapter, we have adapted the algorithm

Features	Balanced Acc	Sensitivity	Specificity	AUC	F-measure
Models learnt on DISORDER723 by our algorithm and tested on CASP10					
$\{w(PSSM,21)\}$	71.94 ± 0.71	70.71 ± 1.3	73.16 ± 0.32	0.795 ± 0.007	39.47 ± 0.73
$\{w(PSSM,21), sep(SA,21)\}$	74.95 ± 0.69	70.31 ± 1.4	79.59 ± 0.29	0.834 ± 0.006	38.51 ± 0.81
$\{w(PSSM,21), sep(SA,21), h^{local}(AA,60)\}$	77.17 ± 0.67	71.64 ± 1.3	**82.69 ± 0.28**	0.847 ± 0.006	39.95 ± 0.88
$\{w(PSSM,21), sep(SA,21), h^{local}(AA,60), w(SS,11)\}$	77.29 ± 0.66	**74.17 ± 1.3**	80.41 ± 0.29	**0.851 ± 0.006**	**40.24 ± 0.84**
$\{w(PSSM,21), sep(SA,21), h^{local}(AA,60), w(SS,11), w(AA,1)\}$	**77.35 ± 0.65**	72.84 ± 1.3	81.85 ± 0.29	0.850 ± 0.006	39.82 ± 0.87
Baseline performances on CASP10 as published by the CASP10 competition					
metaprdos2 (340)	**77.06 ± 0.92**	64.73 ± 1.4	89.40 ± 0.98	0.8727 ± 0.006	41.24 ± 2.9
PreDisorder (125)	76.86 ± 0.67	**67.19 ± 1.7**	86.34 ± 0.94	0.839 ± 0.006	37.50 ± 1.5
POODLE (216)	76.84 ± 0.78	62.74 ± 1.6	90.94 ± 0.26	0.866 ± 0.006	43.06 ± 1.0
PreDNdisorder [47]	76.55 ± 0.75	61.74 ± 1.8	91.36 ± 0.61	0.864 ± 0.006	43.42 ± 1.5
ZHOU-SPARKS-X (413)	75.68 ± 0.76	64.81 ± 1.4	86.55 ± 0.96	0.859 ± 0.006	36.43 ± 1.9
DNdisorder (424)	75.19 ± 0.71	61.92 ± 1.4	88.46 ± 0.29	0.848 ± 0.006	38.02 ± 1.1
CSpritz (484)	75.13 ± 1.4	66.31 ± 1.3	83.94 ± 2.4	0.822 ± 0.007	33.64 ± 3.7
Espritz (380)	73.16 ± 1.6	59.24 ± 1.4	87.08 ± 2.6	0.846 ± 0.006	34.58 ± 4.7
espritz_nopsi_X	71.98 ± 0.97	53.10 ± 1.5	90.87 ± 0.77	0.815 ± 0.007	37.56 ± 2.4
PrDOS-CNF (369)	70.35 ± 0.88	41.95 ± 1.8	**98.74 ± 0.14**	**0.896 ± 0.005**	**52.50 ± 1.4**
biomine_dr_mixed (478)	69.17 ± 0.68	39.95 ± 1.4	98.40 ± 0.11	0.884 ± 0.006	49.40 ± 1.3
biomine_dr_pdb_c (228)	67.81 ± 1.2	36.88 ± 2.6	**9**8.74 ± 0.15	0.882 ± 0.006	47.65 ± 2.1
iupred_short	63.26 ± 0.70	30.68 ± 1.5	95.84 ± 0.25	0.664 ± 0.007	32.34 ± 1.2

TABLE 6.6: **Accuracy evaluation on the CASP10 dataset.** Top : the scores obtained when evaluating CASP10 on models learnt on DISORDER723 through the relevant feature functions found on DISORDER723. Bottom : comparison of a number of predictors, which participated in or evaluated their model to the 10th CASP experiment. These results were reported by [47]. In parenthesis : the group number of the methods that participated in the CASP10 experiment. The standard deviations were calculated by a bootstrapping procedure in which 80% of the dataset was sampled 1000 times, as it was done by [47].

presented in Chapter 5 on disulfide bridge prediction in order to identify the best way to represent protein residues in order to be usable by disordered region predictors. To this end, we used extremely randomized tree ensembles as an 'off-the-shelf' base learner in our feature function selection pipeline. We applied our approach to the DISORDER723 dataset from the literature, so as to select relevant subsets of feature functions and to build simple residue-wise disorder prediction models.

Our experiments have shown that the combination of the feature functions $w(PSSM,21)$ (a local window of size 21 of evolutionary information), $sep(SA,21)$ (a window of 21 of the separation profile of predicted solvent accessibility), $h^{local}(AA,60)$ (a local histogram of size 60 of primary structure) and $w(SS,11)$ (a local window of size 11 of predicted secondary structure) is a relevant representation of protein residues in the context of disordered regions prediction.

From a biological point of view, the major contribution of this paper is the discovery of

Method	Balanced Acc	Sensitivity	Specificity	AUC	F-measure
Our method	$\mathbf{80.36 \pm 4.8e^{-2}}$	$\mathbf{82.67 \pm 9.3e^{-2}}$	$78.06 \pm 2.3e^{-2}$	$0.8835 \pm 4.5e^{-4}$	$33.12 \pm 6.3e^{-2}$
At 94.7% of specificity	$76.73 \pm 5.1e^{-2}$	$58.79 \pm 10.1e^{-2}$	$94.67 \pm 1.3e^{-2}$	$0.8835 \pm 4.5e^{-4}$	$49.89 \pm 8.4e^{-2}$
DISOPRED2 [135]	$76.96 \pm 5.7e^{-2}$	$60.01 \pm 11.3e^{-2}$	$93.90 \pm 1.3e^{-2}$	$0.8658 \pm 4.9e^{-4}$	$48.40 \pm 8.8e^{-2}$
ESpritz [132]	$78.49 \pm 5.6e^{-2}$	$62.26 \pm 11.3e^{-2}$	$\mathbf{94.71 \pm 1.3e^{-2}}$	$\mathbf{0.8856 \pm 4.4e^{-4}}$	$\mathbf{52.20 \pm 9.2e^{-2}}$
IUPred [41]	$74.99 \pm 5.8e^{-2}$	$55.98 \pm 11.4e^{-2}$	$93.99 \pm 1.4e^{-2}$	$0.8363 \pm 5.6e^{-4}$	$46.13 \pm 8.9e^{-2}$

TABLE 6.7: **Evaluation on the PDB30 dataset.** Predictive performances of three freely and easily downloadable methods on PDB30. The standard deviations were calculated over the same 100 bootstrap copies of the whole dataset. Given the huge size of the dataset, all differences (even if they are sometimes tiny) are statistically significant. Notice that (except for the AUC calculation), our method uses a classification threshold that was selected on the training dataset (Disorder723) so as to maximize the balanced accuracy, which explains its difference in (sensitivity, specificity) pattern, as compared to the other methods. Changing the threshold so as to yield a 94.7% specificity on PDB30, would reduce its sensitivity to 58.8%.

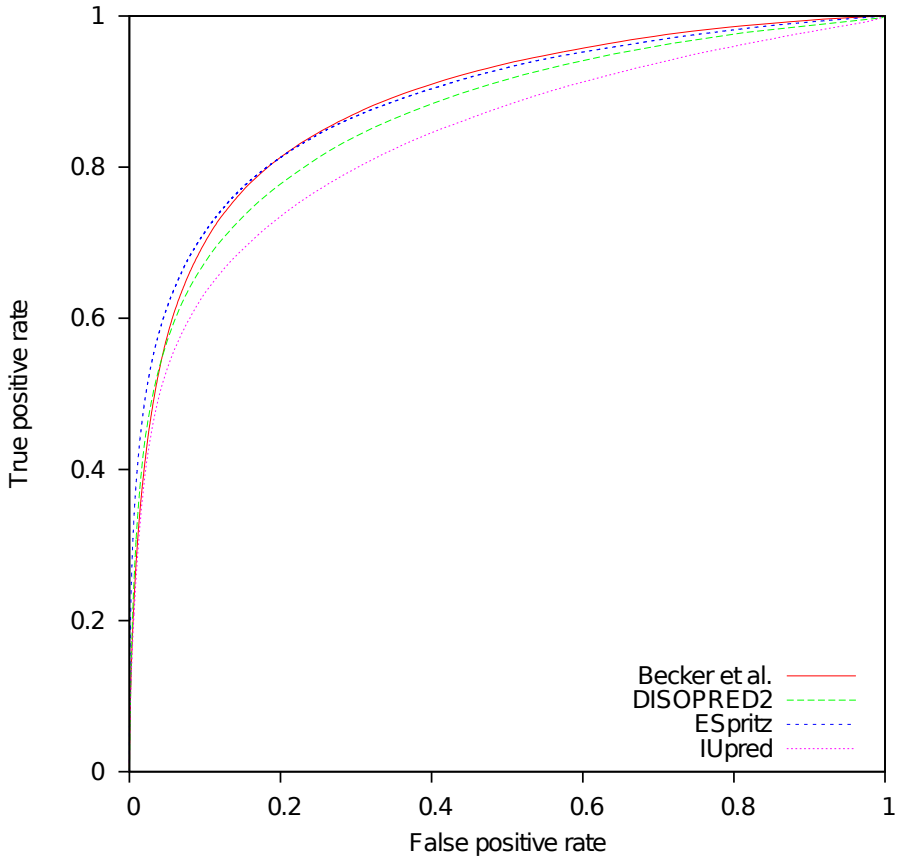

FIGURE 6.5: **ROC curves on PDB30 dataset.** ROC curve of our method (Becker *et al.*) and three freely downloadable predictors : DISPRED2 [135], ESpritz [132] and IUPred [41].

the $sep(SA, \cdot)$ feature function, which has - to our best knowledge - never been highlighted as important in this context. This observation suggests that the proximities (in terms of

amino acid distances) between consecutive exposed (and consecutive buried) residues should play a role in the formation of disordered regions and, consequently, in protein structure-function relationships.

To validate these observations with respect to the state-of-the-art in disorder prediction, we also evaluated our model on the set of proteins used in the CASP10 competition. On CASP10, our model constructed on the DISORDER723 dataset turned out to obtain a very competitive assessment in terms of various predictive accuracy indicators, in spite of the fact that our work was focusing on feature identification rather than accuracy maximization. Since CASP10 is a small dataset that does not capture the whole universe of protein disorder, we further assessed our model on the independent and very large PDB30 dataset, which contains 12,090 proteins and 2,991,008 residues. On PDB30, our model obtained as well very competitive results with respect to three state-of-the-art methods, by clearly beating two of them and being at a tie with the third one.

From a methodological point of view, our paper also shows that the systematic feature family selection pipeline proposed in [11] and adapted here, is a viable and robust approach to yield interpretable information about relevant representations for protein structure inference and allows at the same time to build predictors with state-of-the-art accuracy. Still, it might be the case that extremely randomized tree ensembles with their defaults settings are not the best classifiers for disordered regions prediction. Also, in our predictors we treated each residue independently, *i.e.*, without taking advantage of the structured nature of the problem. Therefore, a main direction for future research is to evaluate more sophisticated classifiers using the feature functions highlighted by the present study.

CHAPITRE 7

Perspectives and Conclusions

Contents

In this manuscript, we have tackled several surrogate problems related to protein structure prediction either by solving them together in a multi-task context or by identifying relevant representations of the information for machine learning approaches.

In the first part of our research (Chapter 4), we noted a gap between the methods used in the bioinformatics literature, which mostly treated these problems independently, and the field of machine learning, which has investigated multi-task methods in the recent years. To fill this gap, we have developed a framework called *iterative multi-task learning sequence labeling* for jointly solving a group of five sequence labeling problems derived from protein structures.

In the second part of our research (Chapters 5 and 6), we focused on an old topic in machine learning : the feature generation and selection problem. Indeed, the way to encode complex and structured objects, such as sequences or graphs, into vectors of features typically has a major impact on the predictions. To determine the best way to perform this encoding, we have developed a pipeline, called *forward feature function selection*, for identifying the minimal set of relevant feature functions among a large list of candidate ones.

Section 7.1 of this chapter provides a discussion around our contributions and our results. We subsequently discuss some directions and perspectives of future researches in Section 7.2.

7.1 Conclusions

Section 7.1.1 provides a conclusion of the first part of our research while Section 7.1.2 provides a conclusion of the second part of our research.

7.1.1 Multi-task learning

Secondary structure prediction, solvent accessibility prediction and disordered regions prediction are all surrogate annotation problems derived from protein structure prediction. Although these prediction tasks are closely related, they have mostly been treated independently, *i.e.*, none of the tasks took advantage of the predictions made on the other tasks.

From a methodological point of view. Our major contribution has been to introduce a conceptually simple but powerful framework to jointly solve multiple related tasks, which can take advantage of any sequence labeling algorithm (called base-model), from simple classification-based approaches to modern structured prediction approaches. Indeed, the concept of our multi-task approach consists in iteratively re-estimating each target using the predictions of the last learned base-models. Moreover, as a matter of fact, our iterative multi-task approach is not restricted to predicting sequence labels. It may be extended to many other complex application domains (text processing, image analysis, network monitoring and control, robotics), where data is available about several related tasks and where synergies could similarly be exploited to enhance machine learning solutions.

From a biological point of view. Our empirical experiments have shown that our methodology is very powerful. When considering a set of five protein annotation tasks and a linear SVM trained by stochastic gradient descent as base-learner, our approach significantly outperformed the state-of-the-art accuracy by +2.1% for secondary structure prediction. More importantly, we demonstrated that our multi-task approach systematically outperformed models learnt in single-tasks fashion, *i.e.*, models that treat each task independently.

7.1.2 Feature function selection

In bioinformatics, it is often the case that the comparison of the conclusions of different works is difficult because these studies rely on different learning algorithms and slightly differ in their experimental protocol. For example, it is common to see authors that build and evaluate a predictor on a recent dataset and then compare their predictive performance with a study that used a different dataset and even sometimes that used different evaluation criteria.

From a methodological point of view. Our major contribution has been to develop a forward feature function selection algorithm to identify a set of relevant feature encoding functions among an extensive set of candidate ones. We applied this methodology to the problems of disulfide connectivity pattern prediction (in Chapter 5) and of disordered regions prediction (in Chapter 6).

Our selection method departs from traditional feature selection with respect to three unique aspects :

- *Feature function selection* : At each iteration, it selects feature functions rather than individual features. By proceeding in this way, the resulting features are more consistent and less prone to overfitting.
- *Insertion in a pipeline* : It optimizes the scoring measure of the whole pipeline rather than the predictive performance of the model alone. It is typically the case in disulfide pattern prediction, where a disulfide bonding predictor with higher accuracy can lead to worse disulfide pattern predictions when combined with the graph matching algorithm, and conversely.
- *Interpretability* : It not only aims at constructing a pipeline maximizing scoring function, but also at drawing more general scientific conclusions on the relevance of various annotations of the primary structure.

As a minor contribution, we have shown, particularly in Chapter 5, that tree-based ensemble methods are very promising to structural bioinformatics problems despite the fact that they were surprisingly rarely applied in this context.

From a biological point of view. The interpretability of our approach highlighted several important key-aspects in our understanding of both disulfide connectivity patterns and protein disordered regions.

In Chapter 5, we have shown that a very limited number of carefully selected feature functions are sufficient to reach a very high performing disulfide pattern predictor : $w(PSSM, 15)$ (a local window of size 15 on the evolutionary information) and $csp(17)$ (a window of size 17 on the cysteine separation profile). Indeed, our model using ETs with these two feature functions outperforms the state of the art with a +3.2% improvement of correctly predicted patterns on the $SPX+$ dataset. In this study, we also investigated the questions of how to couple a disulfide pattern predictor with a pre-filtering step based on either disulfide chain classifications or cysteine bonding state predictions. However, we believe that one direction of research to strongly improve the number of correctly predicted patterns is to develop more sophisticated approaches to better couple the cysteine bonding state prediction task with the pattern prediction task.

In Chapter 6, the major contribution is undoubtedly the discovery of the $sep(SA, \cdot)$ feature function, which suggests that the proximities (in terms of amino acid distances) between consecutive exposed (and consecutive buried) residues should play a role in the formation of disordered regions and, consequently, in protein structure-function relationships. The assessment of our selected feature functions coupled with ETs on the set of proteins used in the CASP10 competition confirmed their relevance for disordered regions prediction and also showed that our model is very competitive and performs similar predictive performance than the state-of-the-art.

Table 7.1 summarizes the feature functions that were selected for the four tasks that we considered in our studies.

According to these results, we conclude this section by affirming that our systematic feature family selection pipeline is a viable and robust approach to yield interpretable information about relevant representations for protein structure inference and allows at the same time to build predictors with state-of-the-art accuracy.

Task	Selected feature functions
Disulfide chain classification	$\{h^{global}(PSSM), h^{global}(AA)\}$
Cysteine bonding state prediction	$\{w(PSSM, 11), h^{global}(PSSM), n_C\}$
Disulfide pattern prediction	$\{w(PSSM, 15), csp(17)\}$
Disordered regions	$\{w(PSSM, 21)$, $sep(SA, 21)$, $h^{local}(AA, 60)$, $w(SS, 11)\}$

TABLE 7.1: **Selected feature functions of the four tasks.** The feature functions were determined by the application of our selection algorithm on top of extremely randomized trees, using the SPX− dataset for chain classification and cysteine bonding state prediction, the SPX+ dataset for disulfide pattern prediction and the DISORDER723 dataset for disordered regions prediciton.

7.2 Perspectives

Protein structure prediction is one of the most venerable *holy grails* in bioinformatics. In this manuscript, we believe that there exists an extremely complex function that given an amino acid sequence determines a unique structure with respect to their environmental conditions. We also believe that this function cannot be solely resumed to a search of lowest energy.

To progress towards the prediction of tertiary structures, our long-term direction of research is to design and apply a *unified iterative multi-task architecture* using annotation-specific set of feature functions. The first section presents the unified architecture as an extension of our iterative multi-task sequence labeling framework in which tertiary structure is treated as any other of its surrogate annotation tasks. However, we believe that to better encompass the physical constraints exerted on tertiary structure, we have to introduce a refinement step, which should be based on physical force-field models. The second section presents how this extra level should be coupled with the unified architecture.

7.2.1 A unified architecture for protein structure prediction

The aim of the unified architecture is to share and re-estimate the predicted information of considered tasks regardless of their structured nature. For example, Figure 7.1 illustrates this unified architecture for 10 protein annotation problems. Each of these tasks is represented by a predictor, whose inputs are the primary structure, the predictions made by the predictor of the other tasks or its own previous predictions. According to the structured nature of the problems, we can group the prediction tasks into four groups : global property prediction (*e.g.*, the function of the protein and its cell localisation within the cell), sequence labeling (*e.g.*, secondary structure, solvent accessibility and disordered regions prediction), contact map prediction (*e.g.* disulfide pattern, β-bridge alignment and residue-residue contact prediction) and tertiary structure prediction (*e.g.*, conformational angles prediction or atom-coordinates prediction).

Although this architecture is very similar to our iterative multi-task sequence labeling algorithm proposed in Chapter 4, the main difference resides in the ability to treat several kinds of structured objects (single values, sequences, graphs, matrices, 3D structure).

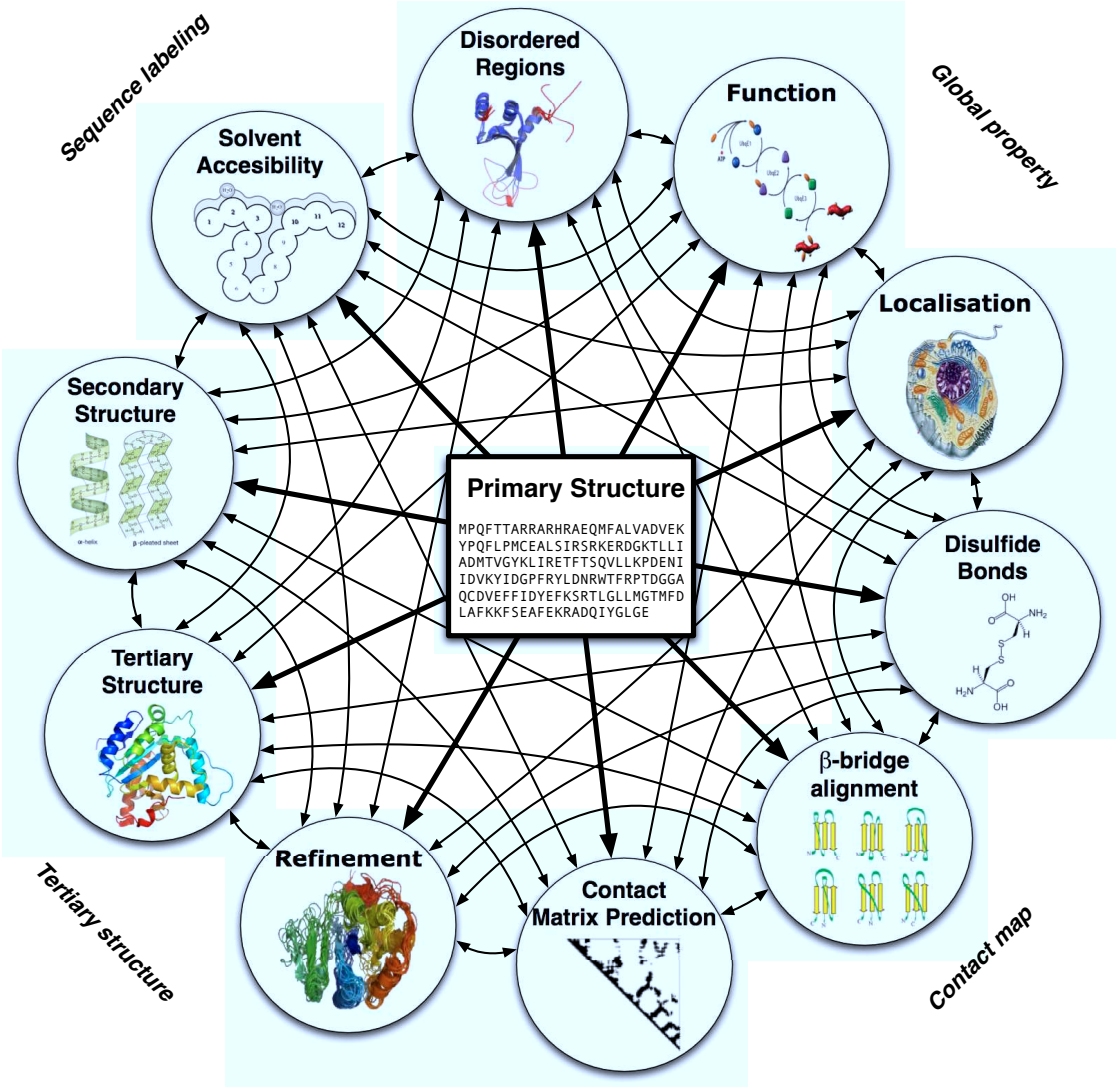

FIGURE 7.1: **A unified architecture for protein structure prediction.** Each circle is a protein annotation predictor, whose inputs are the primary structure and the predictions made by the others tasks. The tasks are grouped according to their structured nature.

We now detail the key-aspects that this unified architecture should satisfy.

Feature function engineering. The most important point to successfully deal with structured objects is undoubtedly the feature generation, *i.e.*, the manner to encode the structured information into vector of features in order to be usable by machine learning algorithms. In the studies presented in this manuscript, we developed a number of feature functions to describe : global properties of proteins (*e.g.*, global label histograms and length of proteins), residue environments (*e.g.*, local label histograms and sliding windows), residue-pair environment (*e.g.*, interval label histograms and position difference of residues) and cysteine-pair specific environment (*e.g.*, cysteine separation profile).

Distinct kinds of structured objects require specific feature functions and the higher is

the dimensionality of the structured objects, the larger is the number of possible manners to project them into one dimensional feature vectors. Indeed, we remark that usually objects of dimensionality $d+1$ can re-use feature functions of objects of dimensionality d. For example, in disulfide bond prediction, a feature function that aims at describing the cysteine-pair environment may be constructed by concatenating the residue environment the two cysteines.

In addition to the feature functions presented in this manuscript, the development of the unified architecture requires a new collection of feature functions able to encode graphs, matrices, 3D coordinates or angles, and any other structures that are considered suitable for future experiments.

Feature function selection. The second most important point to achieve an efficient unified architecture is certainly to determine the best annotation-specific set of feature functions. Indeed, according to the results on feature function selection obtained in Chapters 5 and 6, we now strongly believe that the inputs of each annotation problem must be encoded using a set of feature functions, which is specific to the problem.

Note that this opinion is in contrasts with what we did in our multi-task investigations (Chapter 4), which exploited the same feature encoding scheme regardless of the annotation problem. In this study, we have shown that learning several protein annotation tasks in a multi-task context systematically outperforms single-task learning. We therefore guess that, by carefully selecting the feature representation, the unified architecture will lead to significant improvements, in terms of predictive accuracy, on each one of the considered tasks.

We recommend future experiments to progress towards the mapping of relevant feature functions to annotations and hence to complete Table 7.1. For this purpose, we suggest to further exploit our forward feature function selection algorithm as it demonstrated to be a viable and robust approach to yield interpretable information about relevant representations.

Two-stage feature function selection. Without using sophisticated structured learning algorithm, the forward feature function selection method is not capable to take advantage of the structured nature of the problem. It was typically the case in our study about disordered regions prediction. We did not take into account the key fact that disordered regions are segments and hence that consecutive residues tend to share the same label.

In fact, it is important to note that the unified iterative multi-task architecture should need two sets of feature functions per annotation task. Indeed, due to the iterative nature of the algorithm, the joint distribution $P(\mathcal{X}, \mathcal{Y})$ of input-output spaces of models at first iteration and those at next iterations is different. Therefore, the feature functions selected for a model at a first iteration, which only benefits of the primary structure and the predictions (if they exist) made by the other predictors, and the feature functions selected for a model at subsequent iterations, which benefits of the primary structure and the predictions made by predictors of the previous iterations (including its own task), may/should be different.

Until now, we have determined the feature function sets for predicting disordered regions and disulfide bridges at the first iteration of an iterative multi-task approach. A

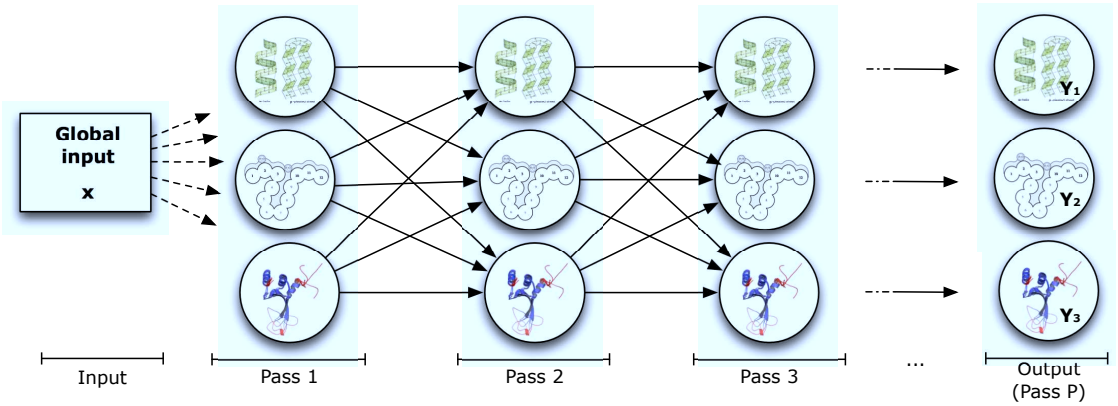

FIGURE 7.2: **Parallel iterative multi-task learning.** Each step (circle) is a distinct base-model, whose inputs are the global input x. Each pass is composed of three distinct steps, one per task. The output is directly derived from the state at the end of the last step.

direction for future research is therefore to establish relevant feature functions using the predictions made by our current models with the hope to improve the predictive accuracy.

To tackle this problem, we propose the following two-stage feature function selection protocol :

1. Apply the feature function selection algorithm on primary structure and predictions made by the other predictors ;
2. Construct a model using the selected feature function ;
3. Make predictions ;
4. Apply the selection algorithm using additional feature function devoted to describe the predictions made in step (3).

Another direction of future research is to evaluate more sophisticated classifiers using the feature functions highlighted by our algorithm such as conditional random fields, recursive neural networks or post-filtering steps.

Speed-up the multi-task algorithm. As a small improvement of our iterative multi-task algorithm, depicted in Figure 4.1, which consists in sequentially learning base-models one-by-one in a predefined order, we propose to parallelize each iteration (or pass). According to the number C of cores of the computer, the number P of iterations and the number T of tasks that the user wants to treat at each iteration, this extension could drastically reduce the computational time from $O(T.P)$ to $O(\lceil T/C \rceil.P)$

Figure 7.2 illustrated a model of P iterations and three tasks in a parallel context. In this view of the algorithm, the tasks of a same pass are independent and can be learnt simultaneously. In few of our preliminary experiments, we did not observe significant predictive performance differences with respect to the sequential version of the algorithm after the two first iterations.

ETs as base-models. Note that despite the fact that several studies have shown that tree-based ensemble methods often reach state-of-the-art results in supervised learning (see e.g. [24]), these methods were surprisingly rarely applied to structural bioinformatics problems yet. According to the results obtained in Chapters 5 and 6, we believe that ETs

provide a general supervised learning approach that can be applied to a wide range of protein related prediction problems, in particular, as a base-model for the unified architecture.

Moreover, ETs are easy to tune (the default value of their hyper-parameters are usually sufficient) and thanks to their use of decision trees, they benefit from good scaling properties, making them applicable to large sets of training proteins and large sets of features.

7.2.2 A semi-unified architecture for protein structure prediction

In the previous section, we presented the general guidelines that the *unified iterative multi-task architecture* should satisfy to progress towards the prediction of tertiary structures. In this perspective, the prediction of the tertiary structures of proteins is implemented as any other tasks, *i.e.*, as an iterative process that makes predictions solely based on the primary structure and outcomes of the previous iterations.

However, this unified architecture solely relies on base-models trained by machine learning algorithms without consideration for our knowledge of the laws of physics, in particular the principles of thermodynamics, which mainly drive the protein folding process. To better encompass the physical constraints exerted on tertiary structure, we therefore suggest a *semi-unified architecture* similar to what we now call the *fully-unified architecture*.

Figure 7.3 illustrates a semi-unified architecture issuing from the adaptation of the fully-unified architecture presented by Figure 7.1. The semi-unified architecture should be composed of three key-components :

- *A fully-unified architecture* : An iterative multi-task architecture that tackles a number of surrogate protein structure prediction problems.
- *A protein structure predictor* : A machine learning approach that creates a rough structure from the predictions made by the fully-unified architecture.
- *A refinement stage* : A physical model that applies a force-field on the coarse macromolecule structure provided by the protein structure predictor.

Since the knowledge of the tertiary structure implies the knowledge of each of its surrogate annotations, we believe that the refined structure may strongly help to better predict each of the surrogate problems considered in the fully-unified architecture, and thus, lead to a more accurate seed for the refinement step. We therefore suggest to iteratively re-inject the refined structure into the whole pipeline.

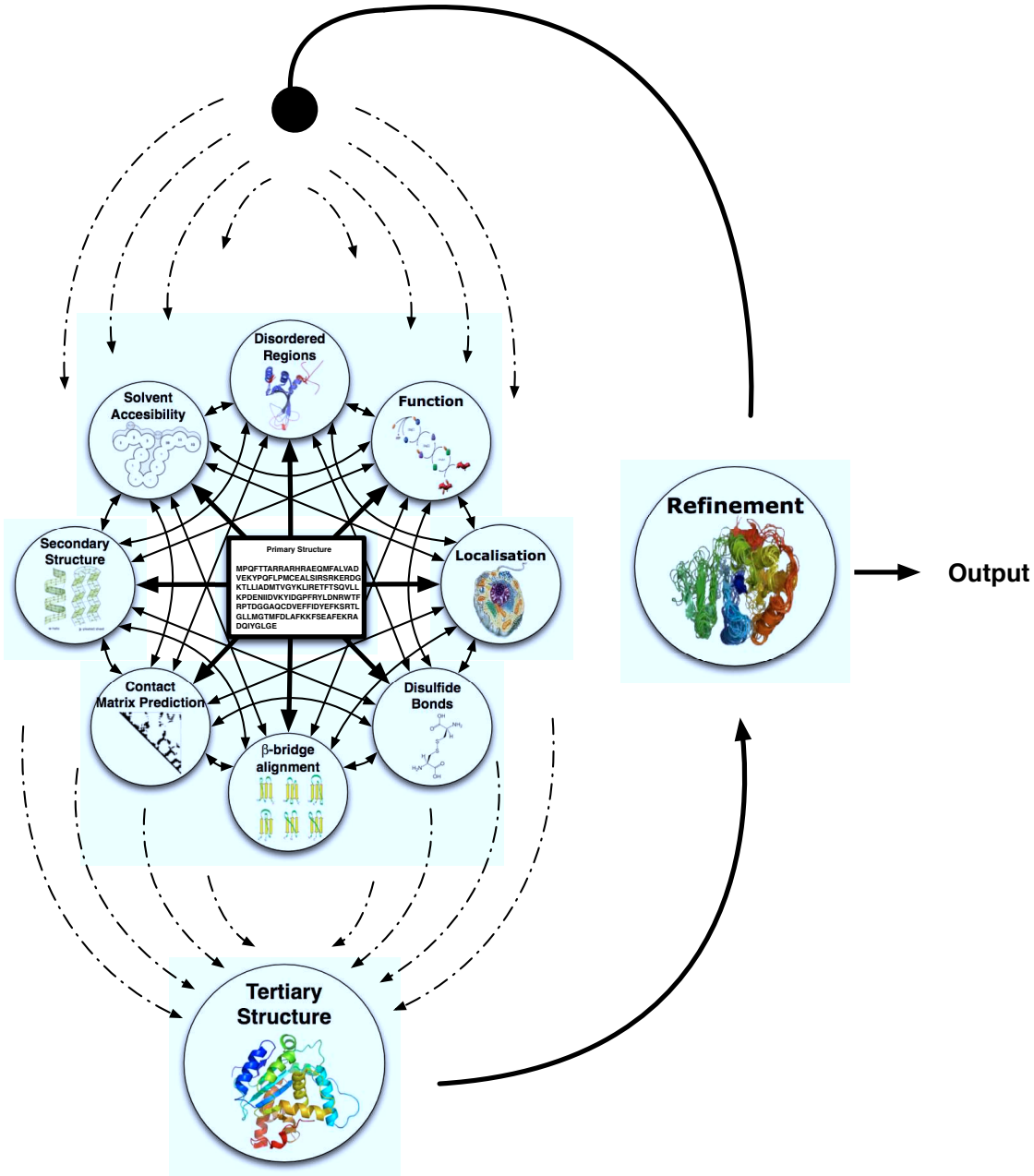

FIGURE 7.3: **A semi-unified architecture for protein structure prediction.** Each circle is a protein annotation predictor, whose inputs are the primary structure and the predictions made by the others tasks. Bottom dashed arrows represent the outcome flow from the protein annotation predictors to a tertiary structure predictor. The top dashed arrows represent the refined tertiary structure that is re-injected as input of the protein annotation predictors.

Bibliographie

[1] R. Adamczak, A. Porollo, and J. Meller. Accurate prediction of solvent accessibility using neural networks–based regression. *Proteins : Structure, Function, and Bioinformatics*, 56(4) :753–767, 2004.

[2] R. Adamczak, A. Porollo, and J. Meller. Combining prediction of secondary structure and solvent accessibility in proteins. *Proteins*, 2005.

[3] S. Ahmad and M. Gromiha. Netasa : neural network based prediction of solvent accessibility. *Bioinformatics*, 18(6) :819–824, 2002.

[4] S. Ahmad, M. Gromiha, and A. Sarai. Real value prediction of solvent accessibility from amino acid sequence. *Proteins : Structure, Function, and Bioinformatics*, 50(4) :629–635, 2003.

[5] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped blast and psi-blast : a new generation of protein database search programs. *Nucleic Acids Research*, 25(17) :3389–3402, 1997.

[6] C. Ambroise and G. J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, 99(10) :6562–6566, 2002.

[7] C. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(96) :223–30, 1973.

[8] D. Baker and D. Agard. Kinetics versus thermodynamics in protein folding. *Biochemistry*, 33(24) :7505–7509, 1994. PMID : 8011615.

[9] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11) :937–946, 1999.

[10] P. Baldi, J. Cheng, and A. Vullo. Large-scale prediction of disulphide bond connectivity. In *Advances in Neural Information Processing Systems*, pages 97–104. MIT Press, 2005.

[11] J. Becker, F. Maes, and L. Wehenkel. On the relevance of sophisticated structural annotations for disulfide connectivity pattern prediction. *PLoS One*, 8(2) :e56621, 2013.

[12] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1) :235–242, 2000.

[13] S. Betz. Disulfide bonds and the stability of globular proteins. *Protein Sci*, 2(10) :1551–8, 1993.

[14] C. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. springer New York, 2006.

[15] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[16] L. Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[17] L. Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[18] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[19] L. Brocchieri and S. Karlin. Protein length in eukaryotic and prokaryotic proteomes. *Nucleic Acids Research*, 33(10) :3390–3400, 2005.

[20] C. Bystroff, V. Thorsson, D. Baker, et al. Hmmstr : a hidden markov model for local sequence-structure correlations in proteins. *Journal of molecular biology*, 301(1) :173, 2000.

[21] H. Cai, W. Cong, S. Ji, S. Rothman, S. Maudsley, and B. Martin. Metabolic dysfunction in alzheimers disease and related neurodegenerative disorders. *Current Alzheimer Research*, 9(1) :5–17, 2012.

[22] A. Camproux, R. Gautier, and P. Tuffery. A hidden markov model derived structural alphabet for proteins. *Journal of molecular biology*, 2004.

[23] R. Caruana. Multitask learning. *Machine learning*, 28(1) :41–75, 1997.

[24] R. Caruana and A. Niculescu. An empirical comparison of supervised learning algorithms. In *In Proc. 23 rd Intl. Conf. Machine learning (ICML'06)*, pages 161–168, 2006.

[25] C. N. Cavasotto and S. S. Phatak. Homology modeling in drug discovery : current trends and applications. *Drug discovery today*, 14(13) :676–683, 2009.

[26] A. Ceroni, A. Passerini, A. Vullo, and P. Frasconi. Disulfind : a disulfide bonding state and cysteine connectivity prediction server. *Nucleic Acids Research*, 34(suppl 2) :W177–W181, 2006.

[27] C.-C. Chang and C.-J. Lin. LIBSVM : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2 :27 :1–27 :27, 2011.

[28] K. Chen and N. Rajewsky. The evolution of gene regulation by transcription factors and micrornas. *Nature Reviews Genetics*, 8(2) :93–103, 2007.

[29] J. Cheng, A. Z. Randall, M. J. Sweredoski, and P. Baldi. Scratch : a protein structure and structural feature prediction server. *Nucleic acids research*, 33(suppl 2) :W72–W76, 2005.

[30] J. Cheng, H. Saigo, and P. Baldi. Large-scale prediction of disulphide bridges using kernel methods, two-dimensional recursive neural networks, and weighted graph matching. *Proteins : Structure, Function, and Bioinformatics*, 62(3) :617–629, 2006.

[31] J. Cheng, M. J. Sweredoski, and P. Baldi. Accurate prediction of protein disordered regions by mining protein structure data. *Data Mining and Knowledge Discovery*, 11(3) :213–222, 2005.

[32] Y. Cheng, T. LeGall, C. J. Oldfield, J. P. Mueller, Y.-Y. J. Van, P. Romero, M. S. Cortese, V. N. Uversky, and A. K. Dunker. Rational drug design via intrinsically disordered protein. *Trends in biotechnology*, 24(10) :435–442, 2006.

[33] W. Cohen and V. Carvalho. Stacked sequential learning. In *International Joint Conferences on Artificial Intelligence*, pages 671–676, 2005.

[34] R. Collobert and J. Weston. A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[35] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20 :273–297, 1995. 10.1007/BF00994018.

[36] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[37] L. Cruzeiro-Hansson and P. Silva. Protein folding : thermodynamic versus kinetic control. *J. Biol. Phys*, 27 :S6–S9, 2001.

[38] X. Deng, J. Eickholt, and J. Cheng. Predisorder : ab initio sequence-based prediction of protein disordered regions. *BMC bioinformatics*, 10(1) :436, 2009.

[39] X. Deng, J. Eickholt, and J. Cheng. A comprehensive overview of computational protein disorder prediction methods. *Molecular BioSystems*, 8(1) :114–121, 2012.

[40] I. Dondoshansky and Y. Wolf. Blastclust (ncbi software development toolkit). *NCBI, Bethesda, Md*, 2002.

[41] Z. Dosztanyi, V. Csizmok, P. Tompa, and I. Simon. The pairwise energy content estimated from amino acid composition discriminates between folded and intrinsically unstructured proteins. *Journal of molecular biology*, 347(4) :827–839, 2005.

[42] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann Publishers, Inc., 1995.

[43] J. Duarte, R. Sathyapriya, H. Stehr, I. Filippis, and M. Lappe. Optimal contact definition for reconstruction of contact maps. *BMC Bioinformatics*, 11(1) :283, 2010.

[44] W. Duch. Filter methods. *Feature Extraction*, pages 89–117, 2006.

[45] A. K. Dunker and V. N. Uversky. Drugs for 'protein clouds' : targeting intrinsically disordered transcription factors. *Current opinion in pharmacology*, 10(6) :782–788, 2010.

[46] H. J. Dyson and P. E. Wright. Intrinsically unstructured proteins and their functions. *Nature Reviews Molecular Cell Biology*, 6(3) :197–208, 2005.

[47] J. Eickholt and J. Cheng. Dndisorder : predicting protein disorder using boosting and deep networks. *BMC Bioinformatics*, 14(1) :88, 2013.

[48] P. Elumalai, J. Wu, and H. Liu. Current advances in disulfide connectivity predictions. *Journal of the Taiwan Institute of Chemical Engineers*, 41(5) :525 – 539, 2010.

[49] E. Faraggi, B. Xue, and Y. Zhou. Improving the prediction accuracy of residue solvent accessibility and real-value backbone torsion angles of proteins by guided-learning through a two-layer neural network. *Proteins : Structure, Function, and Bioinformatics*, 74(4) :847–856, 2008.

[50] E. Faraggi, Y. Yang, S. Zhang, and Y. Zhou. Predicting continuous local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. *Structure*, 17(11) :1515–1527, 2009.

[51] P. Fariselli and R. Casadio. Prediction of disulfide connectivity in proteins. *Bioinformatics*, 17(10) :957–964, 2001.

[52] P. Fariselli, P. Riccobelli, and R. Casadio. Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins. *Proteins : Structure, Function, and Bioinformatics*, 36(3) :340–346, 1999.

[53] F. Ferrè and P. Clote. Disulfide connectivity prediction using secondary structure information and diresidue frequencies. *Bioinformatics*, 21(10) :2336–2346, 2005.

[54] A. Fiser, M. Cserzö, E. Tüdös, and I. Simon. Different sequence environments of cysteines and half cystines in proteins application to predict disulfide forming residues. *FEBS Letters*, 302(2) :117 – 120, 1992.

[55] P. Frasconi, A. Passerini, and A. Vullo. A two-stage svm architecture for predicting the disulfide bonding state of cysteines. *Neural Networks for Signal Processing*, pages 25 – 34, 2002.

[56] H. Gabow. An efficient implementation of edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23 :221–234, April 1976.

[57] A. Garg, H. Kaur, and G. Raghava. Real value prediction of solvent accessibility in proteins using multiple sequence alignment and secondary structure. *PROTEINS : Structure, Function, and Bioinformatics*, 61(2) :318–324, 2005.

[58] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63 :3–42, 2006. 10.1007/s10994-006-6226-1.

[59] S. Govindarajan and R. Goldstein. On the thermodynamic hypothesis of protein folding. *Proceedings of the National Academy of Sciences*, 95(10) :5545–5549, 1998.

[60] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning : data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2) :83–85, 2005.

[61] S. Hirose, K. Shimizu, S. Kanai, Y. Kuroda, and T. Noguchi. Poodle-l : a two-level svm prediction system for reliably predicting long disordered regions. *Bioinformatics*, 23(16) :2046–2053, 2007.

[62] L. Hyafil and R. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1) :15–17, 1976.

[63] T. Ishida and K. Kinoshita. Prediction of disordered regions in proteins based on the meta approach. *Bioinformatics*, 24(11) :1344–1348, 2008.

[64] D. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2) :195–202, 1999.

[65] D. Jones and J. Ward. Prediction of disordered regions in proteins from position specific score matrices. *Proteins : Structure, Function, and Bioinformatics*, 53(S6) :573–578, 2003.

[66] K. Joo, S. Lee, and J. Lee. Sann : Solvent accessibility prediction of proteins by nearest neighbor method. *Proteins : Structure, Function, and Bioinformatics*, 2012.

[67] W. Kabsch and C. Sander. Dictionary of protein secondary structure : pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12) :2577–2637, 1983.

[68] H. Kim and H. Park. Prediction of protein relative solvent accessibility with support vector machines and long-range interaction 3d local descriptor. *Proteins : Structure, Function, and Bioinformatics*, 54(3) :557–562, 2003.

[69] H. Kim and H. Park. Protein secondary structure prediction based on an improved support vector machines approach. *Protein Engineering*, 16(8) :553–560, 2003.

[70] T. Klink, K. Woycechowsky, K. Taylor, and R. Raines. Contribution of disulfide bonds to the conformational stability and catalytic activity of ribonuclease a. *European Journal of Biochemistry*, 267(2) :566–572, 2000.

[71] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1) :273–324, 1997.

[72] V. Kolmogorov. Blossom v : A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1) :43–67, 2009.

[73] L. Kozlowski and J. Bujnicki. Metadisorder : a meta-server for the prediction of intrinsic disorder in proteins. *BMC bioinformatics*, 13(1) :111, 2012.

[74] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[75] T. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. *Feature Extraction*, pages 137–165, 2006.

[76] J. Lee, S. Wu, and Y. Zhang. Ab initio protein structure prediction. In *From protein structure to function with bioinformatics*, pages 3–25. Springer, 2009.

[77] C. Levinthal. Are there pathways for protein folding ? *Journal of Medical Physics*, 65(1) :44–45, 1968.

[78] H. Lin and L. Tseng. Prediction of disulfide bonding pattern based on support vector machine with parameters tuned by multiple trajectory search. *WSEAS Transactions on Computers*, 8 :1429–1439, September 2009.

[79] H. Lin and L. Tseng. Dbcp : a web server for disulfide bonding connectivity pattern prediction without the prior knowledge of the bonding state of cysteines. *Nucleic Acids Research*, 38(suppl 2) :W503–W507, 2010.

[80] H.-H. Lin and L.-Y. Tseng. Prediction of disulfide bonding pattern based on a support vector machine and multiple trajectory search. *Information Sciences*, 199(0) :167 – 178, 2012.

[81] L. Lin, S. Yang, and R. Zuo. Protein secondary structure prediction based on multi-svm ensemble. In *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pages 356–358. IEEE, 2010.

[82] J. Liu, H. Tan, B. Rost, et al. Loopy proteins appear conserved in evolution. *Journal of molecular biology*, 322(1) :53–64, 2002.

[83] C. Lu, Y. Chen, C. Yu, and J. Hwang. Predicting disulfide connectivity patterns. *Proteins : Structure, Function, and Bioinformatics*, 67(2) :262–270, 2007.

[84] F. Maes, J. Becker, and L. Wehenkel. Iterative multi-task sequence labeling for predicting structural properties of proteins. *17th European Symposium on Artificial Neural Networks*, 2011.

[85] F. Maes, J. Becker, and L. Wehenkel. Prédiction structurée multitâche itérative de propriétés structurelles de protéines. *7e Plateforme AFIA : Association Française pour l'Intelligence Artificielle, Chambéry, 16 au 20 mai 2011*, page 279, 2011.

[86] F. Maes, S. Peters, L. Denoyer, and P. Gallinari. Simulated iterative classification : A new learning procedure for graph labeling. In *European Conference on Machine Learning*, 2009.

[87] V. Marc, P. Andrea, L. Matthieu, and F. Paolo. A simplified approach to disulfide connectivity prediction from protein sequences. *BMC Bioinformatics*, 9(1) :20, 2008.

[88] P. Martelli, P. Fariselli, and R. Casadio. Prediction of disulfide-bonded cysteines in proteomes with a hidden neural network. *Proteomics*, 4(6) :1665–1671, 2004.

[89] M. Matsumura, G. Signor, and B. Matthews. Substantial increase of protein stability by multiple disulphide bonds. *Nature*, 342 :291–293, 1989.

[90] J. Mercer. Functions of positive and negative type and their connection with the theory of intregral equations. In *Philos. Trans. Roy. Soc. London*, volume A209, pages 415–446, 1909.

[91] M. Mizianty, W. Stach, K. Chen, K. Kedarisetti, F. Disfani, and L. Kurgan. Improved sequence-based prediction of disordered regions with multilayer fusion of multiple information sources. *Bioinformatics*, 26(18) :i489–i496, 2010.

[92] B. Monastyrskyy, K. Fidelis, J. Moult, A. Tramontano, and A. Kryshtafovych. Evaluation of disorder predictions in casp9. *Proteins : Structure, Function, and Bioinformatics*, 79(S10) :107–118, 2011.

[93] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. Scop : a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4) :536–540, 1995.

[94] O. Noivirt-Brik, J. Prilusky, and J. Sussman. Assessment of disorder predictions in casp8. *Proteins*, 77 Suppl 9 :210–6, 2009.

[95] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, and J. Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8) :1093–1109, 1997.

[96] M. Ouali and R. King. Cascaded multiple classifiers for secondary structure prediction. *Protein Science*, 9(06) :1162–1176, 2000.

[97] K. Peng, S. Vucetic, P. Radivojac, J. CELESTE, A. Dunker, and Z. Obradovic. Optimizing long intrinsic disorder predictors with protein evolutionary information. *Journal of bioinformatics and computational biology*, 3(01) :35–60, 2005.

[98] W. Pirovano and J. Heringa. Protein secondary structure prediction. *Methods Mol. Biol*, 609 :327–348, 2010.

[99] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press, 1999.

[100] G. Pollastri and A. Mclysaght. Porter : a new, accurate server for protein secondary structure prediction. *Bioinformatics*, 21(8) :1719–1720, 2005.

[101] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins : Structure, Function, and Bioinformatics*, 47(2) :228–235, 2002.

[102] K. Pruitt, T. Tatusova, G. Brown, and D. Maglott. Ncbi reference sequences (refseq) : current status, new features and genome annotation policy. *Nucleic Acids Research*, 40(D1) :D130–D135, 2012.

[103] Y. Qi, M. Oja, J. Weston, and W. Noble. A unified multitask architecture for predicting local protein properties. *PloS one*, 7(3) :e32235, 2012.

[104] N. Qian and T. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of molecular biology*, 202(4) :865–884, 1988.

[105] G. Raghava. Protein secondary structure prediction using nearest neighbor and neural network approach. *CASP*, 4 :75–76, 2000.

[106] G. Rama, A. Shilton, M. Parker, and M. Palaniswami. Disulphide bridge prediction using fuzzy support vector machines. *International Conference on Intelligent Sensing and Information Processing*, 0 :48–54, 2005.

[107] G. Rose, A. Geselowitz, G. Lesser, R. Lee, and M. Zehfus. Hydrophobicity of amino acid residues in globular proteins. *Science*, 229(4716) :834–838, 1985.

[108] B. Rost. Review : protein secondary structure prediction continues to rise. *Journal of structural biology*, 134(2-3) :204–218, 2001.

[109] B. Rost and C. Sander. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins : Structure, Function, and Bioinformatics*, 19(1) :55–72, 1994.

[110] B. Rost and C. Sander. Conservation and prediction of solvent accessibility in protein families. *Proteins : Structure, Function, and Bioinformatics*, 20(3) :216–226, 2004.

[111] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19) :2507–2517, Sept. 2007.

[112] J. Sanchez-Ruiz. Protein kinetic stability. *Biophysical Chemistry*, 148(1–3) :1 – 15, 2010.

[113] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins : Structure, Function, and Bioinformatics*, 9(1) :56–68, 1991.

[114] C. Savojardo, P. Fariselli, M. Alhamdoosh, P. L. Martelli, A. Pierleoni, and R. Casadio. Improving the prediction of disulfide bonds in eukaryotes with machine learning methods and protein subcellular localization. *Bioinformatics*, 27(16) :2224–2230, 2011.

[115] A. Schlessinger, J. Liu, and B. Rost. Natively unstructured loops differ from other loops. *PLoS computational biology*, 3(7) :e140, 2007.

[116] A. Schlessinger, C. Schaefer, E. Vicedo, M. Schmidberger, M. Punta, and B. Rost. Protein disorder—a breakthrough invention of evolution ? *Current opinion in structural biology*, 21(3) :412–418, 2011.

[117] S. Schmidler, J. Liu, and D. Brutlag. Bayesian segmentation of protein secondary structure. *Journal of computational biology*, 7(1-2) :233–248, 2000.

[118] K. Shimizu, S. Hirose, and T. Noguchi. Poodle-s : web application for predicting protein disorder by using physicochemical features and reduced amino acid set of a position-specific scoring matrix. *Bioinformatics*, 23(17) :2337–2338, 2007.

[119] K. Shimizu, Y. Muraoka, S. Hirose, K. Tomii, and T. Noguchi. Predicting mostly disordered proteins by using structure-unknown protein data. *BMC bioinformatics*, 8(1) :78, 2007.

[120] J. Sim, S. Kim, and J. Lee. Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method. *Bioinformatics*, 21(12) :2844–2849, 2005.

[121] J. Söding and M. Remmert. Protein sequence comparison and fold recognition : progress and good-practice benchmarking. *Current opinion in structural biology*, 21(3) :404–411, 2011.

[122] J. Song, Z. Yuan, H. Tan, T. Huber, and K. Burrage. Predicting disulfide connectivity from protein sequence using multiple sequence feature vectors and secondary structure. *Bioinformatics*, 23(23) :3147–3154, 2007.

[123] M. Sven and R. Burkhard. Uniqueprot : creating representative protein sequence sets. *Nucleic Acids Res*, pages 3789–3791, 2003.

[124] The UniProt Consortium. Ongoing and future developments at the universal protein resource. *Nucleic Acids Research*, 39(suppl 1) :D214–D219, 2011.

[125] R. Tisch and H. McDevitt. Insulin-dependent diabetes mellitus : review. *Cell*, 85 :291–7, 1996.

[126] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

[127] V. N. Uversky. The mysterious unfoldome : structureless, underappreciated, yet vital part of any given proteome. *BioMed Research International*, 2010, 2009.

[128] V. N. Uversky, C. J. Oldfield, and A. K. Dunker. Showing your id : intrinsic disorder as an id for recognition, regulation and cell signaling. *Journal of Molecular Recognition*, 18(5) :343–384, 2005.

[129] V. Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5) :988–999, 1999.

[130] A. Vullo, O. Bortolami, G. Pollastri, and S. Tosatto. Spritz : a server for the prediction of intrinsically disordered regions in protein sequences using kernel machines. *Nucleic Acids Research*, 34(suppl 2) :W164–W168, 2006.

[131] A. Vullo and P. Frasconi. Disulfide connectivity prediction using recursive neural networks and evolutionary information. *Bioinformatics*, 20(5) :653–659, 2004.

[132] I. Walsh, A. J. Martin, T. Di Domenico, and S. C. Tosatto. Espritz : accurate and fast prediction of protein disorder. *Bioinformatics*, 28(4) :503–509, 2012.

[133] L. Wang and U. H. Sauer. Ond-crf : predicting order and disorder in proteins conditional random fields. *Bioinformatics*, 24(11) :1401–1402, 2008.

[134] J. Ward, L. McGuffin, B. Buxton, and D. Jones. Secondary structure prediction with support vector machines. *Bioinformatics*, 19(13) :1650–1655, 2003.

[135] J. J. Ward, J. S. Sodhi, L. J. McGuffin, B. F. Buxton, and D. T. Jones. Prediction and functional analysis of native disorder in proteins from the three kingdoms of life. *Journal of molecular biology*, 337(3) :635–645, 2004.

[136] W. Wedemeyer, E. Welker, M. Narayan, and H. Scheraga. Disulfide bonds and protein folding. *Biochemistry*, 39(15) :4207–4216, 2000.

[137] J. Wei, E. Zaika, and A. Zaika. p53 family : Role of protein isoforms in human cancer. *Journal of Nucleic Acids*, 2012.

[138] J. C. Wootton. Non-globular domains in protein sequences : automated segmentation using complexity measures. *Computers & chemistry*, 18(3) :269–285, 1994.

[139] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5 :975–1005, Dec. 2004.

[140] B. Xue, R. L. Dunbrack, R. W. Williams, A. K. Dunker, and V. N. Uversky. Pondr-fit : a meta-predictor of intrinsically disordered amino acids. *Biochimica et Biophysica Acta (BBA)-Proteins & Proteomics*, 1804(4) :996–1010, 2010.

[141] Z. Yang, R. Thomson, P. McNeil, and R. Esnouf. Ronn : the bio-basis function neural network technique applied to the detection of natively disordered regions in proteins. *Bioinformatics*, 21(16) :3369–3376, 2005.

[142] Z. Yuan, K. Burrage, and J. Mattick. Prediction of protein solvent accessibility using support vector machines. *Proteins : Structure, Function, and Bioinformatics*, 48(3) :566–570, 2002.

[143] Z. Yuan and B. Huang. Prediction of protein accessible surface areas by support vector regression. *Proteins : Structure, Function, and Bioinformatics*, 57(3) :558–564, 2004.

[144] H. Zhang, T. Zhang, K. Chen, K. Kedarisetti, M. Mizianty, Q. Bao, W. Stach, and L. Kurgan. Critical assessment of high-throughput standalone methods for secondary structure prediction. *Briefings in bioinformatics*, 12(6) :672–688, 2011.

[145] H. Zhang, T. Zhang, K. Chen, S. Shen, J. Ruan, and L. Kurgan. Sequence based residue depth prediction using evolutionary information and predicted secondary structure. *BMC bioinformatics*, 2008.

[146] T. Zhang, E. Faraggi, B. Xue, A. Dunker, V. Uversky, and Y. Zhou. Spine-d : accurate prediction of short and long disordered regions by a single neural-network based method. *Journal of Biomolecular Structure and Dynamics*, 29(4) :799–813, 2012.

[147] E. Zhao, H. Liu, C. Tsai, H. Tsai, C. Chan, and C. Kao. Cysteine separations profiles on protein sequences infer disulfide connectivity. *Bioinformatics*, 21(8) :1415–1420, 2004.