Alma Mater Studiorum - Univeristà di Bologna

DOTTORATO DI RICERCA IN

INFORMATICA

Ciclo XXV

Settore Concorsuale di afferenza: 01/B1 - Informatica
Settore Scientifico disciplinare: INF/01 - Informatica

TITOLO TESI

MACHINE-LEARNING METHODS FOR STRUCTURE PREDICTION OF β-BARREL MEMBRANE PROTEINS

Presentata da: Castrense Savojardo

Coordinatore Dottorato:
Prof. Maurizio Gabbrielli

Relatore:
Dott. Piero Fariselli

Esame finale anno 2013

Abstract

Different types of proteins exist with diverse functions that are essential for living organisms. An important class of proteins is represented by transmembrane proteins which are specifically designed to be inserted into biological membranes and devised to perform very important functions in the cell such as cell communication and active transport across the membrane. Transmembrane β-barrels (TMBBs) are a sub-class of membrane proteins largely under-represented in structure databases because of the extreme difficulty in experimental structure determination. For this reason, computational tools that are able to predict the structure of TMBBs are needed. In this thesis, two computational problems related to TMBBs were addressed: the detection of TMBBs in large datasets of proteins and the prediction of the topology of TMBB proteins. Firstly, a method for TMBB detection was presented based on a novel neural network framework for variable-length sequence classification. The proposed approach was validated on a non-redundant dataset of proteins. Furthermore, we carried-out genome-wide detection using the entire *Escherichia coli* proteome. In both experiments, the method significantly outperformed other existing state-of-the-art approaches, reaching very high PPV (92%) and MCC (0.82). Secondly, a method was also introduced for TMBB topology prediction. The proposed approach is based on grammatical modelling and probabilistic discriminative models for sequence data labeling. The method was evaluated using a newly generated dataset of 38 TMBB proteins obtained from high-resolution data in the PDB. Results have shown that the model is able to correctly predict topologies of 25 out of 38 protein chains in the dataset. When tested on previously released datasets, the performances of the proposed approach were measured as comparable or superior to the current state-of-the-art of TMBB topology prediction.

Acknowledgements

First of all, I would like to thank my advisor Dott. Piero Fariselli for his invaluable scientific support and guidance during the last five years I spent working with him.

A special thanks goes to Prof. Rita Casadio for giving me the opportunity to work at Bologna Biocompunting Group and for her precious contribution to my work and wise guidance.

I would also like to thank my tutor Prof. Lorenzo Donatiello, for being part of my supervisory committee and for its valuable advices and help.

During these years at Biocompunting Group, I worked with an extraordinary reasearch team of whose immense scientific value I extremely benefited. Thanks go to Dott. Pier Luigi Martelli, Dott. Pietro Di Lena, Dott. Gianluca Tasco, Dott. Lisa Bartoli, Dott. Marco Vassura, Dott. Raffaele Fronza, Dott. Andrea Pierleoni, Dott. Ivan Rossi, Damiano Piovesan, Valentina Indio, Giuseppe Profiti and all other colleagues and co-authors I had the fortune to work with.

Thanks go to Prof. Pierre Baldi and Prof. David Jones for having accepted to serve as external referees of this thesis and for their sharp comments on this work.

During my Ph.D., I spent six months abroad at the Bioinformatics Centre (Binf), Copenhagen University. I would like to thank the director of the Binf, Prof. Anders Krogh for giving me the opportunity to join his lab and for introducing me to very interesting and exciting research issues in the field of Bioinformatics. It was an honour to work with him and his group.

I would like to express my thankfulness to the current and former Ph.D. program coordinators, respectively, Prof. Maurizio Gabbrielli and Prof. Simone Martini.

An immense gratitude goes to Maguy, for supporting me during the inevitable

frustations which naturally accompany a difficult journey as a Ph.D. program is.

Finally, thanks to my family and all friends I met during these years in Bologna and Copenhagen.

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Protein folding is a physical process by which the unstructured linear chain of amino acids of a protein assumes its three-dimensional structure. The structure or *conformation* of a protein has been selected by evolution to accomplish its specific task or function. In this perspesctive, the knowledge of the protein structure represents a mandatory step to acquire essential information about its function.

The vast majority of proteins are known at the sequence level lacking of a characterization at the structural and functional levels. With the emergence of new technologies that allow massive genome sequencing, a tremendous amount of sequence data has become available. However, in spite of significant improvements in experimental methods for protein structure determination, the number of proteins whose structures are known at atomic level still occupies a minimal fraction of known sequences (this fraction is estimated around the 1%).

The design of computational methods for protein structure prediction can be seen as an endevour to fill this gap. What make this effort worthwhile is the observation, enclosed in the celebrated Anfinsen's thermodynamic hypothesis, that the information about the three-dimensional structure of a protein is completely encoded in its linear sequence of amino acids (under physiological conditions).

Proteins are macromolecular compounds which are involved in almost every biological process. Different types of proteins exist with diverse functions that are essential for living organisms. *Enzymes* are proteins whose specific function is to catalyze biochemical reactions and they are fundamental in metabolism. *Structural proteins* are the building blocks that constitute the structural components of the cell. Other proteins are essential as transport systems (e.g. haemoglobin), in the immune response, gene transcription and regulation and many others vital functions.

An important class of proteins is represented by *transmembrane proteins*. These proteins are specifically designed to be inserted into biological membranes and they are devised to perform very important functions in the cell such as communication with the external environment, signal transduction (e.g. G protein-coupled receptors) and active transport across the membrane (e.g. ion channels). Therefore, transmembrane proteins have a great pharmaceutical importance and they are among the most common drug targets.

Transmembrane proteins typically assume very specific conformations strongly related to their functions. Two different classes can be identified according to their fold: α-helical bundles and β-barrel transmembrane proteins. α-helical proteins are relatively more abundant than β-barrels in structure databases. For this reason, computational tools that are able to detect and to predict the structure of β-barrels are needed.

In the field of structure prediction of β-barrel proteins, several computational problems are typically addressed. A first task consists in the discrimination between β-barrels and other types of proteins such as globular proteins or α-helical bundles. The ability to detect β-barrels by computational methods can help in identifying potential target proteins on which further computational or experimental validation can be performed. This problem is challenging because of the cryptic nature of β-barrels which make them difficult to be detected by simply analysing amino acid compositions. For this reason, supervised machine learning methods are typically adopted in order to capture non-trivial relationships in the sequence to improve the classification accuracy. In the most common setting, these methods are used to scan datasets of protein sequences typically as large as an entire proteome.

Another important computational problem related to β-barrels is the prediction of the protein *topology*, namely the identification along the sequence of transmembrane segments and their position and orientation with respect to the membrane. The knowledge of the protein topology can be used to define a first coarse model of the protein structure. Essentially, topology prediction is a sequence labelling problem where the protein sequence is segmented in contiguous regions that correspond to the three different locations of amino acids with respect to the membrane: intra-cellular side, membrane spanning and extra-cellular side. In this task, ma-

chine learning methods are extensively applied and the best performing approaches are based on probabilistic methods for sequence analysis such as Hidden Markov Models.

Main contributions

In this thesis, methods for β-barrel detection and topology prediction based on machine learning are presented. The main contributions of this work are both applicative and methodological and can be summarized as follows:

- a method for large-scale β-barrel detection based on a novel machine learning framework for variable-length sequence classification. This framework, called N-to-1 Extreme Learning Machine, is based on the combination of two recent developments in the field of Artificial Neural Networks: N-to-1 neural networks for structured input classification and the Extreme Learning Machine, a recently introduced algorithm for training feed-forward neural networks. The proposed method for β-barrel detection significantly outperforms other existing approaches, reaching a high accuracy and a very low false positive rate;

- a method for β-barrel topology prediction based on a new probabilistic framework called Grammatical-Restrained Hidden Conditional Random Field (GRHCRF) devised to address prediction tasks which typically arise in Computational Biology and Bioinformatics. The topology predictor described in this thesis has been validated using different datasets of experimentally determined β-barrels and compared to other methods developed for the same task. The performance of the proposed method has been measured as comparable with the current state-of-the-art of β-barrel topology prediction;

- although tested on the two tasks described above, the machine learning models developed in this thesis are quite general and not limited to these specific applications. In particular, the GRHCRF framework can be used to face other annotation tasks in biological sequence analisys such as protein secondary structure prediction, coiled-coil prediction and signal/target peptide prediction.

Structure of this document

This thesis is structured as follows:

- **Chapter 1** provides a background on proteins and an overview of the main approaches to protein structure prediction.
- **Chapter 2** provides a general background on transmembrane proteins with a focus on the sub-class of β-barrels. A literature review of the main methods previously developed for β-barrel detection and topology prediction is provided.
- In **Chapter 3**, probabilistic models for sequence analisys are discussed, particularly focusing on well-established frameworks used in Computational Biology to analyse biological sequences such as Hidden Markov Models and Conditional Random Fields. This chapter serves as a background for the next chapter.
- **Chapter 4** describes in details the Grammatical-Restrained Hidden Conditional Random Field framework. In this chapter the probabilistic framework is presented in general terms by discussing parameter estimation and inference algorithms and the main advantages with respect to other frameworks such a HMMs and linear-chain CRFs.
- **Chapter 5** describes all the aspects of the N-to-1 Extreme Learning Machine framework for variable-length sequence classification. In the first part of the chapter the basic concepts of feed-forward neural networks are introduced. In the second part is presented the combination between the N-to-1 NN architecture and the Extreme Learning Machine algorithm.
- In **Chapter 6** the method for β-barrel detection based on N-to-1 ELM is presented. Data, results and comparison with other approches are discussed in detail.
- In **Chapter 7** the application of the GRHCRF framework to β-barrel topology prediction is described and experimental results presented.

Chapter 1

Structural bioinformatics

1.1 Proteins: sequence and structure

From a chemical point of view, proteins are variable-length chains made up by fundamental subunits called *amino acids*, held togheter by covalent or *peptide* bonds. These chains are arranged to form a three-dimensional structure usually referred to as the *native conformation* that is always associated to a specific function of the protein. In a polar solvent, proteins spontaneously fold into this stable and active structure. The process by which the unstructured chain of amino acids acquires its specific biologically functional three-dimensional structure is called *protein folding*. The Anfinsen's thermodynamic hypothesis states that all the information about the native conformation of a protein can be obtained from its amino acid sequence [4]. However, although this observation hypothesizes the existence in nature of an algorithm to determine protein structure from sequence, the full understanding of the folding process still remains an open problem in Molecular Biology.

Biologist distinguish four different levels of organization in the structure of a protein [1]:

1. the *primary structure* is simply the linear sequence of amino acids;

2. the protein *secondary structure* consists in streches of the chain which are locally organized in regular sub-structures or *motifs* (i.e. α-helices or β-sheets);

3. the *tertiary structure* is the full three-dimensional organization of the protein;

Name	Abbreviation	Name	Abbreviation
Alanine	A	Leucine	L
Arginine	R	Lysine	K
Asparagine	N	Methionine	M
Aspartic acid	D	Phenylalanine	F
Cysteine	C	Proline	P
Glutamine	Q	Serine	S
Glutamic acid	E	Threonine	T
Glycine	G	Tryptophan	W
Histidine	H	Tyrosine	Y
Isoleucine	I	Valine	V

Table 1.1: List of the 20 different amino acids and corresponding abbreviations.

4. the *quaternary structure* refers to the complete structure of a protein complex formed by two or more protein chains.

Distinct from these four, of central importance is also another unit of organization called *protein domain*, a sub-structure obtained by a part of a single protein chain which independently folds into a compact and stable three-dimensional structure. The different domains of a protein are often associated with different functions [1].

Secondary structure elements can be also organized in *supersecondary structures*, namely compact three-dimensional structures which are typically smaller than protein domains. In the above hierarchy, these structures can be placed between the secondary and the tertiary structure.

Some proteins can also slightly change their structures while performing their biological function. These modifications are usually referred to as *conformational changes*.

1.1.1 The amino acid sequence

There are 20 different types of amino acids in proteins which differ in chemical properties. In Table 1.1 is reported the list of all amino acids with the corresponding single-letter abbreviations.

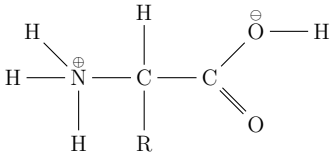


Figure 1.1: Chemical structure of a generic amino acid.

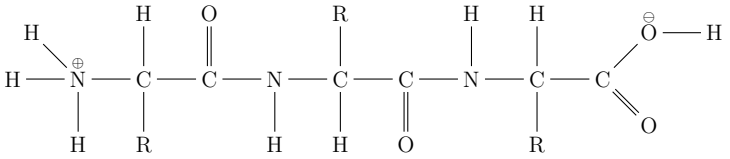


Figure 1.2: A protein chain.

A protein is made up of a chain of these amino acids linked togheter with covalent peptide bonds. The chemical structure of a single generic amino acid is shown in Figure 1.1.

The central carbon atom (typically referred to as the C_α atom) is bonded to an amino group, a carboxyl group, an hydrogen atom and a *side chain* or *residue* (represented with R in figure). All amino acids share a common portion of the molecule which is involved in forming peptide bonds. This repeating sequence of atoms along the core of a protein chain is reffered to as the *backbone*. Attached to the backbone are side chains which are not involved in peptide bonds and uniquely characterize the chemistry of each amino acid. In Figure 1.2 a polypeptide chain of length 3 is shown.

The peptide bond is formed between the carboxyl and the amino groups of two consecutive amino acids. The two ends of the chain are referred to as the N-terminal end or amino terminus (N-terminus) and the C-terminal end or carboxyl terminus (C-terminus) based on the nature of the free group at each extremity. By convention, protein sequences are always written specifying the linear sequence of its residues starting from the N-terminus to the C-terminus.

According to the *central dogma of Molecular Biology*, proteins are the final products of the expression of *genes*. Amino acids are synthesized directly by the process of *translation* from messenger RNA (mRNA) which is, in turn, obtained by *transcription* from the genomic DNA. The translation of the mRNA in the amino acid

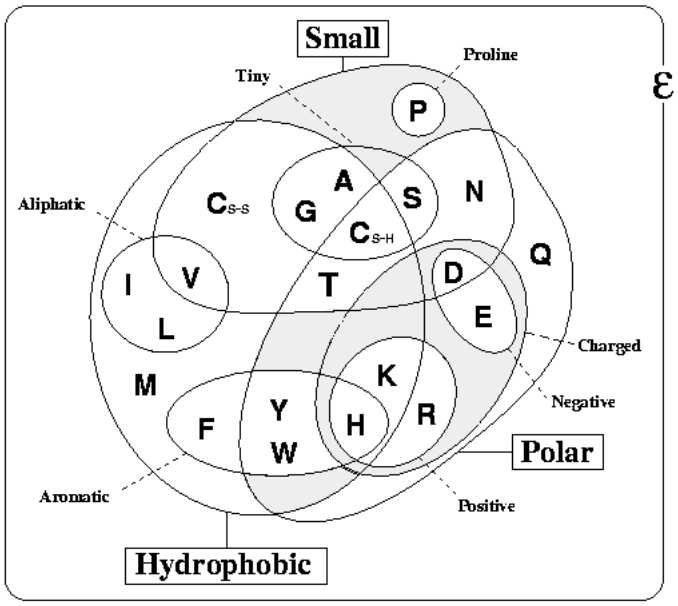


Figure 1.3: Venn diagram grouping amino acids according to physicochemical properties [71].

sequence is performed according to the *genetic code*. By this, sequences of three nucleotides in the mRNA are translated into a single amino acid during protein synthesis. This process is mediated by a RNA-protein complex called *ribosome* [24].

Amino acids have different chemical characteristics. Figure 1.3, adapted from [71], shows a Venn diagram which groups amino acids according to their different physicochemical properties.

1.1.2 The secondary structure

Although the conformation of a protein is unique, by analysing the three-dimensional structures of many proteins it can be noted that two regular folding patterns are often found in part of them. These regular sub-structures are called α-helices and β-sheets and have been discovered more than 50 years ago [1].

An α-helix is formed when a segment of protein chain twists around itself forming a rigid cylinder. To mantain this local arrangement an additional chemical bond is formed every four peptide bonds, linking the C=O of one bond to the N-H of another (see Figure 1.1). This give rise to a helix with a complete turn every 3.6 amino acids,

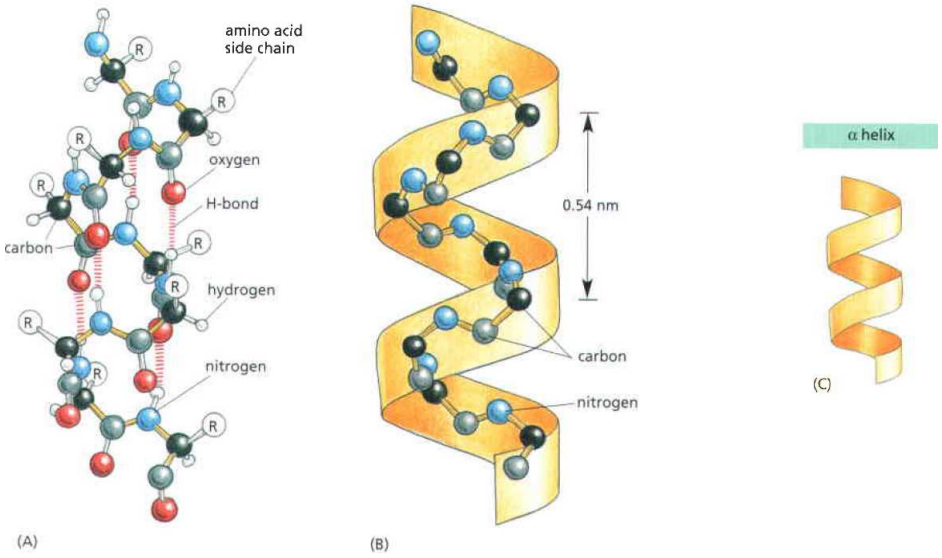


Figure 1.4: The structure of an α-helix [1].

as shown in Figure 1.4 (image taken from [1]).

β-sheets are rigid structures formed by spatially adjacent streches of a polypeptide chain [1]. Given the directionality of a protein chain determined by its N and C termini, two streches can run in opposite directions (antiparallel) or toward the same direction (parallel). In general a β-sheet can assume different structural motifs on the basis of the relative orientation of its strands. In all cases, hydrogen bonds are formed between peptide bonds giving rise to a rigid structure, as shown in Figure 1.5. The side chains of amino acid involved in a β-sheet alternatively point up and down with respect to the plane of the sheet.

1.1.3 The tertiary structure

The folding process is driven by many different forces which togheter come into play in determining the final native conformation of the protein. Several constraints such as the requirements that no two atoms overlap, greatly reduce the possible bond angles in a polypeptide chain. These constraints as well as other steric interactions can significantly reduce the space of possible conformations of the protein. However,

[1] β-sheets can be also formed by streches coming from different polypeptide chains.

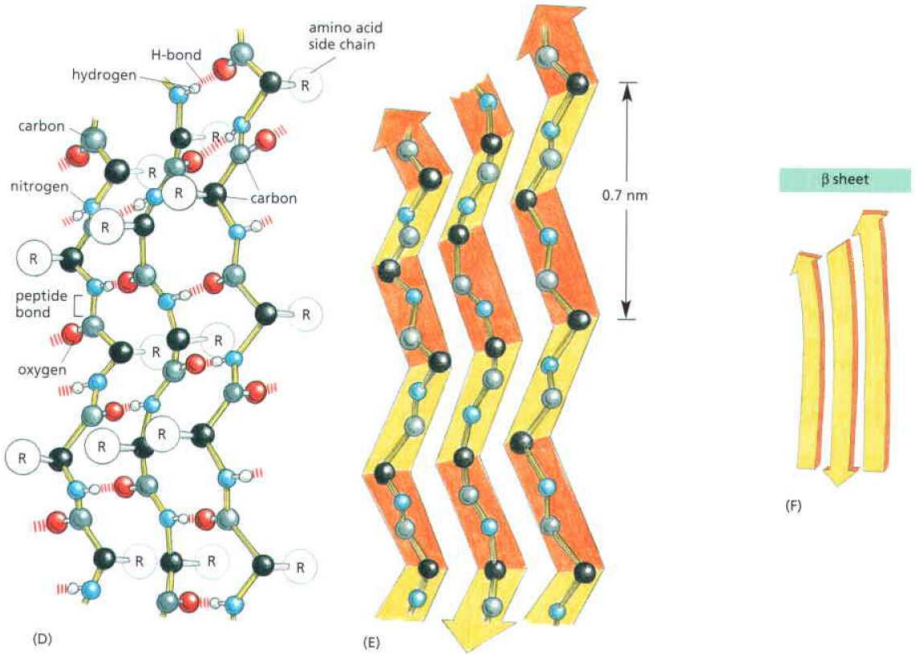


Figure 1.5: The structure of a β-sheet [1].

this space still remain huge, especially for long chains which can still fold in an enormous number of ways.

Other forces are also involved in the folding process such as hydrogen bonds, electrostatic intereactions and van der Waals attractions. Furthemore, also the interactions with the solvent play a key role in the folding, affecting the distribution of its polar and non-polar amino acids. In particular, nonpolar amino acids (which are hydrophobic) tend to cluster in the interior of the molecule whereas polar amino acids tend to occupy positions on the surface of the protein.

As a final and cumulative result of all these interactions proteins have a particular three-dimensional structure which is determined by the order of their amino acids. The native conformation is generally the one that minimizes its *free energy*. This is a consequece of the second law of thermodynamics which states that a system at constant temperature and pressure find an equilibrium state to give a minimum *Gibbs free energy*:

$$G = H - TS \tag{1.1}$$

where H is the enthalpy, S is the entropy and T is the absolute temperature [60].

1.2 Protein structure prediction

The structure of a protein can be experimentally determined. Two techniques are mainly used for this purpose:

- *X-ray crystallography*: the protein must be purified (i.e. isolated from the complex mixture in which it is naturally immerged) and crystallized. Beams of x-rays are passed through the crystal and atoms in the protein scatter the x-rays which produce a diffraction pattern in a photographic film. This method is applicable without any restriction on the protein length.

- *Nuclear Magnetic Resonance (NMR) spectroscopy*: a solution of the protein is inserted in a magnetic field. The effects of the radio frequencies on the resonance of different atoms is measured. This technique can be applied only to small and soluble proteins.

Although by means of experimental methods it is possible to obtain information about the protein structure at atomic level, they are still very expensive both in terms of time and costs and their applicability cannot be extended to all possible proteins. For instance, NMR spectroscopy cannot be applied to long proteins (> 120 residues) and X-ray crystallography of entire classes of protein (e.g. membrane proteins) is still a challenge [75]. In spite of the efforts in structural genomics projects, experimental protein structure determination still remains a bottleneck, especially considering the massive amount of sequence data produced by modern DNA sequencing technologies [107].

For these reasons, prediction of protein structure from sequence using computational methods has emerged in the past decades as an interdisciplinary research field bringing together efforts from Biology, Physics and Computer Science.

Traditionally, computational approaches to protein structure prediction are classified in two different categories [66]:

- *ab-initio* or *de-novo* methods which try to predict protein tertiary structure using only basic physical principles to reproduce the inter-atomic interactions;

- *comparative protein modelling* involve the adoption of knowledge-based methods which exploit the possible similarity of a target protein sequence to known structures in order to build a reasonably good model of the tertiary structure.

In general, whenever it is possible to identify for a terget protein similar sequences of known structures in structural databases, comparative protein modelling are likely to produce a sufficiently good three-dimensional model and should be preferred to ab-initio methods. However, it is not always possible to find sequences that are similar enough to build a model. In such cases, ab-initio methods are the only choice.

1.2.1 Ab-initio protein modelling

In ab-initio methods, the conformational space of a target protein sequence is searched in order to find a suitable native conformation. The two main approaches in ab-initio protein structure prediction are *Molecular Dynamics* and *Markov chain Monte Carlo*(MCMC) simulations.

1.2.1.1 Molecular Dynamics simulations

Molecular Dynamics(MD) aims at simulating the actual physical trajectories at atomic level in a biological molecule [112]. MD works by calculating the forces acting on individual atoms and iteratively solving motion equations based on accelerations resulting from these forces. The system evolves in discrete time steps, and the size of the step should be chosen suffieciently small to ensure the accuracy of the simulations. Central in MD is the definition of an energy function or *force field* which should reflect the real potential energy in the system [112].

MD methods are quite computationally demanding and for this reason can be used to simulate the evolution of a biologcial molecule in very small time scale, typically in the order of nanoseconds. Proteins generally fold in a time range of microseconds or seconds [32]. This limits the applicability of MD to simulate only short molecules or small conformational changes in the folded state.

1.2.1.2 MCMC simulations

Rather than trying to simulate the trajectories of all atoms in biological molecule as in MD, MCMC simulations aim at *sampling* the state of a biological molecule. The *target distribution* is the *Bolzmann distribution* with a energy function based on classical force fields also used in MD. In the most common approach, samples from the target distribution are obtained using the *Metropolis-Hastings* algorithm. At each iteration, a new state of the system is sampled using a *proposal distribution*. This new state, that can depend on the previous state (meaning that it is obtained only by small perturbations of the previous state), is *accepted* with a given probability. Usually, if the new state corresponds to a lower energy conformation, it is accepted with probability 1. Otherwise, the *acceptance probability* depends both on the target distribution and the proposal distribution.

The major advantage of MCMC for ab-initio structure prediction is that, since the proposal distribution can assume any form, the conformational space is sampled independently and at any time scale leading to more efficient exploration of the conformational space than possible with MD. However, the effectiveness is strongly dependent on the quality of the proposal distribution.

1.2.2 Comparative protein modelling

In comparative protein modelling methods, a model of the structure of a target protein sequence is built on the basis of similar experimentally known structures. Two approaches are possible: *homology modelling* [77] and *protein threading* [19, 58]. Figure 1.6 provides an overview of compariative protein modelling.

1.2.2.1 Homology modelling

These methods can be applied when the level of similarity between a target protein sequence and one or more protein sequences of known structure is above a given threshold (approximately 40% of sequence identity). Typically, these techniques are able to produce a sufficiently good three-dimensional model based on the known structures.

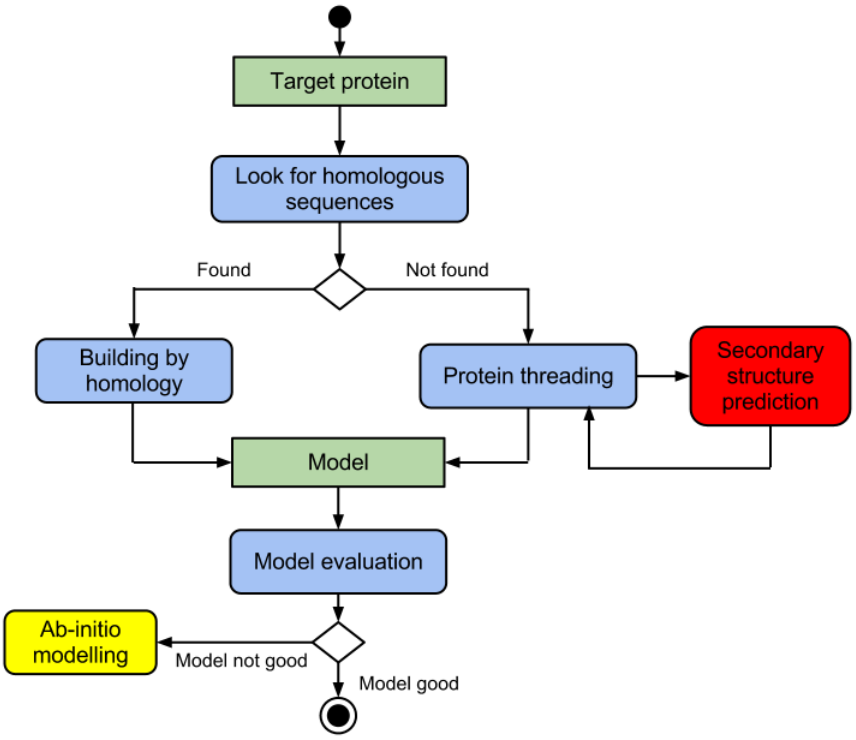


Figure 1.6: The workflow of compariative protein modelling.

In homology modelling the target protein sequence is firstly aligned to the similar sequences of known structure. Typically, insertion and deletions will occur in regions corresponding to loops whereas secondary structure elements are generally more conserved. A model for the backbone of the entire target sequence is then obtained using information about conserved regions, variable loops and side chains [17]. The model is then quality checked (manually or automatically). Finally, the model is refined by exploring the space of small conformational changes using limited energy minimazation. This step is used to fix up the exact geometry of the model.

1.2.2.2 Protein threading

These methods are well-suited when no similar sequences are found for the target protein. Starting from a library of known three-dimensional structures, protein threading techniques try to find a suitable structural model for a target protein by measuring the *compatibility* between the target sequence and the known structures.

A scoring function is used to evaluate the compatibility. These methods are also known as *fold recognition* techniques. Protein secondary structure prediction may play a key role in fold recognition. Indeed, the calculation of the scoring function may be improved by also considering the compatibility between secondary structure elements.

1.2.3 Secondary structure prediction

Methods for secondary structure prediction attempt to identify in a target protein sequence regions that are likely to form α-helices or β-strands [90]. Typically a three-class scheme is adopted: H for helices, E for strands and C for none of the two.

Prediction of secondary structure is one of the protein structure prediction subfields where machine-learning based classification methods have been extensively applied. The most powerful methods are based on neural networks [56, 99] and support vector machines [45]. In a general framework, the selected classifier is fed with a feature vector corresponding to each amino acid in the primary sequence. The output of the classifier is an assignment of the amino acid to one of the three classes. This basic scheme can be extended and improved using information derived from symmetric sliding windows of amino acid in the sequence. In this case the classifier is fed with a symmetric window of size $w = 2k + 1$ centered on each amino acid in the sequence. The prediction of central residue is therefore performed by exploting its correlation with neighboiring positions in the sequence.

1.2.4 The evolutionary information

A major advance in secondary structure prediction (and also in other prediction problems in Computational Biology) has occurred with the observation that *Multiple Sequence Alignments* (MSA) contain much more information compared to a single primary sequence.

Given a query protein sequence of length L, this can be used to scan a large sequence database in order to find homologous sequences. Several algorithms have been developed in the past decades for this purpose [70, 2]. Homologous sequences along with the query sequence can be used to build an *alignment profile*, namely

Multiple Sequence Alignment

```
TALPAFNVNPNSVSVSGLASGGYMAAQLG
-ALGAYNVDPNSISVSGLSSGGFMSAQL-
-SLGAYNVDPNSVSVSGMSAGGFMAA-LG
-SLPAYGADPGQTSVSGLSSGAFMAVQLQ
----AYGADAGRTFVTGLSAGGAMT-VML
-TIQQYNADTSKVFVTGSSSGAMMT-VMA
----KYNIDSSRIFTTGMSNGGFMSSRLL
------DIDPNRVIIGGCSNGGYMTMEMV
---QTYSVDTNRVYATGVSMGGYGTWEL-
----RWPVDESRIYACGQSSGGMMTTTLA
```

Alignment Profile

	V	L	I	M	F	W	Y	G	A	P	S	T	C	H	R	K	Q	E	N	D
T	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L	0.0	0.8	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.34	0.0	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.33	0.0	0.0	0.0
A	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.56	0.0	0.0	0.11	0.0	0.0	0.11	0.11	0.11	0.0	0.0	0.0
F	0.0	0.0	0.0	0.0	0.11	0.11	0.78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.1
										...										

Figure 1.7: An alignment profile and the corresponding MSA.

a $L \times 20$ matrix which compactly represent information derived from a MSA. An example is shown in Figure 1.7 where an alignment profile is shown along with the corresponding MSA.

The alignment profile is the matrix $\mathbf{M}_{L\times 20}$ defined as:

$$\mathbf{M}_{ij} = \frac{N_{ij}}{N_i} \tag{1.2}$$

where N_{ij} is the number of times amino acid type j is found at the i-th aligned position in the MSA and N_i is the number of aligned sequences without gaps at position i. It is also possible to consider gaps in the alignment profile definition. In this case, the matrix dimension becomes $L \times 21$, where the 21st column accounts for gaps and N_i in Equation 1.2 is always the total number of aligned sequences (with or without gaps at position i).

Although some information is lost in switching from the MSA to the profile, the information concerning the degree of conservation of a residue on a given position is directly available from the alignment profile.

The alignment profile of a protein sequence represents the so-called *evolutionary information*. Since sequence homology can be explained by sharing a common ancestor, homologous sequences are derived from random mutations occurred in

this ancestral sequence. Assuming evolution eliminates unessential elements, important structural and functional elements should be conserved among homologous sequences. This information can be in part reconstructed using alignment profiles.

1.2.5 Protein structure prediction from contact maps

An alternative approach to protein structure prediction consists in adopting a representation of a protein three-dimensional structure based on *contact maps* [10]. A protein structure is classically described by the coordinates of its atoms in the three-dimensional space. For a protein with n atoms, $3n$ numbers are needed. In alternative, the *distance matrix* can be used to describe a protein structure. This is a symmetric matrix that contains the distances between each pair of atoms in the molecule. With n atoms we need n^2 elements. The two representation are equivalent: it is possible to reconstruct the coordinate-based representation from the distance matrix and vice versa[2] [10].

A more compact and simplified representation can be obtained in two ways. Firstly, not all atoms of the protein are taken into account but only a represetative atom for each residue (e.g. the C_α). Secondly, rather than storing actual distances, only the binary information about residue contacts is retained. Two residue are considered in contact if their distance is below a fixed distance cut-off. By this, the distance matrix is transformed into the contact map, a symmetric binary matrix whose non-zero components represent contacts between residue. The contact map is a $m \times m$ matrix where m is the number of residues of the protein. Note that the contact map is not equivalent to the distance matrix since lot of information is lost in switching from real-valued distances to binary values.

Protein structure prediction based on contact maps consists of the following two steps:

- contact map prediction from sequence;
- reconstruction of the three-dimensional structure from the predicted contact map.

[2]Of course, reconstruction of the coordinates from the distance matrix is less intuitive than the opposite.

Predicting residue contacts from sequence is a problem that has been addressed several time in the past decades. The first attempts were based on correlated mutations analysis [36, 85], namely the study of pair of residues which mutate in concert in a given set of related proteins (i.e. a *family* of proteins). The rationale behind these approaches is that the conservation of protein functions constrains the evolution of protein sequences. In this view, if a residue mutates in given position, it is likely that a residue in contact with it will mutate as well in order to *compensate* the previous change. Recently, methods have been introduced which perform correlated mutation analysis with advanced techniques such as sparse inverse covariance estimation to reach very high accuracies in contact prediction [79, 57].

Alternative approaches to contact prediction are based on machine-learning methods. Several methods have been used such as neural networks [29, 91] or Hidden Markov Models [21].

The problem of reconstructing the three-dimensional structure from a (predicted or real) contact map has been proven to be NP-hard [116]. Nonetheless, heuristics algorithms exist to obtain approximate solutions for this problem [116, 117, 18, 92].

1.3 Summary

Proteins are important molecules involved in several vital functions in living organisms. Proteins are linear chains made up by 20 different fundamental units called amino acids. The unstructured chain folds into a functional three-dimensional structure that always corresponds to a specific function of the protein.

Protein structure prediction aims at determining the three-dimensional structure of the protein starting from the linear sequence of amino acids using computational methods. Different approaches are possible. Ab-initio methods try to predict the protein structure from scratch by simulating the physics of the folding process. In contrast, comparative protein modelling techniques use available structural information about similar proteins to build a model for a target protein sequence. Both approaches can take advantage of predicted secondary structure, a task that is typically addressed by means of machine-learning based methods, usually enriched with evolutionary information extracted from MSAs. Another approach to protein struc-

ture prediction is based on the representation of protein structures by means of contact maps.

Chapter 2

Transmembrane proteins

2.1 Biological membranes

The *cell* represents the basic structural and functional unit in all living organisms. The boundary that determines what is *inside* and what is *outside* the cell is defined by the *plasma membrane* that encloses the cell. Besides acting as a physical border for the cell, the plasma membrane also helps to mantain the essential difference between the interior of cell, referred to as the *cytosol* and the *extracellular environment* [1]. In eukaryotic cells, characterized by a high level of compartmentalization, similar membranes enclose the various *organelles* such as mithocondria and chloroplasts.

All biological membranes share a common structure: they are essentially made up by *lipid* molecules, arranged to form a double layer usually referred to as the *lipid bilayer*. The lipid bilayer serves as an impermeable envelope to block the passage of most water-soluble melecules [1].

Most of the membrane functions are mediated by *membrane proteins*. It is estimated that almost half of the total membrane mass is constituted by membrane proteins. The following tassonomy of membrane proteins can be defined according to the type of interaction with the membrane:

- *Transmembrane proteins* completely cross the lipid bilayer and are permanently attached to the membrane.

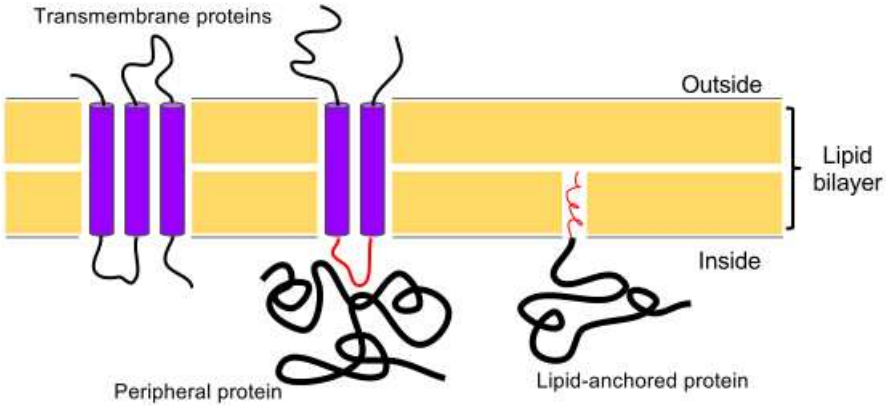


Figure 2.1: Schematic representation of a biological membrane.

- *Lipid-anchored proteins* are attached to the lipid bilayer only by a covalent attachment of a lipid group (e.g. the *Glycosylphosphatidylinositol (GPI)* anchor).

- *Peripheral membrane proteins* are located on the membrane surface and typically do not interact with the core of the lipid bilayer, but instead with other transmembrane proteins.

A schematic representation of a biological membrane is shown in Figure 2.1.

2.2 Transmembrane proteins

Transmembrane proteins are designed to entirely span biological membranes. This class of proteins is devised to perfom several important functions. One of the main roles of transmembrane proteins is communication with the external environment. *Receptors* are able to sense the external environment for the presence of specific substrate molecules. Once a substrate binds the receptor, this in turn changes its conformation also in the intra-cellular domain which activate a cascading singnaling process. By this, the cell can react to the external environment.

Another important function of transmembrane proteins is controlling the exchange of material across membranes. Transmembrane proteins such as *porins* assume structures specifically designed to create channels or pores within the mem-

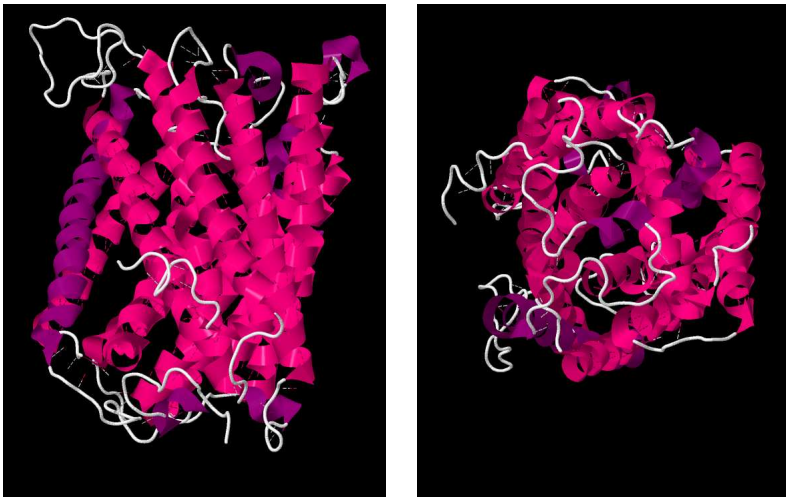


Figure 2.2: Structure of the Cytochrome C Oxidase (1ar1:A) from *Paracoccus denitrificans* [86]

brane to allow the transport of substances. The functions of these channels are typically mediated by other proteins which control the status of the channel (open or closed).

Transmembrane proteins can be divided in two different structural classes according to the specific conformation assumed: α-helical bundles and β-barrel transmembrane proteins.

2.2.1 α-helical bundles

Transmembrane α-helices (TMAH) proteins, as the *Paracoccus denitrificans* Cytochrome C Oxidase (1ar1:A) [86] shown in Figure 2.2, span the membrane with regions formed by 15-35 residue-long hydrophobic helices [120]. These proteins are found in cell membranes of eukaryotic cells and bacterial inner membranes and serve as cell receptors, ion channels, active and passive transporters, and membrane-bound enzymes. [120].

2.2.2 β-barrels

The class of transmembrane β-barrels (TMBBs) is central in this thesis. These proteins have membrane spanning segments formed by anti-parallel β-sheets organized in a closed structure similar to a barrel [105]. These proteins are mainly found in the outer membrane of the gram-negative bacteria, even though β-barrels can be also found in the outer membranes of mitochondria or chloroplasts[1]. Gram-negative bacteria are characterized by the presence of two membranes: an inner membrane cytoplasmic membrane and an outer membrane facing the extracellular space. The region comprised between the two membranes is called the periplasmic space or simply *periplasm*.

TMBBs are endowed with functions relevant to the entire cell metabolism and including active ion transport, passive nutrient intake and defense against attack proteins [105]. TMBBs are estimated to be encoded by as many as 2-3% of the genes in Gram-negative bacteria [105, 125, 22]. However, very few TMBB structures are available at atomic resolution from Gram-negative organisms. The typical shape of TMBBs is shown in Figure 2.3 where the structure of the OpcA outer membrane Adhesin/Invasin from *Neisseria meningitidis* (1k24:A) is shown [93].

2.3 Structure prediction of β-barrel proteins

Structure prediction of β-barrel protein can be tackled using one the techniques discussed in Chapter 1. However, since TMBB proteins are quite underepresented in structural databases, the applicability of homology modelling is limited. Indeed, in most cases the available data can be insufficient to build a satisfactory template library for model construction. On the other hand, ab-initio methods are still far from providing good solution to the protein folding problem. An alternative to these approaches consists in trying to solve the folding problem by identifying a set of sub-problems such as secondary structure, solvent accessibility or contact map prediction.

In the particular case of TMBBs, three intersting computational problems can

[1] A fact that can be explained by the endosymbiotic theory [1].

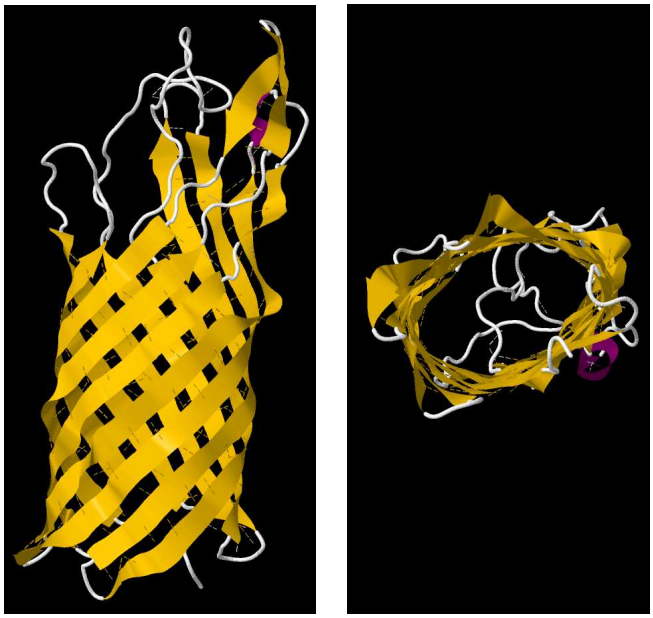


Figure 2.3: Structure of the OpcA outer membrane Adhesin/Invasin from *Neisseria meningitidis* (1k24:A) [93]

be identified:

- discrimination or detection of TMBBs in a set of proteins;
- topology or secondary structure prediction;
- prediction of inter-strand residue contacts;

2.4 Detection of TMBBs

A first, important problem consists in the detection of TMMBs in a set of protein typically as large as the whole genome. Experimental determination of TMBBs (and in general of all membrane proteins) is difficult and expensive [75]. The availability of computational approaches specifically designed to detect a putative TMBB starting from sequence can prove valuable since they can help identifying those candidate genes on which experimental validations can be performed.

The problem of TMBB detection has been addressed by many researchers using different techniques. Methods available can be roughly divided in three main classes:

- statistical approaches;
- machine learning approaches;
- sequence homology approaches.

2.4.1 Statistical methods

Statistical methods [124, 12, 33] are essentially based on statistics about physico-chemical properties of TMBBs. The Wimley algorithm [124] is based on an analysis of structure and composition of experimentally determined TMBBs. Statistical data about amino acid abundances are obtained from known structures. Using a sliding window approach, the sequence is scanned and a β-strand score is assigned to each residue using abundances in a window of neighboring residues. These β-strand scores are therefore used to compute β-hairpin scores scanning the sequence again. Finally, β-hairpin scores which are above an empirically determined threshold are summed to obtain an overall β-barrel score used for the final prediction. A slightly modified version of this method with various improvements, the Freeman-Wimley (FW) algorithm [33], has been recently introduced and tested on a non-redundant dataset of proteins. The FW algorithm is at the moment one of the best performing methods for TMBB detection.

Other approaches based on statistical analyses are essentially extensions of the original Wimley algorithm [124]. The BOMP method [12] use the same β-barrel score as described above. Furthermore, the method recognizes TMBBs relying on a pattern extracted from the last 10 residues in the sequence. This pattern has been extracted from known TMBB sequences and, according to authors, can be used as a fingerprint for identifying candidate TMMBs.

2.4.2 Machine-learning based approaches

Methods in the second class try to address the problem by means of standard machine learning techniques [40, 87, 44, 13, 76]. In many of this approaches the protein sequence is entirely described in terms of overall amino acid composition [40] or di-peptide composition [87, 44]. Therefore, standard machine learning is applied

using these features as input. In [40] different machine-learning methods such as bayesian networks, neural networks and logistic regression have been tested. Among the various methods, neural networks outperformed the others. A k-nearest neighbor approach has been introduced in [44] to assign the protein to one of seven distinct classes corresponding to different locations in the cell. In [87] Radial Basis Function Networks (RBFNets) are used with an extended input including Position Specific Scoring Matrices (PSSMs).

Very popular are methods based on Hidden Markov Models (HMMs) [13, 76]. These approaches are typically designed to tackle the problem of topology prediction and adapted to discriminate TMBBs. The basic idea is to build a probabilistic topological model using HMMs and derive a discrimination score on the basis of the probability of a new sequence in the model. Since the HMM has been trained only on positive TMBB examples, non-TMBB proteins should be assigned a low probability under the model. Different strategies can be used to compute the score from the raw probability assigned by the HMM. In [13] the score is obtained as:

$$S = \log\left(\frac{p(O|\lambda)}{p(O|\text{null})}\right) \tag{2.1}$$

where $p(O|\lambda)$ is the probability of a protein O given the HMM model and $p(O|\text{null})$ is the probability of the protein in the null model i.e. a HMM with emissions probability set to the background distribution of residues in the dataset being predicted. Another approach [76] is to simply normalize the probability over the sequence length L:

$$S = \frac{\log p(O|\lambda)}{L} \tag{2.2}$$

2.4.3 Sequence homology methods

HMMs are also used in approaches based on sequence homology. In [98], starting from protein sequences annotated as TMBBs, a dataset of proteins is built by searching for homologous sequences in the NCBI non-redundant database of sequences. After clusterization, the retrieved proteins are used to build several profile HMMs [28], one for each cluster. Information about predicted secondary structure and trans-membrane topology is added to HMMs as well. For a new protein, a

profile HMM is built and prediction is performed by comparison to the database of HMMs previously generated.

2.5 TMBB topology prediction

A second prediction problem related to TMBBs and in general to all types of transmembrane proteins is the prediction of the *transmembrane topology*. By definition, the topology of a membrane protein refers to the specification of the number and orientations of transmembrane α-helices or β-strands across a membrane in the cell [121]. In the context of TMBBs, the prediction problem can be stated as follows: given a protein sequence that is known to be inserted in the outer-membrane of a gram-negative bacterial cell, one wants to predict the number and the location with respect to the membrane plane of the membrane-spanning segments.

Transmembrane protein topology prediction has been addressed for many years in bioinformatics, although most of the effort has been spent on TMAH topology prediction. One reason for this is that TMAH topology prediction is much easier than TMBB prediction using computational methods because membrane spanning segments in TMAHs are characterized by strong hydrophobic signals that can be easily detected by simple rules such as the *positive-inside rule* [119]. Secondly, known TMAHs are more abundant in both sequence and structure databases compared to TMBBs. As a consequence of this discrepancy, the number of available methods for TMAH topology prediction has increased at a faster pace with respect to methods for TMBB topology prediction.

2.5.1 Early methods

Early methods for TMBB topology prediction were based on simple physico-chemical analisys of the amino acid sequence. A distinguishing structural feature of TMMBs, and in general of all β-sheets, is represented by the *dyad repeat* i.e. a pattern where alternating residues point toward alternating directions of the sheet. Since TMBB are inserted into the membrane, those β-residues that are oriented toward the lipid bilayer are usually hydrophobic whereas the others that are oriented

toward the interior of the barrel are more hydrophilic. As a consequence, about half of the residues of a transmembrane β-strand are expected to be hydrophobic while the others are hydrophilic.

This alternating pattern can be then exploited in order to identify membrane-spanning segments. In particular, through analyses of known three-dimensional structure of TMBBs, two physicochemical properties were identified:

- membrane-spanning β-strands typically corresponds to peaks of hydrophobicity, although these peaks are not generally as high as those of α-helical transmembrane segments;

- the alternating pattern of hydrophobic/hydrophilic residues led to observable peaks of *amphipathicity* [2] in correspondence of transmembrane β-strands.

A variety of algorithms have been developed for membrane-spanning segments identification using the above analisys [25, 39, 127]. However, when the regular alternation between hydrophobic and hydrophilic residues was not respected, as happens in specific sub-classes of β-barrels such as maltoporins, all these methods were not able to reliably detect transmembrane segments. Thus, the simple dyad repeat is not sufficient for idenfying transmembrane β-strands [105].

2.5.2 Methods based on probabilistic models

At a high level of abstraction, topology prediction of TMBBs is essentially a sequence labelling or segmentation problem where a protein sequence is segmented in contiguous regions corresponding to different locations with respect to the outer membrane: periplasmic side or *inner loops*, transmembrane β-strand and extracellular side or *outer loops*. Probabilistic models for sequence analisys are routinely applied. Traditionally, the best performing models are based on HMMs [5].

Several studies in the past decades have investigated the architecture of β-barrels idenfying general construction principles dictated by the β-barrel geometry [81, 105,

[2] An molecular compound is called *amphipathic* if it contains both a water-loving polar part (hydrophilic) and a water-hating non-polar part (hydrophobic).

115]. From these results it is possible to derive the following rules which should be taken into consideration in building a topological model:

- all β-strands are antiparalell and locally connected to their closest neighbors;
- both termini of the protein are at the periplasmic side restricting the strand number n to even values;
- the extracellular loops are typically long loops whereas inner loops are generally minimum-length turns.

In addition transmembrane-segment lengths are distributed accordingly to a probability density distribution that can be experimentally determined and must be taken into account.

From the topology prediction point-of-view, all these constraints can be incorporated into a topological model which is typically described in terms of a regular grammar [31, 13, 76]. The physicochemical and geometrical characteristics of the three types of segments as deduced by the available structures in the PDB [11] suggest how to build a grammar (or the corresponding automaton) for the prediction of the topology.

For these reasons, probabilistic modelling based on HMMs has been widely adopted for TMBB topology prediction [76, 13, 6, 42]. All these approaches typically adopt a HMM architecture which reflects the construction rules of β-barrels.

2.5.3 Other approaches based on machine learning

Other methods have addressed the problem in two steps. These approaches first predict a per-residue class membership score and then apply a refinement method to enforce a predicted topology to be compatible with general construction rules. Various types of neural networks [55, 38, 37, 96] or support vector machines [82] have been employed to predict class membership. For instance, in [55] a dataset of 11 TMBBs has been used to train and test a feed-forward neural network for class membership prediction. An algorithm based on dynamic programming use the network outputs to locate the transmembrane β-strands along the protein sequence by model

optimization. The algorithm takes advantage of the different construction rules and minumum/maximum segment lengths to refine the network output. Minimal and maximal lengths are derived from the database of selected proteins [55]. Similar two-step schemes have been employed in other methods [38, 37]. The TMBpro suite is a pipeline of methods for TMBB topology, β-contacts and tertiary structure prediction [96]. TMBpro improves class membership prediction using a 1D-Recursive Neural Network (1D-RNN) architecture [9]. Even in this case, the network output has been refined using a dynamic programming algorithm which incorporates β-barrel construction rules [96].

2.6 Inter-strand contact prediction

Topology prediction can be used to build a first, coarse model of a TMBB protein structure. However, the sole topology does not provide enough information to build a low-resolution three-dimensional model of the target protein. Additional information can be provided by the knowledge of the exact inter-strand connectivity. Indeed, as described in Section 1.1.2, hydrogen bonds are formed between adjacent strands in a β-sheet. Identifying these bonds allows to construct a β-contact map which in turn can be used to reconstruct a three-dimensional structure as can be done for globular proteins [116]. Similar approaches have been succesfully applied for de-novo prediction of transmembrane α-helices [84]. β-contact maps are often used in combination with template-based modelling or ab-initio methods to predict TMBB tertiary structure [96].

2.7 Resources for transmembrane proteins

The main resource concerning structural information about proteins is the Protein Data Bank (PDB) [11]. This database contains to date approximately 80000 entries corresponding to atomic level descriptions of protein structures. These data has been obtained by different experimental methods: approximately the 85% from X-ray christallography, the 10% from NMR spectroscopy and the rest by other methods (electron microscopy, hybrid methods etc.). About 0.1% of all the structures

from Gram-negative organisms in the PDB are TMBB proteins.

Besides the PDB, several derived databases exist specifically dedicated to transmembrane proteins. The PDBTM databank is derived from the PDB and it is a curated resource containing information about transmembrane proteins [113]. PDBTM entries are obtained from PDB by applying the TMDET method [114], an algorithm which is able to discriminate between transmembrane and globular proteins and to identify transmembrane spanning segments using structural information from PDB entries. By this, the PDBTM provide a comprensive resource which include detailed information about both the topology and the three-dimensional structure of transmembrane proteins. To date, PDBTM contains a redundant dataset of approximately 200 TMBB proteins.

Another database of transmembrane proteins is the Orientation of Proteins in Membranes (OPM) databank [73] which uses a different method to determine transmembrane spanning regions and location of the membrane with respect to the coordinate system of the molecule [72]. The number of (redundant) TMBB proteins in the OPM database is about 130.

PDBTM and OPM databases largely overlap, although they both contain unique entries. Since they use different methods to identify transmembrane segments, topology annotation for the same protein chain may also slightly differ in the two databases.

2.8 Summary

Transmembrane proteins represent an important class of proteins that perform critical functions in cells such as signal transduction and material transport across membranes. Two different types of transmembrane proteins exist, characterized by a different three-dimensional organisation: α-helical bundles (TMAHs) and β-barrels (TMBBs). Several computational methods for predicting the structure of TMAHs have been developed in the past decades. In contrast, few TMBBs are known at atomic level and this has limited the development of computational methods for predicting their structure. Three different computational problems related to TMBBs can be identified: (i) TMBB detection or discrimination from other types

of proteins; (ii) TMBB topology prediction; (iii) TMBB β-contacts prediction. All these tasks are typically addressd using machine-learning based approaches.

Chapter 3

Probabilistic models for sequence analysis

The modelling of complex phenomena often requires to account for random aspects of the system being modelled. In computational molecular biology a large amount of data is available often accompanied by a substantial lack of theory and understanding. In this context, characterized by a very high degree of uncertainty, the probabilistic framework provides a very effective tool to build accurate models based on available data and to reason about them.

Most frequently, biological data are available in the form of *sequences*, either nucleic acid (DNA or RNA) or protein sequences. Typical problems of biological sequence analysis concern about discovering the nature of the sequence being analyzed, e.g. assigning a protein to a given *family* or annotating the sequence with respect to a feature of interest. In general, the common approach consists in building a probabilistic model of the problem being addressed on the basis of the previous knowledge [28].

This chapter is about probabilistic models for sequence analysis with special attention to frameworks that are well-established in the area of biological sequence analysis. In Section 3.1 the general problem of *annotation of sequences*, of central importance in Bioinformatics, is formulated. Section 3.2 contains a brief introduction to *graphical models*, a formalism that allow to graphically represent multivariate probability distributions and that will be used throughout this thesis. Section 3.3 describes the *Hidden Markov Model* (HMM) [95] widely employed for sequence analysis in many scientific fields. Historically, the HMM has been extensively used as

probabilistic framework for biological seqeuence analysis [28]. Section 3.4 provides an introduction to the *Conditional Random Field* (CRF) [63] which has emerged as a promising framework to sequence data analysis in the field of Computational Linguisitics and, more recently, also in Computational Biology. Finally, in Section 3.5 the *generative* and the *discriminative* modelling approaches are discussed in general terms for sake of higlighting the main differences between HMMs and CRFs.

3.1 Annotation of sequences

As mentioned above, one of the main tasks in Bioinformatics consists in providing position-specific annotations for nucleic acid or protein sequences such as protein secondary structure prediction or membrane protein topology prediction. The problem of sequence labelling arises also in other scientific fields such as Computational Linguistics. In this section we formally define the problem and intruduce the notation that will be adopted throughout the entire chapter.

Let $\mathbf{x} = (x_1, \ldots, x_L)$ be a sequence of length L. The elements x_j of the sequence belong to a given set $\mathcal{X}$. The sequence $\mathbf{x}$ is usually referred as *observation sequence*. The sequence annotation or *sequence labelling* problem can be stated follows: given an observation sequence $\mathbf{x}$ find a *label sequence* $\mathbf{y} = (y_1, \ldots, y_L)$ with elements y_j in some set $\mathcal{Y}$ that better explain the sequence $\mathbf{x}$ with respect to a feature of interest. For instance, in protein secondary structure prediction, the set $\mathcal{X}$ will be the set of all possible amimo acids while the set $\mathcal{Y}$ will be the set of all possible secondary structure elements (i.e. the set $\mathbf{H}, \mathbf{E}, \mathbf{C}$ for *helix*, *strand* or *coil* in the most basic setting).

Throughout this thesis, a sequence of objects will be always denoted in boldface (e.g $\mathbf{x}$) while elements of the sequence will be denoted in normal font with a subscript corresponding to the position inside the sequence (e.g. $\mathbf{y} = (y_1, \ldots, y_L)$).

3.2 Graphical models

Probabilistic models can be coherently described using the framework of *graphical models* [122, 15, 80] which provides a unifying language to graphically repre-

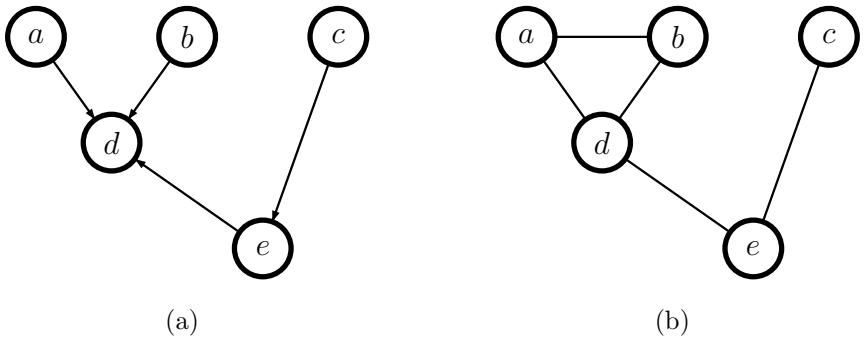


Figure 3.1: (a) Directed graphical model for a joint distribution over five variables. (b) Undirected graphical model over the same set of variables.

sent probability distributions defined on a set of random variables. In particular, a probabilistic graphical model is a formalism that allows to describe a multivariate probability distribution by means of a graph. Each node in the graph is associated to a random variable in the model while edges encode dependencies between variables. As a whole, the graph provides a complete description of the model in terms of random variables and probabilistic relationships that subsist between them [15].

Probabilistic graphical models exist in two flavours which differ on the nature of the underlying graph they adopt to represent the model distribution. *Directed graphical models* also known as *Bayesian Networks* (BN) [88] employ a representation based on directed acyclic graphs. On the other hand, *Undirected Graphical Models* or *Markov Random Fields* (MRF) are based on undirected graphs [15]. Directed models are well suited to describe direct *causal relationship* between random variables, while undirected models allow to express looser constraints on the model. Figure 3.1 shows examples of graphical models over a set of five random variables.

The graph structure in a graphical model captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of variables. In other words, the graph represents a *factorization* of the associated joint distribution. The main difference between directed and undirected models lies in the way each factor is defined. The next two paragraphs provide descriptions of directed and undirected graphical models, respectively.

3.2.1 Bayesian Networks

In a Bayesian Network, the joint distribution factorizes as a product of conditional distribution for each node conditioned on the variables corresponding to the parents of that node on the graph. Thus, for a graph with K nodes, the joint distribution is given by [88]:

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k \mid \pi_k) \tag{3.1}$$

where π_k is the set of parent nodes of x_k and $\mathbf{x} = (x_1, \ldots, x_k)$. Consider for instance the example in Figure 3.1(a). Nodes a, b and c have no parent nodes, node d is a child of a, b and e, node e is a child of c. Therefore the associated joint distribution decomposes as follows:

$$p(a, b, c, d, e) = p(a)p(b)p(c)p(d|a, b, e)p(e|c) \tag{3.2}$$

An important and elegant feature of graphical models (both directed and undirected) is that conditional independence properties of the model can be read directly from the graphical representation. Let a, b and c be three random variables: a is conditionally independent of b given c (in shorthand notation $a \perp\!\!\!\perp b \mid c$) if $p(a, b|c) = p(a|c)p(b|c)$, i.e. the joint distribution of a and b given c factorizes into the product of marginal distributions of a and b (both conditioned on c). Conditional independence may also refer to sets rather than individual variables. In directed graphical model, the conditional independence between two subsets of nodes given another subset is captured by the notion of *d-separation* [88].

Definition 3.2.1. *Two nodes u and v of a directed cyclic graph $G = (V, E)$ are d-separated by a susbset of nodes $Z \subset V$ if, for each trail P (undirected paths) between u and v, one of the following holds:*

1. *P contains a chain $x \to m \to y$ such that $m \in Z$*
2. *P contains a chain $x \leftarrow m \leftarrow y$ such that $m \in Z$*
3. *P contains a fork $x \leftarrow m \to y$ such that $m \in Z$*

4. P contains a collider $x \rightarrow m \leftarrow y$ such $m \notin Z$ and no descendant of m is in Z

Two subsets $A, B \subset V$ are d-seprated by $Z \subset V$ if all possible pairs of nodes $u \in A$ and $v \in B$ are d-separated by Z. It can be proven [64] that if $Z \subset V$ d-seprates $A \subset V$ and $B \subset V$ then $A \perp\!\!\!\perp B \mid Z$. For example, considering the graph in Figure 3.1(a), the subsets $A = \{b, d\}$ and $B = \{c\}$ are conditionally independent given the subset $Z = \{e\}$ because the trail $b \rightarrow d \leftarrow e \leftarrow c$ contains the collider $b \rightarrow d \leftarrow e$ and $d \notin Z$ whereas the trail $d \leftarrow e \leftarrow c$ is a chain with $e \in Z$.

3.2.2 Markov Random Fields

Undirected graphical models or Markov Random Fields (MRF) represent the second class of graphical models in which the underlying graph is not directed as in BNs. Before discussing the factorization properties it is convenient to introduce the notion of conditional independence. In the case of a MRF, the conditional independence follows directly from simple graph separation and hence it is easier to highlight than in BNs. More formally, given an undirected graph $G = (V, E)$ and three disjoint subsets of nodes $A \subset V$, $B \subset V$ and $C \subset V$, C separates B and A if every path that connects nodes in A and nodes in B contains at least one node in C. Then, if C seprates A and B the conditional independence property hold:

$$A \perp\!\!\!\perp B \mid C \tag{3.3}$$

Considering the example in Figure 3.1(b), according to the above definition, the subsets $\{a, b\}$ and $\{e\}$ are conditionally independent given the subset $\{d\}$.

As for BNs, the factorization of the joint distribution in a MRF is defined in terms of *local factors* defined over subsets of variables. The notion of locality used in the factorization should be consistent with the notion of conditional independence. Firstly, note that two nodes that are not directly connected are conditional independent given the rest of nodes in the graph. Intuitively, one can expect that this two nodes should not appear in the same local factor. In general, each local factor should contain only variables that are directly connected. This condition can be met by factorizing according to maximal *cliques* in the graph such that the joint

distribution can be expressed as folllows:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \Psi_C(\mathbf{x}_C) \tag{3.4}$$

where $\Psi_C(\mathbf{x}_C)$ are real-valued *local* or *potential functions* each defined over a maximal clique C, $\mathbf{x}_C$ is the subset of random variables corresponding to the clique and Z is the *partition function* that ensures the correct normalization of the joint distribution:

$$Z = \sum_{\mathbf{x}} \prod_C \Psi_C(\mathbf{x}_C) \tag{3.5}$$

.

Typically a MRF is expressed as a *log-linear*[1] model by setting potential function to have the form:

$$\Psi_C(\mathbf{x}_C) \quad = \quad \exp\left(\sum_k \lambda_{C,k} f_{C,k}(\mathbf{x}_C)\right) \tag{3.6}$$

where the index k runs over all possible configurations of the variables in the clique C and $\{f_{C,k}\}$ are *feature functions* defined as follows:

$$f_{C,k}(\mathbf{x}_C) = \begin{cases} g_k(\mathbf{x}_C) & \text{if } \mathbf{x}_C \text{ is in the } k\text{-th configuration} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The function $g_k : \mathbb{R}^{|C|} \to \mathbb{R}$ can be any real function, although usually the constant function $g_k(\mathbf{x}_C) = 1$ is used. In this case, a feature is simply an indicator function of a specific configuration of the variables. The parameters of the model $\Theta = \{\lambda_{C,k}\}$ are weights associated to features that reflect the strength of the contribution of each feature to the joint probability.

In contrast to BNs, where local factors are conditional probability distributions, here local functions have not necessarily a direct probabilistic interpretation (even though they can have it in some cases [88]). As a consequence, their product is not in general normalized to have a probabilistic meaning. Hence the adoption of the normalization constant. Furthermore, local functions act as constraints defined

[1] A log-linear model is a mathematical model that takes the form of a function whose logarithm is a first-degree polynomial function of the parameters of the model.

Figure 3.2: Graphical structure of a first-order Markov chain.

over subsets of variables and then influence the global distribution: high probability configurations meet more constraints than low probability configurations.

The formal connection between the factorization given by Equation 3.4 and the notion of conditional independence in a MRF is provided by the following *Hammersley-Clifford* theorem:

Theorem 3.2.1 ([23])**.** *Let $\mathcal{UI}$ the set of all probability distributions defined over a fixed set of variables corresponding to nodes of an undirected graph and consistent with the set of conditional independence assertions derived from the graph by means of the notion of graph separation. Let $\mathcal{UF}$ by the set of such distributions that can be expressed as a factorization given by Equation 3.4 with respect to maximal cliques of the graph. Then the sets $\mathcal{UI}$ and $\mathcal{UF}$ are identical.*

3.3 Hidden Markov Models

A Markov model represent the simplest form of probabilistic model for sequential data. Let $\{y_t\}$ denote the value of a random variable at time t and let the *state space* $\mathcal{Y}$ be the countable set of possible value for y values. The random variable is a *Markov process of order 1* if the transition probabilities between different values in the state space depend only the current state of the random variable, namely:

$$p(y_t|y_1, \ldots, y_{t-1}) = p(y_t|y_{t-1}) \tag{3.8}$$

A *Markov chain* refers to a sequence $\mathbf{y} = (y_1, \ldots, y_L)$ of random variables generated by a Markov process. Figure 3.2 shows a graphical model for a first-order Markov chain. Higher-order Markov chains can be obtained by considering dependencies on more than one previous state. The Markov chain is said *time-homogeneous* when the conditional distributions $p(y_t|y_{t-1})$ are independent of t.

A Hidden Markov Model (HMM) is an extension of Markov chains in which the stochastic process governing the state change is not directly observable, but, as the name suggests, it is *hidden*. The process can be only investigated through an additional set of stochastic processes that *generate* or *emit* some observable symbols depending on the current state of the process. More formally, an HMM is completely described by the following elements:

- a finite state space $\mathcal{Y}$
- a finite alphabet of symbols $\mathcal{X}$
- a conditional probability distribution $p(y|y')$ that represent the *transition probability* from state y' to state y, where $y, y' \in \mathcal{Y}$
- a conditional probability distribution $p(x|y)$ that represent the *emission probability* of a symbol x in state y, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
- a probability distribution for the initial state $p(y)$

At time t the system is in a given state y_t from which a symbol x_t is emitted according with the emission probability distribution $p(x_t|y_t)$. Then the system enters to another state with transition probability $p(y_{t+1}|y_t)$. The initial state y_1 of the system is governed by the initial probability distribution $p(y_1)$. Altogether, transition, emission and initial probability distributions are the parameters of the model, often overall referred to as λ.

A HMM can be represented as a directed graphical model as shown in Figure 3.3. Nodes of the graph that correspond to states $\mathbf{y}$ are depicted in dark grey to emphasize the fact they are hidden variables. The graphical structure also highlights the conditional independence assumptions made in a HMM:

- state y_t is conditional independent from all other states given the preceding state y_{t-1} (i.e. the state sequence is Markov);
- the observation symbol x_t is conditionally independent from other observations given the current state y_t.

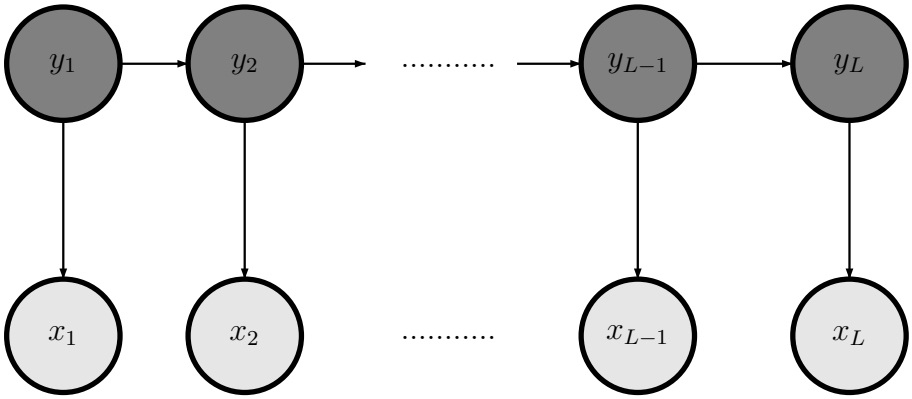


Figure 3.3: Graphical structure of a first-order HMM.

These two assumptions can be easily verified on the graph using the notion of d-separation discussed in Section 3.2.1.

Given these independence assumptions and the graphical structure in Figure 3.3, the joint probability distribution associated to an HMM is then defined as follows:

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1|y_1) \prod_{t=2}^{L} p(y_t|y_{t-1})p(x_t|y_t) \tag{3.9}$$

where $\mathbf{y} = (y_1, \ldots, y_L)$ and $\mathbf{x} = (x_1, \ldots, x_L)$.

Let $\mathbf{x}$ be a sequence of observation symbols. The following three basic problems are associated with a HMM:

1. *evaluation*: given a model with parameters λ, compute the probability of the sequence given the model $p(\mathbf{x}|\lambda)$;

2. *decoding*: given a model with parameters λ, compute the most probable sequence of states $\mathbf{y}$ that generate $\mathbf{x}$, i.e. solving:

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{p(\mathbf{y}, \mathbf{x}|\lambda)}{p(\mathbf{x}|\lambda)} \tag{3.10}$$

3. *training*: estimating the parameters λ of the HMM such that the joint probability of $\mathbf{y}$ and $\mathbf{x}$ given the model is maximized, i.e:

$$\underset{\lambda}{\operatorname{argmax}}\, p(\mathbf{y}, \mathbf{x}|\lambda) \tag{3.11}$$

The three problems just described can be solved efficiently using dynamic-programming algorithms. For the evaluation and the training problems the *forward-backward* algorithm is used. For the decoding problem the *Viterbi algorithm* is adopted. Since these algorithms are very similar to the ones adopted for linear-chain Conditional Random Fields, they will be discussed in the following sections.

3.3.1 HMMs for sequence labelling

HMMs are often used for sequence labelling tasks. In the problem-specific context, a correspondence between labels and states of the HMM is established. In this view the elements of the observation sequence $\mathbf{x}$ are considered as *generated* by the corresponding label.

The label sequence of an observation can be found solving the docoding problem as stated in the previous section. Given an observation sequence $\mathbf{x}$, one wants to find the label sequence $\mathbf{y}$ so that:

$$\mathbf{y} = \underset{\mathbf{y}'}{\operatorname{argmax}}\, p(\mathbf{y}'|\mathbf{x}) = \underset{\mathbf{y}'}{\operatorname{argmax}}\, \frac{p(\mathbf{y}, \mathbf{x})}{p(\mathbf{x})}. \tag{3.12}$$

In other words, we assing to $\mathbf{x}$ the sequence $\mathbf{y}$ that maximize the joint probability distribution defined by the HMM model.

3.4 Linear-chain Conditional Random Fields

Conditional Random Fields (CRFs) [63] are *discriminative* probabilistic models introduced for the modelling of sequence data. In this section linear-chain CRFs will be introduced and compared to HMMs described in the previous section. For sake of clarity, linear-chain CRFs will be mathematically derived from the HMM formulation. This to highlight analogies and differences between the two frameworks.

3.4.1 From HMMs to CRFs

HMMs and linear-chain CRFs are closely related. A natural way to present linear-chain CRFs is to show how they can be defined considering the conditional

probability $p(\mathbf{y}|\mathbf{x})$ associated to the HMM joint probability [109]. The HMM joint probability distribution defined in Equation 3.9 can be written in a more general form that is well-suited for generalizations:

$$p(\mathbf{y},\mathbf{x}) = \frac{1}{Z}\prod_{i=1}^{L}\exp\left(\sum_{s,t}\tau_{st}\mathbf{1}_{\{s=y_i\}}\mathbf{1}_{\{t=y_{i-1}\}}\sum_{s,a}\mu_{sa}\mathbf{1}_{\{s=y_i\}}\mathbf{1}_{\{a=x_i\}}\right) \tag{3.13}$$

where $\Theta = \{\tau_{st}, \mu_{sa}\}$ are the parameters of the distribution and can take any real value and $\mathbf{1}_{\{s=t\}}$ is the indicator function defined as:

$$\mathbf{1}_{\{s=t\}} = \begin{cases} 1 & \text{if } s == t \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

Every HMM can be written in this log-linear form by setting the parameters τ_{st} and μ_{sa} to be *log-probabilities* of transition and emission probabilities, respectively:

$$\tau_{y_{i-1},y_i} = \log(p(y_i|y_{i-1})) \tag{3.15}$$

$$\mu_{y_i,x_i} = \log(p(x_i|y_i)) \tag{3.16}$$

However, since parameters Θ are not in general required to be log-probabilities, an explicit normalization constant Z is introduced in Equation 3.13 to guarantee the meaning of joint probability. In spite of the added flexibility it can be shown that Equation 3.13 describes exactly the same class of HMMs of Equation 3.9.

The notation can be simplified using the concept of *feature functions*. In the case of Equation 3.13 we need one feature for each transition (s,t) and one feature for each emission (s,a):

$$f_{s,t}(y_i, y_{i-1}, x_i) = \mathbf{1}_{\{s=y_i\}}\mathbf{1}_{\{t=y_{i-1}\}} \tag{3.17}$$

$$f_{s,a}(y_i, y_{i-1}, x_i) = \mathbf{1}_{\{s=y_i\}}\mathbf{1}_{\{a=x_i\}} \tag{3.18}$$

Then Equation 3.13 for the HMM can be written in the following compact form:

$$p(\mathbf{y},\mathbf{x}) = \frac{1}{Z}\prod_{i=1}^{L}\exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_i, y_{i-1}, x_i)\right) \tag{3.19}$$

where feature functions and parameters are indexed using a single index k that runs over all possible state-state pair (s,t) and state-observation pair (s,a).

Linear-chain CRFs can be obtained from Equation 3.19 considering the associated *conditional* distribution $p(\mathbf{y}|\mathbf{x})$:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{y},\mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}',\mathbf{x})} \\ &= \frac{\prod_{i=1}^{L} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right)}{\sum_{\mathbf{y}'} \prod_{i=1}^{L} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y'_i, y'_{i-1}, x_i)\right)} = \\ &= \frac{\prod_{i=1}^{L} \Psi_i(y_i, y_{i-1}, x_i)}{\sum_{\mathbf{y}'} \prod_{i=1}^{L} \Psi_i(y'_i, y'_{i-1}, x_i)} \end{aligned} \tag{3.20}$$

where we adopted the notation based on potential functions introduced for undirected graphical models $\Psi_i(y_i, y_{i-1}, x_i) = \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right)$.

Equation 3.20 represent a specific instance of linear-chain CRFs, namely the one obtained by considering feature functions defined over the current observation x_i. In general feature functions can depend on richer properties of the whole observation sequence $\mathbf{x}$ such as symmetric sliding windows or global descriptors. In other words, we allow more general features over the observation seqeuence other than the simple indicator function $\mathbf{1}_{\{s=y_i\}}\mathbf{1}_{\{a=x_i\}}$. In order to include this we need to slightly modify the notation by allowing potential functions to depend on the whole observation sequence

$$\Psi_i(y_i, y_{i-1}, \mathbf{x}) = \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, \mathbf{x})\right) \tag{3.21}$$

We are now able to formally define a linear-chain CRF:

Definition 3.4.1. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be random variables over observation and label sequences, respectively. Let* $\Theta = \{\lambda_k\} \in \mathbb{R}^K$ *be a parameter vector and* $F = \{f_k(y, y', \mathbf{x})\}_{k=1}^{K}$ *be a set of real-valued feature functions defined over labels pairs and the entire observation. A linear-chain Conditional Random Field is a conditional probability distribution of the form*

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{L} \Psi_i(y_i, y_{i-1}, \mathbf{x}) \tag{3.22}$$

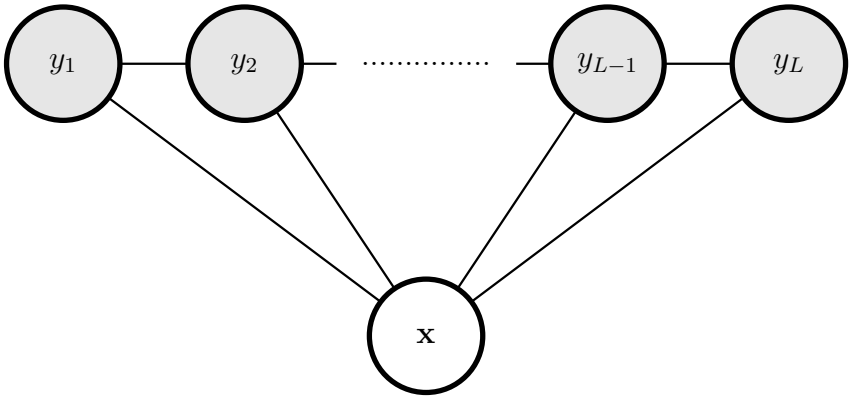


Figure 3.4: Graphical structure of a first-order linear-chain CRF.

where $Z(\mathbf{x})$ *is an observation specific normalization factor of partition function*

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{i=1}^{L} \Psi_i(y'_i, y'_{i-1}, \mathbf{x}) \tag{3.23}$$

3.4.2 Graphical structure of linear-chain CRFs

Linear-chain CRFs can be represented using an undirected graphical model, as showed in Figure 3.4. From the graphical structure one can desume that the set of variables corresponding to the label sequence $\mathbf{y}$ is *globally* conditioned on the entire observation sequence $\mathbf{x}$. The single node corresponding to $\mathbf{x}$ is depicted in white background to indicate that the internal structure of the observation is not modelled probabilistically but *only* its interactions with the labels. As a consequence of this fact, in CRFs the conditional probability distribution is defined in terms of general feature functions that are able to exploit arbitrary properties of the whole observation. Furthermore, as follows directly from the conditional independence notion of undirected graphical models, the label at time j is conditionally independent of any past label given the label at time $j-1$. In particular, being a linear-chain CRF an instance of an undirected graphical model in which the associated graph forms a chain, the distribution defined factorizes as in Equation 3.4. Indeed, in the simple assumption of a first-order model, maximal cliques correspond simply to edges of the chain and potential functions are then defined accordingly in Equation 3.21. In general, models of higher order can be also defined. Figure 3.5 shows the structure

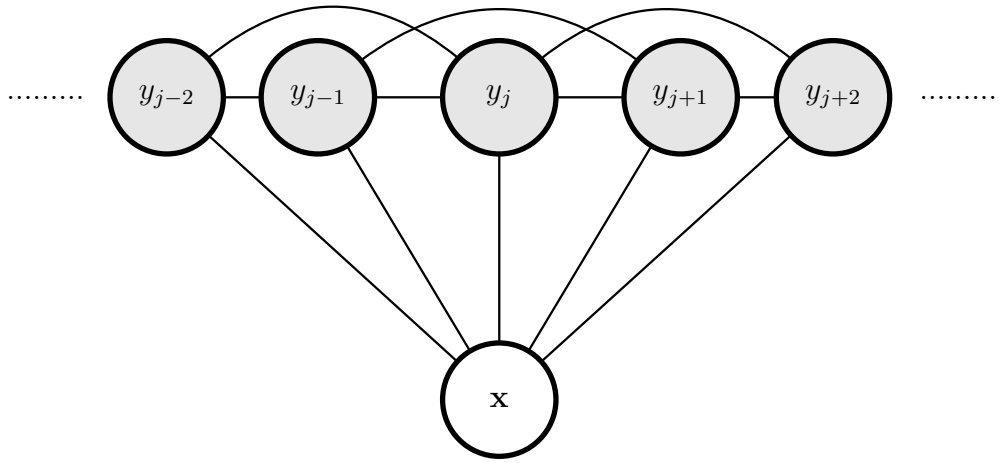


Figure 3.5: Graphical structure of a second-order linear-chain CRF.

of a second-order linear-chain CRF. In this case potential functions would be defined over triplets of labels to capture second-order dependencies:

$$\Psi_i(y_i, y_{i-1}, y_{i-2}, \mathbf{x}) = \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, y_{i-2}, \mathbf{x})\right) \tag{3.24}$$

3.4.3 Training CRFs

The problem of *training* a CRF can be stated as follows: given training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathcal{X}$ and $\mathbf{y}^{(i)} \in \mathcal{Y}$ and a feature set $F = \{f_k\}_{k=1}^{K}$ estimate the parameter vector $\Theta \in \mathbb{R}^K$. Training pairs $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ are identically and independently distributed whereas the i.i.d. assumption has been relaxed *within* each sequence.

This parameter estimation problem is usually performed by *regularized maximum likelihood*. In the case of linear-chain CRFs the following conditional log-likelihood is typically considered as objective function:

$$\ell(\Theta; \mathcal{D}) \;=\; \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}) \tag{3.25}$$

Substituting Equation 3.22 into the log-likelihood Equation 3.25 we obtain the fol-

lowing:

$$\ell(\Theta;\mathcal{D}) = \sum_{i=1}^{N}\sum_{j=1}^{L}\sum_{k=1}^{K}\lambda_k f_k(y_j^{(i)}, y_{j-1}^{(i)}, \mathbf{x}^{(i)}) - \log Z(\mathbf{x}) \quad (3.26)$$

Maximizing directly $\ell(\Theta;\mathcal{D})$ as defined in Equation 3.26 may lead to *overfitting* the training set $\mathcal{D}$. In order to reduce overfitting, $\ell(\Theta;\mathcal{D})$ is often *regularized* to penalize parameter vectors whose norm is too large:

$$\ell(\Theta;\mathcal{D}) = \sum_{i=1}^{N}\sum_{j=1}^{L}\sum_{k=1}^{K}\lambda_k f_k(y_j^{(i)}, y_{j-1}^{(i)}, \mathbf{x}^{(i)}) - \log Z(\mathbf{x}) - \rho\sum_{k=1}^{K}\lambda_k^2 \quad (3.27)$$

where the regularization is based on the Euclidean norm of Θ with a *regularization parameter* ρ governing the strength of the penalty term. In alternative to the Euclidean norm, the L_1 norm can be also used. In this case the regularization tends to favour *sparse* parameter vectors. Also combinations of the two norms are possible [3].

The objective function decribed in Equation 3.27 cannot in general be maximized in closed form. Numerical optimization is used instead. After some simple computations, the gradient of the log-likelihood can be obtained as follows:

$$\begin{aligned}\frac{\partial \ell}{\partial \lambda_k} &= \sum_{i=1}^{N}\sum_{j=1}^{L} f_k(y_j^{(i)}, y_{j-1}^{(i)}, \mathbf{x}^{(i)}) - \\ &\quad - \sum_{i=1}^{N}\sum_{j=1}^{L}\sum_{y,y'} f_k(y, y', \mathbf{x}^{(i)}) p(y_j = y, y_{j-1} = y' | \mathbf{x}^{(i)}) \end{aligned} \quad (3.28)$$

The first term is the expectation of the feature f_k under the empirical distribution, i.e. the value of the feature f_k computed over the entire training set. The second term is the expectation of the feature f_k under the model distribution $p(\mathbf{y}|\mathbf{x})$. In the case of unregularized likelihood, at the maximization solution (where the gradient is zero) the two expectations are equal and this corresponds to fit the model perfectly to the empirical distribution. Therefore, the regularization is needed to avoid overfitting.

From an optimization perspective, the function described in Equation 3.27 is strictly concave. This follows from the convexity of log-sum-exp functions. The concavity of likelihood function ensures it has a *global optimum*. Several methods can be used to perform the optimization:

- iterative scaling tecniques [63];
- simple steepest ascent along the gradient;
- Newton's method;
- quasi-Newton methods such as BFGS [83] and limited-memory BFGS (L-BFGS) [20];
- conjugate gradient;
- stochastic gradient methods [118].

Each method has its pros and cons. A common choice is to use the L-BFGS algorithm [20] that computes an approximation of the Hessian matrix using only the first-order derivative. Furthermore, L-BFGS uses limited memory to avoid storing the full Hessian approximation.

3.4.4 Inference algorithms for CRFs

Regardless of the specific optimization algorithm used, the evaluation of the gradient in Equation 3.28 requires the computation of marginal probabilities $p(y_j = y, y_{j-1} = y'|\mathbf{x})$. Secondly, computing the likelihood requires the partition function $Z(\mathbf{x})$. Finally, labelling a new observation sequence $\mathbf{x}$ requires the computation of the most likely label sequence $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x})$. All these tasks can be performed efficiently and exactly using variants of dynamic-programming algorithms of HMMs.

3.4.4.1 Computing marginal probabilities

The computation of marginal probabilities and the partition function is performed by means of the *forward-backward algorithm*. This algorithm is an instance of more general *sum-product algorithm* for inference in graphical models [15]. The

basic idea is to compute marginal transition probabilities at time j as:

$$\begin{aligned} p(y_j = y, y_{j-1} = y'|\mathbf{x}) &= \frac{\sum_{\{\mathbf{y}: y_j = y, y_{j-1} = y'\}} p(\mathbf{y}|\mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x})} \\ &= \frac{\sum_{\{\mathbf{y}: y_j = y, y_{j-1} = y'\}} \frac{1}{Z(\mathbf{x})} \prod_{j=2}^{L} \Psi_j(y_j, y_{j-1}, \mathbf{x})}{\sum_{\mathbf{y}'} \frac{1}{Z(\mathbf{x})} \prod_{j=2}^{L} \Psi_j(y'_j, y'_{j-1}, \mathbf{x})} \\ &= \frac{\sum_{\{\mathbf{y}: y_j = y, y_{j-1} = y'\}} \prod_{j=2}^{L} \Psi_j(y_j, y_{j-1}, \mathbf{x})}{\sum_{\mathbf{y}'} \prod_{j=2}^{L} \Psi_j(y'_j, y'_{j-1}, \mathbf{x})} \end{aligned} \tag{3.29}$$

where $\{\mathbf{y} : y_j = y, y_{j-1} = y'\}$ is the set of all label sequence having labels y' and y at positions $j-1$ and j, respectively. The numerator of Equation 3.29 is sum of scores of all label sequences which have a transition between labels y' and y at time j, whereas the denominator is the partition function, i.e. the sum of the scores of all label sequences. Equation 3.29 can be written in the following equivalent form:

$$p(y_j = y, y_{j-1} = y'|\mathbf{x}) = \frac{\alpha_{j-1}(y') \cdot \Psi_j(y, y', \mathbf{x}) \cdot \beta_j(y)}{Z(\mathbf{x})} \tag{3.30}$$

where:

$$\alpha_{j-1}(y') = \sum_{y_1, \ldots, y_{j-2}} \Psi_j(y', y_{j-2}, \mathbf{x}) \prod_{k=2}^{j-2} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \tag{3.31}$$

is the sum of scores of partial label sequences $y_1, \ldots, y_{j-2}, y'$ from time 1 to time $j-1$ ending with the label y', and:

$$\beta_j(y) = \sum_{y_{j+1}, \ldots, y_L} \Psi_{j+1}(y_{j+1}, y, \mathbf{x}) \prod_{k=j+2}^{L} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \tag{3.32}$$

is the sum of scores of partial label sequences from time j to L starting with the label y. $\alpha_{j-1}(y')$ is called the *forward variable* at time $j-1$ while $\beta_j(y)$ is the *backward variable* at time j. Figure 3.6 provides an illustration of Equation 3.30.

Forward and backward variables can be computed using dynamic programming.

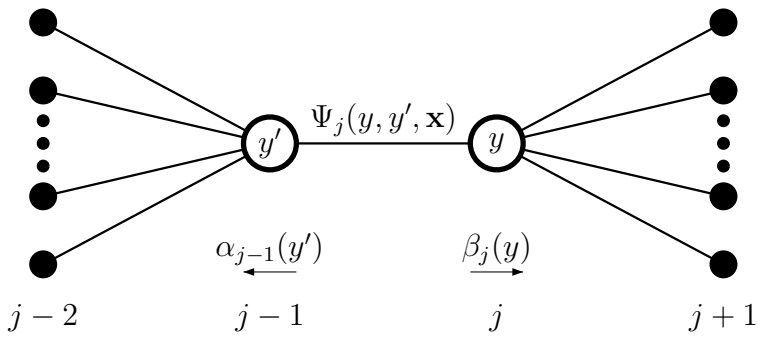


Figure 3.6: Illustration of the forward-backward procedure.

For forward variables $\forall y, j$ we have that:

$$\begin{aligned}
\alpha_j(y) &= \sum_{y_1,\ldots,y_{j-1}} \Psi_j(y, y_{j-1}, \mathbf{x}) \prod_{k=2}^{j-1} \Psi_k(y_k, y_{k-1}, \mathbf{x}) = \\
&= \sum_{y_1,\ldots,y_{j-2}} \sum_{y'} \Psi_j(y, y', \mathbf{x}) \Psi_{j-1}(y', y_{j-2}, \mathbf{x}) \prod_{k=2}^{j-2} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \sum_{y'} \Psi_j(y, y', \mathbf{x}) \sum_{y_1,\ldots,y_{j-2}} \Psi_{j-1}(y', y_{j-2}, \mathbf{x}) \prod_{k=2}^{j-2} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \sum_{y'} \Psi_j(y, y', \mathbf{x}) \alpha_{j-1}(y') \qquad (3.33)
\end{aligned}$$

Therefore $\alpha_j(y)$ can be obtained from values computed at time $j-1$ with initialization $\alpha_1(y) = \frac{1}{|\mathcal{Y}|}$. Similarly, backward variables $\beta_j(y)$ can be obtained from values computed at time $j+1$:

$$\begin{aligned}
\beta_j(y) &= \sum_{y_{j+1},\ldots,y_L} \Psi_{j+1}(y_{j+1}, y, \mathbf{x}) \prod_{k=j+2}^{L} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \sum_{y'} \Psi_{j+1}(y', y, \mathbf{x}) \sum_{y_{j+2},\ldots,y_L} \Psi_{j+2}(y_{j+2}, y', \mathbf{x}) \prod_{k=j+3}^{L} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \sum_{y'} \Psi_{j+1}(y', y, \mathbf{x}) \beta_{j+1}(y') \qquad (3.34)
\end{aligned}$$

with initialization $\beta_L(y) = \frac{1}{|\mathcal{Y}|}$. Finally, the partition function $Z(\mathbf{x})$ can be computed from forward or backward variables as follows:

$$Z(\mathbf{x}) = \sum_y \alpha_L(y) = \sum_y \beta_1(y) \qquad (3.35)$$

Algorithm 1 The forward algorithm for linear-chain CRFs.

for y **in** $\mathcal{Y}$ **do**
    $\alpha_1(y) \leftarrow \frac{1}{|\mathcal{Y}|}$
for $j = 2$ **to** L **do**
    for y **in** $\mathcal{Y}$ **do**
        $\alpha_j(y) \leftarrow 0$
        for y' **in** $\mathcal{Y}$ **do**
            $\alpha_j(y) \leftarrow \alpha_j(y) + \alpha_{j-1}(y') * \Psi_j(y', y, \mathbf{x})$

Algorithm 2 The backward algorithm for linear-chain CRFs.

for y **in** $\mathcal{Y}$ **do**
    $\beta_L(y) \leftarrow \frac{1}{|\mathcal{Y}|}$
for $j = L - 1$ **downto** 1 **do**
    for y **in** $\mathcal{Y}$ **do**
        $\beta_j(y) \leftarrow 0$
        for y' **in** $\mathcal{Y}$ **do**
            $\beta_j(y) \leftarrow \beta_j(y) + \beta_{j+1}(y') * \Psi_{j+1}(y', y, \mathbf{x})$

The pseudocodes of forward and backward algorithms are listed in Algorithm 1 and Algorithm 2, respectively.

3.4.4.2 Computing the most likely label sequence

Let $\mathbf{x}$ be an unseen observation sequence. Given a trained CRF model we would like to compute:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) \tag{3.36}$$

namely the most likely label sequence under the model.

The sequence $\mathbf{y}^*$ can be computed efficiently by the *Viterbi algorithm* which is an instance of the most general *max-product algorithm* for graphical models [15]. Let $\nu_j(y)$ be the score of the most probable label sequence of length j and ending in

label y:

$$\begin{aligned}
\nu_j(y) &= \max_{y_1,\ldots,y_{j-1}} \Psi_j(y, y_{j-1}, \mathbf{x}) \prod_{k=2}^{j-1} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \max_{y_1,\ldots,y_{j-2}} \max_{y'} \Psi_j(y, y', \mathbf{x}) \Psi_{j-1}(y', y_{j-2}, \mathbf{x}) \prod_{k=2}^{j-2} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \max_{y'} \Psi_j(y, y', \mathbf{x}) \max_{y_1,\ldots,y_{j-2}} \Psi_{j-1}(y', y_{j-2}, \mathbf{x}) \prod_{k=2}^{j-2} \Psi_k(y_k, y_{k-1}, \mathbf{x}) \\
&= \max_{y'} \Psi_j(y, y', \mathbf{x}) \nu_{j-1}(y') \qquad (3.37)
\end{aligned}$$

Even in this case, we obtain $\nu_j(y)$ using values computed in the previous step. This recursion is exactly the as same the one in Equation 3.33 except for the summation which is replaced by a maximization. Along with $\nu_j(y)$ we also need to mantain the actual state which maximize the score at time j:

$$\pi_j(y) = \operatorname*{argmax}_{y'} \Psi_j(y, y', \mathbf{x}) \nu_{j-1}(y') \qquad (3.38)$$

After computing the two values up to $j = L$ we can then apply a backtrace procedure to obtain the most probable label sequence $\mathbf{y}^*$ as follows:

$$y_L^* = \operatorname*{argmax}_{y} \nu_L(y) \qquad (3.39)$$

$$y_j^* = \pi_{j+1}(y_{j+1}^*) \qquad (3.40)$$

The pseudocode of the Viterbi algorithm is listen in Algorithm 3.

3.5 Generative and discriminative models

One of the main differences between HMMs and linear-chain CRFs is that HMMs belong to the class of *generative* models while CRFs are *discriminative* models. In general, a generative model provides a joint probability distribution over observations and labels $p(\mathbf{x}, \mathbf{y})$. This means that a generative model also includes a model for the distribution $p(\mathbf{x})$. On the other hand, discriminative models define conditional probability distributions of the form $p(\mathbf{y}|\mathbf{x})$.

Algorithm 3 The Viterbi algorithm for linear-chain CRFs.

for y **in** $\mathcal{Y}$ **do**
    $\nu_1(y) \leftarrow \frac{1}{|\mathcal{Y}|}$
for $j = 2$ **to** L **do**
    for y **in** $\mathcal{Y}$ **do**
        $\nu_j(y) \leftarrow 1$
        for y' **in** $\mathcal{Y}$ **do**
            if $\nu_j(y) * \nu_{j-1}(y') * \Psi_j(y, y', \mathbf{x}) > \nu_j(y)$ **then**
                $\nu_j(y) \leftarrow \nu_j(y) * \nu_{j-1}(y') * \Psi_j(y, y', \mathbf{x})$
                $\pi_j(y) = y'$
$y_L^* \leftarrow \max_y \nu_L(y)$
for $j = L - 1$ **downto** 1 **do**
    $y_j^* \leftarrow \pi_{j+1}(y_{j+1}^*)$

The main problem in modelling $p(\mathbf{x})$ derives from the fact that it may contain several highly dependent features that are difficult to model. In other words, providing an explicit and realistic probabilistic model of the observation $\mathbf{x}$ as in HMMs is not always possible, unless independence assumptions are made on elements of $\mathbf{x}$. For instance, in protein secondary structure prediction, the different positions in the protein are often highly dependent between each other. Providing a model of these dependencies using an HMM is very difficult and in general they are simply neglected.

The main advantage of using discriminative models such as CRFs is that it is very easy to incorporate into the model rich overlapping features of the observation without spending efforts in modelling these dependencies probabilistically. In other words, CRFs made independence assumptions on the label sequence $\mathbf{y}$ and *relax* independence assumptions on the observation $\mathbf{x}$.

As we saw in Section 3.4.1, HMMs and linear-chain CRFs are closely related since we can derive a CRF as the associated conditional distribution of a HMM. This fact can be stated by saying that HMMs and CRFs are a *discriminative-generative* pair. The same argument can be applied, for instance, to Naive Bayes and Logistic

Regression models [109]. However, a CRFs describes a broader class of distributions than HMMs since it can incorporate several features of the observation without take care of providing for them a probabilistic model.

3.6 Summary

Often, biological data representing proteins or nucleic acids are available in the form of sequences. In these contexts, sequence analysis tasks aim at providing position-specific annotations for the sequence being analyzed. These annotations typically correspond to features of interest of the sequence such as secondary structure or transmembrane topology.

Probabilistic models for sequence analysis are very powerful tools and have been extensively applied in Computational Biology. Historically, Hidden Markov Models represent the most popular probabilistic framework in many fields and in particular in Bioinformatics. HMMs are generative models of the joint probability distribution $p(\mathbf{y}, \mathbf{x})$ of a label sequence and an observation sequence. An alternative approach is represented by linear-chain Conditional Random Fiedls. In contrast to HMMs, CRFs are discriminative models that allows to overcome several limitations of generative approaches. In particular, a CRF provide the conditional probability $p(\mathbf{y}|\mathbf{x})$ of a label sequence given an observation sequence. By this, since no explicit probabilistic model of the observation is provided, CRFs are well-suited to exploit arbitrary interdependent features of the observation without introducing additional complexity into the model. In many applications, discriminative approaches have been proven to be much more effective than generative ones.

Chapter 4

Grammatical-Restrained Hidden CRFs

4.1 Introduction

Linear-chain CRFs introduced in Chapter 3 are very powerful tools and they have been applied in several scientific fields to solve sequence labelling tasks [106, 89, 106, 68, 27, 102].

A limitation of linear-chain CRFs is that they are *fully observable* models. This means that they require that label sequences are available and completely observable at training time. In other words liner-chain CRFs do not include any *hidden variable* into the model. This limitation can seriously affect the modelling capability of CRFs, especially on real-world problems that can be easily modelled by considering hidden sub-structures underlying the label dynamics which are usually not directly observable at training time. These problems include Object or Gesture Recognition in Computer Vision [94, 123] and several labelling tasks in Computational Biology [31].

For this reason, extensions of linear-chain CRFs exist to include into the original model an additional set of hidden variables. Hidden Conditional Random Fields (HCRF) have been introduced in the field of Object Recognition [94]. In this classification task, the goal is to assign a given image to a category associated to the (main) object present in the image. Authors have showed how the discriminative approach with hidden variables modelling spatial dependencies between different parts of the image could outperform both generative approaches and standard CRFs [94].

Similar discriminative hidden-varaibles models have been adopted for gesture recognition [123] and phone classification [41].

Another limitation related to the fully observability of CRFs models is that they are not well-suited for those sequence analisys problems which can be successfully addressed only by designing a regular grammar in order to provide meaningful results. These problems typically arise in Computational Biology. For instance in gene prediction tasks exons must be linked in such a way that the donor and acceptor junctions define regions whose length is a multiple of three (according to the genetic code), and in protein secondary structure prediction, helical segments shorter than 4 residues should be consider meaningless, being this the shortest allowed length for a protein helical motif [28, 8].

In this kind of problems, the training sets generally consist of pair of observed and label sequences and very often the number of the different labels representing the experimental evidence is small compared to the grammar requirements and the length distribution of the segments for the different labels. Then, fully-observable models such as CRFs result in poor predictive performances. On the contrary, when in HMMs hidden states and labels are decoupled, it is possible to model a huge number of concurring paths compatible with the grammar and with the experimental labels without increasing the time and space computational complexity [62, 28].

In order to overcome these limitations, we introduced Grammatical-Restrained Hidden CRFs(GRHCRFs) [31]. The model is obtained as an extension of HCRFs and the main properties can be summarized as follows:

- differently from HCRFs which are well-suited to solve *structured input classification* problems i.e. mapping a structured object into an *unique* class label, GRHCRFs are discriminative hidden-variable models designed to map observation sequences to label sequences;

- the GRHCRF is restricted in order consider only hidden-state sequences that are in agreement with a *regular grammar*. In other words, the model *accepts* state sequences that belong to the language generated by a grammar $\mathcal{G}$;

- in order to take advantage of the hidden-state layer of variables introduced, a different decoding algorithm is introduced for GRHCRFs.

A brief introduction to HCRFs is provided in Section 4.2. The mathematical formulation of the model and the parameter estimation procedure are discussed in Section 4.3 and Section 4.4, respectively. Decoding algorithms for GRHCRFs are presented in Section 4.5. Concluding remarks are discussed in Section 4.7.

4.2 Hidden CRFs

The idea of extending the original linear-chain CRF with hidden variables has been introduced for the first time in the field of Object Recognition [94]. In this section a short description of HCRF is provided on the basis of this first formulation.

We consider the general problem of learning a mapping from a structured object $\mathbf{x} \in \mathcal{X}$ (e.g. a protein sequence or an image) to a *single* class label $y \in \mathcal{Y}$. The input object $\mathbf{x}$ is typically a collection of L elements $(x_1, \ldots, x_L)$. For instance if $\mathbf{x}$ is a protein sequence, x_j represent the j-th amino acid in the sequence while if $\mathbf{x}$ is an image, $(x_1, \ldots, x_L)$ can be the set of *patches* or *local feature vectors* of the image. The training data consist of labeled objects $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$. We also assume the presence of hidden variables $(h_1, \ldots, h_L)$ that are not observed at training time. Each hidden state belongs to a finite set $\mathcal{H}$ of possible states. In object recognition an hidden state can represent a *part label* assigned to each image patch [94]. In protein sequences hidden states are typically used to identify varaible-length regions of the sequence with different physicochemical properties e.g. segments with different secondary structure. Although in general $\mathbf{x}$ and $\mathbf{h}$ may have an arbitrary internal structure (e.g. trees or graphs), we restrict our attention to the case in which both $\mathbf{x}$ and $\mathbf{h}$ are sequences.

A linear-chain HCRF can be represented with the undirected graphical model in Figure 4.1. Maximal cliques in this structure are triples (h_j, h_{j-1}, y) assuming a linear-chain first-order assumption. Consequently, according with the MRF formulation described in Chapter 3, potential functions are defined as follows:

$$\Psi_j(h_j, h_{j-1}, y, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(h_j, h_{j-1}, y, \mathbf{x})\right) \tag{4.1}$$

Given the above definitions, a conditional distribution over labels y and hidden

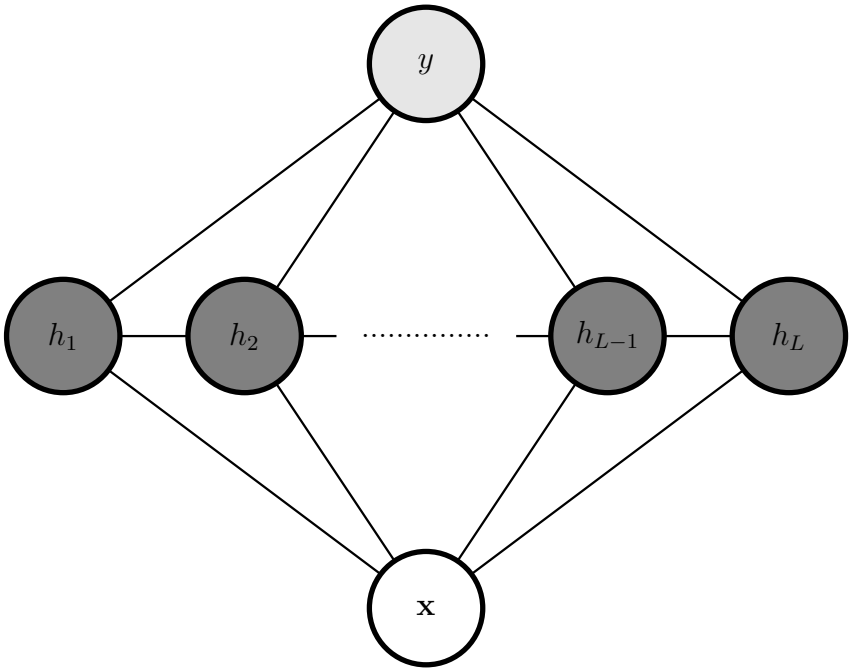


Figure 4.1: Graphical structure of a first-order linear-chain HCRF.

states $\mathbf{h}$ given an observation $\mathbf{x}$ has the following form:

$$\begin{aligned} p(y, \mathbf{h}|\mathbf{x}) &= \frac{\prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y, \mathbf{x})}{\sum_{y', \mathbf{h}'} \prod_{j=1}^{L} \Psi_j(h'_j, h'_{j-1}, y', \mathbf{x})} \\ &= \frac{1}{Z(\mathbf{x})} \prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y, \mathbf{x}) \end{aligned} \tag{4.2}$$

Formally, a Hidden CRF is then defined as follows:

Definition 4.2.1. *Let* $\mathbf{x}$, $\mathbf{h}$ *and* y *be random variables over observation and hidden state sequences and labels, respectively. Let* $\Theta = \{\lambda_k\} \in \mathbb{R}^K$ *be a parameter vector and* $F = \{f_k(h, h', y, \mathbf{x})\}_{k=1}^{K}$ *be a set of real-valued feature functions defined over hidden state pairs, labels and the entire observation. A linear-chain Hidden Conditional*

Random Field is a conditional probability distribution of the form:

$$\begin{aligned} p(y|\mathbf{x}) &= \sum_{\mathbf{h}} p(y, \mathbf{h}|\mathbf{x}) = \\ &= \frac{1}{Z(\mathbf{x})} \sum_{\mathbf{h}} \prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y, \mathbf{x}) = \\ &= \frac{Z(y, \mathbf{x})}{Z(\mathbf{x})} \end{aligned} \tag{4.3}$$

where $Z(y, \mathbf{x})$ and $Z(\mathbf{x})$ are instance-specific partition functions:

$$Z(y, \mathbf{x}) = \sum_{\mathbf{h}} \prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y, \mathbf{x}) \tag{4.4}$$

$$Z(\mathbf{x}) = \sum_{y} \sum_{\mathbf{h}} \prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y, \mathbf{x}) \tag{4.5}$$

.

4.3 Grammatical-Restrained Hidden CRFs: model formulation

Grammatical-Restrained HCRFs extend the HCRF model to handle sequences of labels $\mathbf{y} = (y_1, \ldots, y_L)$ rather than a single class y. Furthermore, in GRHCRFs the model is constrained in order to consider only sequences of hidden states that are in agreement with a regular grammar. This section is devised to the description of the model.

We address the problem of mapping an observation sequence $\mathbf{x} = (x_1, \ldots, x_L)$ to a label sequence $\mathbf{y} = (y_1, \ldots, y_L)$. As done for HCRFs, we also introduce hidden-state sequences $\mathbf{h} = (h_1, \ldots, h_L)$. Each label in the sequence belongs to a finite set $\mathcal{Y}$ of possible labels. Similarly, hidden states are members of a set of possible states $\mathcal{H}$. Furthermore, labels are associated to disjoint sets of states. In other words, each state $h \in \mathcal{H}$ belongs to a subset $\mathcal{H}_y \subset \mathcal{H}$ of states associated to the label $y \in \mathcal{Y}$ and we have:

$$\mathcal{H} = \bigcup_{y \in \mathcal{Y}} \mathcal{H}_y \tag{4.6}$$

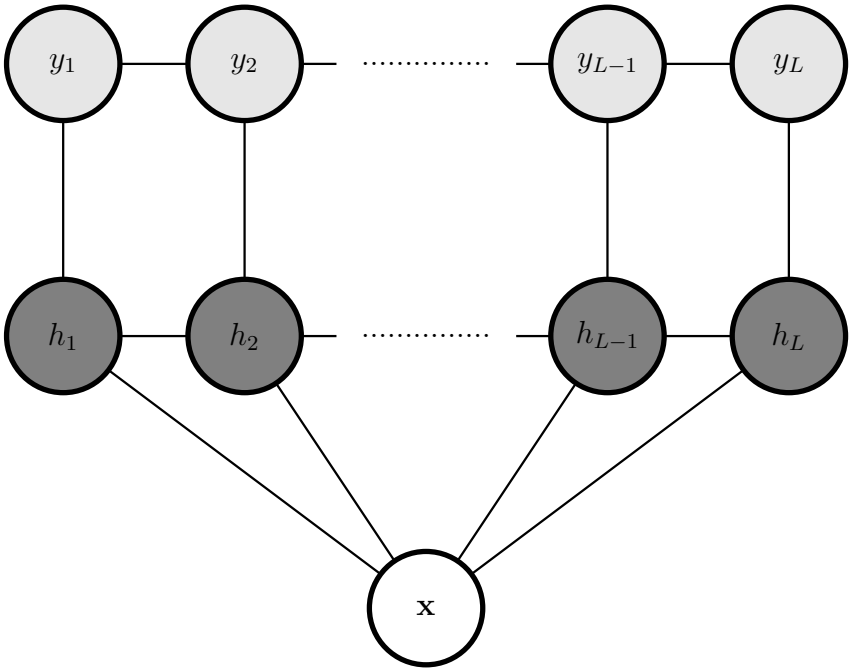


Figure 4.2: Graphical structure of a first-order linear-chain GRHCRF.

We mantain the sets $\mathcal{H}_y$ disjoint in order to keep the inference problem in the model tractable.

We want to restrict our model so that a sequence of hidden states is allowed only if it is in agreement with a regular grammar $\mathcal{G}$. We define grammar constraints as follows:

$$\Gamma(h, h') = \begin{cases} 1 & \text{if the transition } h' \rightarrow h \text{is allowed by } \mathcal{G} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

The GRHCRF model is represented graphically in Figure 4.2. With this graphical structure, being cliques defined over (h_j, h_{j-1}, y_j), we define potential functions which take into consideration grammatical constraints:

$$\Psi_j(h_j, h_{j-1}, y_j, \mathbf{x}) \;=\; \exp\left(\sum_k \lambda_k f_k(h_j, h_{j-1}, \mathbf{x})\right) \cdot \Gamma(h_j, h_{j-1}) \cdot \mathbf{1}_{\{h_j \in H_{y_j}\}} \tag{4.8}$$

where, using the indicator function $\mathbf{1}_{\{h_j \in H_{y_j}\}}$ we also require that the state at time j is compatible with the corresponding label.

The probability distribution of label sequence given an observable sequence $p(\mathbf{y}|\mathbf{x})$ can be defined including hidden states as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y},\mathbf{h}|\mathbf{x})}{p(\mathbf{h}|\mathbf{y},\mathbf{x})} \tag{4.9}$$

The joint probability of a label sequence $\mathbf{y}$ and an hidden-state sequence $\mathbf{h}$ given an observation sequence $\mathbf{x}$ is:

$$p(\mathbf{y},\mathbf{h}|\mathbf{x}) = \frac{\prod_{j=1}^{L}\Psi_j(h_j,h_{j-1},y_j,\mathbf{x})}{Z(\mathbf{x})} \tag{4.10}$$

where, in analogy with HCRFs, $Z(\mathbf{x})$ is a partition function obtained summing over all possible label and states sequences as follows:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}}\sum_{\mathbf{h}}\prod_{j=1}^{L}\Psi_j(h_j,h_{j-1},y_j,\mathbf{x}) \tag{4.11}$$

The probability of an hidden-state sequence given a label sequence and an observation sequence is:

$$\begin{aligned} p(\mathbf{h}|\mathbf{y},\mathbf{x}) &= \frac{p(\mathbf{y},\mathbf{h}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})} = \\ &= \frac{\prod_{i=1}^{L}\Psi(h_i,h_{i-1},y_j,\mathbf{x})}{Z(\mathbf{y},\mathbf{x})} \end{aligned} \tag{4.12}$$

where the partition function $Z(\mathbf{y},\mathbf{x})$ is obtained by keeping fixed the label sequence and summing over all possible corresponding state sequences as follows:

$$Z(\mathbf{y},\mathbf{x}) = \sum_{\mathbf{h}}\prod_{j=1}^{L}\Psi_j(h_j,h_{j-1},y_j,\mathbf{x}) \tag{4.13}$$

Given the above distributions we formally define the Grammatical-Restrained Hidden CRF:

Definition 4.3.1. *Let* $\mathbf{x}$, $\mathbf{h}$ *and* $\mathbf{y}$ *be random variables over observation, hidden state and label sequences, respectively. Let* $\Theta = \{\lambda_k\} \in \mathbb{R}^K$ *be a parameter vector and* $F = \{f_k(h,h',y,\mathbf{x})\}_{k=1}^{K}$ *be a set of real-valued feature functions defined over*

hidden state pairs, labels and the entire observation. A linear-chain Grammatical-Restrained Hidden Conditional Random Field is a conditional probability distribution of the form:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{y},\mathbf{h}|\mathbf{x})}{p(\mathbf{h}|\mathbf{y},\mathbf{x})} = \\ &= \frac{Z(\mathbf{y},\mathbf{x})}{Z(\mathbf{x})} \end{aligned} \tag{4.14}$$

where $Z(\mathbf{y},\mathbf{x})$ *and* $Z(\mathbf{x})$ *are instance-specific partition functions*

$$Z(\mathbf{y},\mathbf{x}) = \sum_{\mathbf{h}} \prod_{j=1}^{L} \Psi_j(h_j, h_{j-1}, y_j, \mathbf{x}) \tag{4.15}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{\mathbf{h}} \prod_{j=1}^{L} \Psi(h_j, h_{j-1}, y_j, \mathbf{x}) \tag{4.16}$$

.

4.3.1 Difference with linear-chain CRFs

The main difference between standard linear-chain CRFs and GRHCRFs lies in the fact that the former may enforce grammatical constraints on label sequences whereas the latter separates hidden states and labels and apply grammatical rules on hidden state sequences. To better highlight this difference and how it affects the capabilities of the two models it is useful to think about both GRHCRFs and linear-chain CRFs in terms of *Finite-State Transducers* (FST). In contrast to ordinary Finite-State Automata (FSA) which have a single tape [1], a FST is a finite-state machine with two tapes, called the *input* and the *output* tapes. An FST works by *transducing* (i.e. translating) the content of the input tape into the output tape. By this, the machine computes a relation between two regular languages.

In this view, a standard linear-chain CRF is equivalent to the FST shown in Figure 4.3. In the case of linear-chain CRF, the FST is defined such that the number of states equals the number of different labels (in the example there are three labels) and the function computed by the machine is simply the identity function. In other

[1] This single tape can be alternatively seen as an input tape in *recognizers* or as an output tape for generative automata

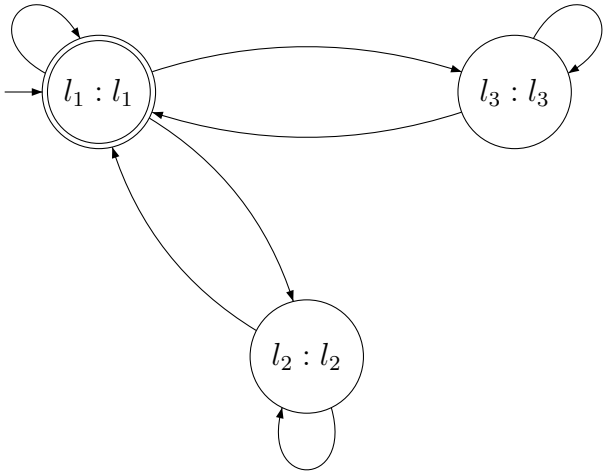


Figure 4.3: A FST associated to a linear-chain CRF.

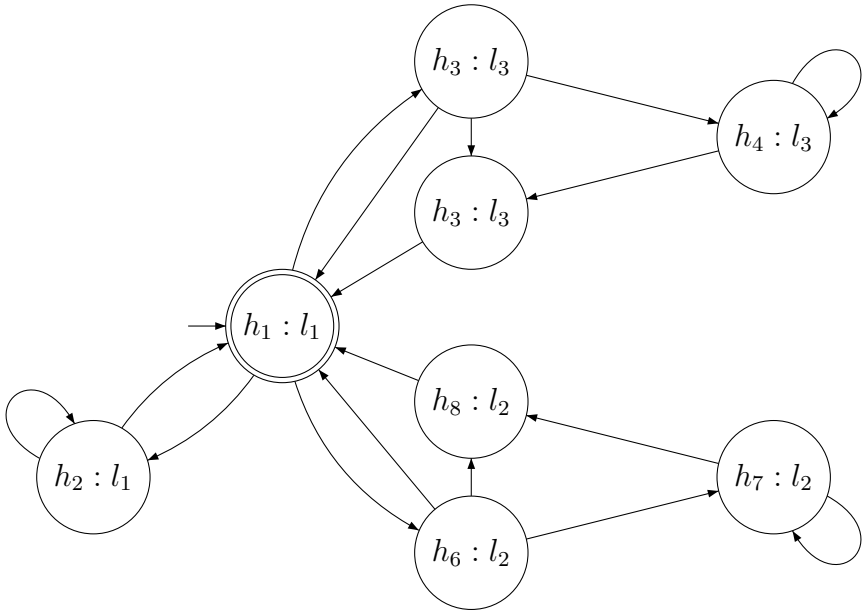


Figure 4.4: A FST associated to a GRHCRF.

words, the FST simply copies a label sequence from the input tape into the same sequence in the output tape. On the contrary, the GRHCRF model *decouples* hidden states from labels, meaning that the corresponding FST reads a sequence of hidden states on the input tape and transduces it into a sequence of labels. Furthermore, this translation is performed so that a single label sequence can be generated by multiple hidden-state sequences. An example is shown in Figure 4.4. By this, the

capability of the model is improved since arbitrary problem-specific FSTs can be defined with the only restriction that they must be deterministics, namely a given hidden-state sequence on the input tape may generate a unique label sequence. This restriction is represented by the condition in Equation 4.6 and the requirement that the sets $\mathcal{H}_y$ are disjoint.

In summary, any linear-chain CRF can be defined as a GRHCRF where the relation computed by the corresponding FST is a bijection, namely a one-to-one mapping between state and label sequences. However, GRHCRF models can be defined such that this condition is not met, meaning the set of linear-chain CRF models is a subset of all possible models that can be defined using the more expressive GRHCRF framework.

4.4 Parameter estimation

Given a training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ of independent and identically distributed labelled observation sequences, the parameters Θ of a GRHCRF model can be obtained by maximum likelihood estimation. The log-likelihood of data is:

$$\begin{aligned}
\ell(\Theta; \mathcal{D}) &= \log \prod_{i=1}^{N} p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \\
&= \log \prod_{i=1}^{N} \frac{Z(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}{Z(\mathbf{x}^{(i)})} \\
&= \sum_{i=1}^{N} \log Z(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) \qquad (4.17)
\end{aligned}$$

Taking the first derivative with respect to parameter λ_k of the objective function we obtain:

$$\frac{\partial \ell(\Theta; \mathcal{D})}{\partial \lambda_k} = \underbrace{\sum_{i=1}^{N} \frac{\partial}{\partial \lambda_k} \log Z(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}_{\mathcal{C}} - \underbrace{\sum_{i=1}^{N} \frac{\partial}{\partial \lambda_k} \log Z(\mathbf{x}^{(i)})}_{\mathcal{F}} \qquad (4.18)$$

where, in analogy with the Boltzmann machines and HMMs for labelled sequences [62], $\mathcal{C}$ and $\mathcal{F}$ can be seen as *clamped* and *free* phases. After simple computations we can

rewrite the derivative as:

$$\frac{\partial \ell(\Theta; \mathcal{D})}{\partial \lambda_k} = E_{p(\mathbf{h}|\mathbf{y},\mathbf{x})}[f_k] - E_{p(\mathbf{h},\mathbf{y}|\mathbf{x})}[f_k] \tag{4.19}$$

where the $E_{p(\mathbf{h}|\mathbf{y},\mathbf{x})}[f_k]$ and $E_{p(\mathbf{h},\mathbf{y}|\mathbf{x})}[f_k]$ are the expected values of the feature function f_k computed in the clamped and free phases, respectively.

To avoid overfitting, we regularize the objective function using a Gaussian prior, so that the function to maximize has the form of:

$$\ell(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \log Z(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) - \sum_{k} \frac{{\lambda_k}^2}{2\sigma^2} \tag{4.20}$$

and the corresponding gradient is:

$$\frac{\partial \ell(\Theta; \mathcal{D})}{\partial \lambda_k} = E_{p(\mathbf{h}|\mathbf{y},\mathbf{x})}[f_k] - E_{p(\mathbf{h},\mathbf{y}|\mathbf{x})}[f_k] - \frac{\lambda_k}{\sigma^2} \tag{4.21}$$

4.4.1 Computing the expectations

Differently from the linear-chain CRF, both expectations in Equation 4.19 must be computed using the forward and backward algorithms. These algorithms are adaptations of the algorithms presented in the previous chapter for linear-chain CRFs that take into consideration the grammar constraints.

Let be **BEGIN** and **END** to special hidden states that are used to model the beginning and the end of a sequence. Hidden states that are *initial* states will have an allowed incoming transition from **BEGIN**. States that are *final* will have an allowed transition outgoing to **END**. In this way we virtually extends sequences of length L to $L + 2$ adding at the beginning and at the end the two special states.

In both the clamped and the free phases we want the algorithms to consider only grammatically correct hidden state paths. Furthermore, in the clamped phase the hidden state path must be compatible with the experimental labelling. According to these observations, we introduce two different definitions of potential functions,

one for the clamped phase and one for the free phase:

$$\Psi_j^{\mathcal{C}}(h_j, h_{j-1}, y_j, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(h_j, h_{j-1}, \mathbf{x})\right) \cdot \Gamma(h_j, h_{j-1}) \cdot \mathbf{1}_{\{h_j \in H_{y_j}\}} \tag{4.22}$$

$$\Psi_j^{\mathcal{F}}(h_j, h_{j-1}, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(h_j, h_{j-1}, \mathbf{x})\right) \cdot \Gamma(h_j, h_{j-1}) \tag{4.23}$$

where for the clamped phase we use exactly the same definition of Equation 4.8 whereas for the free phase we exploit the fact that each hidden state is uniquely associated to a label and hence all possible label sequences can be simply obtained by considering all possible grammatically correct hidden state sequences[2].

Using these definitions, the recursion rules of the forward algorithm for the free phase are then defined as follows:

$$\alpha_j^{\mathcal{F}}(h) = \sum_{h' \in \mathcal{H}} \alpha_{j-1}^{\mathcal{F}}(h') \Psi_j^{\mathcal{F}}(h, h', \mathbf{x}) \tag{4.24}$$

with the following initialization:

$$\alpha_0^{\mathcal{F}}(h) = \begin{cases} 1 & \text{if } h = \textbf{BEGIN} \\ 0 & \text{otherwise} \end{cases} \tag{4.25}$$

The clamped phase require a slight modification to enforce the hidden state at time j to be compatible with the corresponding experimental label:

$$\alpha_j^{\mathcal{C}}(h|\mathbf{y}) = \sum_{h' \in \mathcal{H}} \alpha_{j-1}^{\mathcal{C}}(h') \Psi_j^{\mathcal{C}}(h, h', y_j, \mathbf{x}) \tag{4.26}$$

with initialization:

$$\alpha_0^{\mathcal{C}}(h|\mathbf{y}) = \begin{cases} 1 & \text{if } h = \textbf{BEGIN} \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

Analogously, the backward variables can be computed for the free phase as follows:

$$\beta_j^{\mathcal{F}}(h) = \sum_{h' \in \mathcal{H}} \beta_{j+1}^{\mathcal{F}}(h') \Psi_{j+1}^{\mathcal{F}}(h', h, \mathbf{x}) \tag{4.28}$$

[2]To better understand this fact, note also that the partition function $Z(\mathbf{x})$ in Equation 4.11 can be alternatively written as $Z(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{\mathbf{h}:\forall j, h_j \in \mathcal{H}_{y_j}} \Psi_j^{\mathcal{F}}(\cdot) = \sum_{\mathbf{h}} \Psi_j^{\mathcal{F}}(\cdot)$, where in the last step we used the observation above.

and initialized as:

$$\beta^{\mathcal{F}}_{L+1}(h) = \begin{cases} 1 & \text{if } h = \textbf{END} \\ 0 & \text{otherwise} \end{cases} \tag{4.29}$$

Finally for the clamped phase we fix the label at time $j+1$:

$$\beta^{\mathcal{C}}_{j}(h|\mathbf{y}) = \sum_{h' \in \mathcal{H}} \beta^{\mathcal{C}}_{j+1}(h') \Psi^{\mathcal{C}}_{j+1}(h', h, y_{j+1}, \mathbf{x}) \tag{4.30}$$

and initialized as:

$$\beta^{\mathcal{C}}_{L+1}(h|\mathbf{y}) = \begin{cases} 1 & \text{if } h = \textbf{END} \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

The expectations of the feature functions ($E_{p(\mathbf{h}|\mathbf{y},\mathbf{x})}[f_k]$, $E_{p(\mathbf{h},\mathbf{y}|\mathbf{x})}[f_k]$) are then computed as:

$$\begin{aligned} E_{p(\mathbf{h}|\mathbf{y},\mathbf{x})}[f_k] &= \sum_{j=1}^{L+1} \sum_{h',h \in \mathcal{H}} f_k(h, h', \mathbf{x}) p(h_j = h, h_{j-1} = h'|\mathbf{y}, \mathbf{x}) = \\ &= \sum_{j=1}^{L+1} \sum_{h',h \in \mathcal{H}} f_k(h, h', \mathbf{x}) \frac{\alpha^{\mathcal{C}}_{j-1}(h'|\mathbf{y}) \Psi^{\mathcal{C}}_{j}(h, h', y_j, \mathbf{x}) \beta^{\mathcal{C}}_{j}(h|\mathbf{y})}{Z(\mathbf{y}, \mathbf{x})} \end{aligned} \tag{4.32}$$

$$\begin{aligned} E_{p(\mathbf{h},\mathbf{y}|\mathbf{x})}[f_k] &= \sum_{j=1}^{L+1} \sum_{h',h \in \mathcal{H}} f_k(h, h', \mathbf{x}) p(h_j = h, h_{j-1} = h'|\mathbf{x}) \\ &= \sum_{j=1}^{L+1} \sum_{h',h \in \mathcal{H}} f_k(h, h', \mathbf{x}) \frac{\alpha^{\mathcal{F}}_{j-1}(h') \Psi^{\mathcal{F}}_{j}(h, h', \mathbf{x}) \beta^{\mathcal{F}}_{j}(h)}{Z(\mathbf{x})} \end{aligned} \tag{4.33}$$

Partition functions are obtained from forward or backward algorithms as:

$$\begin{aligned} Z(\mathbf{y}, \mathbf{x}) &= \alpha^{\mathcal{C}}_{L+1}(\textbf{END}|\mathbf{y}) = \beta^{\mathcal{C}|\mathbf{y}}_{0}(\textbf{BEGIN}) \\ Z(\mathbf{x}) &= \alpha^{\mathcal{F}}_{L+1}(\textbf{END}) = \beta^{\mathcal{F}}_{0}(\textbf{BEGIN}) \end{aligned} \tag{4.34}$$

4.5 Decoding algorithms

Decoding is the task of assigning labels $\mathbf{y}$ to an unknown observation sequence $\mathbf{x}$. Viterbi algorithm is routinely applied as decoding for the linear-chain CRFs, since

it finds the most probable path of an observation sequence given a CRF model [63]:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) \tag{4.35}$$

In CRF models with hidden variables, the Viterbi algorithm is used to search the hidden state space rathan than the label space. In this context, the algorithm is particular effective when there is a single strong highly probable hidden state path, while when several paths compete (have similar probabilities), *posterior decoding* may perform significantly better. Whit this approach, for each position the label is assigned according to its posterior probability $p(y_j = y|\mathbf{x})$ as follows:

$$y_j^* = \max_{y} p(y_j = y|\mathbf{x}) \tag{4.36}$$

which is equivalent to:

$$\mathbf{y}^* = \underset{\mathbf{y}'}{\operatorname{argmax}} \prod_{j=1}^{L} p(y_j = y'_j|\mathbf{x}) \tag{4.37}$$

However, the selected label sequence of the posterior decoding may not be allowed by the grammar. A simple solution of this problem is provided by the posterior-Viterbi decoding, that was previously introduced for HMMs [30]. Posterior-Viterbi, exploits the posterior probabilities and at the same time preserves the grammatical constraint.

The algorithm works by considering, for any position j in the sequence and label y, the posterior probability of reaching some state h associated with the label y. This probability is given as follows:

$$p(h_j \in \mathcal{H}_y|\mathbf{x}) = \sum_{h \in \mathcal{H}_y} p(h_j = h|\mathbf{x}) \tag{4.38}$$

i.e. summing over posterior probabilities $p(h_j = h|\mathbf{x})$ of all possible states $h \in \mathcal{H}_y$.

After this step, at any given position, we have the posterior probabilities of reaching different disjoint regions of the hidden state space that correspond to the different labels. By applying a Viterbi search over these posteriors we can obtain the labelling as in Equation 4.36 and at the same time preserving the grammatical constraints.

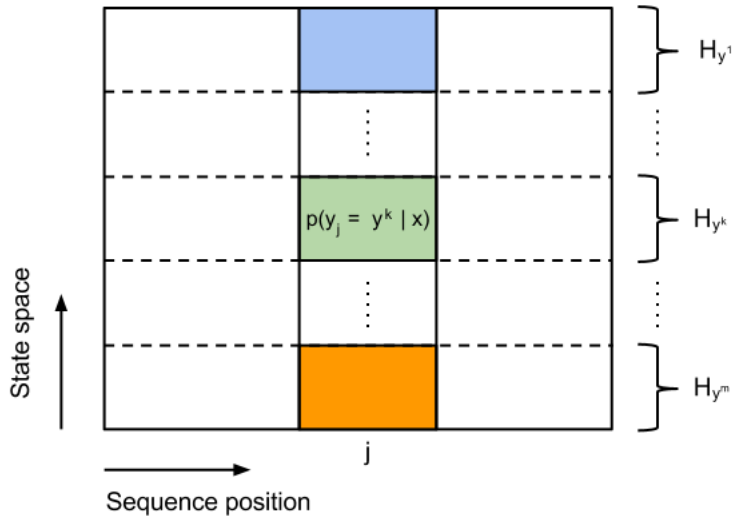


Figure 4.5: An illustration of the posterior matrix M.

The first step can be accomplished using the Forward-Backward algorithm as described for the free phase of parameter estimation. In particular, the posterior probability of a hidden state is obtained as:

$$p(h_j = h|\mathbf{x}) \quad = \quad \frac{\alpha_j^{\mathcal{F}}(h)\beta_j^{\mathcal{F}}(h)}{Z(\mathbf{x})} \tag{4.39}$$

Let $M(h, j)$ be the matrix obtained as:

$$M(h, j) \quad = \quad \sum_{h' \in \mathcal{H}_{\Lambda(h)}} p(h_j = h'|\mathbf{x}) \tag{4.40}$$

where $\Lambda(h) = y$ is the function that returns for each state the associated label y. The matrix M is such that if h, h' are associated to the same label y then $M(h, j) = M(h', j), \forall j$ as shown in Figure 4.5.

By performing a grammatical constrained Viterbi search over the matrix M we can solve the maximization problem in Equation 4.36. We define $\rho_j(h)$ as the product of posterior probabilities of the partial state sequence $h_1, \ldots, h_{j-1}, h_j = h$ of length j ending in state h while $\pi_j(h)$ is a traceback pointer. The algorithm proceeds as follows:

1. Initialization:

$$\rho_0(h) = \begin{cases} 1 & \text{if } h = \textbf{BEGIN} \\ 0 & \text{otherwise} \end{cases} \tag{4.41}$$

2. Recursion:

$$\rho_j(h) = \max_{h'} \rho_{j-1}(h')\Gamma(h', h)M(h, j) \tag{4.42}$$

$$\pi_j(h) = \operatorname*{argmax}_{h'} \rho_{j-1}(h')\Gamma(h', h)M(h, j) \tag{4.43}$$

3. Termination and Traceback

$$h^*_{n+1} = \textbf{END} \tag{4.44}$$

$$h^*_j = \pi_{j+1}(h^*_{j+1}) \quad for \quad j = n, n-1, \ldots, 1 \tag{4.45}$$

$$h^*_0 = \textbf{BEGIN} \tag{4.46}$$

The labels are assigned to the observed sequence according to the state path $\mathbf{h}^*$.

4.6 Experiments

Simple experiments on synthetic data were performed in order to test the different modelling capabilities of GRHCRFs and linear-chain CRFs. For sake of simplicity, the *occasionally disohonest casino* problem was chosen as test case in experiments. In this toy problem, we model the situation of a casino that uses two kind of dice:

- *fair* dice with uniform probability over the six possible outcomes;
- *loaded* dice with biased probability to throw one of the numbers;

An instance of the disohonest casino is shown in Figure 4.6, where the problem is modelled using an HMM with five states with different emission and transition probabilities (only the latter are shown in figure as edge labels). The state f_1 corresponds to the (unique) fair die with uniform emission probabilities whereas states l_i represent loaded dice. The HMM model has been defined so that the loaded die l_i emits the outcome i with probability 0.6 and outcomes $j \neq i$ with probability 0.08. The complete emission probability matrix is shown in Table 4.1. The system

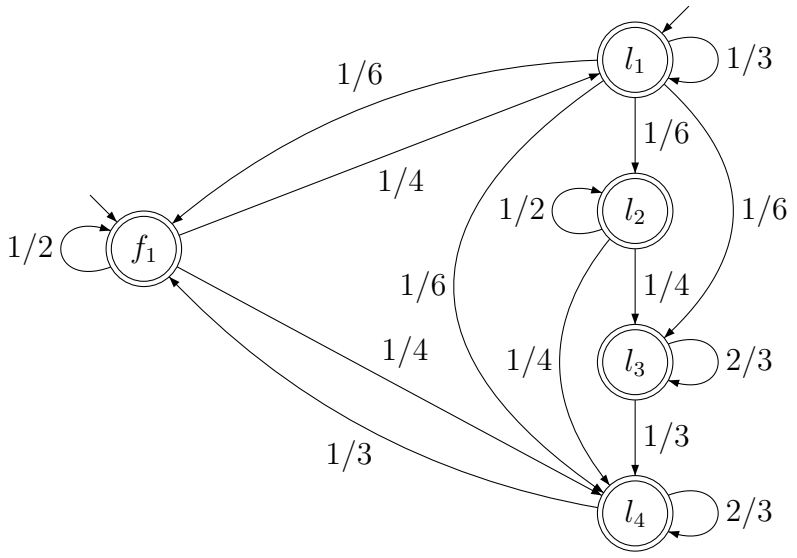


Figure 4.6: The generative HMM model used to generate synthetic data.

State	Outcome					
	1	**2**	**3**	**4**	**5**	**6**
f_1	1/6	1/6	1/6	1/6	1/6	1/6
l_1	3/5	2/25	2/25	2/25	2/25	2/25
l_2	2/25	3/5	2/25	2/25	2/25	2/25
l_3	2/25	2/25	3/5	2/25	2/25	2/25
l_4	2/25	2/25	2/25	3/5	2/25	2/25

Table 4.1: Emission probability matrix for the dishonest casino HMM.

can transit from the fair die to loaded dice and vice versa according to the shown transition scheme and probabilities. Each state has associated a label corresponding to the type of die: F for the fair die and L for loaded dice.

The HMM in Figure 4.6 was used to generate synthetic data for the disohonest casino problem. To make the experiments more effective, a mixed-order source was modelled where the emission probabilities at time t are obtained as:

$$p(x_t|y_t) = \alpha p(x_t|y_t) + (1-\alpha)p(x_t|y_t, x_{t-1}) \tag{4.47}$$

Die outcome	5	5	3	1	4	6	2	2	3	4	6	3	5	4	2
Hidden state	l_1	l_4	f_1	f_1	f_1	l_1	l_1	l_2	l_3	l_4	f_1	l_4	l_4	l_4	f_1
Observable label	L	L	F	F	F	L	L	L	L	L	F	L	L	L	F

Figure 4.7: An example of HMM-generated sequence for the dishonest casino toy problem.

and transition probabilities as:

$$p(y_t|y_{t-1}) = \alpha p(y_t|y_{t-1}) + (1-\alpha)p(y_t|y_{t-1}, y_{t-2}) \tag{4.48}$$

By this, a more expressive mixed first/second order model was used as source data generator. The parameter $0 \leq \alpha \leq 1$ was used to trade-off between first ans second order contributions. First-order transition and emission probabilities were defined as described above in Figure 4.6 and Table 4.1. Second-order probabilities are listed in Tables 4.2 and 4.3.

One hundred synthetic training/test datasets were generated for $\alpha \in \{0.05, 0.1, 0.2, 0.4, 0.8\}$. Each training dataset consisted of 5000 sequences of length comprised between 50 and 150 elements. Testing set size was set to 500 sequences. Each element in a sequence was annoted with the label associated to the corresponding emitting die as shown in Figure 4.7. By this, the actual sequence of states was considered as hidden and the unique information available at training time was the associated label sequence.

The two different GHRCRFs shown in Figure 4.8 were tested. Both models had the same number of states of the generating HMM. The model in Figure 4.8(a) adopts exactly the same transition scheme of the generator HMM whereas the model in Figure 4.8(b) has a slightly modified architecture whit missing and added allowed transitions between states.

Linear-chain CRF models for the disohonest casino problem could be defined such that there was a single state sequence for each different label sequence as explained in Section 4.3.1. Finite-state machines in Figure 4.9 both satisfy this condition. In Figure 4.9(a) the simplest model is defined with two states, one fair and one loaded, respectively, and a fully-connected transition topology. This represents the most common way linear-chain CRFs are defined in applications. In the second model in

x_{t-1}	y_t	Outcome					
		1	**2**	**3**	**4**	**5**	**6**
1	f_1	1/6	1/6	1/6	1/6	1/6	1/6
2	f_1	1/6	1/6	1/6	1/6	1/6	1/6
3	f_1	1/6	1/6	1/6	1/6	1/6	1/6
4	f_1	1/6	1/6	1/6	1/6	1/6	1/6
5	f_1	1/6	1/6	1/6	1/6	1/6	1/6
6	f_1	1/6	1/6	1/6	1/6	1/6	1/6
1	l_1	2/5	3/25	3/25	3/25	3/25	3/25
2	l_1	3/5	2/25	2/25	2/25	2/25	2/25
3	l_1	3/5	2/25	2/25	2/25	2/25	2/25
4	l_1	3/5	2/25	2/25	2/25	2/25	2/25
5	l_1	3/5	2/25	2/25	2/25	2/25	2/25
6	l_1	3/5	2/25	2/25	2/25	2/25	2/25
1	l_2	3/5	2/25	2/25	2/25	2/25	2/25
2	l_2	2/5	3/25	3/25	3/25	3/25	3/25
3	l_2	3/5	2/25	2/25	2/25	2/25	2/25
4	l_2	3/5	2/25	2/25	2/25	2/25	2/25
5	l_2	3/5	2/25	2/25	2/25	2/25	2/25
6	l_2	3/5	2/25	2/25	2/25	2/25	2/25
1	l_3	3/5	2/25	2/25	2/25	2/25	2/25
2	l_3	3/5	2/25	2/25	2/25	2/25	2/25
3	l_3	2/5	3/25	3/25	3/25	3/25	3/25
4	l_3	3/5	2/25	2/25	2/25	2/25	2/25
5	l_3	3/5	2/25	2/25	2/25	2/25	2/25
6	l_3	3/5	2/25	2/25	2/25	2/25	2/25
1	l_4	3/5	2/25	2/25	2/25	2/25	2/25
2	l_4	3/5	2/25	2/25	2/25	2/25	2/25
3	l_4	3/5	2/25	2/25	2/25	2/25	2/25
4	l_4	2/5	3/25	3/25	3/25	3/25	3/25
5	l_4	3/5	2/25	2/25	2/25	2/25	2/25
6	l_4	3/5	2/25	2/25	2/25	2/25	2/25

Table 4.2: Second-order emission probabilities used for the mixed-order source HMM.

Figure 4.9(b), the same number of states of GRHCRF and HMM models above is used. The transition scheme was defined in order to meet the one-to-one mapping between state and label sequences.

y_{t-2}	y_{t-1}	y_t				
		f_1	l_1	l_2	l_3	l_4
f_1	f_1	1/3	1/3	-	-	1/3
f_1	l_1	1/9	1/3	1/9	1/9	2/9
f_1	l_4	3/7	-	-	-	4/7
l_1	f_1	1/2	1/4	-	-	1/4
l_1	l_1	9/40	1/10	9/40	9/40	9/40
l_1	l_2	-	-	1/2	1/6	1/3
l_1	l_3	-	-	-	2/3	1/3
l_1	l_4	3/7	-	-	-	4/7
l_2	l_2	-	-	1/6	5/12	5/12
l_2	l_3	-	-	-	2/3	1/3
l_2	l_4	3/7	-	-	-	4/7
l_3	l_3	-	-	-	1/3	2/3
l_3	l_4	3/7	-	-	-	4/7
l_4	f_1	1/2	1/4	-	-	1/4
l_4	l_4	4/7	-	-	-	3/7

Table 4.3: Second-order transition probabilities used for the mixed-order source HMM.

All described models were trained until convergence using the generated training datasets. Testing datasets were used to score the performances. In Figure 4.10 testing error and Matthews Correlation Coeffiecient (MCC) of the GRHCRF model in Figure 4.9(a) is plotted against the same indices evaluated for the fully-connected linear-chain CRF model in Figure 4.9(a). Each point in the plots represents a single independent experiment. Different colours are associated to different values of the trade-off parameter α used in generating the corresponding training/test pair. It is clear from the plots, especially from MCC values, that the GRHCRF model which was able to capture the hidden substructure of the problem, performed much better than the simple linear-chain CRF. This discrepancy in performances was somewhat expected since the structure of the simple 2-state linear-chain CRF model was obviously too far from the source model. Both models suffered in modelling the mixed-order nature of data as suggested by low performances obtained on datasets where second-order contributions were stronger ($\alpha < 0.5$).

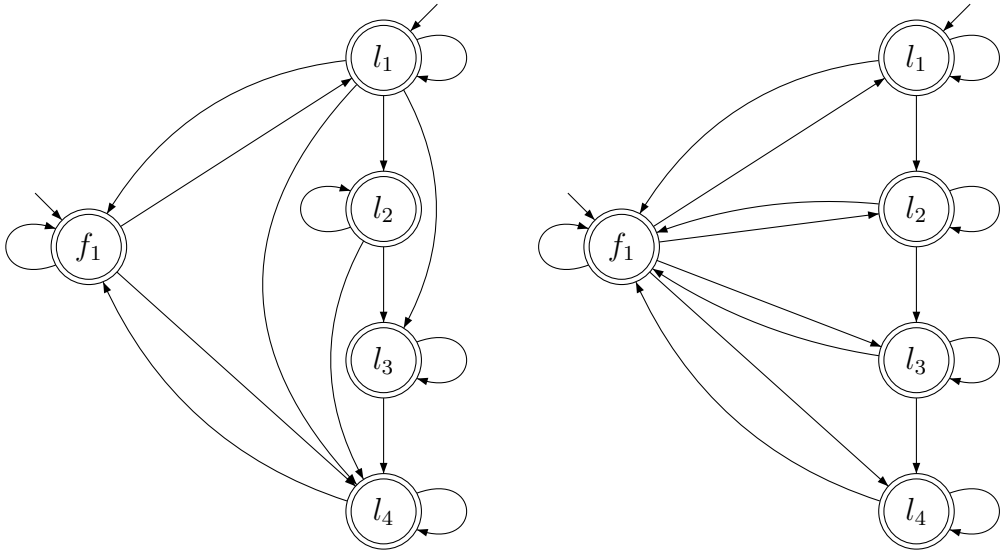
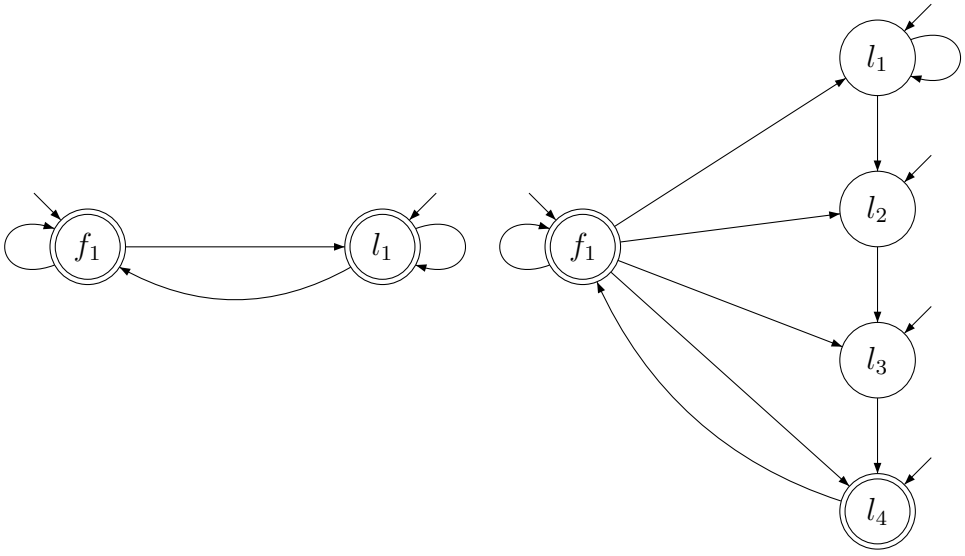


Figure 4.8: Two different GRHCRF models for the occasionally dishonest casino problem.

In Figure 4.11 the GRHCRF model in Figure 4.8(b) is compared to the linear-chain CRF model in Figure 4.9(b). In this case, the GRHCRF model structure deviated from the true hidden substructure of the HMM. However, prediction performances are essentially stable with respect to the GRHCRF model adopted for the previous experiment. On the contrary, the more complex structure of the linear-chain CRF model performed better than the simple fully-connected 2-state model. However, also in this experiment, the GRHCRF outperformed the linear-chain CRF, suggesting that decoupling hidden states from labels can effectively improve modelling capabilities.

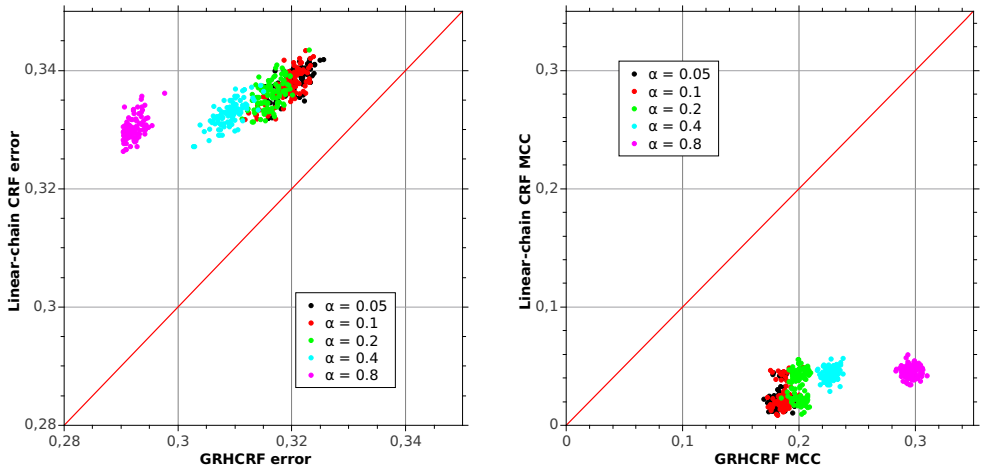
4.7 Summary

Discriminative models are very powerful and offer several advantages over generative approaches such as the possibility to exploit arbitrary features of the observation sequence without adding complexity. Linear-chain CRFs have gained a lot of atten-



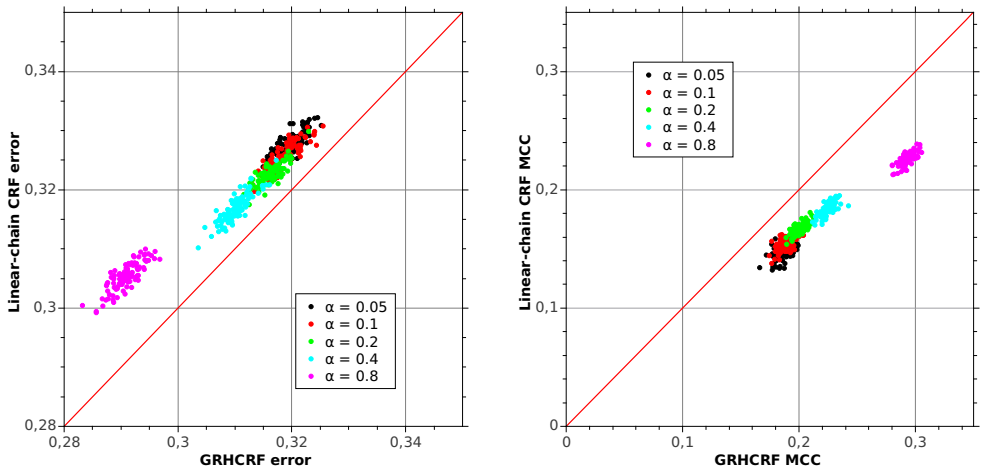
(a) (b)

Figure 4.9: Two different linear-chain CRF models for the occasionally dishonest casino problem.



(a) (b)

Figure 4.10: Comparison between GRHCRFs and linear-chain CRFs on synthetic data.



(a) (b)

Figure 4.11: Comparison between GRHCRFs and linear-chain CRFs on synthetic data.

tion in the last years and they have been chosen in many fields to solve sequence labelling problems.

In this chapter an extension of standard linear-chain CRFs, called the Grammatical-Restrained Hidden CRF (GRHCRF), was introduced. This extended framework improves the expressive power of linear-chain CRFs in two ways. Firstly, the GRHCRF introduces hidden states as a separate layer of variables into the model. By this, it is possible to model information that is not available as part of the training data. This information is precisely captured by the hidden layer. Secondly, the introduction of the hidden states allows to improve the way prior knowledge is inserted into the model. Indeed, by the separation of hidden states from labels, arbitrary regular grammars can be defined over hidden-state sequences. In this way, multiple hidden state sequences compatible with the grammar are associated to a single observed label sequence. The advantage derived from this many-to-one mapping consists in the fact that a single label sequence can be modelled as obtained from multiple sub-structures not observed at training time. In other words, this allows to design models that automatically learn relations which are not directly learnable from the experimental labelling. In contrast, the fully observability of

linear-chain CRFs limits the capability of the model to what is observed at training time, meaning that the model can enforce grammars only on the label sequences.

The advantages of GRHCRFs over linear-chain CRFs were investigated in the present chapter using synthetic datasets generated for a simple toy problem. In Chapter 7 a GRHCRF model is tested on the bionformatics problem of β-barrel topology prediction. In general, GRHCRFs are well-suited for many different problems in Computational Biology such as protein secondary structure prediction or coiled coil prediction. Typically, these tasks have been addressed in literature using HMMs in which label and state were decoupled similarly to what was done in GRHCRFs. These HMMs are usually referred to as *Class HMMs* (CHMMs) [62]. A strength of the GRHCRF over the CHMM lies in the discriminative nature of the former and all the advantages this brings over generative approaches.

Chapter 5

N-to-1 Extreme Learning Machines for sequence classification

5.1 Introduction

In machine learning, a typical problem consists in learning to map *structured objects* into a finite number of predefined classes. Indeed, the need to represent and to operate on structured objects is prevalent in many application areas of machine learning such as DNA/protein classification, speech recognition, intrusion detection and text classification. Typical structured domains are sequences, trees or graphs.

Sequence data represent the simplest and most common form of structured domain. In many fields, sequence data are of central importance such as in Computational Biology and Bioinformatics where nucleic acid and protein sequences are the raw material par excellence. A core problem in biological sequence analysis is the annotation of new protein sequences with structural or functional features. In this framework, several annotation tasks can be identified such as detection of protein distant homologuous, protein fold recognition and trans-membrane proteins discrimination (a problem that will be addressed in the next chapter). In all these tasks, a major issue consists in finding a suitable fixed-size feature representation of the variable-length sequence. Indeed, traditional machine learning algorithms for classification and regression such as Artificial Neural Networks (ANNs) or Support Vector Machines (SVMs) are designed to work on fixed-size real-valued vectors which are

provided as input. However, when dealing with sequence data, the objects of interest are typically of variable size.

Several techniques can be used to map a variable-length sequence into a representation that can be handled by traditional classifiers. The strategy adopted often depends on the type of classification problem that is addressed. A common approach is to extract from the sequence a fixed number of elements and then using them as fixed-sized feature vectors provided as input for a standard classification algorithm. The process of feature extraction, by which the variable-sized sequence is transformed into a fixed-sized feature vector, in many cases is carried out manually using prior knowledge about the problem at hand.

Other approaches derive fixed-length feature representation of sequence data using probabilistic generative models such as HMMs [54, 16]. String and profile-based kernels used in conjunction with max-margin classifiers such as SVMs represent another strategy for sequence data classification. In bioinformatics, string kernels have been adopted for protein annotation tasks [67, 97].

In this chapter a novel machine learning method for sequence classification is described. The method is based on ANNs and combines two recent developments in the neural network research area. On one side, a novel feed-forward network architecture called N-to-1 Neural Network [78] that is specifically designed for non-linear feature selection on variable-length sequence data. On the other side, the Extreme Learning Machine (ELM) [52, 51], an alternative framework to train feed-forward networks that has several advantages over traditional gradient-based learning algorithms.

In this chapter the method is introduced in general. The next chapter describes an application to the discrimination of β-barrels from other types of proteins. The present chapter is structured as follows. Section 5.2 introduces the basic concepts about ANNs by focusing on the most common network architecture, namely the Feed-Forward Neural Network (FFNN). Section 5.3 is about training algorithms for FFNNs. Standard techniques such as gradient-descent (Section 5.3.1) as well as the Extreme Learning Machine framework (Section 5.3.2) are described. Finally, the new method based on the combination of N-to-1 NNs and ELMs is presented in Section 5.5.

5.2 Feed-forward Neural Networks

An Artificial Neural Network (ANN) is a mathematical model that draws inspiration from mechanisms for information processing in biological nervous systems, in particular the human brain [14, 15]. From a pure mathematical perspective, an ANN can be regarded as a non-linear function:

$$f : \mathbb{R}^n \to \mathbb{R}^m \tag{5.1}$$

which transform a set of n input variables to a set of m output varibales. This transformation is carried out by means of a set of *interconnected* computation units called *artificial neurons* (in what follows simply neurons) in analogy to real *biological neurons* which are responsible of the information processing in the brain. Connections between different neurons are organized in a specific topology which determines the form of the function f. In the most typical architecture, ANNs are structured in variable-sized *layers* of neurons such that each neuron of the i-th layer is fully connected to all neurons of the $(i+1)$-th layer. Each layer has an input and an output. Basically, a layer performs a *non-linear* transformation of the output of the previous layer and provides it as input for the next layer. By this, the overall network function f is obtained as a combination of successive transformations performed by layers. The connections between layers have associated *weights* or *parameters* that govern the specific form of the transformation.

The kind of network just described is known as *Feed-Forward Neural Network* (FFNN) and the underlying mathematical model can be formally defined as follows.

Let $\mathbf{x} \in \mathbb{R}^n$ be an input vector. In what follows, for sake of simplicity, a network topology as shown in Figure 5.1 is assumed. The network consists of an *input* layer of size n whose output is simply the input vector $\mathbf{x}$, a single internal or *hidden* layer of size d and an output layer of size m which provides the final network output. This typical architecture is usually referred in literature to as Single hidden-Layer Feedforward Network (SLFN). The generalization to multiple hidden-layer fedd-forward newtorks is straightforward.

The i-th neuron of the hidden layer performs a non-linear transformation of the input $\mathbf{x}$ (graphically represented in Figure 5.2). The neuron operates by firstly

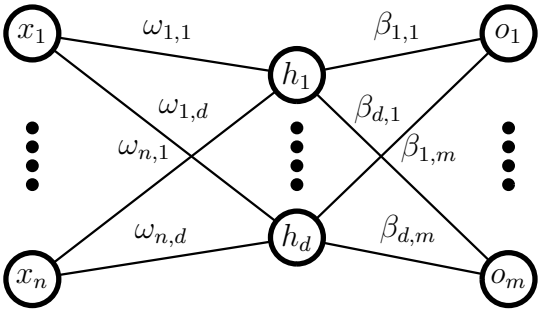


Figure 5.1: A graphical representation of a Single hidden-Layer Feedforward Network (SLFN).

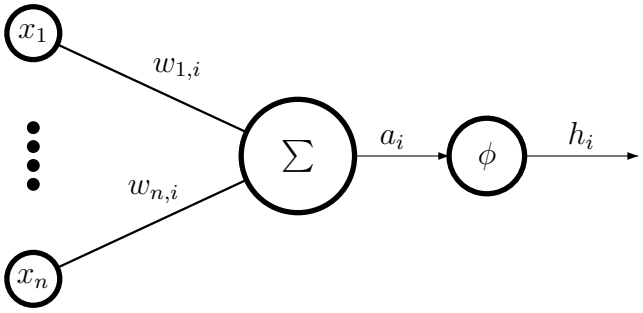


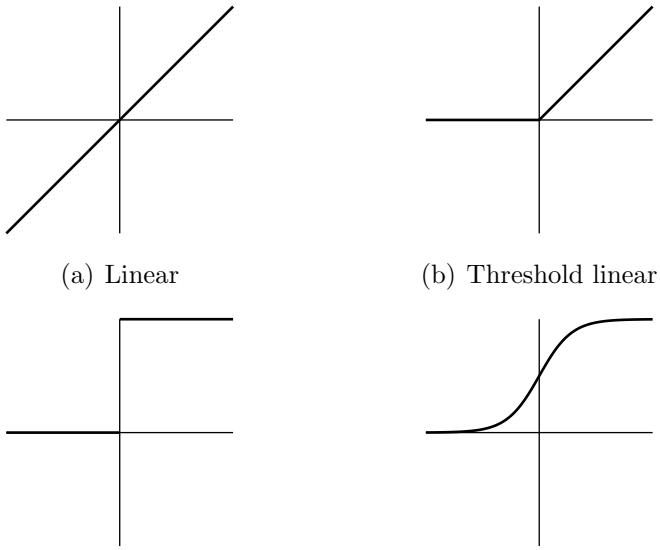
Figure 5.2: The non-linear transformation at the i-th hidden neuron.

computing a linear combination of the input $\mathbf{x}$ as follows:

$$a_i = \sum_{j=1}^{n} \omega_{j,i} x_j + b_i \tag{5.2}$$

where $\omega_{j,i} \in \mathbb{R}$ is the weight associated to the connection between the j-th input neuron and the i-th hidden neuron and $b_i \in \mathbb{R}$ is the threshold or *bias* of the i-th hidden neuron. To make the notation uncluttered, the bias b_i can be absorbed into the summation by regarding it as a special case of weight $\omega_{0,i}$ from an extra input x_0 whose value is permanently set to 1. Therefore, Equation 5.2 can be simplified writing:

$$a_i = \sum_{j=0}^{n} \omega_{j,i} x_j \tag{5.3}$$



(a) Linear (b) Threshold linear

(c) Threshold (d) Sigmoid

Figure 5.3: Shapes of typical activation functions.

The value a_i is then transformed using a (non-linear) function $\phi : \mathbb{R} \to \mathbb{R}$ usually referred to as *activation function* giving the final neuron output h_i:

$$h_i \quad = \quad \phi(a_i) = \phi\left(\sum_{j=0}^{n} \omega_{j,i} x_j\right) \tag{5.4}$$

In general, the value h_i is called simply the *activation* of the i-th hidden neuron. The activation function is typically a sigmoid function of the form:

$$\phi(x) = \sigma(x) \quad = \quad \frac{1}{1 + \exp(-\lambda x)} \tag{5.5}$$

Other functions can be also used such as step functions or the linear function. The shapes of some of these functions are showed in Figure 5.3. We assume that the same activation function is used at each hidden neuron.

On the whole, the hidden layer $\mathbf{h} = (h_1, \ldots, h_d)$ performs a non-linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^d$ of the input $\mathbf{x}$.

Similarly, at each neuron of the output layer a transformation is performed as follows:

$$o_k \quad = \quad \psi\left(\sum_{i=0}^{d} \beta_{i,k} h_i\right) \tag{5.6}$$

where $\beta_{i,k} \in \mathbb{R}$ is the weight connecting the i-th hidden neuron and the k-th output neuron[1] and $\psi : \mathbb{R} \to \mathbb{R}$ is a generic activation function.

Overall, for any choice of weights $\omega_{j,i}$ and $\beta_{i,k}$, the network implements a mapping $f : \mathbb{R}^n \to \mathbb{R}^m$ given by:

$$\mathbf{o} \quad = \quad \Psi \left(\sum_{i=0}^{d} \boldsymbol{\beta}_i \phi \left(\langle \boldsymbol{\omega}_i, \mathbf{x} \rangle \right) \right) \tag{5.7}$$

where $\boldsymbol{\omega}_i = (\omega_{1,i}, \ldots, \omega_{n,i})$ is the vector of weights connecting the input layer to the i-th hidden neuron, $\boldsymbol{\beta}_i = (\beta_{i,1}, \ldots, \beta_{i,m})$ is the vector of weights connecting the i-th hidden neuron and the output layer, $\Psi : \mathbb{R}^m \to \mathbb{R}^m$ defined as:

$$\Psi(y_1, \ldots, y_m) = (\psi(y_1), \ldots, \psi(y_m)) \tag{5.8}$$

is the vector-valued activation function, $\mathbf{o} = (o_1, \ldots, o_m)$ is the m-dimensional network output and $\langle \cdot, \cdot \rangle$ represents the inner product between two vectors.

The evaluation of the output of a FFNN given by Equation 5.7 comprising all the computations involved such as evaluation of dot products and activation functions is often referred to as *forward propagation* since it can be regraded as a flow of information through the network.

5.3 Supervised learning of SLFNs

Consider for simplicity a SLFN with linear output activation $\psi(x) = x$. Given N arbitrary distinct training samples $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)}) \in R^n$ are input vectors and $\mathbf{t}^{(i)} = (t_1^{(i)}, \ldots, t_m^{(i)}) \in \mathbb{R}^m$ are corresponding target vectors, a standard SLFN with d hidden neurons and non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$ can approximate these N samples with zero error if there exist parameters:

$$\boldsymbol{\Theta} = \begin{cases} \boldsymbol{\beta}_k = (\beta_{0,k}, \ldots, \beta_{d,k}) \in \mathbb{R}^{d+1} & k = 1, \ldots, m \\ \boldsymbol{\omega}_i = (\omega 0, i, \ldots, \omega_{n,i}) \in \mathbb{R}^{n+1} & i = 1, \ldots, d \end{cases} \tag{5.9}$$

such that:

$$\sum_{i=0}^{d} \boldsymbol{\beta}_i \phi \left(\langle \boldsymbol{\omega}_i, \mathbf{x}^{(i)} \rangle \right) \quad = \quad \mathbf{t}^{(i)}, \quad i = 1, \ldots, N \tag{5.10}$$

[1] Also in this case the bias parameter has been absorbed into the summation.

The above N equations can be written compactly as:

$$\mathbf{HB} = \mathbf{T} \tag{5.11}$$

where

$$\mathbf{H} = \begin{bmatrix} \phi\left(\langle \boldsymbol{\omega}_1, \mathbf{x}^{(1)} \rangle\right) & , \cdots, & \phi\left(\langle \boldsymbol{\omega}_d, \mathbf{x}^{(1)} \rangle\right), & 1 \\ \vdots & \cdots & \vdots & \vdots \\ \phi\left(\langle \boldsymbol{\omega}_1, \mathbf{x}^{(N)} \rangle\right) & , \cdots, & \phi\left(\langle \boldsymbol{\omega}_d, \mathbf{x}^{(N)} \rangle\right), & 1 \end{bmatrix}_{N \times (d+1)} \tag{5.12}$$

$$\mathbf{B} = \begin{bmatrix} \beta_{0,1} & , \cdots, & \beta_{0,m} \\ \vdots & \cdots & \vdots \\ \beta_{d,1} & , \cdots, & \beta_{d,m} \end{bmatrix}_{(d+1) \times m} \tag{5.13}$$

and

$$\mathbf{T} = \begin{bmatrix} t_1^{(1)} & , \cdots, & t_m^{(1)} \\ \vdots & , \cdots, & \vdots \\ t_1^{(N)} & \cdots & t_m^{(N)} \end{bmatrix}_{N \times m} \tag{5.14}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network [46]. The i-th column of $\mathbf{H}$ is the i-th hidden neuron output with respect to inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$.

Since the number of distinct training examples is usually much greater than the number of hidden neurons ($N \gg d$), the $\mathbf{H}$ matrix is a rectangular matrix and tuning the parameters $\boldsymbol{\Theta}$ in order to obtain a unique solution for the system $\mathbf{HB} = \mathbf{T}$ is not always possible.

Gradient-based learning algorithms [15] try to find a solution that minimizes the cost or error function:

$$\begin{aligned} E(\mathcal{D}; \boldsymbol{\Theta}) &= \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{o}^{(i)} - \mathbf{t}^{(i)}\|^2 = \\ &= \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{m} (o_k^{(i)} - t_k^{(i)})^2 \end{aligned} \tag{5.15}$$

where $\mathbf{o}^{(i)}$ is the output of the network when the training sample $\mathbf{x}^{(i)}$ is provided as input vector. Consider that Equation 5.15 corresponds to a specific choice of the error function, namely the *sum-of-squares error*. In general is possible to use different error functions such as the *cross-entropy* error [14].

Regardless of the specific error used, the smallest value of the cost function will occur in a point of the parameter space such that the gradient of the function is zero:

$$\nabla E(\mathcal{D}; \mathbf{\Theta}) \quad = \quad 0 \tag{5.16}$$

Of course, there are many points at which the gradient is zero. In general, since $E(\mathcal{D}; \mathbf{\Theta})$ has a highly non-linear dependence on the parameters $\mathbf{\Theta}$, there will be multiple inequivalent stationary points. A minimum that corresponds to the smallest value of the error function is said to be a *global minimum*. Any other minima at which the gradient vanishes are *local minima*. The goal in network training is to find the global minimum. However, in general it will not be known if a global mimimum has been reached. Therefore, it may be necessary to compare different local mimima to obtain a sufficiently good solution.

Since there is no way to find an analytical solution for the Equation 5.16, numerical iterative optimization procedures are often used. Most techniques involve choosing some initial value $\mathbf{\Theta}^0$ for the parameters and then explore the parameter space with a succession of steps. The simplest approach is to perform each step in the negative direction of the gradient such as:

$$\mathbf{\Theta}^{\tau+1} \quad = \quad \mathbf{\Theta}^{\tau} - \eta \nabla E(\mathcal{D}; \mathbf{\Theta}^{\tau}) \tag{5.17}$$

where $\eta > 0$ is a hyper-parameter called *learning rate* that governs the magnitude of the learning step. Small learning rates can lead to a more accurate but very slow traning of the network. Big learning steps may cause the procedure to indefinitely oscillate without finding a solution. In general, the optimal value for η must be found using appropiate validation mechanisms such as cross-validation.

In the above procedure, the gradient of the error function must be evaluated once in every step. The function $E(\mathcal{D}; \mathbf{\Theta})$ depends on the entire training set and the update in Equation 5.17 is performed after the entire training set has been processed. This strategy is called *batch training*. However, it also possible to update the parameters using a so-called *online training*. Typically, error functions are defined as sums of errors computed on each of the training examples as follows:

$$E(\mathcal{D}; \mathbf{\Theta}) = \frac{1}{2} \sum_{i=1}^{N} E_i(\mathcal{D}; \mathbf{\Theta}) \tag{5.18}$$

In online learning the weight update takes place just after a single example has been processed:

$$\mathbf{\Theta}^{\tau+1} = \mathbf{\Theta}^{\tau} - \eta \nabla E_i(\mathcal{D}; \mathbf{\Theta}^{\tau}) \tag{5.19}$$

This training strategy is also known as *sequential gradient descent*. On-line training is in general more appropriate to handle the (possible) redundancy in the training set. Furthermore, by using on-line techniques it is possible to escape from local mimima since a stationary point for the overall error function is not in general a stationary point for errors computed on each data point individually [15].

5.3.1 The Error Back-Propagation algorithm

The computation of the gradient of the error function in Equation 5.15 can be efficiently carried out in different ways. The most simple approach is to use the method finite differences. By this each weight is perturbated in turn and the derivatives approximated using the following expression:

$$\frac{\partial E}{\partial \omega_{j,i}} = \frac{E(\omega_{j,i} + \epsilon) - E_i(\omega_{j,i})}{\epsilon} + O(\epsilon) \tag{5.20}$$

where $E(\omega_{j,i} + \epsilon)$ is the error function computed after a perturbation of the weight $\omega_{j,i}$ by a small $\epsilon \ll 1$.

If M is the number of parameters $\omega_{j,i}$ in the first layer of the network, each evaluation of the error function require $O(M)$ time. This derives from the fact that typically the number of parameters is much greater than the number of neurons[2]. As a consequence, most of the computational effort in the forward propagation is concerned with the evaluation of inner products in Equation 5.3, with evaluation of activation functions and the output layer representing a small overhead. Since each weight must be perturbated individually the overall computational complexity is quadratic in M.

The Error Back-Propagation (EBP) algorithm [100] provides an efficient alternative to compute the required derivatives. The required time in this case scales

[2]In general the number of parameters is always greater than the number of neurons except for networks with very sparse connections.

with M. To achieve this goal, EBP uses a local message passing scheme that exploits the network topology to reduce the computational complexity. The procedure can be applied to compute the gradient for any type of FFNN. In this section we restrict for simplicity our attention to SLFNs with differentiable activation function g for the hidden layer. In order to make the notation uncluttered, we also assume a linear output layer. Furthermore, since we consider a sum-of-squares error function that comprises sum of terms as in Equation 5.18 we implicitly focus on evaluating $\nabla E_i(\mathcal{D}; \mathbf{\Theta}^\tau)$ for one of such terms. This may be in turn used directly in online training or accumulated for batch training.

The EBP algorithm actually begins by forward propagate a vector $\mathbf{x}$ through the network and computing all activations of hidden and output neurons. We now define the following quantities δ_k^o and δ_i^h as:

$$\delta_k^o = \frac{\partial E}{\partial o_k} \tag{5.21}$$

$$\delta_i^h = \frac{\partial E}{\partial a_i} \tag{5.22}$$

where a_i is defined as in Equation 5.3. These quantities can be informally seen as variations in the error function that are due to variations in o_k and a_i, respectively. For this reason they are often referred to as *local errors* computed at each neuron[3].

Evaluating the gradient of the error function with respect to a generic output weight $\beta_{i,k}$ we have:

$$\begin{aligned} \frac{\partial E}{\partial \beta_{i,k}} &= \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial \beta_{i,k}} = \\ &= (o_k - t_k)h_i \\ &= \delta_k^o h_i \end{aligned} \tag{5.23}$$

where we simply used the chain rule for partial derivatives. Then for output neurons we have

$$\delta_k^o = (o_k - t_k) \tag{5.24}$$

[3]There a δ for each hidden and output neuron in the network.

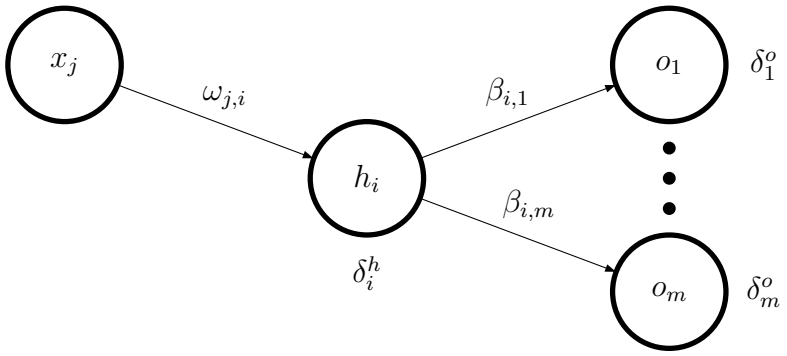


Figure 5.4: Illustration of the back-propagation formula.

For hidden neurons it holds the following:

$$\begin{aligned} \frac{\partial E}{\partial \omega_{j,i}} &= \frac{\partial E}{\partial a_i}\frac{\partial a_i}{\partial \omega_{j,i}} = \\ &= \delta_i^h \frac{\partial a_i}{\partial \omega_{j,i}} \end{aligned} \tag{5.25}$$

where:

$$\frac{\partial a_i}{\partial \omega_{j,i}} = x_j \tag{5.26}$$

and:

$$\delta_i^h = \sum_{k=1}^{m} \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial a_i} \tag{5.27}$$

In writing Equation 5.27 we apply again the chain rule and the fact that variations in the error function due to a_i are possible only through variations in o_r.

Using Equations 5.21 and 5.6 we obtain the *back-propagation* formula (shown graphically in Figure 5.4) as follows:

$$\delta_i^h = \phi'(a_i)\sum_{k=1}^{m} \beta_{i,k}\delta_k^o \tag{5.28}$$

where $\phi'(a_i)$ is the first derivative of the activation function ϕ. The EBP algorithm is summarized in Algorithm 4.

As can be easily verified, the computational complexity of the EBP algorithm is $O(M)$ making EBP more efficient than the method of finite differences.

Algorithm 4 The Error Back-Propagation (EBP) Algorithm [100]

1: forward propagate an input vector $\mathbf{x}$ to compute all activations h_i and o_k of hidden and output neurons
2: evaluate $\delta_k^o = (o_k - t_k)$ for all output neurons
3: back-propagate δ_k^o to compute δ_i^h using Equation 5.27
4: evaluate the derivative as: $\frac{\partial E}{\partial \beta_{i,k}} = \delta_k^o h_i$ and $\frac{\partial E}{\partial \omega_{j,i}} = \delta_i^h x_j$

5.3.2 The Extreme Learning Machine (ELM)

Traditional learning algorithms for SLFNs, like gradient descent, have several disadvantages. Firstly, the training procedure is strongly dependent on the learning rate and the initial set of parameters chosen. As already mentioned above there is no way to determine the optimal values of these hyper-parameters. In practice the network need to be trained several times and the learning rate and initial parameters must be selected using some validation procedure in order to obtain a sufficiently good solution. However, the overall procedure can be very time-consuming especially in the case of large datasets of training.

Secondly, gradient based training suffers the problem of local minima. Trapping in a local minimum can lead to poor performance in several applications, especially when it is located far away the global optimum in the error function surface.

Finally, SLFNs training using gradient based techniques may lead to overfitting of training data. In order to reduce this phenomenon, appropriate mechanisms such as regularization or *early stopping* [15] must be adopted. These mechanisms introduce new hyper-parameters such as the regularization coefficient that must be finely tuned to make these procedures really effective.

Recently, in order to overcome these issues, a new framework for training SLFNs has emerged as promising alternative to standard gradient based techniques. This new training algorithm is called Extreme Learning Machine [51, 52] and it based on a simple but effective idea that drastically reduces both the training time and the number of hyper-parameters in the model.

Traditional, gradient based algorithms require to tune *all* the parameters of SLFNs, namely input weights $\omega_{j,i}$ and output weights $\beta_{i,k}$. In their work, Huang

Algorithm 5 The Extreme Learning Machine Algorithm [51]

1: assign randomly input weights $\omega_{j,i} \quad \forall j = 0, \ldots, n, i = 1, \ldots, d$
2: calculate the hidden layer output matrix $\mathbf{H}$
3: calculate the output weights $\mathbf{B}$ as : $\mathbf{B}^* = \mathbf{H}^\dagger \mathbf{T}$

et el. [51, 48] provide a rigorous proof to support the idea that input weights $\omega_{j,i}$ can be chosen *randomly* and keep fixed during training without affecting the approximation capabilities of the network. As a consequence, assuming fixed input weights, the training of a SLFN simply reduces to find a solution $\mathbf{B}^*$ of the linear system of equations:

$$\mathbf{HB} = \mathbf{T} \tag{5.29}$$

This is achieved by adopting the least-square solution of the above linear system as:

$$\mathbf{B}^* = \mathbf{H}^\dagger \mathbf{T} \tag{5.30}$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$.

The three main steps involved in ELM algorithm can be summarized as in Algorithm 5.

The solution $\mathbf{B}^*$ obtained by ELM has two important properties:

- $\mathbf{B}^* = \mathbf{H}^\dagger \mathbf{T}$ is one of the least-square solutions of the linear system $\mathbf{HB} = \mathbf{T}$. This means that the minimum training error is achieved by this special solution.
- Furthermore, the solution $\mathbf{B}^*$ is the least-square solution with the smallest norm and this reduces over-fitting.

5.4 Interpolation and approximation capabilities of NNs

From a mathematical point of view, approximation capabilities of multy-layered feed-forward neural networks have been investigated in two different aspects: univer-

sal approximation on compact input sets and interpolation of a finite set of training samples.

Most of the research on universal approximation of NNs builds on a classical Kolmogorov's theorem [61] concerning general function approximation. Although this result is not strictly related to feed-forward NNs, it states that any multivariate continuous function have an *exact representation* in terms of a finite compositions and superpositions of a *small* number of univariate function of the form:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{2n+1} g_i \left(\sum_{j=1}^{n} \phi_{i,j}(x_j) \right) \tag{5.31}$$

where g_i are continuous functions of one variable and $\phi_{i,j}$ are continuous monotocally increasing functions independent of f. This theorem can be interpreted as providing theoretical support for feed-forward networks implementing such functions, although many authors have also rejected this interpretation [35].

An important, groundbreaking result on universal approximation capabilities of NNs is the rigorous formal proof that feed-forward networks with a *single* hidden layer and any continuous sigmoidal activation function can approximate any continuous function on $[0, 1]^n$ provided that no restrictions are placed on the number of hidden neurons and on the magnitude of parameters. This result is due to Cybenko [26], Hornik *et al.* [43] and Funahashi [34] who independently provided proofs of this fact. Using the notation from Cybenko [26] this can be stated as follows:

Theorem 5.4.1 (Universal Approximation Theorem [26])**.** *Let σ be any sigmoidal bounded function. Then finite sums of the form:*

$$G(\mathbf{x}) = \sum_{j=1}^{N} \boldsymbol{\beta}_j \sigma(\langle \boldsymbol{\omega}_i, \mathbf{x} \rangle) \tag{5.32}$$

are dense in the space of continuous functions on $[0, 1]^n$. In other words, given any f continuous on $[0, 1]^n$ and $\epsilon > 0$, there is a sum $G(\mathbf{x})$ of the above form, such that

$$|G(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \qquad \forall \mathbf{x} \in [0, 1]^n \tag{5.33}$$

Subsequently other researchers further extended this theorem to any continuous function f on $\mathbb{R}^n$ and to more general activations than sigmoidal functions [108, 7, 69, 65].

On real-world problems, NNs are trained on finite set of training examples. It has been shown that N distinct samples can be learned with zero error by a SLFN with N hidden neurons and *threshold* activation functions [53, 101]. In subsequent reasearch, interpolation with SLFNs has been further investigated by proving that a SLFN with N hidden neurons and *any non-linear arbitrary bounded activation function* which has a limit at one infinity can exactly learn N distinct samples. These activation functions include sigmoidal, threshold, threshold linear, radial basis, cosine squasher and many other non-regular activation functions [47].

In all the above discussed results, it is assumed that *all* the parameters of the feed-forward network are tuned, including input layer weights and biases. Interstingly, similar results about approximation and interpolation capabilities have been obtained also for randomized networks where input layer weights and biases are chosen randomly and only output weigths are tuned as in the ELM training framework. Learning capabilities and generalization performances of *adjustable* SLFN with respect to randomized networks were originally investigated in the context of Radial Basis Function (RBF) networks[4] [74]. Although lacking of a formal proof, the main conclusion of this work was that, according to experiments, non-linear optimization of the centers and impact factors of the RBF network was beneficial only when a *minimal network* (with a minimum number of hidden neurons) was required, since comparable generalization performances were obtained by using more randomly fixed centers and impact factors and optimize only the second linear layer of weights [74].

As the ELM framework has been introduced, several studies focused more thoroughly on the learning capabilities of randomized SLFNs [48, 49, 52, 50]. The main results can be summarized in following theorems:

Theorem 5.4.2 (Approximation on a finite set of samples [52])**.** *Given any small $\epsilon > 0$, any infinitely differentiable activation function $\psi : \mathbb{R} \to \mathbb{R}$, and N arbitrary distinct samples $\{(\mathbf{x}^{(i)} \in \mathbb{R}^n, \mathbf{t}^{(i)} \in \mathbb{R}^m)\}_{i=1}^N$, there exists $L \leq N$ such that for any randomly generated $\{\boldsymbol{\omega}_i \in \mathbb{R}^n\}_{i=1}^L$ drawn from any continuous probability distribution*

[4]RBF Nets can be considered a special type of SLFN where hidden neurons are of the form $\frac{\psi_j(\|\mathbf{x}-\boldsymbol{\omega}_i\|)}{\omega_{i,0}}$ where ψ_j is a radial basis function and $\boldsymbol{\omega}_i$ and $\omega_{i,0}$ are centers and impact factor of the RBF kernel.

the following holds with probability one:

$$\|\mathbf{H}_{N\times L}\mathbf{B}_{L\times m} - \mathbf{T}_{N\times m}\| < \epsilon \tag{5.34}$$

where $\mathbf{H}$ *is the hidden layer output matrix,* $\mathbf{B}$ *is the matrix of output weights and* $\mathbf{T}$ *is the target matrix.*

Theorem 5.4.3 (Exact interpolation of a finite set of samples [52])**.** *Given any infinitely differentiable activation function* $\psi : \mathbb{R} \to \mathbb{R}$*, and* N *arbitrary distinct samples* $\{(\mathbf{x}^{(i)} \in \mathbb{R}^n, \mathbf{t}^{(i)} \in \mathbb{R}^m)\}_{i=1}^N$*, for any randomly generated* $\{\boldsymbol{\omega}_i \in \mathbb{R}^n\}_{i=1}^N$ *drawn from any continuous probability distribution the following holds with probability one:*

$$\|\mathbf{H}_{N\times N}\mathbf{B}_{N\times m} - \mathbf{T}_{N\times m}\| = 0 \tag{5.35}$$

where $\mathbf{H}$ *is the hidden layer output matrix,* $\mathbf{B}$ *is the matrix of output weights and* $\mathbf{T}$ *is the target matrix.*

Theorems 5.4.2 and 5.4.3 state that for a broad class of activation functions which include, sigmoid, radial basis, sine, cosine, exponential and many other functions, a SLFN trained in the ELM framework is able to interpolate with arbitrary error a finite set of samples provided enough hidden neurons are available. Theorem 5.4.3 in particular provides an upper bound of N in the number of neurons that are necessary to *exactly* interpolate a training set of size N.

For the next theorem about universal approximation of ELMs, the following definitions are needed:

Definition 5.4.1. *A function* f *is said to be piecewise continuous if it has only a finite number of discontinuities in any interval and its left and right limits are definied at each discontinuity.*

Definition 5.4.2. *Let* $L^2(\mathbb{R}^n)$ *(simply denoted by* L^2*) be a space of functions* f *in* $\mathbb{R}^n$ *such that* $|f|^2$ *is integrable, i.e.* $\int_{\mathbb{R}^n} |f(\mathbf{x})d\mathbf{x} < \infty$

Theorem 5.4.4 (Universal approximation of ELMs [50])**.** *Given any nonconstant, piecewise continuous activation function* $\psi : \mathbb{R} \to \mathbb{R}$ *such that* $\text{span}\{\psi(\langle\boldsymbol{\omega}, \mathbf{x}\rangle) : \boldsymbol{\omega} \in \mathbb{R}^n\}$ *is dense in* L^2*, for any continuous target function* f *and the function* f_L

realized by a SLFN with randomly generated input weights $\boldsymbol{\omega}_i$*, L hidden neurons and activation function* ψ *the following holds with probability one:*

$$\lim_{L\to\infty} \|f - f_L\| = 0 \tag{5.36}$$

if the output weights $\mathbf{B}$ *are determined by ordinary least square estimation:*

$$\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5.37}$$

where $\mathbf{H}$ *is the hidden layer output matrix,* $\mathbf{B}$ *is the matrix of output weights and* $\mathbf{T}$ *is the target matrix.*

Theorem 5.4.4 formally define the connection between the class of functions realized by ELM-trained SLFNs and continuous functions. The only condition imposed on activation functions is that $\text{span}\{\psi(\langle\boldsymbol{\omega}, \mathbf{x}\rangle) : \boldsymbol{\omega} \in \mathbb{R}^n\}$ is dense in L^2. The following lemma better characterizes the class of activation functions for which Theorem 5.4.4 is valid:

Lemma 5.4.1. *Given* $\psi : \mathbb{R} \to \mathbb{R}$*,* $\text{span}\{\psi(\langle\boldsymbol{\omega}, \mathbf{x}\rangle) : \boldsymbol{\omega} \in \mathbb{R}^n\}$ *is dense in* L^p *for every* $p \in [1, \infty)$ *if and only if* ψ *is not a polynomial.*

5.5 N-to-1 Extreme Learning Machines

5.5.1 N-to-1 Neural Networks

N-to-1 Neural Networks (N-to-1 NNs) are a new formulation of feed-forward networks aimed at solving variable-length sequence classification tasks [78]. As mentioned above, in this task the goal is to learn a mapping from variable-length sequences $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)$ where $\mathbf{x}_l \in \mathbb{R}^n$ to target vectors $\mathbf{t} \in \mathbb{R}^m$. In the context of classification, m is the number of predefined classes and the target vector $\mathbf{t} = (t_1, \ldots, t_m)$ represents a single class with orthogonal encoding[5]. Alternatively, target vectors can represent normalized membership probabilities of the sequence for each of the m different classes.

[5]The i-th class is represented using a vector $\mathbf{t}$ such that $t_i = 1$ and $t_{j\neq i} = 0$.

The basic idea of N-to-1 NNs is to encode into a single feature vector the piece-wise information defined by all the vectors in the sequence. Differently from the standard approach where the classification is tackled by extracting global features from predefined positions of the sequence, in the N-to-1 NN framework the entire information available is mapped non-linearly into a predefined number of features. By this, the network adapts itself in order to discover signals in the input sequence that are conducive for the prediction task and to discard non-relevant information [78].

In a more formal way, given a sequence of length L and a non-linear activation function ψ (for simplicity we assume the same activation function for all non-linear layers), the overall mapping from $\mathbb{R}^{L\times n}$ to $\mathbb{R}^m$ performed by the network can be seen as constitued by two different sub-mappings. In the *sequence-to-feature* mapping, the entire input sequence is mapped non-linearly into a feature vector $\mathbf{f} = (f_1, \ldots, f_{N_f}) \in \mathbb{R}^{N_f}$ independent of the sequence length L by *enrolling* a SLFN with h_{sf} hidden neurons and linear output through the sequence as follows:

$$\mathbf{f} \;=\; \rho \sum_{l=1}^{L} \sum_{j=1}^{h_{sf}} \boldsymbol{\alpha}_j \psi\left(\langle \boldsymbol{\nu}_j, \mathbf{x}_l \rangle\right) \tag{5.38}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^{N_f}$ and $\boldsymbol{\nu}_j \in \mathbb{R}^n$ are the input and output weights of the SLFN [6], respectively, the $\langle \cdot, \cdot \rangle$ represents the inner product between two vectors and ρ is a constant. In principle ρ may take any value. For instance, if ρ is set to $\frac{1}{L}$, the feature vector $\mathbf{f}$ represents the average feature representation the entire sequence.

By means of the second sub-mapping, called the *feature-to-property* mapping, the extracted feature vector is mapped into the property vector $\mathbf{o} \in \mathbb{R}^m$ using of SLFN with h_{fp} hidden neurons and linear output as follows:

$$\mathbf{o} \;=\; \sum_{i=1}^{h_{sp}} \boldsymbol{\beta}_i \psi\left(\langle \boldsymbol{\omega}_i, \mathbf{f} \rangle\right) \tag{5.39}$$

where $\boldsymbol{\beta}_j \in \mathbb{R}^m$ and $\boldsymbol{\omega}_j \in \mathbb{R}^{N_f}$ are the input and output weights of the SLFN, respectively.

Considered together, the sequence-to-feature and the feature-to-property networks form a unique multi-layered feed-forward network. The feature vector $\mathbf{f}$ can

[6] For simplicity bias parameters are neglected.

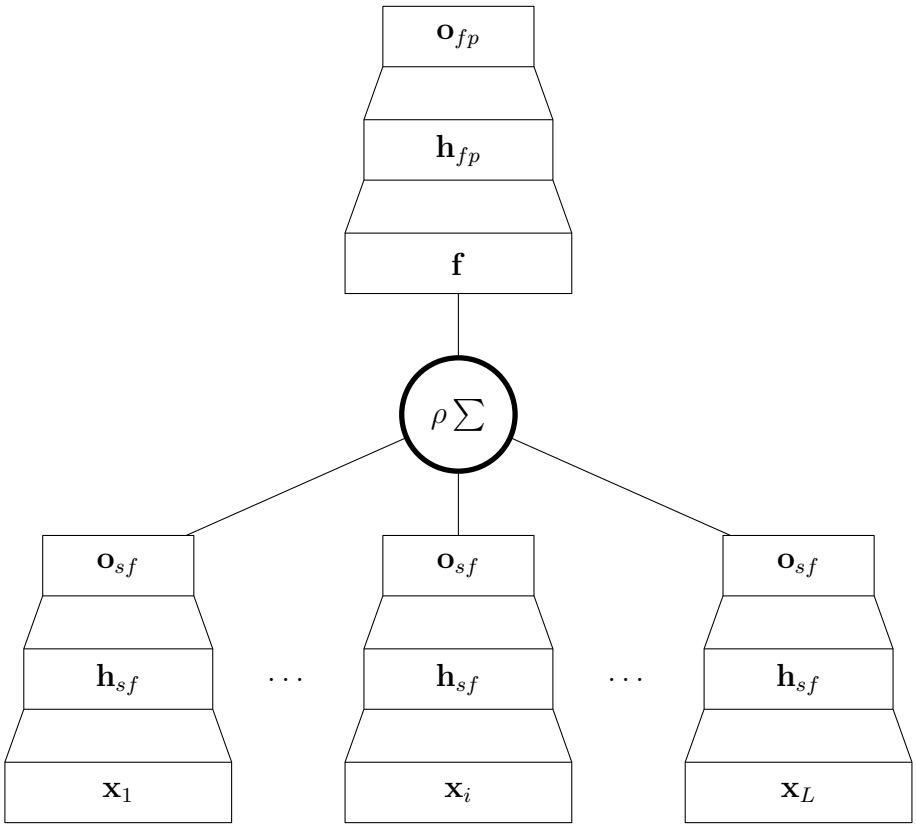


Figure 5.5: The general architecture of a N-to-1 Neural Network.

be seen as a compression of the entire sequence into N_f real-valued descriptors. These descriptors are adaptively tuned in order to minimize the training error and therefore to be the most informative to predict the property of interest. Hence, if training is successful the vector $\mathbf{f}$ is such that different predictive targets (i.e. properties) induce different representation/compression of a sequence. An illustration of the general architecture of a N-to-1 NN is shown in Figure 5.5.

Although lengths of input sequences are arbitrary, the total number of parameters M of the N-to-1 network is independent of sequence lengths and is given by (considering also biases):

$$M = (n+1)h_{sf} + (h_{sf}+1)N_f + (N_f+1)h_{fp} + (h_{fp}+1)m \tag{5.40}$$

5.5.2 Training N-to-1 NNs by gradient-descent

Traditional gradient-descent procedures can be applied to estimate the parameters of N-to-1 NN from training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{N}$ consisting of N pairs of variable-length sequences and relative target vectors. For each sequence of length L, the sequence-to-feature mapping can be interpreted as the application of L identical SLFNs whose contributions to the gradient are simply added together. By this, the gradient of a sum-of-squares error function $E(\mathcal{D}; \boldsymbol{\Theta})$ with respect to parameters in the different layers of the overall network is obtained as follows:

$$\frac{\partial E}{\partial \beta_{i,k}} = \delta_k^{o,fp} \psi\left(\langle \boldsymbol{\omega}_i, \mathbf{f} \rangle\right) \tag{5.41}$$

$$\frac{\partial E}{\partial \omega_{j,i}} = \delta_i^{h,fp} f_j \tag{5.42}$$

$$\frac{\partial E}{\partial \alpha_{r,j}} = \rho \sum_{l=1}^{L} \delta_j^{o,sf} \psi\left(\langle \boldsymbol{\nu}_r, \mathbf{x}_l \rangle\right) \tag{5.43}$$

$$\frac{\partial E}{\partial \nu_{s,r}} = \rho \sum_{l=1}^{L} \delta_{r,l}^{h,sf} x_{l,s} \tag{5.44}$$

where the different back-propagation deltas are given by:

$$\delta_k^{o,fp} = (o_k - t_k) \tag{5.45}$$

$$\delta_i^{h,fp} = \psi'\left(\langle \boldsymbol{\omega}_i, \mathbf{f} \rangle\right) \sum_{k=1}^{m} \beta_{i,k} \delta_k^{o,fp} \tag{5.46}$$

$$\delta_j^{o,sf} = \sum_{i=1}^{h_{fp}} \omega_{j,i} \delta_i^{h,fp} \tag{5.47}$$

$$\delta_{r,l}^{h,sf} = \psi'\left(\langle \boldsymbol{\nu}_r, \mathbf{x}_l \rangle\right) \sum_{j=1}^{nf} \alpha_{r,j} \delta_j^{o,sf} \tag{5.48}$$

5.5.3 Coupling N-to-1 NNs and ELMs

As described above, the whole N-to-1 NN model can be seen a feed-forward network with two non-linear hidden-layers, one in the sequence-to-feature network and one in the feature-to-property networks, respectively. In these models the number of adjustable parameters depends on the dimensions of the different layers of the

whole network as described in Equation 5.40. On real applications, this number can be very high and this can lead to overparametrization and over-fitting of training data.

It is possible to reduce the total number of free parameters by considering N-to-1 models that perform a single non-linear transformation rather than two. In these simplified networks, whose architecture is shown in Figure 5.6, an entire sequence $\mathbf{x}$ of length L can be non-linearly mapped *directly* into the feature vector $\mathbf{f} \in \mathbb{R}^{N_f}$ whose i-th component is given by:

$$f_i = \rho \sum_{l=1}^{L} \psi\left(\langle \boldsymbol{\omega}_i, \mathbf{x}_l \rangle\right) \tag{5.49}$$

where $\boldsymbol{\omega}_i$ is a weight vector which connects the i-th feature component (i.e. hidden neuron) to the input layer. Therefore, the feature-to-property mapping is performed by a linear combination of features as follows:

$$\begin{aligned} \mathbf{o} &= \sum_{i=1}^{N_f} \boldsymbol{\beta}_i f_i = \\ &= \sum_{i=1}^{N_f} \boldsymbol{\beta}_i \left(\rho \sum_{l=1}^{L} \psi\left(\langle \boldsymbol{\omega}_i, \mathbf{x}_l \rangle\right) \right) \end{aligned} \tag{5.50}$$

The whole network defined in Equation 5.50 is then a feed-forward neural network with a single hidden layer. This formulation can be refered to as N-to-1 Single hidden-Layer Feed-forward Networks (N-to-1 SLFNs). The total number of parameters in this model is given by:

$$M = (n+1)N_f + (N_f+1)m \tag{5.51}$$

which is in general less than the number reported in Equation 5.40 at the cost of using a single non-linear mapping rather than two as in N-to-1 NNs.

Besides having a reduced number of free parameters and preserving function approximation capabilities, N-to-1 SLFNs also have the advantage to be trainable in the Extreme Learning Machine framework described in Section 5.3.2. The resulting model, obtained by coupling N-to-1 SLFNs and the ELM learning framework is called N-to-1 ELM [103].

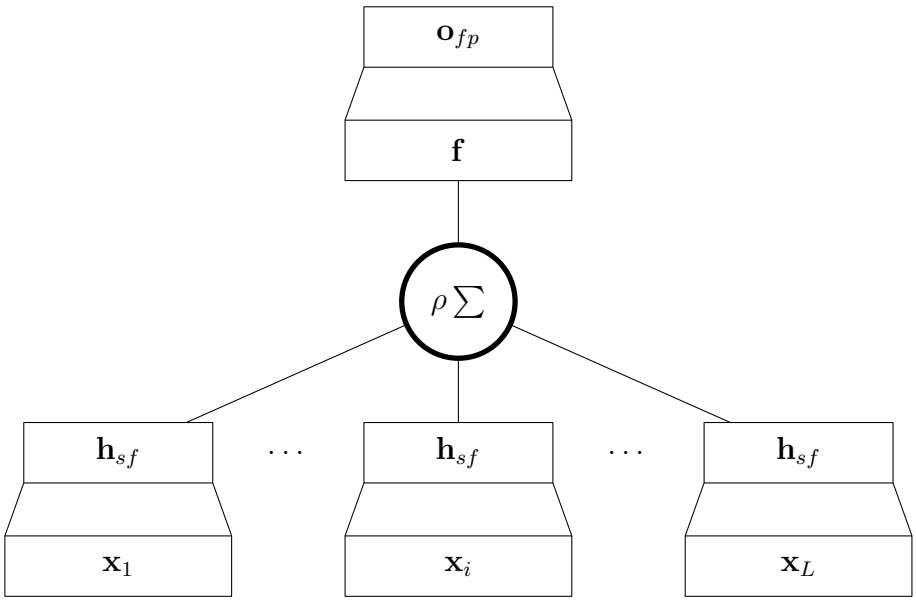


Figure 5.6: The architecture of a simplified N-to-1 network with a single hidden layer.

In N-to-1 ELMs the input weight matrix $\mathbf{\Omega}$ is chosen randomly. Therefore, the output weights can be analitically obtained by least-squares estimation. In particular, given training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{N}$, the hidden layer output matrix $\mathbf{H}$ (see Equation 5.12) can be defined as follows:

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(N)} \end{bmatrix} = \\ &= \begin{bmatrix} \rho \sum_{l=1}^{L_1} \psi \left(\langle \boldsymbol{\omega}_1, \mathbf{x}_l^{(1)} \rangle \right) & , \cdots , & \rho \sum_{l=1}^{L_1} \psi \left(\langle \boldsymbol{\omega}_{N_f}, \mathbf{x}_l^{(1)} \rangle \right) \\ \vdots & \cdots & \vdots \\ \rho \sum_{l=1}^{L_N} \psi \left(\langle \boldsymbol{\omega}_1, \mathbf{x}_l^{(N)} \rangle \right) & , \cdots , & \rho \sum_{l=1}^{L_N} \psi \left(\langle \boldsymbol{\omega}_{N_f}, \mathbf{x}_l^{(N)} \rangle \right) \end{bmatrix}_{N \times N_f} \end{aligned} \tag{5.52}$$

The i-th row of the matrix $\mathbf{H}$ is feature vector representation $\mathbf{f}^{(i)}$ of the i-th input sequence $\mathbf{x}^{(i)}$ of length L_i.

The output layer weights of the overall SLFN can be then obtained analytically as:

$$\mathbf{B}^* = \mathbf{H}^{\dagger}\mathbf{T} \tag{5.53}$$

where $\mathbf{H}^{\dagger}$ is the pseudo-inverse matrix of the matrix $\mathbf{H}$ and

$$\mathbf{B} = [\boldsymbol{\beta}_1^{\intercal}, \cdots, \boldsymbol{\beta}_m^{\intercal}]_{d \times m} \tag{5.54}$$

and

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}^{(1)} \\ \vdots \\ \mathbf{t}^{(N)} \end{bmatrix}_{N \times m} \tag{5.55}$$

are the output weight and the target matrices, respectively.

As described in Section 5.3.2, by using the ELM training algorithm it is possibile to achieve at the same time the minimum training error and the minumum norm for output weights using a procedure that does not necessitate of any hyper-parameter tuning.

5.6 Summary

Classification problems in structured domains arise in many scientific fields. In these contexts, typical objects of interest have an intrinsic complex structures. For instance in Computational Biology data are often available as variable-length sequences (DNA or protein sequences) and methods for protein sequence annotation are extremely important.

In this chapter a novel machine learning method for variable-length sequence classification has been presented. The method, called N-to-1 ELM, is pretty general and can be used for several classification tasks which involve sequence data. The main advantage of the N-to-1 ELM lies in the ability to perform non-linear feature selection by exploiting the entire information contained in the input sequence. In addition, the framework also takes advantage of the powerful ELM training algorithm. Several biological sequence classification tasks can be addressed using N-to-1 ELMs. In the next chapter, an application of the N-to-1 ELM framework that improves TMBB detection in genomic data is described.

Chapter 6

Detection of transmembrane beta-barrel proteins in genomic data

Discrimination of β-barrels (TMBB) from other types of proteins such as α-helical transmembrane or globular proteins is a challenging problem. A major obstacle is due to the cryptic nature of the TMBB structure compared with that of the all-α membrane proteins [124]. Secondly, the available information for the discrimination is quite limited due to the scarcity of known TMBB structures.

In this chapter, the problem of the detection of TMBB proteins from primary sequence is addressed. A machine-learning method is described, based on N-to-1 ELMs introduced in the previous chapter. The performance of the method was measured and compared to other existing approaches using a non-redundant dataset of proteins previously released [33]. In addition, the method was tested on a large-scale dataset of proteins consisting of the entire proteome of the *Escherichia Coli* organism.

The chapter is organized as follows. Section 6.1 provides an overview of the proposed method. In Section 6.2, datasets used, scoring measures and evaluation procedures are described in details. Results and comparisons with other existing approaches for TMBB detection are reported in Section 6.3. Finally, concluding remarks are discussed in Section 6.4.

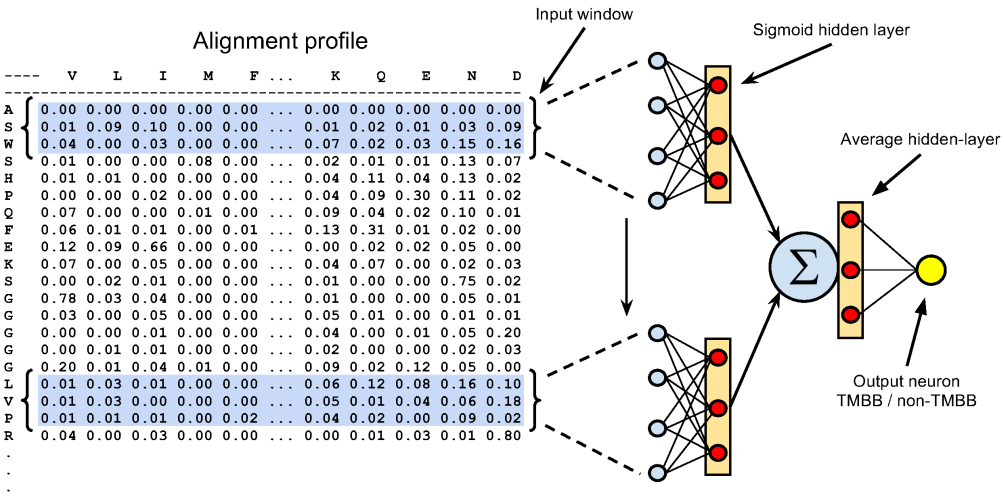


Figure 6.1: Overview of the TMBB detection method.

6.1 Methods

The detection of TMBBs is a typical structured classification problem where a variable-length protein sequence has to be mapped into a single class-membership property: TMBB or non-TMBB. As described Chapter 2, most of the available approaches for TMBB detection have tried to address this problem by extracting information from specific regions of the sequence or by compressing the entire protein into a fixed number of features such as amino acid or dipeptide compositions. These features are then processed by simple statistical predictors or by more advanced machine-learning algorithms.

In Chapter 5, the N-to-1 ELM was presented as a framework to address sequence classification tasks using Single hidden-Layer Feedforward Networks (SLFNs) and Extreme Learning Machines [103]. The method described in this chapter adopts N-to-1 ELMs to encode in the hidden layer of a SLFN the piece-wise information defined by the entire protein sequence as shown in Figure 6.1. More precisely, for a protein chain of length L in the form of an alignment profile of dimension $L \times 20$ (see Section 1.2.4) the hidden layer of a SLFN was used to encode the information of L input vectors obtained by sliding a symmetric window of size $s = 2r + 1$ along the protein sequence, centered each time on a different residue in the sequence. The

hidden layer was computed as follows:

$$h_i = \frac{1}{L}\sum_{l=1}^{L}\sigma\left(\langle \mathbf{w}_i, \mathbf{x}_l\rangle\right) \tag{6.1}$$

where σ is a sigmoid activation function, $\mathbf{w}_i$ are input weights connecting the i-th hidden neuron to the input layer and $\mathbf{x}_l$ is the l-th symmetric window of size s centered at the residue at position l and $\langle\cdot,\cdot\rangle$ represents the inner product between two vectors. With these choices, for a given protein sequence, the hidden layer represented the average sigmoidal transformation of all the motifs (i.e. sliding windows) in the sequence and it was independent of the sequence length.

The output of the SLFN was used to assign a protein to one of the two classes: TMBB or non-TMBB. A sinlge, sigmoidal output neuron was used. The overall function implemented by the network was then defined as follows:

$$o = \sigma\left(\sum_{i=1}^{d}\beta_i\frac{1}{L}\sum_{l=1}^{L}\sigma\left(\langle \mathbf{w}_i, \mathbf{x}_l\rangle\right)\right) \tag{6.2}$$

where β_i is the output weight connecting the i-th hidden neuron to the output neuron and d is the number of hidden neurons. A protein sequence was predicted as belonging to the TMBB class if the network output o was above a fixed predictive threshold.

According to the N-to-1 ELM formulation, all the input weights $\mathbf{w}_i$ of the SLFN were chosen randomly and the network was trained using the ELM training algorithm to obtain the output weigths $\boldsymbol{\beta}$ (refer to Chapter 5 for details).

6.1.1 Model ensembles

For a given choice of the window size s and number of hidden neurons d, different models which exploited a different amount of information from the alignment profile were obtained. In order to leaverage the predictive power of two or more models, ensembles were defined by simply averaging network outputs as follows:

$$o = \frac{1}{N_m}\sum_{i=1}^{N_m} o_s^d \tag{6.3}$$

where N_m is the number of models in the ensemble and o_s^d is the output of the model obtained using a window and an hidden layer of sizes s and d, respectively.

Statistic	Value
Number of chains	14238
Number of TMBB chains	48
Number of non-TMBB chains	14190
Internal homology	50%
Average chain length	240.4

Table 6.1: Statistics about the NRPDB dataset.

6.2 Experimental setup

6.2.1 Datasets

In order to test the performance of the method, two different dataset of proteins were used. The first dataset, NRPDB, was obtained from previous works [33] and it was adopted in order to reliably compare the method with existing approaches on the same data. Furthermore, a large-scale genomic screening was also carried out using the entire proteome of *Escherichia Coli*. Below, these two datasets are described in details.

6.2.1.1 The NRPDB dataset

Statistics about this dataset are reported in Table 6.1. According to authors [33], the NRPDB dataset was generated from all sequences available at the Protein Data Bank. An additional filtering procedure was applied in order to obtain a dataset with at most the 50% of sequence identity between any pair of proteins. Furthermore, the dataset was refined to consider only proteins with lengths between 60 and 4000 residues. This because the detection algorithm described in [33] could only handle sequences in this range of lengths. The NRPDB contains in total 14238 chains, where 48 are true TMBB and 14190 are non-TMBB. Therefore the dataset is really unbalanced in favour of the non-TMBB class and this corresponds to the realistic situation. In addition, the non-TMBB portion covers the full range of SCOP protein fold classes as showed in Figure 6.2. This make the dataset a stringent test case which

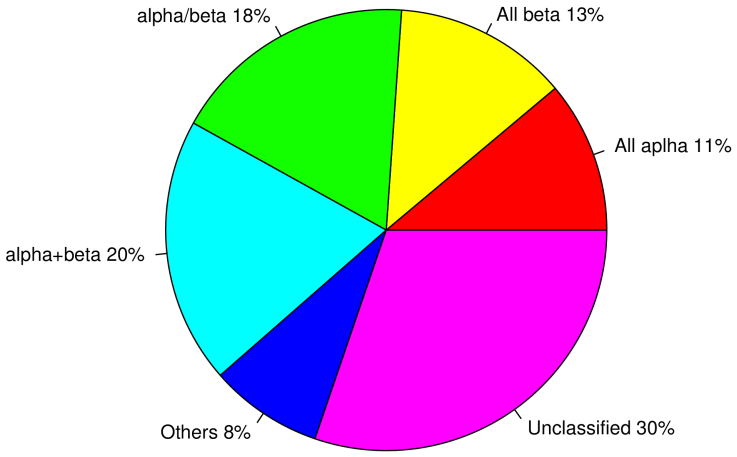


Figure 6.2: Distribution of SCOP protein fold classes in the non-TMBB portion of the NRPDB dataset.

Statistic	Value
Number of chains	2545
Number of TMBB chains	30
Number of non-TMBB chains	2515
Average chain length	339

Table 6.2: Statistics about the E.Coli dataset.

simulate a real full proteome where proteins are distributed into a wide variety of supersecondary structures.

6.2.1.2 The Escherichia Coli proteome

The method was evaluated on the genome of Escherichia coli (K12 strain), which is one of the most comprehensively annotated genomes available. From the UniProt database [111] all protein sequences from E.coli K12 were extracted. Entries annotated as hypothetical or putative were filtered out. Similarly of what has been done for the NRPDB database, proteins that were shorter than 60 or longer than 4000 residues were not included in the dataset. After this selection procedure, a dataset

comprising 30 TMBBs and 2515 non-TMBBs proteins was obtained. Statistics about the E.Coli dataset are showed in Table 6.2.

6.2.2 Scoring indices

Standard scoring indices adopted in literature to evaluate TMBB detection methods were computed to score the performance of the various approaches considered in this work.

In what follows, let TP, TN, FP and FN be, respectively, the true positives, the true negatives, the false positives and the false negatives with respect to the TMBB class (i.e. the positive class). The following scoring indices were evaluated:

- Accuracy (AC) evaluates the number of correctly predicted TMBB proteins divided by the total number of proteins:

$$\mathrm{AC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{6.4}$$

- Specificity (SP) is the number of correctly predicted proteins in the non-transmembrane (i.e. the negative) class divided by the total number of proteins in the class:

$$\mathrm{SP} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{6.5}$$

- Sensitivity (SN) is the number of correctly predicted TMBB proteins divided by the total number of observed TMBB proteins:

$$\mathrm{SN} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6.6}$$

- Positive predicted value (PPV), also referred to as *probability of correct prediction*, measures the probability that TMBBs are correctly assigned among the predicted TMBBs. PVV is defined as the number of correctly predicted TMBBs divided by the total number of predicted TMBBs:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{6.7}$$

- The F1 score is defined as the harmonic mean of PPV and SN:

$$\mathrm{F1} = \frac{2 \times \mathrm{PPV} \times \mathrm{SN}}{\mathrm{PPV} + \mathrm{SN}} \tag{6.8}$$

- The Matthews correlation coefficient (MCC) is:

$$\mathrm{MCC} = \frac{(\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN})}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TP} + \mathrm{FN}) \times (\mathrm{TN} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FN})}} \tag{6.9}$$

Among the different indices, the F1 score and the MCC in particular are the most informative since they provide a balanced measure of the overall performance of a classification method, especially in TMBB detection where datasets are often highly unbalanced in favour of the negative class.

6.2.3 Model selection and cross-validation

In order to fairly evaluate the method a 10-Fold cross-validation procedure was carried out as follows. Firstly, since the NRPDB dataset contained a significant amount of internal sequence homology (precisely the 50%), the 10 subsets were generated by taking into account this information. In particular, protein chains were clustered by local similarity using the BLAST program [2] and a threshold of 25% of sequence homology. The 10 cross-validation subsets were then created assigning entire clusters to each subset. By this, sequence identity between pairs of proteins belonging to two different subsets was <25% and, as a consequence, the homology between training and testing sets on each cross-validation run was reduced. This ensured a more fair evaluation of the method.

Secondly, since some hyper-parameters (i.e. random initial weights, number of hidden neurons, the input sliding window and the predictive threshold) needed to be optimized, validation sets were used as follows. Given the 10 non-homologous subsets generated as described above, for each cross-validation run, 8 subsets were adopted for training and the remaining two were used for validation and testing, respectively. The hyper-parameters were therofore selected using scoring indices computed on the validation sets and then fixed.

Thirdly, another point of concern was related to the stochastic nature of the ELM training algorithm. Since the performance of the method could vary between random

initialization of input weights, five distinct random sets of weights were generated and the performance was evaluated averaging over five independent cross-validation procedures, one for each random set.

Finally, the method used an ensemble of models generated with different input windows and hidden layer sizes (see Section 6.1). For the ensemble selection an exhaustive procedure was adopted based on MCC scores computed on the validation sets. First, all the N-to-1 ELM models whose MCC scored ≥ 0.77 were selected. Then for each combination of these models, an ensemble was defined by averaging their predictions as described in Equation 6.3. Finally, the best performing ensemble on the validation sets was retained.

The final performance was assessed on the test sets without any further hyper parameter tuning.

6.3 Results

6.3.1 Performance of the method on the NRPDB

Several N-to-1 ELM models were evaluated, differing from each other in the number of neurons of the hidden layer d and in the dimension of the sliding window s. MCC scores for each tested model on the validation subsets are reported in Table 6.3. Each MCC score was obtained by averaging over five different random initializations of the first layer of weights as described in the previous section. From these results, it appeared that MCC values are > 0.70 for nearly any input window in the range of 800-1600 hidden neurons. Table 6.3 also reports (within parentheses) values of MCC standard deviations (SDs) for all the window/hidden neuron combinations. The small variations obtained indicate that the method is quite robust and insensitive to the random initialization. Notably, this was also true for a single residue long sliding window, indicating that N-to-1 ELMs are able to detect significant differences between TMBBs and other proteins even encoding a single residue at a time in the hidden layer.

Among the four models that obtained an average MCC score ≥ 0.77 in Table 6.3, all their possible ensembles were computed as explained in Equation 6.3. The models

s	d					
	100	**200**	**400**	**800**	**1600**	**3200**
1	0.65(±0.01)	0.71(±0.01)	0.71(±0.01)	0.73(±0.01)	0.74(±0.01)	0.64(±0.01)
3	0.51(±0.02)	0.58(±0.01)	0.69(±0.02)	0.78(±0.01)	0.77(±0.01)	0.58(±0.01)
5	0.47(±0.04)	0.55(±0.02)	0.63(±0.02)	0.73(±0.01)	0.76(±0.02)	0.54(±0.02)
7	0.48(±0.01)	0.55(±0.03)	0.64(±0.02)	0.75(±0.02)	**0.78(±0.01)**	0.54(±0.03)
9	0.50(±0.02)	0.55(±0.01)	0.63(±0.02)	0.70(±0.03)	0.75(±0.02)	0.55(±0.01)
11	0.51(±0.03)	0.55(±0.02)	0.64(±0.01)	0.71(±0.02)	**0.77(±0.01)**	0.55(±0.02)
13	0.54(±0.03)	0.56(±0.03)	0.62(±0.02)	0.70(±0.03)	0.74(±0.02)	0.56(±0.03)
15	0.55(±0.04)	0.59(±0.05)	0.64(±0.01)	0.69(±0.02)	0.73(±0.02)	0.59(±0.05)
17	0.57(±0.05)	0.58(±0.03)	0.63(±0.02)	0.69(±0.03)	0.75(±0.02)	0.58(±0.03)
19	0.59(±0.03)	0.60(±0.03)	0.64(±0.03)	0.68(±0.01)	0.71(±0.02)	0.59(±0.03)
21	0.59(±0.03)	0.60(±0.03)	0.64(±0.03)	0.68(±0.01)	0.68(±0.02)	0.59(±0.03)

Table 6.3: MCC as a function of window size and hidden layer dimension.

Method	**MCC**	**SN(%)**	**SP(%)**	**PPV(%)**	**AC(%)**	**F1(%)**
ELM_w7/1600	0.77(±0.02)	64(±2)	100(±0)	92(±3)	100(±0)	75(±2)
ELM_w11/1600	0.75(±0.03)	59(±6)	100(±0)	99(±3)	99(±1)	73(±4)
ELME	0.82(±0.02)	73(±4)	99(±1)	92(±3)	99(±1)	81(±3)

Table 6.4: Performance of the N-to-1 ELM Ensemble and its constituents.

for the ensemble were selected on the validation sets, using the procedure described above (see Section 6.2.3). After an exhaustive search, an ensemble of two models that achieved an MCC score of 0.82 was selected. The two models singled out for the ensemble (highlighted in boldface in Table 6.3) had both 1600 neurons in the hidden layer and input windows of 7 and 11 residues, respectively. The selected ensemble (in what follows simply referred to as ELME), together with its constituent models were then evaluated on the test sets. These results are shown in Table 6.4. Although scores obtained on the test sets were slightly lower than those computed using the validation sets (compare Table 6.3 with Table 6.4), ELME performances were still very high achieving an MCC score of 0.82 with SD equal to 0.02 (Table 6.4).

Method	TP	FP	TN	FN	MCC	SN(%)	SP(%)	AC(%)	PPV(%)	F1(%)
FW	46	599	13591	2	0.26	96	96	96	7	13
ELME	46	88	14102	2	0.57	96	99	99	34	51
FW-MRS	37	161	14029	11	0.38	77	99	99	19	3
ELME	37	12	14178	11	0.76	77	99.9	99.8	76	76
k-NN	41	369	13821	7	0.29	85	97	97	10	18
ELME	41	25	14165	7	0.73	85	99.8	99.8	62	72
RBFNets	46	902	13288	2	0.21	96	94	94	5	9
ELME	46	88	14102	2	0.57	96	99.4	99.4	34	51
BOMP	39	227	13963	9	0.34	81	98	98	15	25
ELME	39	19	14171	9	0.74	81	99.9	99.8	67	74

Table 6.5: Performance comparison with FW algorithms

6.3.2 Comparison with previous approaches

In a previous study [33], authors thoroughly compared performances of their algorithms with other available methods (also based on machine learning approaches) on their newly generated dataset NRPDB. In Table 6.5 scoring indices of different methods on the NRPDB dataset are reported. All reported results in Table 6.5 (except for indices relative to the ELME algorithm) are taken from [33]. The following algorithms were considered for comparison:

- the Freeman-Wimley (FW) statistical algorithm based on the calculation of β-barrel scores [33]. An improvement of this algorithm was also evaluated. This extension used randomized sequence analysis (Fw-MRS) to decrease the false positive rate [33].

- a k-NearestNeighbour (k-NN) based algorithm [44];

- the TMBETADISC-RBF method that is based on Radial Basis Function Networks (RBFNets) [87];

- the BOMP method that, similarly to the FW algorithm, is based on a statistical approach [12].

The comparison was carried out using the following protocol. The ELME method was used to predict, in cross-validation, the entire NRPDB dataset. Since the different methods reached different TP rates, the predictive threshold of the ELME algorithm was tuned in order to match the sensitivity of each evaluated algorithm. By this, it was possible to pair-wise compare the ELME algorithm with other approaches on the same conditions.

From results, it was clear that the ELME algorithm outperformed all other methods when evaluated on the NRPDB dataset. In all cases, at the same level of sensitivity, the ELME algorithm was able to drastically reduce the false positive rate as highlighted by the MCC and the PPV indices which always scored significantly better than other algorithms. When ELME was compared with one the best performing methods developed so far, namely the FW-MRS algorithm, the MCC was doubled and the PPV was even quadrupled.

Reaching high levels of accuracy and, at the same time, maintaining a very low false positive rate is particularly important in genomic TMBB detection where the expected number of TMBB protein is very low compared to the typical size of a proteome. This make the ELME algorithm well-suited for large-scale genomic identification of TMBB proteins.

6.3.3 Genomic analysis

One of the main purposes of a TMBB detection method is to sort out TMBB proteins in genomic databases. As additional benchmark, the ELME method was tested on the entire proteome of E.Coli (the dataset used is described in Section 6.2.1.2). The proteome was scanned using the ensemble of N-to-1 ELMs trained on the NRPDB dataset. Since there was some degree of similarity between the E.coli proteins and those in NRPDB, the following procedure to perform the genomic predictions was adopted:

- for each E.coli protein q, a BLAST search of q against NRPDB to identify the most similar sequence p from NRPDB (i.e. the highest BLAST hit of q in NRPDB) was carried out;
- among the 10 NRPDB cross validation sets, the one that contained p was

Method	MCC	SEN(%)	SPE(%)	PPV(%)	AC(%)	F1(%)
ELM w7/1600	0.93(±0.01)	93(±2)	99(±1)	93(±2)	99(±1)	93(±1)
ELM w11/1600	0.93(±0.01)	90(±2)	99(±1)	96(±2)	99(±1)	93(±1)
ELME	0.95(±0.01)	90(±2)	100(±1)	100(±1)	99(±1)	95(±1)

Table 6.6: Performance of the N-to-1 ELM on the E.Coli genome.

identified and the protein q was predicted using the same training set as done for p.

With this procedure, the case of a genomic analysis on never-seen before proteins was simulated. The results on the E.coli dataset are reported in Table 6.6. From these results it was evident that the ELME algorithm achieved even better performances in analyzing the E.coli annotated proteins than in scoring over the NRPDB dataset (compare Tables 6.4 and 6.6).

6.4 Summary

The ability to detect TMBB proteins in genomic data is an important task in Computational Biology, especially comparing the vast amount of available sequence data with respect to known TMBB structures.

In this chapter a new method for TMBB detection based on N-to-1 ELMs was presented. The method was tested on a non-redundant dataset of proteins released in previous works and compared to other state-of-the-art methods for the same problem. The proposed approach significantly outperformed other methods for TMBB detection. As a large-scale benchmark, complete genome-wide detection of TMBBs was performed on the entire proteome of E.Coli. In this benchmark the performance of the method even improved. The method presented in this chapter is at the moment one of the most accurate predictors available for TMBB detection.

The main disadvantage of the proposed approach lies in the fact that, differently from statistical approaches where different parameters have a direct interpretation in terms of psicochemical properties of the protein [33, 124], parameters of our model lack of this clear interpretation. In general, any machine learning based method for

TMBB detection suffers from this drawback. However, when prediction performance is the major goal, a learning approach can discover hidden relationships which can significantly improve the performance over simple statistical approaches.

Chapter 7

Topology prediction of β-barrel membrane proteins

As described in Chapter 2, TMBB topology prediction is an important step of TMBB protein structure prediction. Indeed, topologcial information about membrane-spanning β-strands, inner and outer loops can be used to build a first, coarse model of a target protein. This problem has been addressed widely in literature using a variety of machine-learning based techniques such as HMMs, Neural Networks and Support Vector Machines [55, 76, 37, 13, 6, 82, 38, 96, 42].

In this chpater a method for TMBB topology prediction based on probabilistic modelling is presented. In particular, a new topological model which incorporate different construction rules of β-barrel is introduced in Section 7.1 by means of a regular grammar. Secondly, the model is incorporated into a Grammatical-Restrained Hidden CRF and discriminatively trained from experimental data. The prediction power of the model is tested on a newly generated non-redundant dataset of proteins derived from the PDB. This dataset and other data used to evaluate the performance of the model are desribed in Section 7.1.1. Furthermore, experiments are carried out to compare the performance of the method with top performing approaches for topology prediction also based on machine learning. Results are reported in Section 7.2.

7.1 Methods

7.1.1 Data

Several datasets used to score the different methods for TMBB topology prediction have been so far introcuded in literature. In general, it is difficult to reliably compare available approaches to TMBB topology prediction because different methods were scored using different protein datasets. For this reason three different datasets were considered in this study. This section is devised to the description of these datasets.

7.1.1.1 TMBB38 dataset

The TMBB38 dataset consisted of 38 outer-membrane proteins from Prokaryotes. All the structures of the proteins in the dataset were experimentally determined at high resolution (≤ 2.5 Å). The internal homology in the dataset was reduced such that sequence identity between any pair of sequences was less than 40%. Topological annotation was obtained using the DSSP program [59] by selecting the β-strands that span the outer membrane. Basic statistics about this dataset are listed in Table 7.1 and the distributions of transmembrane strand, inner and outer loops lengths shown in Figure 7.1.

Looking at the distributions, it was clear that the three segment types had different length distributions. Transmembrane strand lengths ranged from 5 to 26 with an average value of 12.6. As expected, inner loops (average length 4.4) were in general shorter than outer loops (average length 15.3).

7.1.1.2 Other datasets used for comparison

In order to reliably compare with other methods, two additional datasets were taken from literature:

- the setPRED-TMBB dataset that comprised 15 TMBB proteins [6]. This dataset was used for performance comparison with methods PRED-TMBB [6] and TMBpro [96].

Statistic	Value
Number of chains	38
Internal homology	40%
Average chain length	385.9
Number of TM strands	530
Number of inner loops	303
Number of outer loops	265
Average TM strand length	12.6
Average inner loop length	4.4
Average outer loop length	15.3
Number of TM residues	6686
Number of periplasmic residues	3940
Number of extracellular residues	4038

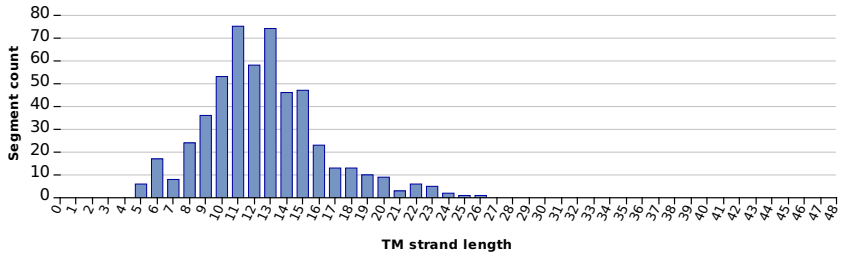
Table 7.1: Statistics about the TMBB38 dataset.

Dataset	PDB id list
setPRED-TMBB	1a0s, 1e54, 1fep, 1i78, 1k24, 1kmo, 1prn, 1qd5, 1qj8, 1qjp, 2fcp, 2mpr, 2omf, 2por
setHMM-B2TMR	1a0s, 1bxw, 1e54, 1ek9, 1fcp, 1fep, 1i78, 1k24, 1kmo, 1prn, 1qd5, 1qj8, 2mpr, 2omf, 2por

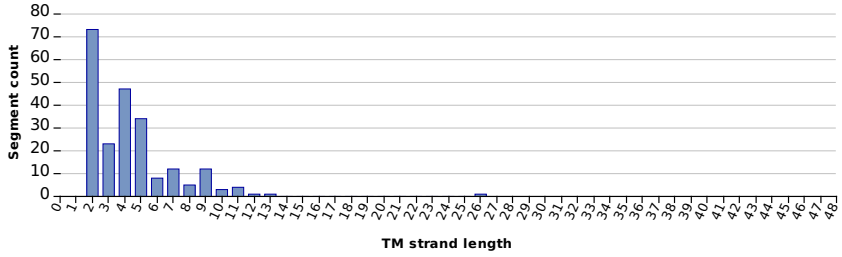
Table 7.2: A summary of the setPRED-TMBB and setHMM-B2TRM datasets.

- the setHMM-B2TMR dataset tht comprised 14 TMBB proteins [76]. This dataset was used for comparison with methods PROFtmb [13] and HMM-B2TMR [76].

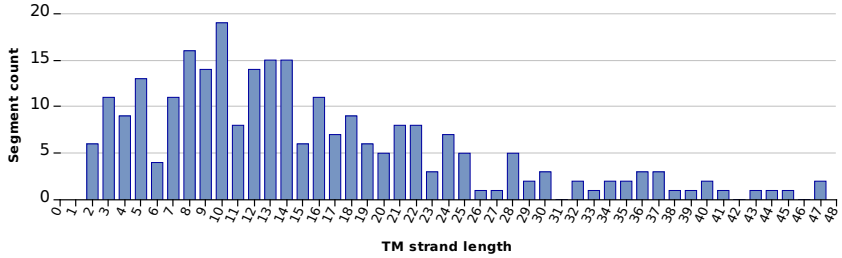
In Table 7.2 a summary of the two dataset is provided. As can be seen, the two datasets mostly overlaped. For each protein, the corresponding topology was obtained using the publicily available PDBTM database [113]. Topological annotations deposited at this database are assigned from PDB structural data using the TM-DET method [114].



(a) TM strands



(b) Inner loops



(c) Outer loops

Figure 7.1: Distributions of TM strand and loop lengths in the TMBB38 dataset.

7.1.2 Input description

It has been shown that evolutionary information can improves the performance of TMBB topology prediction [76]. For this reason, most of the methods available incorporate this information in the form of alignment profiles [76, 13, 42].

For each protein in different datasets, an alignment profile was created using the PSI-BLAST program on the Uniprot non-redundant dataset of sequences (uniref90) [110]. PSI-BLAST runs were performed using a fixed number of cycles set to 3 and an e-value of 0.001.

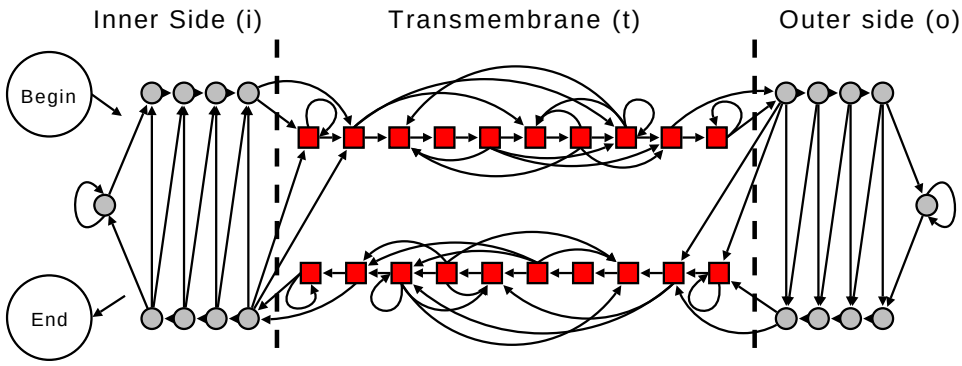


Figure 7.2: Finite-state machine describing the topological model.

7.1.3 The topological model

The topological model used in this work is shown in Figure 7.2 as a Finite-State Transducer (FST). In designing the model, the main construction rules of β-barrels were taken into consideration. These general rules were described in Chapter 2 and can be summarized as follows:

- the closed structure of a β-barrel is defined by antiparalell β-strands, each connected to the closest neighbors;
- the number of β-strands is always even. In other words, both termini of the protein sequence are in the periplasmic side (it is also possible that the initial and/or the final inner loops are missing);
- the three different segment types have different length distributions. Inner loops are in general shorter than outer loops;

The model was essentially based on three different types of states, each associated to one of the three labels: i, representing inner loops, o representing outer loops and t representing transmembrane strands. In Figure 7.2, states associated to transmembrane strands (both in upstream and downstream directions) were drawn as squares while states shown with circles represented loops (both inner and outer). The parity in the number of strands was enforced by confining initial states in inner loops or transmembrane upstream regions and final states only in the inner or

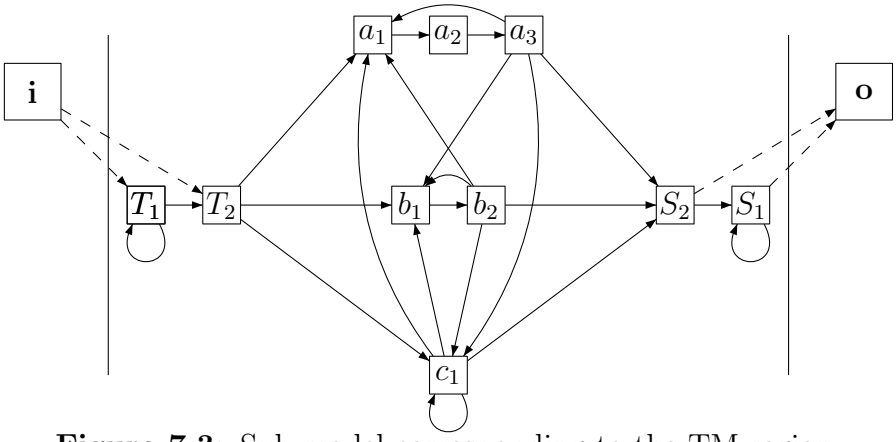


Figure 7.3: Sub-model corresponding to the TM region.

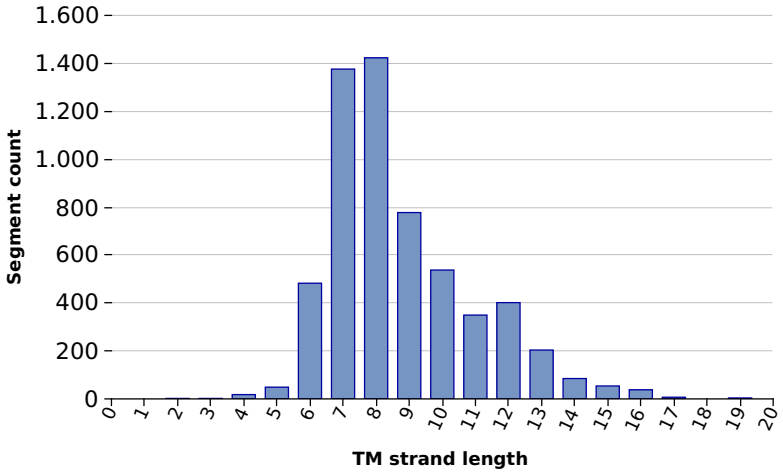


Figure 7.4: Distribution of TM strand lengths in the PDBTM database.

transmembrane downstream regions. By this, it was possible to model also those situations where the initial or the final inner loops were absent.

The sub-model corresponding to the transmembrane region, shown in Figure 7.3, was modelled by taking into consideration the average length of strands in known structures. A simple statistics on gram-negative bacterial TMBBs available at the database of membrane proteins PDBTM [113], revealed that the length of strands ranged from 3 to 19 residues (with an average length of approximately 9 residues). The distribution of strand lengths is shown in Figure 7.4. The shape of the distribution was similar to the one obtained from the TMBB38 dataset (compare Figure 7.1a

and Figure 7.4), slightly shifted to the left.

In the topological model in Figure 7.2, the minimal length was enforced by requiring a minimum number of 3 states ($T_1 \rightarrow c_1 \rightarrow S_2$) to traverse a transmembrane region. Differently from other approaches [13] that used similar models, the maximal strand length was not enforced. In the transmembrane sub-model, three distinct sets of states were included. These corresponded, respectively, to the edge toward the perisplasmic side (states T_1 and $T2$), the strand core (states a_1, a_2, a_3, b_1, b_2 and c_1) and the edge toward the extracellular region (states S_1 and S_2). By this, different physicochemical properties of sub-regions of a single transmembrane strand were captured.

For inner and outer loops no minimal nor maximal lengths were enforced and the corresponding sub-models are essentially the same.

7.1.4 Training the model

The topological model in Figure 7.2 was encoded into a GRHCRF (see Chapter 4). For the specific problem of TMBB topology prediction a GRHCRF was defined as follows:

- input observations corresponded to protein alignment profiles $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)$ where $\mathbf{x}_i \in \mathbb{R}^{20}$ (see Section 1.2.4);

- the set of possible labels was defined as $\mathcal{Y} = \{i, o, t\}$; a TMBB topology for a protein of length L was a sequence of labels $\mathbf{y} = (y_1, \ldots, y_L)$ where each $y_i \in \mathcal{Y}$;

- the set of possible hidden states $\mathcal{H}$ was defined from the set of states of the topological model. In total the model consisted of 38 states. A particular sequence of hidden states $\mathbf{h}$ corresponded to a path in the FST in Figure 7.2;

- hidden states were associated to labels according to the mapping shown in Figure 7.2. This mapping defined a partition of the set $\mathcal{H}$ in three different sets $\mathcal{H}_i$, $\mathcal{H}_o$ and $\mathcal{H}_t$ corresponding to hidden states associated to each of the three labels in the model. By this and given the structure of the automaton, a

sequence of labels $\mathbf{y}$ could be generated by several different sequences of hidden states $\mathbf{h}$.

Features for the GRHCRF model included transition and state features. Given the input encoding based on alignment profiles as described in Section 7.1.2, spatially neighboring state features were defined using symmetric sliding windows centered on each residue. In particular, a single state feature depended on a hidden state h, an offset k in a sliding window of size w and an amino acid type a:

$$s_{h,k,a}(h_j, j, \mathbf{x}) = \begin{cases} x_{j+k,a} & \text{if } h == h_j \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

where k ranges from $-\frac{w}{2}$ to $\frac{w}{2}$, a ranges from 1 to 20 and $x_{j+k,a}$ is the alignment profile frequency of amino acid a at position $j+k$. State features were used to score the compatibility between (the frequency of) an amino acid in a given position and the current hidden state. For a given position j in the sequence the overall feature score was evaluated over the whole window, as follows:

$$\varphi_j(h_j, \mathbf{x}) = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{a=1}^{20} \sum_{h \in \mathcal{H}} \nu_{h,k,a} s_{h,k,a}(h_j, j, \mathbf{x}) \tag{7.2}$$

where $\nu_{h,k,a}$ is the weight associated to the state feature $s_{h,k,a}$.

A transition feature depended on two hidden states h', h:

$$t_{h',h}(h_j, h_{j-1}, j, \mathbf{x}) = \begin{cases} 1 & \text{if } h == h_j \text{ and } h' == h_{j-1} \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

Transition features were used to score transitions between pairs of hidden states. For the position j in the sequence the transition score was defined as:

$$\tau_j(h_j, h_{j-1}, \mathbf{x}) = \sum_{h,h'} \lambda_{h',h} t_{h',h}(h_j, h_{j-1}, j, \mathbf{x}) \tag{7.4}$$

where $\lambda_{h',h}$ is the weight associated to the transition feature $t_{h',h}$. Note that, altought the observation sequence $\mathbf{x}$ was included as an argument of $t_{h',h}$, it did not contribute in any way to the transition score. In general, it is also possible to explicitly include in the transition features a dependency on the observation as done for state features.

Potential functions of the GRHCRF were therefore defined considering transition, window scores, and grammatical constraints, as follows:

$$\Psi_j(h_j, h_{j-1}, y_j, \mathbf{x}) = \exp\left(\tau_j(h_j, h_{j-1}, \mathbf{x}) + \varphi_j(h_j, \mathbf{x})\right) \cdot \Gamma(h_j, h_{j-1}) \cdot \mathbf{1}_{\{h_j \in H_{y_j}\}}$$

The conditional probability of a TMBB topology $\mathbf{y}$ given a protein alignment profile $\mathbf{x}$ was obtained using the GRHCRF formulation:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\sum_{\mathbf{x}} \prod_j \Psi_j(h_j, h_{j-1}, y_j, \mathbf{x})}{\sum_{\mathbf{y},\mathbf{h}'} \prod_j \Psi_j(h'_j, h'_{j-1}, y_j, \mathbf{x})} \tag{7.5}$$

The parameters of the model (transition and state feature weights) were estimated by maximization of the conditional log-likelihood computed on the training set of annotated proteins:

$$\ell = \sum_i \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \tag{7.6}$$

For the maximization the standard L-BFGS algorithm was used [20].

In the decoding phase, a new protein sequence was annotated using alternatively the Viterbi algorithm or the Posterior-Viterbi algorithm as described in Chapter 4.

7.2 Results

7.2.1 Scoring indices

7.2.1.1 Residue-level indices

The accuracy of topology prediction can be evaluated at the level of the single residue using canonical indices for multi-class classification. The following indices were evaluated to score the performance:

- the Q_2 index evaluates the accuracy in a two-class setting, namely, considering a positive transmembrane class t and a single negative non-transmembrane class $\neg t$ which include the other two classes o and i. The Q_2 is obtained as follows:

$$Q_2 = \frac{\mathbf{N}_c(t) + \mathbf{N}_c(\neg t)}{\mathbf{N}} \tag{7.7}$$

where $\mathbf{N}_c(t)$, $\mathbf{N}_c(\neg t)$ and $\mathbf{N}$ are the number of correctly predicted transmembrane residues, the number of correctly predicted non-transmembrane residues (these include also i residues predicted as o and vice versa) and the total number of residues, respectively;

- the $\mathbf{Q}_3$ index evaluates the accuracy in a three-class setting and is computed as follows:

$$\mathrm{Q}_3 = \frac{\mathbf{N}_c(t) + \mathbf{N}_c(i) + \mathbf{N}_c(o)}{\mathbf{N}} \tag{7.8}$$

  where $\mathbf{N}_c(t)$ and $\mathbf{N}$ are defined as above, and $\mathbf{N}_c(i)$ and $\mathbf{N}_c(o)$ are the number of correctly predicted inner-loop residues and the number of correctly predicted outer-loop residues, respectively;

- the sensitivity index evaluate the fraction of correctly predicted transmembrane residues over the total number of real transmembrane residues $\mathbf{N}_r(t)$ and it is defined as:

$$\mathrm{SN} = \frac{\mathbf{N}_c(t)}{\mathbf{N}_r(t)} \tag{7.9}$$

- the positive predictive value index evaluates the fraction of correctly predicted transmembrane residues over the total number of predicted transmembrane residues $\mathbf{N}_p(t)$:

$$\mathrm{PPV} = \frac{\mathbf{N}_c(t)}{\mathbf{N}_p(t)} \tag{7.10}$$

- the Matthews Correlation Coefficient evaluated in a two-class setting as for Q_2:

$$\mathrm{MCC} = \frac{\mathbf{N}_c(t) \times \mathbf{N}_c(\neg t) - \mathbf{N}_w(\neg t) \times \mathbf{N}_w(t)}{\sqrt{(\mathbf{N}_c(t) + \mathbf{N}_w(\neg t)) \times (\mathbf{N}_c(t) + \mathbf{N}_w(t)) \times (\mathbf{N}_c(\neg t) + \mathbf{N}_w(\neg t)) \times (\mathbf{N}_c(\neg t) + \mathbf{N}_w(t))}} \tag{7.11}$$

  where $\mathbf{N}_c(t)$ and $\mathbf{N}_c(\neg t)$ are defined as above, and $\mathbf{N}_w(t)$ and $\mathbf{N}_w(\neg t)$ are the number of wrongly predicted transmembrane and non-transmembrane residues, respectively.

7.2.1.2 Segment-level and protein-level indices

Residue-level indices provide a rough indication of the accuracy of a method for topology prediction. However, in order to obtain an additional insight into the real

performance of the method, segment-level indices are definitely more appropriate. When these indices are calculated, the basic atomic entity is the *segment*, i.e. a contiguous portion of the sequence annotated with the same label.

The first segment-level index evaluated is the the Segment OVerlap (SOV) measure that has been introduced for assessing the performance of methods for protein secondary structure prediction [126]. The SOV globally evaluates the goodness of a prediction considering at the same time different aspects which are typically negleted in residue-level evaluation [126]:

- type and position of segments rather than residue-level assignments;
- natural variation of segment boundaries among families of homologous sequences;
- ambiguity in the position of segment ends due to experimental error or different approaches for secondary structure classification.

The formula for the calculation of the SOV requires some definitions. Let $S_o(l)$ and $S_p(l)$, respectively, the set of observed and predicted segments of type l (in the of TMBB topology prediction there are three types of segments, i, o and t). The set $S(l)$ is defined as the set of all overlapping pairs of segments as follows:

$$S(l) \;=\; \{(s_o, s_p) \;:\; s_o \in S_o(l), s_p \in S_p(l) \text{ and } s_o \cap s_p \neq \emptyset\} \tag{7.12}$$

The set $S'(l) \subseteq S_o(l)$ is the set of observed segments for which there is no ovarlapping segment $s_p \in S_p(l)$:

$$S'(l) \;=\; \{s_o \in S_o(l) \;:\; \forall s_p \in S_p(l), s_o \cap s_p = \emptyset\} \tag{7.13}$$

For the segment type l the $\mathbf{SOV}(l)$ measure is the defined as follows:

$$\mathrm{SOV}(l) \;=\; 100 \times \left(\frac{1}{N(l)} \sum_{(s_o,s_p)\in S(l)} \left[\frac{\mathrm{minov}(s_o, s_p) + \delta(s_o, s_p)}{\mathrm{maxov}(s_o, s_p)} \times \mathrm{len}(s_o) \right] \right) \tag{7.14}$$

where the normalization $N(l)$ is defined as:

$$N(l) \;=\; \sum_{S(l)} \mathrm{len}(s_o) + \sum_{S'(l)} \mathrm{len}(s_p) \tag{7.15}$$

In Equation 7.14, $\text{minov}(s_o, s_p)$ is the actual overlap of the two segments, $\text{maxov}(s_o, s_p)$ is the length of the segment obtained by merging the two overlapping segments s_o and s_p, $\text{len}(s)$ is the length of the segment and $\delta(s_o, s_p)$ is the quantity defined as follows:

$$\begin{aligned}\delta(s_o, s_p) &= \min\Bigg(\text{maxov}(s_o, s_p) - \text{minov}(s_o, s_p), \text{minov}(s_o, s_p), \\ &\quad \lceil\frac{\text{len}(s_o)}{2}\rceil, \lceil\frac{\text{len}(s_p)}{2}\rceil\Bigg)\end{aligned} \tag{7.16}$$

The formula in Equation 7.14 can be easily extended to evaluate multi-type segmentation as follows:

$$\text{SOV}_n = 100 \times \left(\frac{1}{N}\sum_{l}\sum_{(s_o,s_p)\in S(l)}\left[\frac{\text{minov}(s_o, s_p) + \delta(s_o, s_p)}{\text{maxov}(s_o, s_p)} \times \text{len}(s_o)\right]\right) \tag{7.17}$$

where n is the number of different segment types and the normalization is obtained as:

$$N = \sum_{l} N(l) \tag{7.18}$$

Further details about the properties of **SOV** can be found in [126].

Besides the **SOV** measure, another segment-level index, namely the number of observed segments of type l whose location is correctly predicted, was taken into consideration. An observed segment is correctly located if it has a minumum overlap with the corresponding predicted segment. For a given protein this number was obtained as:

$$\text{SCP}(l) = |S_c(l)| = |\{s_o^i \,:\, \text{minov}(s_o^i, s_p^i) > \theta\}| \tag{7.19}$$

where the superscript in s_o^i and s_p^i indicate the i^{th} predicted and observed segments, respectively, and θ is a threshold corresponding to the minimum overlap required.

Protein-level indices globally evaluate the topology. A protein transmembrane topology is said to be correctly predicted if the number of observed and predicted transmembrane segments is the same and all observed segments are predicted at the

correct locations. The Protein OVerlap (POV) is a binary measure defined for a given protein q and segment type t as follows:

$$\mathrm{POV}(q,t) = \begin{cases} 1 & \text{if } |S_o(l)| = |S_p(l)| \text{ and } \mathrm{SCP}(l) = |S_o(l)| \\ 0 & \text{otherwise} \end{cases} \tag{7.20}$$

For a set of proteins Q the average POV provides the percentage of correctly predicted proteins:

$$\mathrm{POV}(l) = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{POV}(q,l) \tag{7.21}$$

The POV depends on the threshold θ used to evaluate the set $\mathrm{SCP}(l)$. Typical thresholds are:

- a fixed threshold value, typically 2 or 3 residues [42];
- the minimum of half-lengths of the two segments:

$$\theta = \min(\frac{\mathrm{len}(s_o)}{2}, \frac{\mathrm{len}(s_p)}{2}) \tag{7.22}$$

- the average half-length:

$$\theta = \frac{1}{2}\left(\frac{\mathrm{len}(s_o)}{2} + \frac{\mathrm{len}(s_p))}{2})\right) \tag{7.23}$$

Among the three options, the average half-length is the most stringent threshold.

7.2.2 Cross-validation

All the experiments on the TMBB38 dataset were carried out by means of a 19-Fold cross-validation procedure. The 19 subsets for the cross-validation experiments were generated such that there was no sequence identity >25% between two elements belonging to disjoint sets. For all tests performed on both datasets setPRED-TMBB and setHMM-B2TMR, given the small number of proteins, a *leave-one-out* cross validation was instead adopted.

Index	Score
$Q_2(\%)$	85
$Q_3(\%)$	86
MCC	0.71
SN(%)	85
PPV(%)	84
$SOV_3(\%)$	89
$SOV(t)(\%)$	93
$SCP(t)(\%)$	69 (367 / 530 segments)
$POV_1(t)(\%)$	66
$POV_{ave}(t)(\%)$	66

Table 7.3: Scoring indices of the GRHCRF model on the TMBB38 dataset.

7.2.3 Performance on the TMBB38 dataset

Scoring measures of the GRHCRF model evaluated on the TMBB38 dataset are listed in Table 7.3.

The indices are defined as described above. SOV_3 refers to the segment overlap index calculated considering all the three segment type (i, o, t) using the formula in Equation 7.17 while $POV_1(t)$ and $POV_{ave}(t)$ refer to the protein overlap index calculated using a threshold of 1 and the average of the half-length (see Equation 7.23), respectively.

Results showed that performances of the model were comparable to other state-of-the-art approaches [5] reaching a MCC of 0.71 and a very high $SOV(t)$ of 93%. Among the 38 protein chains in the dataset, the model was able to correctly identify topologies of 25 proteins. This was true even when the more stringent correctness criterion was applied, as demonstred by a $POV_{ave}(t)$ of 66%.

A detailed evaluation of scoring indices for individual proteins in the TMBB38 dataset whose topology was correctly predicted is shown in Table 7.4. Similarly, Table 7.5 reports scoring indices for wrongly predicted proteins. In both tables, the number of observed N_β and predicted N_β^p transmembrane strands are also reported. Proteins are sorted by number of beta strand in descreasing order.

Protein	N_β	N_β^p	SCP(%)	SOV_3(%)	SOV(%)	Q_2(%)	Q_3(%)	PPV(%)	SN(%)	MCC
1xkhA	22	22	22	98	99	91	91	90	87	0.80
1xkwA	22	22	22	97	99	90	90	95	83	0.80
1fepA	22	22	22	97	98	91	91	87	88	0.80
1kmoA	22	22	22	94	95	87	87	87	80	0.73
2odjA	18	18	18	98	98	90	90	85	95	0.80
2omf_	16	16	16	97	99	89	89	84	96	0.78
1prn_	16	16	16	98	98	90	90	92	89	0.80
1e54A	16	16	16	97	98	88	88	90	89	0.75
1pho_	16	16	16	98	99	90	90	85	98	0.81
1osmA	16	16	16	97	98	87	87	82	95	0.76
2j1nA	16	16	16	98	99	90	90	84	98	0.81
2por_	16	16	16	96	97	87	87	90	86	0.73
2f1cX	14	14	14	90	94	82	82	85	87	0.60
1t16A	14	14	14	95	93	87	87	94	78	0.75
1ildA	12	12	12	95	95	85	85	77	94	0.71
2qomA	12	12	12	98	100	88	88	96	85	0.75
1i78A	10	10	10	80	82	70	70	95	63	0.46
2ervA	8	8	8	93	100	87	87	84	98	0.73
1p4tA	8	8	8	92	93	77	77	96	73	0.54
1ormA	8	8	8	93	100	82	82	80	95	0.61
2f1tA	8	8	8	90	95	79	79	81	82	0.58
2ge4A	8	8	8	99	99	88	88	81	96	0.77
1yc9A	4	4	4	100	100	98	98	98	88	0.92
1tqqA	4	4	4	99	96	96	96	91	71	0.78
1wp1A	4	4	4	100	100	97	97	93	82	0.86

Table 7.4: Detail of performance measures on individual protein chains whose topologies were correctly predicted.

Proteins whose topologies were correctly predicted had a number of strands that ranged from 4 to 22. These covered the entire range in the dataset.

The protein 1k24 chain A had a topology prediction almost completely wrong with a MCC of -0.04 (very close to the random prediction). This could be due to the poor quality of the alignment profile. This quality can be evaluated considering the sequence diversity of the multiple sequence alignment from which the profile was generated. The sequence diversity was measured by the N_{eff} score (nunmber of effective sequences in the alignment) which was computed as the average exponentiated

Protein	N_β	N_β^p	SCP(%)	SOV_3(%)	SOV(%)	Q_2(%)	Q_3(%)	PPV(%)	SN(%)	MCC
2grxA	22	22	21	94	95	86	86	83	82	0.71
2gufA	22	20	3	90	90	87	87	94	79	0.75
2hdfA	22	18	1	84	79	84	83	89	71	0.67
1mprA	18	20	1	84	96	81	81	79	88	0.62
1a0sP	18	20	1	84	96	84	84	79	94	0.69
1af6A	18	20	1	84	97	81	80	80	87	0.61
2qdzA	16	20	0	75	94	83	81	72	84	0.65
2o4vA	16	20	0	69	95	79	72	74	85	0.58
1tlwA	12	10	1	76	77	77	76	83	70	0.55
1uynX	12	14	0	76	98	79	76	81	85	0.54
1k24A	10	10	4	35	50	51	39	62	60	-0.04
1mm4A	8	8	2	58	78	71	55	61	88	0.46
2gr7A	4	6	0	53	80	74	67	55	92	0.54

Table 7.5: Detail of performance measures on individual protein chains whose topologies were wrongly predicted.

entropy of the alignment profile as follows:

$$\mathrm{N}_{eff}(\mathbf{x}) = \frac{1}{L} \exp\left(\sum_i \sum_{j:x_{i,j}\neq 0} -x_{i,j} * \log x_{i,j} \right) \tag{7.24}$$

The N_{eff} ranges from 0 to 20, namely the number of different amino acids, and measures the average information content of rows of the alignment profile. If applied to the alignment profile of the protein 1k24 chain A the above formula gave a N_{eff} value of 1.53 which was very low. This could in part explain why the method poorly performed on this protein.

Besides the above discussed case, the performance of the model was relatively good on almost all proteins, with a MCC and a SOV ranging from 0.46 to 0.92 and from 77% to 100%, respectively.

To gain further insight into topology prediction, the posterior probability estimate of a residue to be in a β-strand was also analyzed. Figures 7.5 and 7.6 illustrate the posterior probability signal along with the observed and the predicted topologies of two proteins with a different number of β-strands. The analyzed proteins were the 1xkh chain A with 22 transmembrane strands and the 1yc9 chain A with

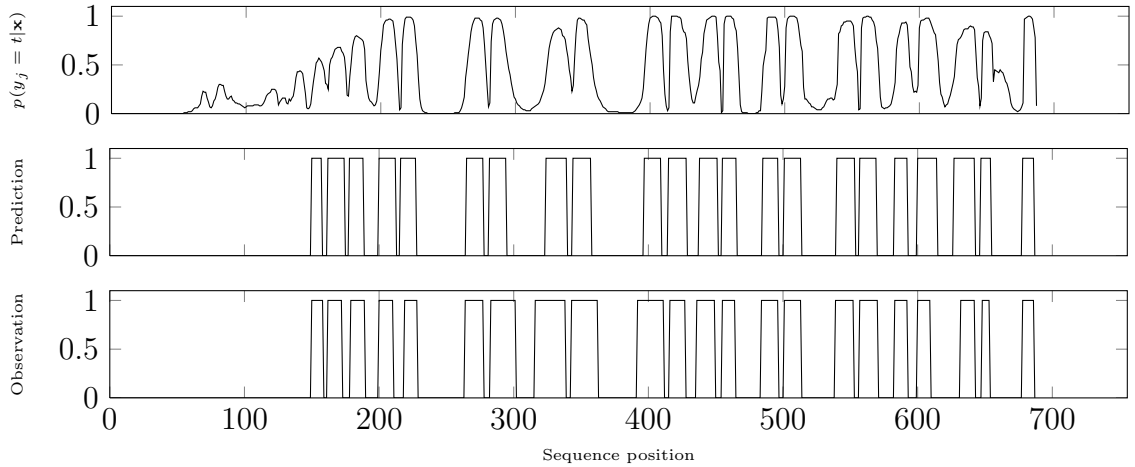


Figure 7.5: Topology prediction of protein 1xkh chain A.

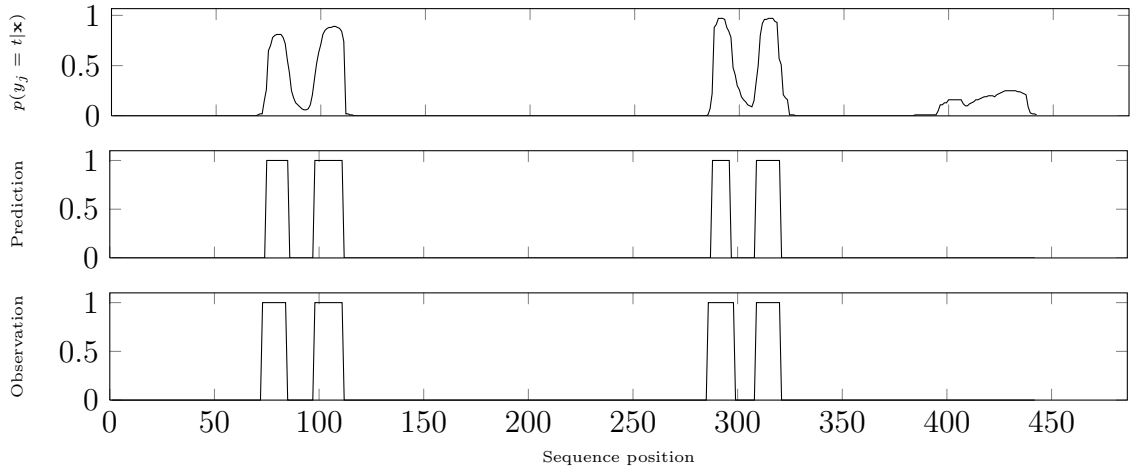


Figure 7.6: Topology prediction of protein 1yc9 chain A.

4 strands. Both topologies were correctly predicted. From this analisys, it could be observed that posterior probability signals were almost perfectly superposable with the observed topologies, with exceptions at the N-terminus and at the C-terminus of 1xkh:A and 1yc9:A, respectively. This demonstrated that the posterior probability is well-estimated by the model.

The posterior probability signal can be also used to try to interpret why some proteins are wrongly predicted. In Figure 7.7 the signal is shown for the protein 2grx chain A. The 22 transmembrane strands of this protein were all correctly located

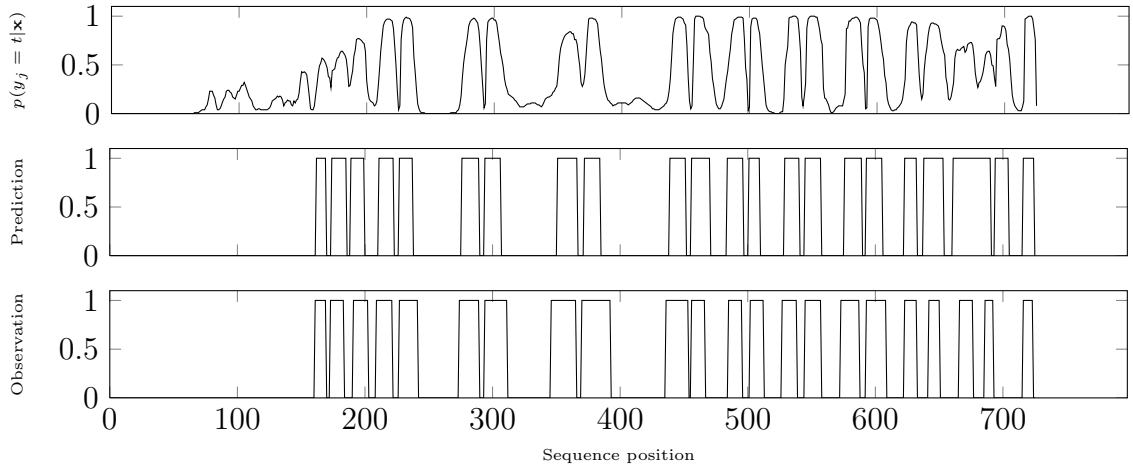


Figure 7.7: Topology prediction of protein 2grx chain A.

except for 21th predicted strand which was shifted with respect to the corresponding observed strand. By observing the probability signal at the positions where the error occurred it could be seen that the probability was not well-estimated and this caused the 20th strand to be predicted too long leading to the inevitable shift at the 21th strand.

Another source of errors could also be due to the presence of β-strands which do not cross the membrane (i.e. they are not part of the protein topology). These strands were detected and predicted as transmembrane segments. For instance, in Figure 7.8 also the secondary structure of the protein 2qdz chain A is shown. As can be seen, the posterior probability was poorly estimated in correspondence of the non-transmembrane strands located at the N-terminus and around position 450 of the protein sequence. This led to overpredicted transmembrane strands at the same locations.

7.2.4 Comparison with linear-chain CRFs

As decribed in Chapter 4, a GRHCRF is essentially an extension of standard linear-chain CRFs. In order to highlight the advantage of using this modelling strategy, GRHCRFs were compared with standard linear-chain CRFs on TMBB topology prediction. The grammar depicted in Figure 7.2 could not be handled

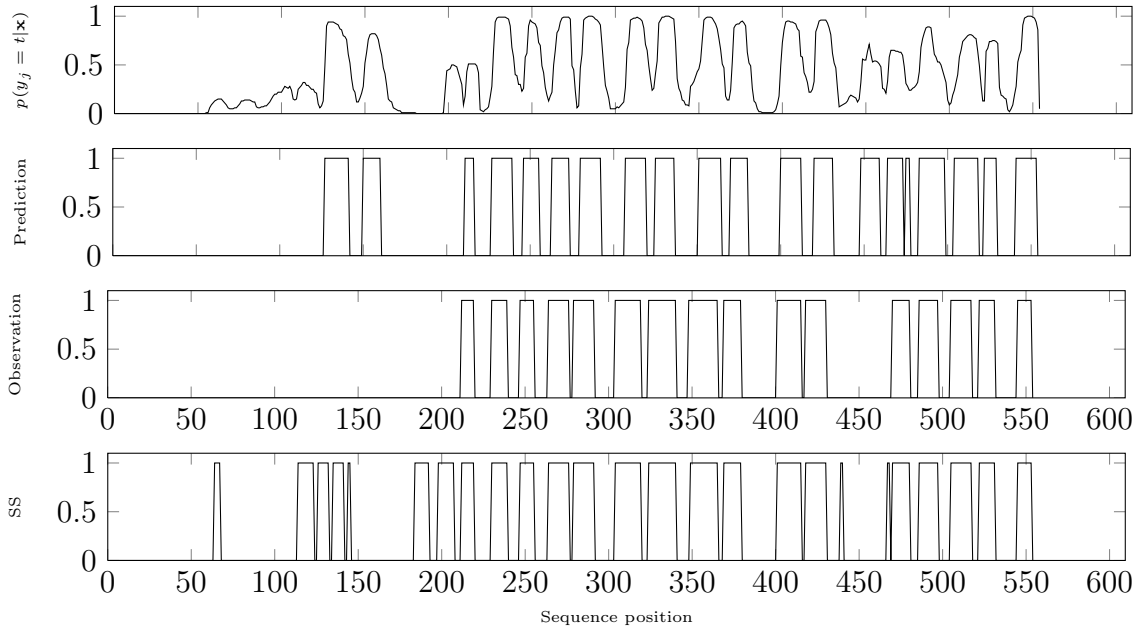


Figure 7.8: Topology prediction of protein 2qdz chain A.

directly by linear-chain CRFs. This was due to the fact that the grammar was defined such that multiple hidden-state paths corresponds to a single label sequence. Since linear-chain CRFs are fully-observable models, they can handle only grammars that are defined over label sequences. In other words, a linear-chain CRF is equivalent to a GRHCRF where the grammar is defined such that there is a one-to-one mapping between label and hidden state sequences. Obviusly, the set of such models is a subset of all possible models that can be defined using GRHCRFs.

For this reason, different linear-chain CRFs were defined by considering simplified topological models derived from the one in Fugure 7.2. In particular, the three models shown in Figure 7.9 were tested.

In defining these models, the number of states was mantained fixed and transitions were removed or added in order to achieve a one-to-one mapping between state and label sequences as discussed above. In both the first and third CRF models in Figure 7.9, sub-models corresponding to the transmembrane strand were modified whereas inner and outer loops (which separately met the one-to-one mapping condition) were not changed. In the second model both loops and transmembrane

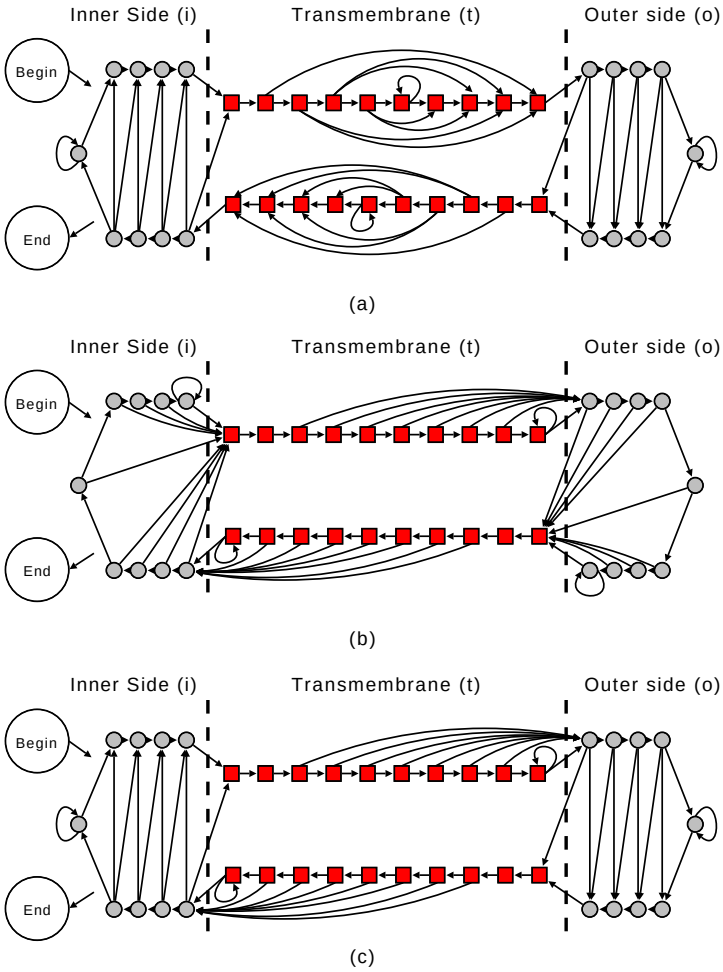


Figure 7.9: Three different topological models for linear-chain CRFs.

sub-models were modified.

The four different topological models were trained and tested on the TMBB38 dataset. Results are listed in Table 7.6.

For GRHCRFs, the decoding phase was performed using the Posterior-Viterbi algorithm. For linear-chain CRFs also the standard Viterbi decoding was tested. From results in Table 7.6, the GRHCRF outperformed the three linear-chain CRF models in all the reported indices. Furthermore, the Posterior-Viterbi always led to better performance compared to the standard Viterbi.

Method	Decoding	POV_{ave}(%)	Q_2(%)	MCC	SN(%)	PPV(%)
CRF-1	Viterbi	26	72	0.47	59	80
CRF-1	PosteriorViterbi	39	77	0.54	71	80
CRF-2	Viterbi	34	76	0.52	63	82
CRF-2	PosteriorViterbi	47	80	0.60	74	82
CRF-3	Viterbi	29	72	0.45	60	79
CRF-3	PosteriorViterbi	45	76	0.52	70	79
GRHCRF	PosteriorViterbi	66	85	0.70	83	84

Table 7.6: Performance of different CRF-based models on the TMBB38 dataset.

7.2.5 Comparison with other approaches

The predictive performances of the GRHCRF model were compared to other state-of-the-art methods for TMBB topology prediction. The following methods were considered:

- PROFtmb [13]
- HMM-B2TMR [76]
- PRED-TMBB [6]
- TMBpro [96]

The first three approaches are all specifically designed to predict TMBB protein topology. These methods are all based on HMMs and alignment profiles whereas they differ in the topological model adopted [76, 13, 6]. TMBpro is a three-stage method which includes a topology predictor, a β-contact predictor and a tertiary structure template-based predictor [96]. In TMBpro topology prediction is carried out using Recursive Neural Networks (RNNs) and dynamic-programming refinement [96].

For sake of comparison, several tests were performed on different datasets, as described in Section 7.1.1. In a first experiment, the GRHCRF model was trained and tested on the dataset setHMM-B2TMR. Results are reported in Table 7.7 as well as scoring indices for PROFtmb and HMM-B2TMR methods obtained on the same dataset. From this experiment, the performance of the GRHCRF appeared in

Method	Q_2(%)	SN(%)	PPV(%)	MCC	SOV(t)(%)	SOV_2(%)	POV_{ave}(%)
PROFtmb	83	80	87	0.69	93	79	-
HMM-B2TMR	84	80	87	0.69	94	91	-
GRHCRF	86	79	84	0.70	94	95	73

Table 7.7: Performances of different methods on the setHMM-B2TMR dataset.

Method	Q_2(%)	SN(%)	PPV(%)	MCC	SOV(t)(%)	POV_1(%)	POV_{ave}(%)
TMBpro	88	97	95	0.75	91	79	-
PRED-TMBB	84	94	95	0.72	-	57	-
GRHCRF	85	81	81	0.70	95	71	71

Table 7.8: Performances of different methods on the setPRED-TMBB dataset.

line with state-of-the-art methods such as PROFtmb and HMM-B2TMR. In spite of a lower PPV(t), the GRHCRF model reached a very high SOV of 94-95% suggesting the method is able to correctly identify protein topologies. In fact the method was able to correctly predict topologies of 11 over 15 proteins in this dataset. Unfortunately, this index was not computed for other methods and hence it could not be used for comparison.

Table 7.8 reports results obtained on the setPRED-TMBB dataset. In this test the method was compared with both TMBpro and PRED-TMBB approaches. In this dataset the GRHCRF model performed slighty worse. The method global per-residue accuracy, evaluated by the Q_2, is of 84%. In segment-level indices a very high SOV(t) (95%) was obtained. The per-protein accuracy was 71%, slightly lower than the one of obtained by TMBpro (79%).

As a final test, different methods were tested on the TMBB38 dataset which contained more proteins (38 chains) and it was updated to more recent data available on the PDB. Performance measures are reported in Table 7.9. Results for HMM-B2TMR and the GRHCRF reported in Table 7.9 were obtained in cross-validation (rows HMM-B2TMR-cv and GRHCRF-cv, respectively). Furthermore, HMM-B2TMR was retrained using the topological model in Figure 7.2. These results were therefore directly comparable. In contrast, PRED-TMBB and TMBpro

Method	$Q_2(\%)$	MCC	$Q_3(\%)$	SOV(t)(%)	$SOV_3(\%)$	$POV_{ave}(\%)$
HMM-B2TMR-cv[1]	80	0.62	83	93	81	58
GRHCRF-cv[1]	85	0.70	93	89	87	66
PRED-TMBB[2]	79	0.58	76	85	79	21
TMBpro[2]	80	0.59	74	89	63	50
GRHCRF-bs[3]	90±2	0.8±0.03	89±2	95±3	92±5	74±7

[1] Results obtained in cross-validation.

[2] Results obtained using the publicily available web server.

[3] Results obtained using the bootstrapping procedure described in text.

Table 7.9: Comparison on the TMBB38 dataset.

were tested using directly the corresponding web interfaces to predict the entire dataset TMBB38 (both methods are available as web applications). In this experiment, the performance of the GRHCRF model significantly outperformed all other methods. However, it should be pointed out these results were hardely comparable because they were obtained in different conditions (GRHCRF and HMM-B2TMR were evaluated in cross-validation while other methods were not).

In order to make the comparison among different methods more fair, an additional test was performed. In principle, other methods could have been re-trained on datasets which overlap with the TMBB38 dataset. Since the respective training sets were in general not known, a possible way to try to reproduce the same testing conditions was to validate the GRHCRF model using the following *bootstrapping* approach. The model was repeatdely trained using randomly selected subsets of the TMBB38 dataset. The size of each random subset was set to 15, which was approximately the size of datasets used to test both PRED-TMBB and TMBpro methods in cross-validation (whose results taken from literature are reported Table 7.7 and Table 7.8, respectively). Each trained GRHCRF model was then used to predict the entire TMBB38 dataset. Scoring indices obtained averaging over 100 independent runs as well as standard deviations are listed in last row of Table 7.9. By taking into consideration standard errors, the performance on this test was comparable to the one obtained by the GRHCRF model in cross-validation and still outperformed

other approaches.

7.3 Summary

In this chapter the problem of topology prediction of transmembrane β-barrel proteins was addressed. A new topological model was described whose paramenters were discriminatively trained using Grammatical-Restrained Hidden CRFs.

The model was tested on a newly generated non-redundant dataset of TMBB proteins derived from high-resolution data in the Protein Data Bank. When validated on this dataset the model was able to correctly identify protein topologies of 25 out of 38 proteins even when the most stringent scoring measures was used in classifying topology predicitions.

To highlight the advantages of using a GRHCRF formulation, the model was compared with three different sub-models trained with standard linear-chain CRFs. These experiments showed that the GRHCRF model significantly outperformed standard CRF models in all reported scoring measures. Furthermore, the above tests also validated the effectiveness of PosteriorViterbi decoding with respect to standard Viterbi for both GRHCRF and CRF models.

The model was further compared with other approaches previously released for topology prediction. Different datasets available in literature were adopted and experiments performed in several different conditions. In all the experiments, the model achieved performances that were comparable or superior to other state-of-the-art methods.

Conclusions

Transmembrane β-barrels (TMBBs) represent an important class of proteins that perform essential functions and they play a key role as potential drug targets. For these reasons, computational methods devised to elucidate TMBB protein structure are of prominent importance. Nonetheless, TMBB protein structure prediciton is difficult because of the low number of known TMBB structures, a fact which limits the applicability of standard comparative protein modelling techniques.

In this thesis machine-learning based methods were described for TMBB protein structure prediction. In particular, two prediction tasks were addressed:

- the detection of TMBBs in large datasets of proteins;
- the prediction of the topology of TMBB proteins.

The TMBB detection method presented in this thesis is based on N-to-1 Extreme Learning Machines, a machine-learning framework devised to address variable-length sequence classification tasks. The proposed approach was validated on a non-redundant dataset of proteins. Furthermore, genome-wide detection was tested using the E.Coli proteome. In both tests, the method significantly outperformed other existing state-of-the-art approaches for TMBB detection. One of the main strength of the proposed approach is that it can detect TMBBs with a very low false positive rate compared to the one achieved by other methods. This makes the method well-suited for genome-wide detection where thousands of proteins are analysed and the datasets are tipically highly unbalanced in favour of the non-TMBB class.

In regard to TMBB topology prediction, a probabilistic method based on Grammatical-Restrained Hidden Conditional Random Fields was presented. In particular,

a novel topological model was defined by taking into consideration construction rules and transmembrane segment length distributions of β-barrels. All these constraints were defined by means of a regular grammar and the parameters of the topological model estimated using the GRHCRF discriminative framework. The proposed topology predictor was tested on a newly-generated dataset of proteins obtained from high-resolution data available at the PDB. When validated on this dataset, the method was able to correctly predict protein topologies of 25 out of 38 proteins in the dataset, outperforming a similar HMM validated under identical testing conditions. The proposed model was also thoroughly compared with other state-of-the-art approaches for TMBB topology prediction. To carry out this comparison, different datasets available in literature were used. Although the comparison was difficult because different methods reported different scoring indices, the performance of the proposed model was measured to be, on average, comparable or superior to the performances of other methods.

Besides the specific applications addressed in this thesis, the machine-learning frameworks described here have a broader applicability. The GRHCRF framework was introduced as an extension of the standard linear-chain CRF to address sequence labelling tasks. The main advantage of GRHCRF lies in the explicit introduction of hidden variables which are not observable at traning time. By this, it is possible to insert into the model prior knowledge by means of a regular grammar defined over hidden state sequences. In contrast, linear-chain CRFs are fully observable models, meaning that all the variables must be observed at training time. This fact can seriously affect the modelling capability of linear-chain CRFs, especially on real-world problems that can be easily modelled by considering hidden sub-structures underlying the label dynamics that are often not directly observable at training time. These problems include many important tasks in Computataion Biology such as protein secondary structure prediction, coiled-coil prediction and signal/target peptide prediction that are usually addressed by means of probabilistic modelling techniques. In all these tasks the GRHCRF framework represents an alternative to standard probabilistic modelling based on HMMs.

Future perspectives

The following are possible future directions of the reaseach conducted in this thesis:

- the performance of β-barrel topology prediction could be improved in several ways. A first, simple idea could be to improve the topological model in order to reduce overpredictions of transmembrane strands. As described in Chapter 7, this is often due to the presence of non-transmembrane β-strands which are detected and predicted as membrane-spanning segments. The model could be improved, for instance, by explicitly introducing states to model the non-transmembrane β-strands. Furthermore, this improvement could be accompained by introducing some sort of non-linear feature selection in order to better discriminate between membrane-spanning and non-transmembrane strands. This and other research directions are at the moment under investigation.

- An interesting direction would be to investigate whether the adoption of different training strategies for GRHCRF could led to improvements in TMBB topology prediction. Since GRHCRFs are latent-variable models, a variety of alternative training schemes could be tested such as Expectation Maximization-like algorithms or sampling techniques.

- In TMBB detection, an improved strategy for ensemble definition and selection could led to better performances by improving the way contributions from different-sized windows are exploited. Several possible strategies are in course of study.

- In this thesis β-contact prediction was not addressed. However, a comprehensive pipeline for TMBB protein structure prediction would eventually also include this step. For this reason, also methods for β-contact prediction are currently under development.

Bibliography

[1] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. *Molecular biology of the cell*, 4 ed. Garland Science Taylor & Francis Group, 2002.

[2] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res. 25*, 17 (Sept. 1997), 3389–402.

[3] Andrew, G., and Gao, J. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning* (New York, NY, USA, 2007), ICML '07, ACM, pp. 33–40.

[4] Anfinsen, C. B. Principles that govern the folding of protein chains. *Science (New York, N.Y.) 181*, 4096 (July 1973), 223–230.

[5] Bagos, P., Liakopoulos, T., and Hamodrakas, S. Evaluation of methods for predicting the topology of beta-barrel outer membrane proteins and a consensus prediction method. *BMC Bioinformatics 6* (2005), 7–20.

[6] Bagos, P. G., Liakopoulos, T., Spyropoulos, I. C., and Hamodrakas, S. J. A Hidden Markov Model method, capable of predicting and discriminating beta-barrel outer membrane proteins. *BMC Bioinformatics 5* (2004), 29.

[7] Baldi, P. Computing with Arrays of Bell-Shaped and Sigmoid Functions. In *NIPS* (1990), R. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., Morgan Kaufmann, pp. 735–742.

[8] Baldi, P., and Brunak, S. *Bioinformatics: The Machine Learning Approach*, 2nd ed. MIT Press, Aug. 2001.

[9] Baldi, P., and Pollastri, G. The principled design of large-scale recursive neural network architectures–dag-rnns and the protein structure prediction problem. *J. Mach. Learn. Res. 4* (Dec. 2003), 575–602.

[10] Bartoli, L., Capriotti, E., Fariselli, P., Martelli, P. L., and Casadio, R. The Pros and Cons of Predicting Protein Contact Maps. In *Protein Structure Prediction*, M. J. Zaki and C. Bystroff, Eds., vol. 413 of *Methods in Molecular Biology*. Humana Press, 2008, pp. 199–217.

[11] Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D., and Zardecki, C. The Protein Data Bank. *Acta Crystallographica Section D 58*, 6 Part 1 (Jun 2002), 899–907.

[12] Berven, F. S., Flikka, K., Jensen, H. B., and Eidhammer, I. BOMP: a program to predict integral beta-barrel outer membrane proteins encoded within genomes of Gram-negative bacteria. *Nucleic Acids Res. 32*, Web Server issue (July 2004), W394–9.

[13] Bigelow, H. R., Petrey, D. S., Liu, J., Przybylski, D., and Rost, B. Predicting transmembrane beta-barrels in proteomes. *Nucleic Acids Res. 32*, 8 (2004), 2566–77.

[14] Bishop, C. M. Neural networks and their applications. *Review of Scientific Instruments 65*, 6 (1994), 1803–1832.

[15] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. 2006. corr. 2nd printing ed. Springer, Oct. 2007.

[16] Blasiak, S., and Rangwala, H. A hidden Markov model variant for sequence classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two* (2011), IJCAI'11, AAAI Press, pp. 1192–1197.

[17] Blundell, T., Sibanda, B., Sternberg, M., and Thornton, J. Knowledge-based prediction of protein structures and the design of novel molecules. *Nature 326*, 6111 (1987), 347–352.

[18] Bohr, J., Bohr, H., Brunak, S., Cotterill, R. M., Fredholm, H., Lautrup, B., and Petersen, S. B. Protein Structures from Distance Inequalities. *Journal of Molecular Biology 231*, 3 (1993), 861–869.

[19] Bowie, J., Luthy, R., and Eisenberg, D. A method to identify protein sequences that fold into a known three-dimensional structure. *Science 253*, 5016 (1991), 164–170.

[20] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput. 16*, 5 (Sept. 1995), 1190–1208.

[21] Bystroff, C., and Shao, Y. Fully automated ab initio protein structure prediction using I-SITES, HMMSTR and ROSETTA. *Bioinformatics 18*, suppl 1 (2002), S54–S61.

[22] Casadio, R., Fariselli, P., Finocchiaro, G., and Martelli, P. L. Fishing new proteins in the twilight zone of genomes: the test case of outer membrane proteins in Escherichia coli K12, Escherichia coli O157:H7, and other Gram-negative bacteria. *Protein Sci. 12*, 6 (June 2003), 1158–68.

[23] Clifford, P. Markov Random Fields in statistics. In *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*, G. Grimmett and D. Welsh, Eds. Oxford University Press, Oxford, 1990, pp. 19–32.

[24] Cooper, G. M., and Robert, E. H. *The cell: a molecular approach*, 5th ed. ASM Press, 2009.

[25] Cowan, S. W., Schirmer, T., Rummel, G., Steiert, M., Ghosh, R., Pauptit, R. A., Jansonius, J. N., and Rosenbusch, J. P. Crystal structures explain functional properties of two E. coli porins. *Nature*, 6389 (1992), 727–733.

[26] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS) 2*, 4 (1989), 303–314.

[27] Dang, T. H., Van Leemput, K., Verschoren, A., and Laukens, K. Prediction of kinase-specific phosphorylation sites using conditional random fields. *Bioinformatics 24*, 24 (2008), 2857–2864.

[28] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1998.

[29] Fariselli, P., and Casadio, R. A neural network based predictor of residue contacts in proteins. *Protein Engineering 12*, 1 (1999), 15–21.

[30] Fariselli, P., Martelli, P., and Casadio, R. A new decoding algorithm for hidden Markov models improves the prediction of the topology of all-beta membrane proteins. *BMC Bioinformatics 6(Suppl.4)*, S12 (2005).

[31] Fariselli, P., Savojardo, C., Martelli, P., Casadio, R., et al. Grammatical-restrained hidden conditional random fields for bioinformatics applications. *Algorithms for Molecular Biology 4*, 1 (2009), 13.

[32] Fersht, A. R. On the simulation of protein folding by short time scale molecular dynamics and distributed computing. *Proceedings of the National Academy of Sciences of the United States of America 99*, 22 (Oct. 2002), 14122–14125.

[33] Freeman, T. C., and Wimley, W. C. A highly accurate statistical approach for the prediction of transmembrane beta-barrels. *Bioinformatics 26*, 16 (Aug. 2010), 1965–74.

[34] Funahashi, K. On the approximate realization of continuous mappings by neural networks. *Neural Networks 2*, 3 (1989), 183–192.

[35] Girosi, F., and Poggio, T. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Comput. 1*, 4 (Dec. 1989), 465–469.

[36] Goebel, U., Sander, C., Schneider, R., and Valencia, A. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics 18*, 4 (1994), 309–317.

[37] Gromiha, M. M., Ahmad, S., and Suwa, M. Neural network-based prediction of transmembrane beta-strand segments in outer membrane proteins. *J Comput Chem 25(5)* (2004), 762–767.

[38] Gromiha, M. M., Ahmad, S., and Suwa, M. TMBETA-NET: discrimination and prediction of membrane spanning beta-strands in outer membrane proteins. *Nucleic Acids Research 33*, Web-Server-Issue (2005), 164–167.

[39] Gromiha, M. M., and Ponnuswamy, P. K. Prediction of transmembrane beta-strands from hydrophobic characteristics of proteins. *Int J Pept Protein Res 42* (1993), 420–31.

[40] Gromiha, M. M., and Suwa, M. Discrimination of outer membrane proteins using machine learning algorithms. *Proteins 63*, 4 (June 2006), 1031–7.

[41] Gunawardana, A., Mahajan, M., Acero, A., and Platt, J. C. Hidden conditional random fields for phone classification. *International Conference on Speech Communication and Technology.* (2005).

[42] Hayat, S., and Elofsson, A. BOCTOPUS: improved topology prediction of transmembrane barrel proteins. *Bioinformatics 28*, 4 (2012), 516–522.

[43] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks 2* (1989), 356–366.

[44] Hu, J., and Yan, C. A method for discovering transmembrane beta-barrel proteins in Gram-negative bacterial proteomes. *Comput Biol Chem 32*, 4 (Aug. 2008), 298–301.

[45] Hua S, S. Z. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of Molecular Biology 308(2)* (2001), 397–407.

[46] Huang, G.-B. Learning Capability and Storage Capacity of Two-Hidden-Layer Feedforward Networks. *IEEE Transaction on Neural Networks 14*, 2 (2003).

[47] Huang, G.-B., and Babri, H. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *Neural Networks, IEEE Transactions on 9*, 1 (jan 1998), 224–229.

[48] Huang, G.-B., and Chen, L. Convex incremental extreme learning machine. *Neurocomputing 70*, 16-18 (2007), 3056–3062.

[49] Huang, G.-B., and Chen, L. Enhanced random search based incremental extreme learning machine. *Neurocomput. 71*, 16-18 (Oct. 2008), 3460–3468.

[50] Huang, G.-B., Chen, L., and Siew, C.-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *Neural Networks, IEEE Transactions on 17*, 4 (july 2006), 879–892.

[51] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)* (2004).

[52] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing 70* (2006), 489–501.

[53] Huang, S., and Huang, Y. Bounds on the number of hidden neurons in multilayer perceptrons. *Neural Networks, IEEE Transactions on 2*, 1 (jan 1991), 47–55.

[54] Jaakkola, T., Diekhans, M., and Haussler, D. A Discriminative Framework for Detecting Remote Protein Homologies. *Journal of Computational Biology 7*, 1-2 (Feb. 2000), 95–114.

[55] Jacoboni, I., Martelli, P., Fariselli, P., De Pinto, V., and Casadio, R. Prediction of the transmembrane regions of beta-barrel membrane proteins with a neural network-based predictor. *Protein Sci 10*, 4 (2001), 779–87.

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

[57] Jones, D. T., Buchan, D. W. A., Cozzetto, D., and Pontil, M. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics 28*, 2 (Jan. 2012), 184–90.

[58] Jones, D. T., Taylort, W., and Thornton, J. A new approach to protein fold recognition. *Nature 358* (1992), 86–89.

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

[60] Kessel, A., and Ben-tal, N. *Introduction to Proteins. Structure, function and motion.* CRC Press, 2010.

[61] Kolmogorov, A. K. On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition. *Doklady Akademii Nauk SSSR 114* (1957), 369–373.

[62] Krogh, A. Hidden Markov Models for Labeled Sequences. In *In Proceedings of the 12th IAPR ICPR'94* (1994), IEEE Computer Society Press, pp. 140–144.

[63] Lafferty, J., McCallum, A., and Pereira, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML01* (2001), pp. 282–289.

[64] Lauritzen, S. L. *Graphical Models (Oxford Statistical Science Series)*. Oxford University Press, USA, July 1996.

[65] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks 6*, 6 (1993), 861–867.

[66] Lesk, A. M. *Introduction to Bioinformatics*, 3rd ed. Oxoford University Press, 2008.

[67] Leslie, C., Eskin, E., and Noble, W. S. S. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* (2002), 564–575.

[68] Li, M.-H., Lin, L., Wang, X.-L., and Liu, T. Protein protein interaction site prediction based on conditional random fields. *Bioinformatics 23*, 5 (2007), 597–604.

[69] Light, W. Ridge Functions, Sigmoidal Functions and Neural Networks. In *In Approximation Theory VII* (1993), Academic Press, pp. 163–206.

[70] Lipman, D. J., and Pearson, W. R. Rapid and sensitive protein similarity searches. *Science 227*, 4693 (Mar 1985), 1435–1441.

[71] Livingstone CD, B. G. Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. *Comput Appl Biosci. 9(6)* (1993), 745–756.

[72] Lomize, A. L., Pogozheva, I. D., and Mosberg, H. I. Anisotropic Solvent Model of the Lipid Bilayer. 2. Energetics of Insertion of Small Molecules, Peptides, and Proteins in Membranes. *Journal of Chemical Information and Modeling 51*, 4 (2011), 930–946.

[73] Lomize, M. A., Lomize, A. L., Pogozheva, I. D., and Mosberg, H. I. OPM: Orientations of Proteins in Membranes database. *Bioinformatics 22*, 5 (2006), 623–625.

[74] Lowe, D. Adaptive radial basis function nonlinearities, and the problem of generalisation. In *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)* (oct 1989), pp. 171–175.

[75] Lundstrom, K. Structural genomics for membrane proteins. *Cellular and Molecular Life Sciences 63* (2006), 2597–2607. 10.1007/s00018-006-6252-y.

[76] Martelli, P., Fariselli, P., Krogh, A., and Casadio, R. A sequence-profile-based HMM for predicting and discriminating beta barrel membrane proteins. *Bioinformatics 18 Suppl 1* (2002), 46–53.

[77] Marti-Renom, M. A., Stuart, A. C., Fiser, A., Sánchez, R., Melo, F., and Sali, A. Comparative protein structure modeling of genes and genomes. *Annual review of biophysics and biomolecular structure 29*, 1 (2000), 291–325.

[78] Mooney, C., Wang, Y.-H., and Pollastri, G. SCLpred: protein subcellular localization prediction by N-to-1 neural networks. *Bioinformatics 27*, 20 (2011), 2812–2819.

[79] Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., Zecchina, R., Onuchic, J. N., Hwa, T., and Weigt, M. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proc. Natl. Acad. Sci. U.S.A. 108*, 49 (Dec. 2011), E1293–301.

[80] Murphy, K. P. An Introduction to Graphical Models. Tech. rep., 2001.

[81] Murzin, A. G., Lesk, A. M., and Chothia, C. Principles determining the structure of beta-sheet barrels in proteins. I. A theoretical analysis. *J Mol Biol 236*, 5 (Mar. 1994), 1369–1381.

[82] Natt, N. K., Kaur, H., and Raghava, G. P. S. Prediction of transmembrane regions of beta-barrel proteins using ANN- and SVM-based methods. *Proteins 56* (2004), 11–18.

[83] Nocedal, J., and Wright, S. J. *Numerical Optimization*. Springer, Aug. 2000.

[84] Nugent, T., and Jones, D. T. Accurate de novo structure prediction of large transmembrane protein domains using fragment-assembly and correlated mutation analysis. *Proc. Natl. Acad. Sci. U.S.A. 109*, 24 (June 2012), E1540–7.

[85] Olmea, O., and Valencia, A. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Folding and Design 2, Supplement 1*, 0 (1997), S25–S32.

[86] Ostermeier, C., Harrenga, A., Ermler, U., and Michel, H. Structure at 2.7 A resolution of the Paracoccus denitrificans two-subunit cytochrome c oxidase complexed with an antibody FV fragment. *Proc Natl Acad Sci U S A 94(20)* (1997), 10547–10553.

[87] Ou, Y.-Y., Gromiha, M. M., Chen, S.-A., and Suwa, M. TMBETADISC-RBF: Discrimination of beta-barrel membrane proteins using RBF networks and PSSM profiles. *Comput Biol Chem 32*, 3 (June 2008), 227–31.

[88] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1 ed. Morgan Kaufmann, Sept. 1988.

[89] Pinto, D., McCallum, A., Wei, X., and Croft, W. B. Table extraction using conditional random fields. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), SIGIR '03, ACM, pp. 235–242.

[90] Pirovano, W., and Heringa, J. Protein Secondary Structure Prediction. In *Data Mining Techniques for the Life Sciences*, O. Carugo and F. Eisenhaber,

Eds., vol. 609 of *Methods in Molecular Biology*. Humana Press, 2010, pp. 327–348.

[91] Pollastri, G., and Baldi, P. Prediction of contact maps by GIOHMMs and recurrent neural networks using lateral propagation from all four cardinal corners. *Bioinformatics 18*, suppl 1 (2002), S62–S70.

[92] Pollastri, G., Vullo, A., Frasconi, P., and Baldi, P. Modular DAG-RNN architectures for assembling coarse protein structures. *J Comput Biol 13*, 3 (Apr. 2006), 631–650.

[93] Prince, S. M., Achtman, M., and Derrick, J. P. Crystal structure of the OpcA integral membrane adhesin from Neisseria meningitidis. *Proceedings of the National Academy of Sciences of the United States of America 99* (2002), 3417–3421.

[94] Quattoni, A., Collins, M., and Darrell, T. Conditional Random Fields for Object Recognition. In *Advances in Neural Information Processing Systems 17* (Cambridge, MA, 2005), L. K. Saul, Y. Weiss, and L. Bottou, Eds., MIT Press, pp. 1097–1104.

[95] Rabiner, L. A tutorial on HMM and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (Feb. 1989), 257–286.

[96] Randall, A. Z., Cheng, J., Sweredoski, M. J., and Baldi, P. TMBpro: secondary structure, beta-contact and tertiary structure prediction of transmembrane beta-barrel proteins. *Bioinformatics 24*, 4 (2008), 513–520.

[97] Rangwala, H., and Karypis, G. Profile based direct kernels for remote homology detection and fold recognition. *Bionformatics 21* (2005), 6–22.

[98] Remmert, M., Linke, D., Lupas, A. N., and Söding, J. HHomp–prediction and classification of outer membrane proteins. *Nucleic Acids Res. 37*, Web Server issue (July 2009), W446–51.

[99] Rost, B., and Sander, C. Prediction of Protein Secondary Structure at Better than 70% Accuracy. *Journal of Molecular Biology 232*, 2 (1993), 584–599.

[100] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Neurocomputing: foundations of research. MIT Press, Cambridge, MA, USA, 1988, ch. Learning internal representations by error propagation, pp. 673–695.

[101] Sartori, M., and Antsaklis, P. A simple method to derive bounds on the size and to train multilayer neural networks. *Neural Networks, IEEE Transactions on 2*, 4 (jul 1991), 467–471.

[102] Sato, K., and Sakakibara, Y. RNA secondary structural alignment with conditional random fields. *Bioinformatics 21*, 2 (2005), 237–242.

[103] Savojardo, C., Fariselli, P., and Casadio, R. Improving the detection of transmembrane -barrel chains with N-to-1 extreme learning machines. *Bioinformatics 27*, 22 (2011), 3123–3128.

[104] Savojardo, C., Fariselli, P., and Casadio, R. BETAWARE: a machine-learning tool to detect and predict transmembrane beta-barrel proteins in prokaryotes. *Bioinformatics 29*, 4 (2013), 504–505.

[105] Schulz, G. E. beta-Barrel membrane proteins. *Curr. Opin. Struct. Biol. 10*, 4 (Aug. 2000), 443–7.

[106] Sha, F., and Pereira, F. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1* (Stroudsburg, PA, USA, 2003), NAACL '03, Association for Computational Linguistics, pp. 134–141.

[107] Shendure, J., and Ji, H. Next-generation DNA sequencing. *Nat Biotechnol 26*, 10 (Oct. 2008), 1135–1145.

[108] Stinchcombe, M., and White, H. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *Neural*

Networks, 1989. IJCNN., International Joint Conference on (1989), vol. 1, pp. 613–617.

[109] Sutton, C., and McCallum, A. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[110] Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., and Wu, C. H. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics 23*, 10 (May 2007), 1282–8.

[111] TheUniProtConsortium. Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Res. 40*, Database issue (Jan. 2012), D71–5.

[112] Tuckerman, M. E., and Martyna, G. J. Understanding Modern Molecular Dynamics: Techniques and Applications. *The Journal of Physical Chemistry B 104*, 2 (2000), 159–178.

[113] Tusnady, G. E., Dosztanyi, Z., and Simon, I. PDBTM: selection and membrane localization of transmembrane proteins in the protein data bank. *Nucleic Acids Res 33* (2005), 275–278.

[114] Tusnady, G. E., Dosztanyi, Z., and Simon, I. TMDET: web server for detecting transmembrane regions of proteins by using their 3D coordinates. *Bioinformatics 21*, 7 (2005), 1276–1277.

[115] Valavanis, I. K., Bagos, P. G., and Emiris, I. Z. Beta-Barrel transmembrane proteins: Geometric modelling, detection of transmembrane region, and structural properties. *Comput. Biol. Chem. 30*, 6 (Dec. 2006), 416–424.

[116] Vassura, M., Margara, L., Di Lena, P., Medri, F., Fariselli, P., and Casadio, R. Reconstruction of 3D Structures From Protein Contact Maps. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on 5*, 3 (july-sept. 2008), 357–367.

[117] Vendruscolo, M., Kussell, E., and Domany, E. Recovery of protein structure from contact maps. *Folding and Design 2*, 5 (1997), 295–306.

[118] Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 969–976.

[119] von Heijne, G. Membrane Protein Topology Prediction. Hydrophobicity analysis and the positive-inside rule. *J Mol Biol 225(2)* (1992), 487–494.

[120] von Heijne, G. Recent advances in the understanding of membrane protein assembly and structure. *Q Rev Biophys 32*, 4 (Nov 1999), 285–307.

[121] von Heijne, G. Membrane-protein topology. *Nat Rev Mol Cell Biol 7*, 12 (Dec 2006), 909–918.

[122] Wainwright, M. J., and Jordan, M. I. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.

[123] Wang, S., Quattoni, A., Morency, L., and Demirdjian, D. Hidden Conditional Random Fields for Gesture Recognition. In *CVPR* (2006), pp. II: 1521–1527.

[124] Wimley, W. C. Toward genomic identification of beta-barrel membrane proteins: composition and architecture of known structures. *Protein Sci. 11*, 2 (Feb. 2002), 301–12.

[125] Wimley, W. C. The versatile beta-barrel membrane protein. *Curr. Opin. Struct. Biol. 13*, 4 (Aug. 2003), 404–11.

[126] Zemla, A., Venclovas, C., Fidelis, K., and Rost, B. A modified definition of SOV, a segment-based measure for protein secondary structure prediction assessment. *Proteins: Structure, Function, and Genetics 34* (1999), 220–223.

[127] Zhai, Y., and Saier, M. H. The beta-barrel finder (BBF) program, allowing identification of outer membrane beta-barrel proteins encoded within prokaryotic genomes. *Protein Science 11*, 9 (2002), 2196–2207.

Appendices

Chapter A

Method implementation and availability

In this appendix I describe the BETAWARE package [104], a software tool that provides an implementation of the two methods described in Chapter 6 and Chapter 7 for transmembrane β-barrel detection and topology prediction, respectively. BETAWARE is available under GPL license as a standalone program as well as web application.

A.1 BETAWARE standalone

The standalone version of BETAWARE is entirely written using the Python programming language to allow high portability. The program takes as input a target protein sequence in FASTA format and the corresponding pre-computed alignement profile. In the basic setting, BETAWARE works in two steps. Firstly, it predicts a score for a protein to be a transmembrane β-barrel (TMBB) using N-to-1 Extreme Learning Machine as described in Chapter 6. Therefore, in case the protein is predicted as TMBB, it assigns the putative topology using the Grammatical-Restrained Hidden CRF model (see Chapter 7).

The output of the program (an example is shown in Figure A.1) reports the sequence name present in the FASTA file, the sequence length, the TMBB prediction (yes or no) and the possible topology prediction. When the topology is assigned, the output also displays the posterior probability of the label for each sequence position (from "a" = highest probability to "j" = lowest probability, see output example).

```
Sequence id     : 1QJ8:A|PDBID|CHAIN|SEQUENCE
Sequence length : 148
Predicted TMBB  : Yes
Topology        : 2-9,23-29,35-46,60-69,78-87,103-114,120-131,135-145
Seq : ATSTVTGGYAQSDAQGQMNKMGGFNLKYRYEEDNSPLGVIGSFTYTEKSRTASSGDYNKN
SS  : iTTTTTTTTooooooooooooTTTTTTiiiiiTTTTTTTTTTTTooooooooooooT
Prob: cbaaaaaaaaaaaaaaaaaccaaaaaaacaaabccaaaaaaabbdaaaaaaaaaacdeb
------------------------------------------------------------------
Seq : QYYGITAGPAYRINDWASIYGVVGVGYGKFQTTEYPTYKNDTSDYGFSYGAGLQFNPMEN
SS  : TTTTTTTTiiiiiiiiTTTTTTTTTooooooooooooooTTTTTTTTTTTiiiiiT
Prob: aaaaaaaaabaaaaaabaaaaaaaaaaaaaaaaaaaaaaaadaaaaaaaaabdaaaaae
------------------------------------------------------------------
Seq : VALDFSYEQSRIRSVDVGTWIAGVGYRF
SS  : TTTTTTTTTToooTTTTTTTTTTiii
Prob: baaaaaaaaaaaaaaaaaaaaaaaaba
---------------------------------
//
```

Figure A.1: BETAWARE example output.

These probabilities can be interpreted as a reliability measure of the prediction at each sequence position.

The standard behaviour of BETAWARE can be changed by means of several options, which include:

- forcing the prediction of the topology even when the detection module does not recognize the sequence as a TMBB protein (-t option);
- tuning the sensitivity threshold of the detection algorithm (-s option). The higher is the threshold the smaller would be the chance to obtain false negatives. However, a high threshold also increases the probability of getting false positives;
- the alignment profile is provided as a frequency matrix of dimension $L \times 20$, where L is the protein sequence length while each column corresponds to one

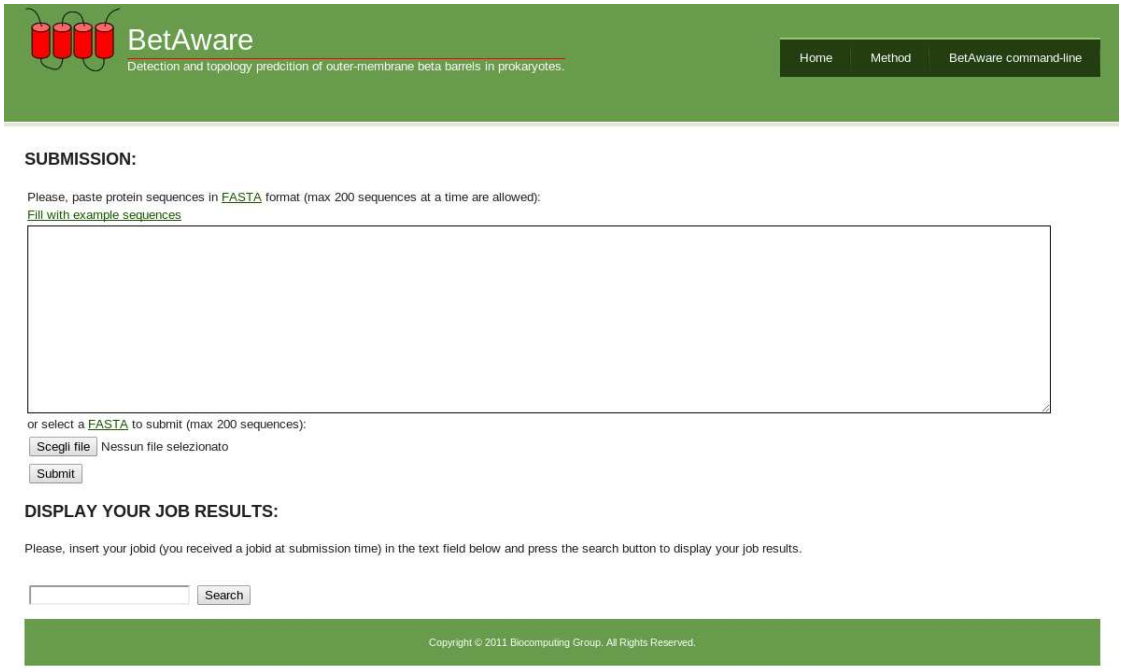


Figure A.2: The BETAWARE web user interface.

of 20 possible amino acids according to a specific amino acid order. A specific option (-a option) can be used to specify the correspondence between amino acids and columns in the alignment profile file. If a different order is specified the program automatically rearrange the matrix columns accordingly;

- directing the output on a file (-o option).

Source code is availbale at the BETAWARE standalone home page `http://www.biocomp.unibo.it/savojard/betawarecl`.

A.2 BETAWARE web server

As mentioned above, the BETAWARE software is also available as web application. A screenshot of the BETAWARE user interface is shown in Figure A.2.

The BETAWARE web server requires protein sequences in FASTA format (up to 200 sequences are allowed for a single job submission). Differently from the stan-

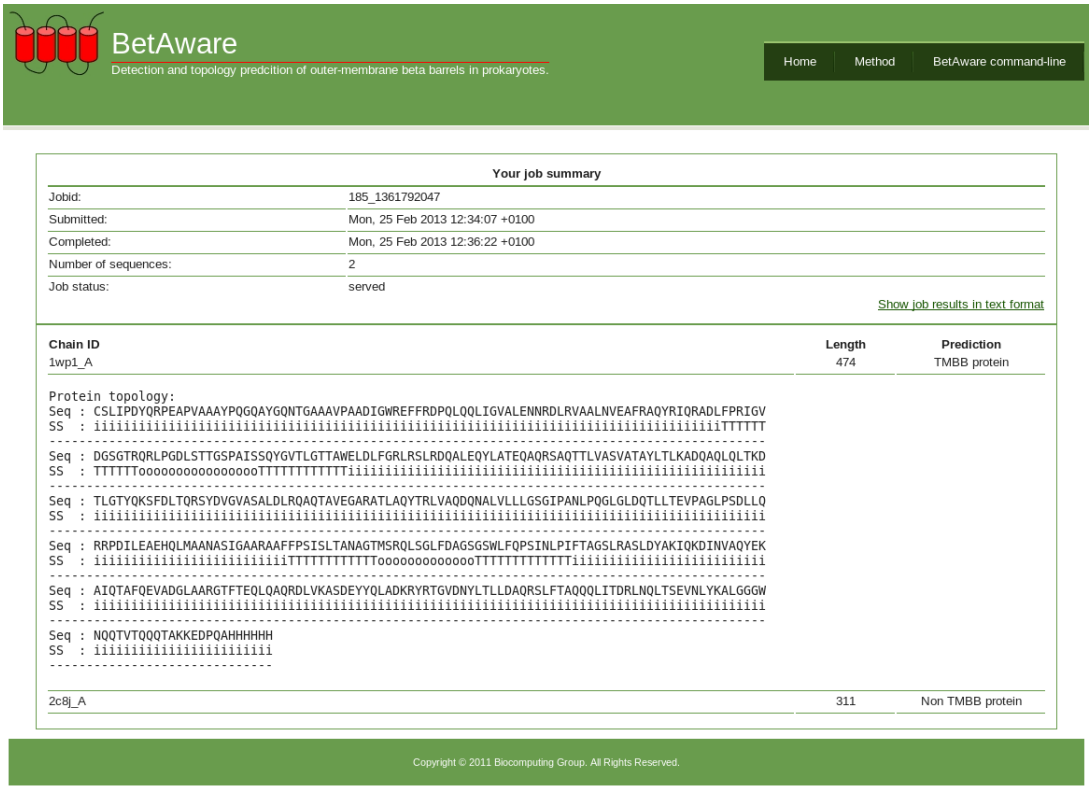


Figure A.3: An example of job result in BETAWARE.

dalone version, corresponding alignement profiles are not needed and they are automatically computed internally by aligning submitted sequences against the Uniref90 sequence database [110]. Upon job submission, the system assign an unique job id which marks each prediction request as a future reference to retrieve prediction results. An example of a job result is shown in Figure A.3. Besides general information about the job (id, number of sequences, submission and completion times), the output includes, for each submitted sequence, the information relative to the predicted protein class (TMBB or non-TMBB) and predicted protein topology. The job result can be also downloaded in text format in order to facilitate subsequent parsing by other tools. User requests and job results are stored in a internal database and conserved for a limited period of time (at least one month).

The BETAWARE web application has been developed using web2py (`http://www.web2py.com/examples/default/index`), a full-stack framework to de-

velop database-drive web-based applications, written and programmable in Python. The BETAWARE web server is available for use at
`http://betaware.biocomp.unibo.it/BetAware`.