Statistical Relational Learning for Proteomics: Function, Interactions, Evolution

Stefano Teso

2013/12/19

International Doctorate School in Information and
Communication Technologies
DIT - University of Trento

Advisor:
Prof. Andrea Passerini
Università degli Studi di Trento

Abstract

In recent years, the field of Statistical Relational Learning (SRL) [1, 2] has produced new, powerful learning methods that are explicitly designed to solve complex problems, such as collective classification, multi-task learning and structured output prediction, which natively handle relational data, noise, and partial information. Statistical-relational methods rely on some First-Order Logic as a general, expressive formal language to encode both the data instances and the relations or constraints between them. The latter encode background knowledge on the problem domain, and are use to restrict or bias the model search space according to the instructions of domain experts. The new tools developed within SRL allow to revisit old computational biology problems in a less *ad hoc* fashion, and to tackle novel, more complex ones. Motivated by these developments, in this thesis we describe and discuss the application of SRL to three important biological problems, highlighting the advantages, discussing the trade-offs, and pointing out the open problems.

In particular, in Chapter 3 we show how to *jointly* improve the outputs of multiple correlated predictors of protein features by means of a very general probabilistic-logical consistency layer. The logical layer — based on grounding-specific Markov Logic networks [3] — enforces a set of weighted first-order rules encoding biologically motivated constraints between the predictions. The refiner then improves the raw predictions so that they least violate the constraints. Contrary to canonical methods for the prediction of protein features, which typically take predicted correlated features as inputs to improve the output *post facto*, our method can jointly refine *all* predictions together, with potential gains in overall consistency. In order to showcase our method, we integrate three stand-alone predictors of correlated features, namely subcellular localization (Loctree[4]), disulfide bonding state (Disulfind[5]), and metal bonding state (MetalDetector[6]), in a way that takes into account the respective strengths and weaknesses. The experimental results show that the refiner can improve the performance of the underlying predictors by removing rule violations. In addition, the proposed method is fully general, and could in principle be applied to an array of

heterogeneous predictions without requiring any change to the underlying software.

In Chapter 4 we consider the multi-level protein–protein interaction (PPI) prediction problem. In general, PPIs can be seen as a hierarchical process occurring at three related levels: *proteins* bind by means of specific *domains*, which in turn form interfaces through patches of *residues*. Detailed knowledge about which domains and residues are involved in a given interaction has extensive applications to biology, including better understanding of the binding process and more efficient drug/enzyme design. We cast the prediction problem in terms of multi-task learning, with one task per level (proteins, domains and residues), and propose a machine learning method that collectively infers the binding state of all object pairs, at all levels, concurrently. Our method is based on Semantic Based Regularization (SBR) [7], a flexible and theoretically sound SRL framework that employs First-Order Logic constraints to tie the learning tasks together. Contrarily to most current PPI prediction methods, which neither identify which regions of a protein actually instantiate an interaction nor leverage the hierarchy of predictions, our method resolves the prediction problem up to residue level, enforcing consistent predictions between the hierarchy levels, and fruitfully exploits the hierarchical nature of the problem. We present numerical results showing that our method substantially outperforms the baseline in several experimental settings, indicating that our multi-level formulation can indeed lead to better predictions.

Finally, in Chapter 5 we consider the problem of predicting drug-resistant protein mutations through a combination of Inductive Logic Programming [8, 9] and Statistical Relational Learning. In particular, we focus on *viral* proteins: viruses are typically characterized by high mutation rates, which allow them to quickly develop drug-resistant mutations. Mining relevant rules from mutation data can be extremely useful to understand the virus adaptation mechanism and to design drugs that effectively counter potentially resistant mutants. We propose a simple approach for mutant prediction where the input consists of mutation data with drug-resistance information, either as sets of *mutations* conferring resistance to a certain drug, or as sets of *mutants* with information on their susceptibility to the drug. The algorithm learns a set of relational rules characterizing drug-resistance, and uses them to generate a set of potentially resistant mutants. Learning a weighted combination of rules allows to attach generated mutants with a resistance score as predicted by the statistical relational model and select only the highest scoring ones. Promising results were obtained in generating resistant mutations for both nucleoside and non-nucleoside HIV reverse transcriptase inhibitors. The approach can be generalized quite easily to learning mutants characterized by

more complex rules correlating multiple mutations.

Contents

List of Tables

List of Figures

Acknowledgments

First of all, I would like to express my gratitude towards my supervisor, Andrea Passerini, for passionately teaching me the many unwritten rules of scientific research, which far too often go unnoticed to obtuse slackers like me. As I came to understand with time, research is a difficult craft, and problem solving can be a frustrating and utterly unrewarding activity. But thanks to Andrea's continuous support, I somehow managed to get over the difficult times, and had the opportunity to enjoy the state of grace that only real numbers in the range $[0, 1]$, written in obscure monochromatic LaTeX tables, can give. Other than that, I would like to thank Andrea deeply and sincerely for being so friendly and generous, despite my embarrassing faults.

I'd also like to thank two paradigmatic figures that I had the chance to meet here in Trento, Jacopo Staiano and Gabriele Catania. They managed to single handedly [*sic.*] transform the incongruous, desolate, fat misplaced lump of contemporary anti-seismic insanity that is our faculty, into an enjoyable workplace. You are like metal water tanks in the middle of the desert: your faucets may be rusty, but inside you are full of wet. What does it mean? I do not know. But I think I got it right.

Then I'd like to thank Duy Tin Truong and Umut Avci, my office mates, for being delightful persons to work with, and for the time spent side-to-side crunching datasets, running experiments, and often sharing the same excited/derailed state of mind that positive/negative results cast upon us.

I'd like to extend my gratitude to Elisa Cilia, at the Université Libre de Bruxelles; to the colleagues at the University of Siena: Claudio Saccà, Salvatore Frandina, and Michelangelo Diligenti; and to Bruno Lepri at the Fondazione Bruno Kessler, for the insightful discussions and support.

I'd also like to thank the Rost Lab at the Technische Universität München, and Gianluca Pollastri at the UCD School of Computer Science and Informatics, for making their software available.

Then there are all my friends, here in Trento and back at my hometown, too many to list here, who directly and indirectly supported my endeavors in scientific research, and provided many occasions for long sessions of

indiscriminate feasting and guzzling. Thank you.

Finally, my most sincere thanks go to my girlfriend Silvia, and to my parents Anna and Antonio, whose patience, reliability and constant support never cease to amaze and flatter me, and who have put so much effort into teaching me how to be a better person — even though I failed, far too often, to pay attention. It is to them that I wish to dedicate this thesis.

You all have taught me, once more, that it is through others that life can be bliss.

Chapter 1

Introduction

1.1 Motivation

The marriage between molecular biology and computer science is a long and fruitful one. Biologists have embraced computer science since the very beginning of the computer era in order to store the results of biological experiments, analyze the corresponding databases, find regularities, test hypotheses, and perform predictions. Due to the particularly complex and fuzzy nature of biological annotations, machine learning in particular has played an important role in biological research, providing many indispensable tools for forecasting properties of genes, proteins, RNA fragments, and other entities of biological interest. Thanks to their wide applicability and general success, predictive models have proved to be a valid support to wet experiments, leading to the development of hundred of computational prediction methods for a variety of protein, DNA and RNA properties, and a florid research field [11, 12].

Thanks to recent advancements in experimental methods, biological information has kept increasing in size and complexity. The rapid accumulation of huge amounts of data on many aspects of cell life has ushered many new, important questions. In general, the subject of molecular biology has broadened from the study of individual genes and proteins to a more integrated view: in the post-genomic era, open problems involve networks of interacting entities and their evolution, and can only be successfully answered by taking into consideration the correlations linking the various types of biological objects.

Unfortunately, the classical tools provided by classical statistical or logical machine learning methods, such as Hidden Markov Models [13] and Support Vector Machines [14], are not ideally suited to deal with these problems,

which are, at once, intimately *relational* and *statistical*. Purely logical models, such as those developed by the sub-field of Inductive Logic Programming [8, 15], do not provide a sound mechanism to handle noisy or incomplete data, and therefore have issues with biological data. On the other hand, Statistical Learning methods [16, 17, 18, 19], while designed to very naturally deal with noise and probabilities, can not be trivially adapted to handle relational data (in a sound manner): most of these methods are based on the assumption that examples are independently and identically distributed, which is violated by relational data. In other words, these two classes of learning methods offer complementary advantages, and are afflicted by complementary faults. As a result, neither of them is entirely appropriate for the new requirements imposed by complex biological prediction tasks.

Prompted by these difficulties, and similar ones arising in other research fields (e.g. network sociology and computational neuroscience), machine learners have developed novel methods, grouped under the umbrella term of Statistical Relational Learning (SRL) [1, 2], which generalize the classical models and merge the respective advantages. SRL methods are explicitly designed to perform predictions of complex objects, such as trees, graphs, multi-graphs and relational databases, and can natively take advantage of the correlations existing between the individual object parts.

Methods in SRL are typically based on some general, expressive, *formal language*, such as First-Order Logic (or subsets thereof), to encode both the data and a set of *relations* or *constraints*. The constraints offer the possibility of including additional human-readable *background knowledge* within the learning task, in order to restrict the model search space according to the instructions of domain experts, with potential gains in predictive accuracy and learnability (especially in cases where data is scarce). Contrary to purely logical methods, where the constraints are deterministic, constraints in SRL can also be *soft*, in order to account for errors in the data or in the constraints. Inference and parameter learning are implemented as logical-probabilistic reasoning over the formal language, and enable sound handling of noise and missing data. Notably, the ability to include prior knowledge in the learning problem goes hand in hand with the formalization effort that is currently pervading biology and medicine, embodied by the development of *formal ontologies*, which are being applied to a growing number of biological databases [20, 21].

Statistical-relational learning offers new tools that allow to revisit old problems in a less *ad hoc* fashion, and to tackle novel, more complex biological problems. Motivated by these developments, in this thesis we describe and discuss three applications of SRL methods to three important biological problems, highlighting the advantages and discussing the tradeoffs and open

problems. In the next few sections we present a high-level overview of the problems tackled in the next chapters.

1.1.1 Protein Feature Predictor Refinement with Markov Logic

Computational methods for the prediction of protein features from sequence are a long-standing focus of bioinformatics. A key observation is that several protein features are closely inter-related, that is, they are conditioned on each other. Researchers invested a lot of effort into designing predictors that exploit this fact. Most existing methods leverage inter-feature constraints by including known (or predicted) correlated features as *inputs* to the predictor, thus conditioning the result.

By including correlated features as inputs, existing methods only rely on one side of the relation: the output feature is conditioned on the input features. In Chapter 3 we show how to *jointly* improve the outputs of multiple correlated predictors by means of a probabilistic-logical consistency layer. The logical layer enforces a set of weighted first-order rules encoding biological constraints between the features, and improves the raw predictions so that they least violate the constraints. In particular, we show how to integrate three stand-alone predictors of correlated features: subcellular localization (Loctree[4]), disulfide bonding state (Disulfind[5]), and metal bonding state (MetalDetector[6]), in a way that takes into account the respective strengths and weaknesses, and does not require any change to the predictors themselves. We compare our methodology against two alternative refinement pipelines based on state-of-the-art sequential prediction methods.

Our refinement framework is able to improve the performance of the underlying predictors by removing rule violations. We show that different predictors offer complementary advantages, and our method is able to integrate them using non-trivial constraints, generating more consistent predictions. In addition, our framework is fully general, and could in principle be applied to a vast array of heterogeneous predictions without requiring any change to the underlying software. On the other hand, the alternative strategies are more specific and tend to favor one task at the expense of the others, as shown by our experimental evaluation. The ultimate goal of our framework is to seamlessly integrate full prediction suites, such as Distill[22] and PredictProtein[23].

1.1.2 Multi-level Protein–Protein Interaction Prediction

Protein–protein interactions can be seen as a hierarchical process occurring at three related levels: *proteins* bind by means of specific *domains*, which in turn form interfaces through patches of *residues*. Detailed knowledge about which domains and residues are involved in a given interaction has extensive applications to biology, including better understanding of the binding process and more efficient drug/enzyme design. Alas, most current interaction prediction methods do not identify which parts of a protein actually instantiate an interaction. Furthermore, they also fail to leverage the hierarchical nature of the problem, ignoring otherwise useful information available at the lower levels; when they do, they do not generate predictions that are guaranteed to be consistent between levels.

In Chapter 4 we formalize the problem as a multi-level learning task, with one task per level (proteins, domains and residues), and propose a machine learning method that collectively infers the binding state of all object pairs. Our method is based on Semantic Based Regularization (SBR), a flexible and theoretically sound machine learning framework that uses First Order Logic constraints to tie the learning tasks together. We introduce a set of biologically motivated rules that enforce consistent predictions between the hierarchy levels.

We study the empirical performance of our method using a standard validation procedure, and compare its performance against the only other existing multi-level prediction technique. We present results showing that our method substantially outperforms the competitor in several experimental settings, indicating that exploiting the hierarchical nature of the problem can lead to better predictions. In addition, our method is also guaranteed to produce interactions that are consistent with respect to the protein–domain–residue hierarchy.

1.1.3 Forecasting Viral Mutants

Viruses are typically characterized by high mutation rates, which allow them to quickly develop drug-resistant mutations. Mining relevant rules from mutation data can be extremely useful to understand the virus adaptation mechanism and to design drugs that effectively counter potentially resistant mutants.

In Chapter 5, we propose a simple statistical relational learning approach for mutant prediction where the input consists of mutation data with drug-resistance information, either as sets of mutations conferring resistance to a certain drug, or as sets of mutants with information on their susceptibility to

the drug. The algorithm learns a set of relational rules characterizing drug-resistance and uses them to generate a set of potentially resistant mutants. Learning a weighted combination of rules allows to attach generated mutants with a resistance score as predicted by the statistical relational model and select only the highest scoring ones.

Promising results were obtained in generating resistant mutations for both nucleoside and non-nucleoside HIV reverse transcriptase inhibitors. The approach can be generalized quite easily to learning mutants characterized by more complex rules correlating multiple mutations.

1.1.4 Publications

The three chapters are taken from the following papers:

- Stefano Teso, Andrea Passerini – "*Joint Probabilistic-Logical Refinement of Multiple Protein Feature Predictors*", BMC Bioinformatics (in press).
- Claudio Saccà, Stefano Teso, Michelangelo Diligenti, Andrea Passerini – "*Improved Multi-level Protein–Protein Interaction Prediction with Semantic-based Regularization*", BMC Bioinformatics (submitted).
- Elisa Cilia, Stefano Teso, Sergio Ammendola, Tom Lenaerts, Andrea Passerini – "*Predicting virus mutations through relational learning*", In ECCB Workshop on Annotation, Interpretation and Management of Mutations (AIMM-2012), 2012.

Chapter 2

Background

2.1 Molecular Biology of Proteins

This thesis is concerned mostly with problems related to *proteins*. Proteins are polypeptides, biological molecules of variable size, often composed of multiple parts or *chains*, which carry out many different functions within the cell, including (but not limited to) metabolism, DNA transcription, translation and synthesis, regulation of gene expression, signal transduction (in response to external stimuli or internal events), and antigenic response. Most of the activities that take place within the cell are enacted by proteins [24, 25], including those involved in the mechanisms of inherited and infectious diseases.

Complete details on the composition, properties and concerted behavior of proteins within the organism (the *proteome*), are a key factor in elucidating the genotype-phenotype relationship, with all the scientific and medical consequences it entails. This observation motivates much of the research done in bioinformatics to *predict* such information, which are generally difficult or expensive to annotate experimentally, from the available data.

In this section we give a short exposition of the molecular biology of proteins, tailored towards the scope of this thesis. We describe the three basic properties of proteins (sequence, structure, and function) and the relation between them; we give details about the means by which proteins interact, and why; and sketch how proteins evolve by natural selection. Of course, our exposition only scratches the surface of protein biology. For a broader and deeper treatment of the subject, we refer the reader to the many excellent textbooks available [24, 25, 26].

2.1.1 Sequence

A protein can be described in terms of its chemical composition. Each protein is uniquely identified, modulo neutral mutations, by a specific *sequence* of twenty canonical amino acids, small organic molecules that have a common chemical component (an amine and a carboxylic acid functional group) and a variable component, the *side chain*. The common part forms the backbone of the protein, while the side chain determines the identity, chemical properties, and size of each amino acid within the protein sequence. From a computational point of view, a protein sequence can be thus described as a string of arbitrary length on an alphabet of twenty symbols — with one character for each amino acid.

Proteins are genetic products. Their sequence is encoded within the genetic material of the cell, the DNA, as a sequence of nucleotides. A region of DNA that encodes for a protein is called *gene*, and is often delimited by special genetic markers in order to be easily identified and processed by the

molecular mechanism that operate on them. Given the genetic sequence, it is possible to derive the corresponding protein sequence by mapping triplets of nucleotides into amino acids — the mapping between the two alphabets is the celebrated *genetic code*. Since whole-genome sequencing is both affordable and very efficient, sequence information currently represents the most abundant form of information on proteins.

The correspondence between genes and proteins, however, is not bijective, due to two mechanisms: *alternative splicing* and *homology*. Alternative splicing is the process by which, during transcription, different sets of exons (coding regions) of a gene are shuffled and adjoined, allowing the gene to encode for multiple proteins. On the other hand, a genome may hold multiple copies of the same gene, i.e. *paralogues*, which encode for the same set of proteins. Despite these issues, there exist rather advanced techniques that — by aligning a newly sequenced genome to already annotated ones — allow to determine the majority of the genes (and therefore proteins) in an organism [27].

A basic assumption in biology is that the sequence, or *primary structure*, of a protein completely determines its three-dimensional (*tertiary*) structure [28], which in turn is responsible for the biological function expressed by the protein itself [29]. While the relationship is thought to be (roughly) deterministic, the exact mechanisms by which the sequence — known for the majority of proteins — maps to the structure and function of the protein itself, are still largely unknown. This state of affairs motivates much of the research done on protein function prediction in computational biology [11].

2.1.2 Structure and Structural Properties

The mismatch between our knowledge of sequence and structure can be at least partially retraced to the technological gap between current experimental methods for sequencing and structure determination. Methods in the latter category, such as nuclear magnetic resonance spectroscopy [30], X-ray crystallography [31], and electron microscopy [32] are not high-throughput, requiring time consuming sample preparation, and are not always applicable (e.g. X-ray crystallography requires the protein to be first crystallize, which is not always feasible). As a result, the number of resolved structures accounts for only a fraction of the sequenced proteins [33].

Another fundamental problem is our current lack of understanding of *folding* [34], that is, the mechanism by which a newly assembled protein molecule assumes its definitive, working tertiary structure [35]. In accordance to the seminal work by Anfinsen [28], folding is assumed to be a thermodynamical energy minimization procedure driven by the amino acid

composition of the protein sequence [36]. Since the major driving forces are physico-chemical in nature (namely the hydrophobic effect, the formation of hydrogen bonds and electrostatic interactions, and the conformational entropy due to restricted degrees-of-freedom of the chain [37]), folding takes place autonomously under suitable conditions. The formation of secondary structure elements [38] and post-translational modifications [39] (such as the addition of disulphide bonds [40] and acetyl functional groups [41]) also play a role in guiding and stabilizing the folding process. Correctly folded structures are thought to fluctuate around the *global* minimum of the free-energy function. In order for proteins to successfully reach said global minimum, the energy landscape is thought [36] to be shaped like a rough funnel, although the exact mechanisms are not yet clear. Folding plays a crucial role in biology: since structure determines function, mis-folded protein may express impaired, null or toxical activity [42]. As a matter of fact, there are many inherited diseases which have been attributed to mutation-induced misfolding, e.g. Alzheimer's and Parkinson's diseases, cystic fibrosis and some neurodegenerative diseases [42, 34].

Several factors contribute to the complexity of the folding problem. An important component are the non-local contacts, i.e. long-range correlations between residues, which constrain and stabilize the folding process [43]; these correlations also render computational simulations of folding very computationally demanding. It is also difficult to experimentally determine the folding intermediates sampled during the process and the alternative final conformations [44]. Furthermore, some proteins are only stabilized by external factors such as post-translational modifications [39], which are hard to capture computationally. Despite the advancements in computer simulations and machine learning methods for structure predictions, these still have limited applicability [45], hindering our ability to automatically determine the structure of newly sequenced proteins.

The tertiary structure of proteins lodges many types of well-defined local structures which have critical biological roles, and can, notably, be inferred independently. These characteristic sub-structures include *secondary structure* elements, *catalytic sites*, *binding sites*, and *disordered regions*. Knowledge about their displacement can be leveraged to reconstruct the global structure of the protein, and define its function; this observation stands at the heart of many hierarchical structure and function prediction methods.

A perfectly folded protein structure can be dissected into roughly two parts: an inner *core*, and an outer solvent-accessible layer. Residues in the core tend to be hydrophobic, and thus form a tightly packed, *buried* cluster, not directly in contact with the solvent medium that surrounds the protein — by definition, their solvent accessible area must be less than 1.4 Å. Due to

its strong structural and functional role, the core is the most evolutionarily conserved region of the protein: mutations that alter the volume of the core often have a negative impact on protein stability [42].

The *secondary structure* (SS) of a protein consists of very recognizable fragments of well defined, stable, and conserved three-dimensional residue arrangements, the SS elements. The two most frequent configurations are α-helices and β-sheets [25], which are ubiquitous in protein space. The full list of secondary structure types is listed in the Dictionary of Protein Secondary Structures (DSSP) [38], which classifies proteins according to their secondary structure composition. Contrary to tertiary structure, which is a global property of the protein sequence, secondary structures are the result of *local* contributions of hydrogen bonds between nearby residues [46], and as such are thought to act as building blocks for intermediates during folding. Secondary structure elements are tailored towards specific functions. For instance, α-helices are commonly found in transmembrane proteins [47], which attach to the lipid bilayer of the cell wall, and are pivotal for enabling transmembrane proteins to participate to signaling and cell-cell recognition.

Catalytic sites play a central role in *enzymes* [48], i.e. proteins that catalyze chemical reactions. Thanks to their physico-chemical properties, catalytic sites can recognize and capture the substrates of a reaction, and carefully position them in order to facilitate (i.e. lower the activation energy of) the reaction. Only a fraction of the residues accessible within the active sites participate in the reaction; the remaining residues may be inert, or only help in capturing the substrate. Upon binding, substrates may induce a conformational change, which may expose other active sites in the protein, or detach other substrates [49]. In general, catalytic residues owe their efficacy to both their chemical composition and spatial arrangement, and are typically highly conserved. Identification of catalytic sites is fundamental for identifying enzymes and characterizing their function. A complete hand curated list of annotated catalytic sites can be found in the Catalytic Site Atlas (CSA) database [48].

Binding sites [50] play a similar role for protein–protein interactions, a fundamental process by which proteins interact with each other and carry out their biological function. Matching binding sites in interacting proteins owe their effectiveness to a combination of shape complementarity, residue affinity, and structural flexibility [50]. The residue propensity of interfaces has been shown to be substantially different than that of protein regions not involved with binding [51]; the same can be said about interfaces mediating different types of interactions (e.g. transient versus obligate, homodimer versus heterodimer). Similarly to catalytic sites, binding sites have a lot of potential as drug targets [52]: drugs may act on a protein by occluding the

contact surface or clogging the hot spots, i.e. those residues that contribute the majority of the binding energy [53].

Finally, we end this section with a note on protein flexibility. Protein structures have long been known to be *dynamic*, as molecular shape variations occur naturally as a consequence of thermal fluctuations [54]. More importantly, many protein — e.g. enzymes — *rely* on their structure to switch between alternate conformations, which have different functional roles. For instance, depending on its current state, a protein may expose different functional regions. However, it has been recently observed that a large fraction of eukaryotic proteins (over 30% [55]) present structural segments that are *inherently disordered*, i.e. segments with no definite secondary or tertiary structure. Even though their exact role is still unclear [56], inherently disordered proteins have been observed to play a role in intracellular signaling and regulatory processes [55], and are known to have strong molecular recognition capabilities [57], enabled by their ability to undergo structural stabilization upon binding [58]. Disordered proteins have been associated with several pathological conditions, such as cancer and cardiovascular diseases [59, 60].

2.1.3 Function

Protein function is a compound term to refer to the overall behavior of a protein and its role within the proteome. Being so diverse, protein function has been defined in different, often conflicting terms over time. Recently, however, there has been growing consensus on formalizing protein function according to the Gene Onotology (GO) [20], a wide vocabulary of shared, controlled terms. GO terms are organized in a hierarchy according to a general-to-specific relation: terms in the GO form a *tree*, where parent terms are strictly more general than their children [20].

The GO includes keywords for three orthogonal aspects of protein function: i) what *molecular function* is displayed by the protein (e.g. enzymatic reaction), ii) what *biological processes* it participates in (e.g. cell cycle, gene regulation, apoptosis), and iii) what *cellular components* it resides within (e.g. nucleus, cytosol, mitochondrion, *etc.*).

Being so central, information about function is crucial for many biomedical and pharmaceutical tasks. Unsurprisingly, functional annotation is not growing as steadily as sequence annotation [61], prompting researchers to develop accurate prediction methods to fill the gap. For a survey on the subject, please see [62, 63, 61].

2.1.4 Interactions

Proteins do not act in isolation. In order to carry out their function, most proteins interact with (bind to) other proteins. A group of proteins and their interactions is called a protein–protein interaction network (PIN). PINs capture a static, protein-level snapshot of the overall topology of protein interactions. The totality of interactions expressed by a proteome is called the *interactome*. Physical interactions are the workhorse of cell life and development [64], and play an extremely important role both in the mechanisms of disease [65] and in the design of new drugs [66].

Physical interactions between proteins can be seen at different levels of detail. At the highest level, proteins interact to perform some joint function [67], which is effectively mediated by the interaction itself [68]. The propagation of function at the network-level forms the basis of biological *pathways*, evolutionarily conserved sub-modules of the PIN that are specialized for a particular function. Pathways can be distinguished in three varieties (metabolic pathways, regulation pathways, and signal transduction pathways), as accurately annotated in the KEGG encyclopedia [69]. In particular, signal transduction pathways consist of groups of interacting proteins that collectively transport information about events in the cell, e.g. apoptosis (programmed cell death).

At a lower level, the same interaction occurs between a pair of specific *domains* appearing in the proteins. Domains are conserved sub-regions of protein sequence/structure that perform a specific function, e.g. partner recognition, catalysis, *etc.* The types of the domains involved in an interaction characterize the functional semantics of the latter [70].

At the lowest level, the interaction is instantiated by the binding of a pair of protein *interfaces*, i.e. patches of solvent accessible residues with compatible shapes and chemical properties [71]. The low-level features of binding sites determine whether the interaction is transient or permanent, whether two proteins compete for interaction with a third one, *etc.*

The topology of the PIN and individual features of interactions are an essential component of a wide range of biological tasks: inferring protein function [61] and localization [72], reconstructing signal and metabolic pathways [73], discovering candidate targets for drug development [65]. Finer granularity predictions at the domain level allow to discover affinities between domain types that can be carried over to other proteins [74, 75]; domain–domain networks have also been assessed as being typically more reliable than their protein counterparts [76]. Finally, residue-level predictions, i.e., interface recognition, enable the detailed study of the principles of protein interactions, and are crucial for tasks such as rational drug de-

sign [66], metabolic reconstruction and engineering [77], and identification of hot-spots [78] in the absence of structure information.

Notwithstanding the increased availability of interaction data, the natural question of whether two arbitrary proteins interact, and why, is still open. The growing literature on protein interaction prediction [79, 80, 81] is symptomatic of the gap separating the amount of available data and the effective size of the interaction network [82]. Furthermore, protein–protein interaction data is under-characterized at the domain and residue levels: the current databases are relatively lacking when compared to the magnitude of the existing body of data about protein-level interactions [76]. At the time of writing, the PDB hosts 84,418 structures, but merely 4,210 resolved complexes[1]. The latter cover only a tiny fraction of the interactions stored in databases such as BioGRID [83] and MIPS [84].

2.1.5 Evolution

During the normal life cycle of the cell, DNA may be affected by *mutations*, i.e. the addition, removal or substitution of some of its bases [85]. Mutations have multiple causes, including naturally occurring errors in DNA replication, mutagens (chemicals, radiation, *etc.*), and pathogens. In an individual mutation event, the number of mutated bases is typically low. In particular, single nucleotide polymorphisms (SNP) comprise the majority of known mutations [86].

Due to the redundancy of the genetic code a mutation may change a nucleotide triplet into a new triplet that encodes the same amino acid. Even though the accumulation of such events, called *synonymous* mutations, is thought to play a significant role in evolution [87], individual synonymous mutations have no immediate effect at the protein level. Conversely, *non-synonymous* mutations can induce different effects, including acquisition and loss of function. Mutant proteins may exhibit changes in physical and chemical properties, folding and stability, activity, structure and function [42, 88, 89, 90]. A single mutation affecting a catalytic site may for instance alter the efficiency of the enzyme. At the network level, mutations may disrupt the affinity with binding partners or form novel binding sites, which in turn influence the interaction network topology and biological pathways the proteins participates into.

In general, mutations occurring in functional regions of a protein are likely to have a deeper impact, and are thus subject to more stringent con-

[1] According to `http://www.rcsb.org/pdb/statistics/holdings.do`, retrieved on 2013/06/20.

straints [91]. Mutations affecting protein stability or folding (which may therefore have immediate detrimental effects on the host), are more likely to be rapidly removed from the genetic pool by natural selection. It is known that many serious inherited diseases, e.g. cancer and some neuromuscular pathologies, are due to mutations [92], and in particular to nsSNPs [93].

While it is true that, at the protein level, evolution rate has changed significantly in the last 1.5 billion years [94], the overall mutation rate of a gene depends crucially on the replication rate of an organism, and on the presence (or rather absence) of genetic error-correction mechanisms. Therefore, viruses represent an excellent source of genetic variation for the study of protein evolution. This is particularly true for RNA viruses, such as HIV-1, whose genetic material is chemically more prone to mutations [95]. Viral mutations are the mechanism by which viruses develop resistance to drugs [96], making the subject extremely important from a medical point of view, as we will see in Chapter 5.

2.2 Machine Learning

Machine learning is a large research field and covers a broad range of methodologies and problems, catering techniques from a number of other disciplines, including (but not limited to) mathematical optimization, statistics, formal languages, and artificial intelligence. In this section we focus on three areas of machine learning, namely *probabilistic graphical models*, *kernel methods*, and *inductive logic programming*, and further provide a short introduction to Statistical Relational Learning, which form the basis for the methods described and used in the next chapters.

In the following we will focus on *supervised* learning, where the goal is to find a *mapping* between inputs and outputs that *generalizes* over unseen inputs. While learning in a mathematical sense has a broadly accepted foundation, the exact meaning of "generalization", as well as the performance measures used to quantify the *generalization ability* of a learning machine, depend on the particular learning framework under examination. We will briefly discuss these details shortly, and refer the reader to any of the several exhaustive books on the subject [19, 18, 16, 17] for a more detailed treatment.

The other major machine learning flavor of machine learning is *unsupervised* learning [97], where the goal is to find "interesting" patterns — e.g. clusters, sub-spaces, latent representations — in the data. We will not make use of unsupervised learning techniques in this thesis. We will also ignore other flavors of learning, such as *reinforcement* learning [98], and postpone a description of *semi-supervised/transductive* learning [99] to a later Chapter.

2.2.1 Statistical Learning

Statistical learning is a sub-field of machine learning that borrows tools from statistics and decision theory to i) formally define the semantics of learning in a mathematical sense, and ii) solve the resulting numerical problems. Both kernel machines and probabilistic graphical models are part of statistical learning.

In *supervised* statistical learning, the learning algorithm is given a training set of input-output pairs $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$, with inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$, where the pairs are drawn i.i.d. (independently and identically distributed) from a fixed but *unknown* joint distribution:

$$(x_i, y_i) \sim p_{X,Y}(X, Y)$$

This is the so-called "natural" distribution of the learning problem. The inputs $x \in \mathcal{X}$ and the outputs $y \in \mathcal{Y}$ can be seen as drawn i.i.d. from fixed,

but (again) unknown distributions:

$$x \sim p_X \qquad y|x \sim p_{Y|X}$$

whose product is the joint[2]. Here we do not impose any restrictions on the form of the input and output domains $\mathcal{X}$ and $\mathcal{Y}$. In classical learning scenarios, the input domain $\mathcal{X}$ is a real-valued vector field $\mathbb{R}^d$ (for some $d \geq 1$), called the *feature space*, and $\mathcal{Y}$ is either the set of binary class labels $\mathcal{Y} = \{0, 1\}$, in which case learning is a *classification* problem, or a real-valued vector field $\mathcal{Y} = \mathbb{R}^{d'}$ (for some $d' \geq 1$), in which case we talk about a *regression* problem.

As already hinted to, learning amounts to finding a (predictive) function (also called *model* or *hypothesis*) $f : \mathcal{X} \to \mathcal{Y}$, taken from a set of possible models $\mathcal{F}$, that is able to *generalize* to unobserved samples: learning is *inductive*, as it aims to discover a general model of the data from a finite set of examples. A model f on a dataset $\mathcal{D}$ has good generalization ability if its output is "close enough" to the real output *for all possible inputs*. The notion of "closeness" can be formalized as a *loss function*.

Definition 2.2.1 (Loss function) *A function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a* loss *function if (i) $\ell(y, y') \geq 0$ for all $y, y' \in \mathcal{Y}$, and (ii) $\ell(y, y) = 0$ for all $y \in \mathcal{Y}$.*

The quantity $\ell(y, y')$ measures the penalty or cost incurred when a model predicts y' while the correct output is y. Here, (ii) ensures that there is no penalty when the predicted output is correct. A loss function implicitly defines the *risk* of a learning machine f, which measures the total error f entails when applied to the full domain $\mathcal{X} \times \mathcal{Y}$, i.e. its generalization ability:

Definition 2.2.2 (Risk) *Given a loss function ℓ, the risk (or total error) incurred by a model f is:*

$$R[f] := \mathbb{E}_{p_{X,Y}}[\ell(Y, f(X)] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, f(x)) p_{X,Y}(x, y) d(x, y)$$

The choice of loss function is problem dependant. Classical loss functions for classification, assuming that the possible classes are $y = 1$ and $y = -1$, are [100] the *zero-one* loss function $\ell(y, y') := \mathbb{I}[y = y']$, the *square loss* $\ell(y, y') := (1 - yy')^2$, and the *hinge* loss $\ell(y, y') := \max(0, 1 - y \cdot y')$. Loss functions for regression include the *square loss* $\ell(y, y') := (y' - y)^2$, the *absolute value loss* $\ell(y, y') := |y - y'|$, and the *ϵ-insensitive loss* $\ell(y, y') := \max(|y' - y| - \epsilon, 0)$.

By seeking the model f with maximum generalization ability — or equivalently minimum risk — we obtain the so-called risk minimization criterion.

[2]In the following we will omit the subscripts when they are clear from the context.

Definition 2.2.3 (Risk minimization) *Given a dataset D and a set of candidate models $\mathcal{F}$, find the model $f_* \in \mathcal{F}$ that minimizes the risk:*

$$f_* := \arg\min_{f \in \mathcal{F}} R[f]$$

The issue is that, of course, the joint $p_{X,Y}$ is unobserved: all the learning method is allowed to learn from is the dataset D, which (likely) captures only a very small portion of all possible combinations of inputs and outputs. This limitation implies that we can not compute the risk functional. Instead, we can resort to estimating an approximation thereof, the *empirical risk*, based on the empirical joint $\hat{p}_{X,Y}$. This reasoning leads to the *empirical risk minimization* procedure.

Definition 2.2.4 (Empirical risk minimization) *Given a loss function ℓ, the empirical risk incurred by a model f over a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ is:*

$$\hat{R}[f] := \mathbb{E}_{\hat{p}_{X,Y}}[\ell(y, f(x))] = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i))$$

Empirical risk minimization amounts to fininding the model $f \in \mathcal{F}$ that has minimizes the empirical risk:

$$f_* := \arg\min_{f \in \mathcal{F}} \hat{R}[f]$$

Despite its many forms, empirical risk minimization (ERM) stands at the heart of many machine learning methods. There are other optimality criteria used in the literature, such as *maximum likelihood* for probabilistic graphical models (which we sketch in Section 2.2.3), which states that the best model is the one most likely to have generated the data (modulo Bayesian priors), and *maximum entropy*, which postulates that the best model is the one that has maximal uncertainty given the observations [101]. All of these criteria, ERM included, boil down to mathematical optimization over the set of candidate functions — and the efficacy of the search algorithm depends crucially on the structure of $\mathcal{F}$ and on the choice of loss function. For the purpose of this section, for simplicity we will focus on the ERM formulation of learning.

Since the empirical risk is just an estimator of the true risk, it does not necessarily represent the true risk. Consider, for instance, a model whose output is always identical to the true output over the training set. Such a model achieves null empirical risk, but may still have an arbitrarily high error rate outside the training set. In other words, low empirical risk does not guarantee learning. This can occur in practice when the joint over the

training set $\hat{p}_{X,Y}$ is not a faithful sample for the joint $p_{X,Y}$ as a consequence of noise or missing data. In such conditions, the model may *overfit* the data.

To lessen the chances of overfitting, it is useful to restrict the set of candidate models $\mathcal{F}$, e.g. by penalizing complex hypotheses. This involves including a *regularization* term (or *prior*) over $\mathcal{F}$. How to do so depends on the method at hand, but the two most common regulariers are the ℓ_2 loss, which can be seen as a Gaussian prior over the model complexity, and the ℓ_1 loss, that corresponds to a Laplacian prior and favors more sparse models.

Notwithstanding these commonalities — namely, learning as optimization, use of regularizers/priors, reliance on i.i.d. assumptions for the data, and requiring data to be formatted in a propositional, or attribute-value, representation —, statistical learning methods vary greatly, and each method offers its particular set of trade-offs. We will omit a detailed treatment of the details, which is far beyond the scope of this work.

Model Selection

In order to perform model selection, i.e. picking a learned model out of many, it is necessary to estimate its generalization ability by evaluating its empirical performance on an independent dataset, called the *test* set. The test set must be drawn from the same distribution $p(X, Y)$ as the training set, but must also be as statistically independent from the latter, as to avoid optimistically inflating the model performance. There are a few alternative procedures to select the test set [16, 17].

The most widely used procedure is *cross-validation* (CV). In CV, the dataset is partitioned into k folds, with $k = 10$ in typical scenarios. The procedure consists of k rounds: in each iteration, a model is learned from of $k - 1$ folds and tested on the remaining one. The overall performance is estimated as the average performance over all rounds. Notably, cross-validation makes use of *all* the data available for training. Two variants on the theme are *leave-one-out*, where there are exactly as many folds as there are data instances, and *stratified* cross-validation, where the proportion of classes is kept fixed between folds — this is particularly important in cases of class unbalance. It has been shown that cross-validation with moderate choices of k reduces the variance and increase the bias [102], and that stratification produces more accurate results.

Another canonical procedure is the *bootstrap*, whereby a dataset with n instances is randomly sampled with replacement to form a test set of n instances, and the procedure can be repeated an appropriate number of times to obtain multiple test sets. Similarly to cross-validation, bootstrap reduces variance, but can show large bias under specific circumstances [102].

Finally, the *holdout* procedure consists of splitting the dataset into two mutually exclusive subsets. The bigger one, typically accounting for 2/3 of the data, is used for training, while the rest (the *holdout*) is used for testing only. The holdout is a pessimistic estimator [102], whose performance depends crucially on the selection of the test set. Moreover, since it does not prescribe repeated trials, the holdout method makes inefficient use of the data. Because of these issues, and since the cross-validation provides (under mild conditions [103]) better accuracy, the holdout has been largely superseded by cross-validation.

2.2.2 Kernel Methods

Kernel methods [104, 105, 106, 107] are one of the most popular classes of methods in machine learning, due to their efficiency, versatility, and theoretical soundness. Support Vector Machines, in particular, have been applied almost universally in the field of computational biology (see e.g. [12]), also thanks to the availability of excellent implementations [108, 109, 110]. Kernel methods come in a multitude of forms, and over time have been adapted to perform a variety of learning tasks.

The original Support Vector Machine (SVM) classifier, which laid the groundwork for the whole field, has been published in Vapnik *et al.* [111] and in Cortes and Vapnik in [14]. Given a dataset of attribute-value inputs $\mathbf{x}_i \in \mathbb{R}^d$ with binary class labels $y_i \in \{-1, 1\}$, the SVM attempts to find a separating *hyperplane* (or *weight vector*) $\mathbf{w} \in \mathbb{R}^d$, with offset $b \in \mathbb{R}$ from the origin, that maximizes the *margin* (hence the term *max-margin*) between instances belonging to the two classes. Given $\mathbf{w}$ and b, predicting the class of a novel input $\mathbf{x}$ amounts to checking on which side of the hyperplane it resides, i.e. the sign of hyperplane equation:

$$f(\mathbf{x}; \mathbf{w}, b) := \sigma(\mathbf{w}^T\mathbf{x} + b)$$

where $\sigma(t) = 1$ if $t \geq 0$, and -1 otherwise. One of the most important aspects of SVM max-margin training is that it guarantees that the true error of the learning machine is probabilistically bounded by the training set error (empirical risk) [112]. This result is one of the key advantages of Support Vector Machines over more heuristic methods.

The primal form of the training procedure [111] for linearly separable classes, where a separation hyperplane always exists, is:

$$\arg\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1, \quad i = 1, \ldots, n$$

By minimizing the modulo $\|\mathbf{w}\|$ of the weight vector, SVMs maximize the separation between the classes. In the non-separable case, the *soft* SVM formulation [14] introduces additional slack variables ξ_i to account for examples that necessarily violate the separation assumption. The primal form of soft SVM can be written as:

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

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

Here C is a hyper-parameter that balances the model complexity and the training set error, and can be used to control overfitting. When written in this form, it is clear that the term $\|\mathbf{w}\|^2$ acts as a regularizer, and behaves analogously to a Gaussian prior.

The optimization problem in the primal is a quadratic program, which can be solved by the Lagrange multiplier method. The Lagrangian dual form of (soft) SVM is:

$$\arg\min \sum_{i=0}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

$$s.t. \quad 0 \leq \alpha_i \leq C, i = 1, \ldots, n; \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

where the α_i are per-instance variables introduced by the dual transformation. The instances for which $\alpha_i > 0$ are called the *support vectors*, and are typically less than n. The hyperplane $\mathbf{w}$ can be reconstructed from the set of support vectors by means of the linear combination $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$. The fact that the max-margin hyperplane only relies on a subset of the instances, namely on the support vectors, renders SVMs more robust to noise.

Most importantly, since the data appears in the dual problem only in the form of dot products — i.e., $\mathbf{x}_i^T\mathbf{x}_j$ —, it is possible to substitute to the canonical dot product of Euclidean space any other dot product. This is the so-called "kernel trick" [105]. For an arbitrary input domain $\mathcal{X}$, a kernel is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ that satisfies an additional condition.

Definition 2.2.5 (Kernels [113]) *Given a function k as described above, and a set of objects $\{x_i\}_{i=1}^n$ the* Gram matrix *$K \in \mathbb{R}^{n\times n}$ is defined as:*

$$K_{ij} := k(x_i, x_j)$$

If the Gram matrix of k is positive semi-definite, i.e. if it satisfies the condition:

$$\sum_{i,j} c_i K_{ij} c_j \geq 0$$

for all vectors $\mathbf{c}$*, then* k *is a* kernel.

Kernel functions are important because they always represent a dot product in some Reproducing Kernel Hilbert Space (RKHS) [113]. Therefore, kernel functions can be used in place of regular dot product in SVMs. Furthermore, the Representer Theorem [113] guarantees that, whenever the function k is a kernel, SVM training is convex, and its optimum — the max-margin hyperplane — can be expressed as a kernel expansion of the form:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

Representer theorems have been devised for a multitude of kernel methods, and stand at the basis of the many efficient algorithms for SVMs, including Sequential Minimal Optimization [114] and the Cutting Plane algorithm [115]

Alternative kernels allow to perform classification with arbitrary objects, including structured and relational ones. The literature flourishes with kernels for different data types. The most basic kernels include the polynomial kernel and radial basis functions [113], which allow to include correlations between instances into the learning problem. Kernels for structured data also exist, for instance for attributed graphs (useful for representing chemical structures) and relational databases [116]. There are also more exotic kernels, like the Fisher kernel [117], which allows to define the similarity between structured objects in terms of a probabilistic graphical model (or stochastic grammar).

As we have seen, kernel methods have a number of advantages: they are efficient, since the training problem is convex, they can handle a plethora of different objects as *inputs* (even infinite dimensional ones) via the kernel trick, and they have a very firm theoretical basis. Support Vector Machines can be currently considered a *stock* learning method, and have been used as a black box throughout the computational biology literature. Furthermore, the core idea of max-margin learning has developed into a wealth of kernel machines, including regression [118], one-class classification [119], multi-class classification [120], transductive learning [121], and structured output prediction [122, 123], among many others. Despite its limitations, attribute-value representations allow kernel machines to easily integrate data from different sources — simply by concatenating the various features in a single feature vector $\mathbf{x}$, although there are more complex schemas, and, as always, caveats — a task often required in computational biology.

Notwithstanding their success, kernel machines of course also have limitations. First, even though SVMs can give a confidence on the prediction (namely, the margin $\mathbf{w}^T\mathbf{x} + b$), it is difficult to associate probabilities to their

predictions. For binary classification, a practical method is the one proposed by Platt [124], who suggests to fit a sigmoid to the predicted margin, and estimate a probability by evaluating the sigmoid. Clearly, this method is heurisic, as the sigmoid is not a density function. Another issue of most kernel methods is that they can not handle relational *outputs*. This restriction has been recently lifted by structured output SVMs [122, 123], but still affects the majority of kernel methods. Other limitations include the inability to natively deal with missing data (requiring the use of heuristics like the Expectation Maximization algorithm or pre-processing, e.g. removing instances with missing features) [125] and unbalanced classes. Finally, contrarily to logic-based methods, the learned SVM parameters are difficult to interpret, even for domain experts.

2.2.3 Probabilistic Graphical Models

Probabilistic graphical models (PGMs) are a principled and elegant framework to compactly represent high-dimensional joint distributions, which combines probability theory and graph theory [126, 127]. Joint distributions can represent arbitrary dependencies between variables, for instance between inputs (i.e. observed variables) and outputs of a prediction problem, and thus offer excellent expressivity and flexibility. Alas, learning a full joint distribution is infeasible: even in the simplest case, i.e. a multinomial over n binary variables, fitting is equivalent to learning 2^n parameters from the data. The main insight of PGMs is that, by selectively inserting independencies in the full joint, we can limit the complexity of the model — i.e., the number of parameters to be estimated —, drastically improving its learnability (especially when there is little data available), while retaining its expressive power. Contrarily to statistical learning methods like kernel machines, PGMs are designed to deal with relational data, and their predictions are natively accompanied by a confidence in the form of probabilities.

The basic idea is to define a joint by means of a graph $G = (X, E)$, whose nodes are random variables, and whose local dependencies are captured by the topology of the graph. In particular, variables connected by an edge are conditioned on each other. The exact semantics then depends on whether the graph is directed or undirected, in which case we talk of Bayesian networks (BNs) and Markov networks (MNs), respectively. The two classes of models differ in what distributions they can represent [128]. While the two definitions are fully general, in the following we will restrict our attention to models with discrete random variables.

Definition 2.2.6 (Bayesian network) *A Bayesian network defined on di-*

rected acyclic graph (DAG) $G = (X, E)$, where $X = \{X_i\}_{i=1}^n$ is a set of random variables and $E \subseteq X \times X$ is a set of directed edges, encodes the joint probabilty distribution:

$$p(X) = \prod_{i=1}^{n} p(X_i \mid \pi(X_i))$$

where $\pi(X_i)$ is the set of parents of node X_i. Here the functional form of the conditional distributions is assumed to be known beforehand.

Intuitively, a BN defines a set of *causality* relations in terms of local influences between a node and its parents. For discrete random variables, the local conditional distributions $p(X_i \mid \pi(X_i))$ are multinomials, parameterized by a conditional probability table (CPT), whose size is proportional to the number of parents and the number of their states. From a computational perspective, this cuts the number of parameters needed to specify a discrete joint distribution on n variables from $O(m^n)$ for a complete graph (for random variables taking values in $\{1, \ldots, m\}$ with n parents each) to a much smaller number that depends only on the graph topology.

Definition 2.2.7 (Markov network) *A Markov network defined on an undirected graph $G = (X, E)$, where $X = \{X_i\}_{i=1}^n$ is a set of random variables, $E \subseteq X \times X$ is a set of undirected edges, and $\mathcal{C}$ is the set of maximal cliques in G, encodes the joint probabilty distribution:*

$$p(X) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(X_C) = \frac{1}{Z} \exp \sum_{C \in \mathcal{C}} \phi_C(X_C)$$

Here X_C is the subset of variables X appearing in a clique C, the $\phi_C := \log \psi_C$ are arbitrary positive real-valued functions, called clique potentials *or* factors*, that evaluate the compatibility of the variables X_C, and Z is a normalization term.*

Rather than on causality, MNs rely on undirected correlations between the nodes. The clique potentials ϕ_C measure the "compatibility" between variables in a clique, and higher values of ϕ_C identify more likely configurations. The potentials need not be conditional distributions. The normalization term Z, also called *partition function*, converts the unnormalized exponential into a proper probability distribution.

It is worth mentioning that the majority of MNs used in the literature restrict the clique potentials to a linear form, and thus effectively encode a log-linear model:

$$p(X) = \frac{1}{Z} \exp \sum_{C \in \mathcal{C}} \mathbf{w}_C^T \mathbf{f}_C(X_C)$$

Here the functions $\mathbf{f}_C = (f_C^1, \ldots, f_C^d)$ act as *feature functions*, and their contribution to the potential is modulated by a set of real-valued weights $\mathbf{w}_C$. Positive weights are associated to features that render the configuration more likely. A natural way to implement the feature function is as indicator functions, that evaluate to 1 if a particular property is satisfied by the configuration of X_C, and to 0 otherwise. This very useful reinterpretation of clique potential has been introduced in the machine learning literature by Lafferty, McCallum and Pereira in [129].

While conceptually similar, BNs and MNs differ in what kinds of distributions they can represent [126]. In particular, BNs are more fit to model complex causal processes, and have been applied in computational biology to problems related to the prediction of bidimensional protein properties (such as secondary structure [130]), to gene regulation [131], and to collective protein function prediction [132], among others. Particular instantiations of BNs include Markov Chains, Hidden Markov Models (HMMs) [13], and Dynamic Bayesian Networks (DBNs) [133]. On the other hand, MNs are more suited to problems with no definite directionality, e.g. spatial and relational data, where the correlations are symmetrical. These cases can not be easily captured with BNs. MNs and their instantiations, such as Conditional Random Fields (CRFs) [129], have been applied to a wealth of problems, in particular in the areas of spatial statistics [134], computer vision [135], natural language processing [136], and many other tasks.

Inference in PGMs has different variants [137]. In general, given a set of observed nodes $X = x$, the *evidence*, the goal is to compute the most likely state y_* of a set of unknown, *query* nodes Y. This form of inference is called *maximum a-posteriori* (MAP), and can be written as:

$$y_* = \arg\max_y p(y \mid x) \tag{2.1}$$

Another type of inference involves computing the conditional probability of a given state $Y = y$ given the observations $X = x$, and can be written as:

$$p(y \mid x) = \frac{p(x, y)}{p(x)} \tag{2.2}$$

where the second equality follows from Bayes theorem. It is clear that, in order to perform the optimization in the first equation, we are required to evaluate the probability of the various candidates y, which in turn relies on being able to solve the second problem. Note also that, if there is a set of nodes W in the graph, which are neither observed nor queried, i.e. if there are *latent* variables in the problem, then W needs to be marginalized out in

order to compute Equation 2.2:

$$p(y \mid x) = \sum_{w} p(y, w \mid x) = \sum_{w} p(y \mid x, w) p(w \mid x) \tag{2.3}$$

This additional summation renders inference more computationally complex.

Similarly to most other probabilistic models, given a set of m examples D, learning in graphical models amounts to maximize the likelihood of the model f with respect to the data D, that is, to compute $\arg\max p(f \mid D)$. As is often the case in machine learning, in order to avoid overfitting, the parameters (i.e. CPTs for BNs and weights for log-linear models) are equipped with suitable priors. Restricting the discussion to log-linear Markov Networks, the maximum (log) likelihood equation can be written as:

$$\arg\max_{\mathbf{w}} \sum_{i=1}^{m} \frac{1}{Z} \exp\left(- \sum_{C \in \mathcal{C}} \mathbf{w}_C^T \mathbf{f}_C(x_C^i) \right)$$

where the x_C^i are the observed values of variables in clique C for example i [137]. The log-likelihood, being a concave function of the weights [126], can be minimized efficiently by stock first-order and second-order continuous optimization methods.

Alas, evaluating the objective function during the optimization procedure is equivalent to performing inference in the model, which turns out to be, in the worst case, intractable [137]: although there are general algorithms to compute the conditional $p(y \mid x)$ (e.g. message passing algorithms over the junction tree [126], which can be applied equally to directed and undirected graphical models), exact inference in PGMs is NP-complete [126]. In particular, the complexity of inference depends on the *treewidth* of the model, i.e. the size of the largest clique in the junction tree. This fact makes inference tractable only for models with low treewidth. As already hinted to, exact weight learning, which employs inference as a subroutine, also learning is intractable.

For this reason, researchers have developed many approximate algorithms [138], such as loopy belief propagation [139], approximate message passing [140], and variational inference [141]. The latter is based, roughly, on fitting a tractable distribution $q(X, Y; \theta)$ to the full joint $p(X, Y)$ and performing inference on the former. Thanks to its strong mathematical foundation and excellent flexibility — i.e. the possibility to carefully engineer the set Q from which q is taken from —, variational inference is routinely employed for hierarchical non-parametric Bayesian models [142], such as the Infinite Hidden Markov model [143], Latent Dirichlet Allocation (LDA) [144] and extensions, which have gained substantial momentum in the latest years. Approximate

inference in PGMs can also be performed with Markov Chain Monte Carlo (MCMC) methods [145], such as Gibbs sampling [145] which allow to draw samples from the joint $p(X, Y)$ and compute approximate expected statistics. This technique has been employed in very complex models, such as Markov Logic Networks [146] and extensions [147].

Overall, notwithstanding the serious computational limitations afflicting exact inference and parameter learning, PGMs offer many important advantages: the they are intimately relational (even if not necessarily first-order), can deal with non-i.i.d. data natively, enable the designer to model very complex probabilistic processes with in a clear and intuitive language, and are naturally suited for tasks such as collective classification. Furthermore, PGMs have been shown to be very successfull in practice. Tractable models such as Hidden Markov Models and pairwise Conditional Random Fields are the workhorse of (computational) gene and protein biology. Intractable models, which are bound to approximate inference (and learning), have also found broad application in computational biology. Moreover, PGMs can be turned into discriminative models, which capture the the conditional probability $p(y \mid x)$ directly rather than the full joint, using alternative learning strategies, such as *conditional* log-likelihood maximization and max-margin methods [148, 122, 123].

Finally, PGMs are naturally *modular*, and we will see that their structure can be defined by means of a formal language. This property can be exploited to perform *templatization*, i.e. to instantiate an arbitrary number of copies of a single PGM template. Using this technique, it is possible to write short model descriptions that can be unrolled over a large set of variables — with shared parameters. We will see later how this technique, when combined with a formal language to guide the instantiation, leads to first-order SRL models like Markov Logic Networks [149]. Another consequence, which we will not explore further, is that templatization by means of a formal language also facilitates the use of lifted algorithms [150], which exploit the layout of the template instances to reduce the complexity of inference.

2.2.4 Relational Learning

Relational learning [151] refers to learning from data that have structure, i.e. collections of objects that have *relations* between them, or collections of inherently relational/structured objects, or both. Examples of relational data include trees, graphs, multi-graphs, and relational databases.

Relational data is fundamentally different from the data considered in statistical learning, which can be easily represented in attribute-value form. On the contrary, relational data are *not* i.i.d, since the individual objects (or

parts thereof) are correlated, and their properties are somehow tied to each other. As a consequence, pure statistical learning methods, which ignore the dependencies between elements, are bound to underperform when applied to relational domains [152]. In particular, Inductive logic programming (ILP) [8, 15, 153] refers to a branch of relational learning that bridges learning and logical programming, and serves as a basis for many Statistical Relational Learning methods. In ILP the goal is, given a dataset, to recover a *logic program* h, called *hypothesis* or *concept*, which "explains" the data. Both the data and the hypothesis are described in First-Order Logic, which also serves as a mechanism to define what we mean by "explanation". In order to elucidate these points, in the following we will briefly outline the required concepts.

First-order Logic

The syntax of FOL accounts for four kinds of symbols: constants, variables, functions and predicates. Given a set of objects, *constants* identify specific objects in the domain, and are written in upper case; *variables* are placeholders for any object in the domain, written in lower case; *functions* map tuples of objects into an object; a *term* is either a constant, a variable, or a function applied to a set of terms.

Predicates are syntactic symbols, with a given *arity*, that describe either properties if a given object (if 1-ary) or relations between objects (if n-ary, with $n > 1$). Examples of predicates are:

$$\texttt{enzyme}(\texttt{p}) \qquad \texttt{bound}(\texttt{p}, \texttt{p}')$$

which capture the fact that a protein `p` may be an enzyme, or that two proteins `p`, `p'` interact, respectively. Predicates are associated with a Boolean *truth value* (`True` or `False`) that represents the fact that the predicate holds or not. For practical reasons, in most learning settings terms are typed.

An *atom* is a predicate instantiated on a tuple of terms. Well-defined *formulae* (or simply formulae) are inductively defined as either i) atoms, or ii) formulae connected by logical operators. The logical connectives define both the syntax and the semantics of the formulae, by determining how the truth value can be inferred from the sub-formulae. Given two formulae `F` and `G`, the logical connectives are:

- *negation.* $\neg$`F` is `True` if and only if `F` is `False`.
- *conjunction.* `F` $\wedge$ `G` is `True` iff both `F` and `G` are `True`.
- *disjunction.* `F` $\vee$ `G` is `True` iff either `F` or `G` is `True`.

- *implication.* $\mathtt{F} \Rightarrow \mathtt{G}$ is `True` iff `G` is `True` or `F` is `False`.
- *double implication.* $\mathtt{F} \Leftrightarrow \mathtt{G}$ is `True` iff `F` and `G` have the same truth value.
- *universal quantifier.* $\exists \mathtt{x}\ \mathtt{F}(\mathtt{x})$ is `True` iff there is at least one object `x` in the domain $\mathcal{X}$ for which $\mathtt{F}(\mathtt{x})$ is `True`.
- *existential quantifier.* $\forall \mathtt{x}\ \mathtt{F}(\mathtt{x})$ is `True` iff $\mathtt{F}(\mathtt{x})$ is `True` for each and every object `x` in the domain $\mathcal{X}$.

Terms, atoms and formulae containing no variables are called *ground*.

A *literal* is an atomic formula or its negation. A formula in conjunctive normal form (CNF) is a conjunction of *clauses*, where each clause is a disjunction of literals. Thus a CNF formula with n clauses and m variables each can be written as:

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} \mathtt{x}_{ij}$$

where the same variable can occur in different clauses. A *Horn clause* is a clause which contains at most one positive literal (the *head*), and can thus be written as:

$$\mathtt{head} \Leftarrow \mathtt{l}_1 \vee \ldots \vee \mathtt{l}_m$$

Here the set of literals $\mathtt{l}_j$ is called the *body* of the clause. A clause with no head is called a *fact*. A *definite logic program*, is a set of Horn clauses with exactly one head each. Definite logic programs are frequently used in ILP to represent the model (hypothesis). A *knowledge base* `KB` is a collection of implicitly conjoined formulae:

$$\mathtt{KB} := \bigwedge_{i} \mathtt{F}_i$$

An *interpretation* is a truth assignment to all predicates. An interpretation is a *model* for a formula `F` if it satisfies the formula, i.e. if it makes it true. Given two formulae `F` and `G`, `F` logically entails `G` if all models of `F` are also models of `G`. Checking whether a formula logically entails another, a common problem in e.g. theorem proving, is called the *satisfiability* problem for First-Order Logic. Explicitly enumerating the models that satisfy `F` in order to check whether they also satisfy `G` is infeasible, so solvers for satisfiability must rely only on the *syntax* of the two formulae. Unfortunately, FOL is only *semidecidable*, meaning that i) if `F` does logically entail `G`, then there exists an algorithmic procedure that produces a proof of the entailment; but ii) if `F` does not entail `G`, then there is no way to decide whether it is. In other words, the proof procedure may never terminate.

The canonical algorithm for solving First-Order satisfiability is the Davis-Putnam-Logemann-Loveland (DPLL) procedure [154, 155].

When the domain of constants is finite, as is in most relational learning tasks, then First-Order inference can be lowered to propositional satisfiability, by merely grounding the First-Order formulae. Given a (propositional) formula $\mathtt{F}$ in conjunctive normal form with clauses $\mathtt{C}_i$ on literals $\mathtt{l}_{ij}$:

$$\mathtt{F} := \bigwedge_i \mathtt{C}_i = \bigwedge_i \bigvee_j \mathtt{l}_{ij}$$

propositional satisfiability (SAT) requires to find an interpretation to the variables such that all clauses are satisfied. SAT is the prototypical NP-complete problem, but can be solved in practice by means of local search heuristics. A more complex extension of SAT, MAX-SAT, involves solving an *optimization* problem for *weighted* clauses, and is also NP-complete. Again, there are heuristic optimization procedures that work well in practice.

Inductive Logic Programming

The high level goal of ILP is to learn a deterministic *hypothesis* or *concept*, in the form of a logic program, that (intensionally) describes the relational data. In other words, the learned hypothesis can be interpreted as a logical theory that entails, or explains, the data at hand.

More formally, given a dataset D that includes n facts x_i, $i = 1, \ldots, n$, annotated with a (positive or negative) class label y_i, a background knowledge B of logical formulae, and a language bias L that describes the hypothesis space, the goal of ILP is to induce a logical program H that entails all positive examples and none of the negatives. That is, the learned hypothesis ought to be both *complete* and *consistent* with respect to the background knowledge B and the data D.

In order to define completeness and consistency, we define the indicator function $covers(H, B, x)$, which evaluates to 1 if the example x logically follows from the data and background knowledge, i.e. if $H \cup B \Rightarrow x$. By extension, we define $covers(B, H, X)$, which returns the set of elements in $X \subseteq D$ that are logically entailed by B and H. Completeness and consistency can then be defined as follows:

Definition 2.2.8 (Completeness) *A hypothesis H is* complete *with respect to background knowledge B and examples D, whose positive part is D^+, if and only if:*

$$covers(B, H, D^+) = D^+$$

Definition 2.2.9 (Consistency) *A hypothesis H is* consistent *with respect to background knowledge B and examples D, whose negative part is D^-, if and only if:*

$$covers(B, H, D^-) = \varnothing$$

Learning in ILP amounts to searching the hypothesis space for a complete and consistent hypothesis. Clearly, as a result of noise or wrong assumptions in the background knowledge, it is not always possible to find one, so ILP methods are restricted to find the best scoring hypothesis according to the number of false positives and false negatives.

In order to evaluate the quality of a candidate hypothesis H, the ILP procedure needs to compute the function $covers(B, H, x)$. Since inference in FOL is only semi-decidable, in order to facilitate computing the *covers* function, the rules in H and B are written in a decidable subset of FOL, typically Horn clauses. The model can thus be interpreted as a series of if-then rules. Chaining procedures or SLD-resolution rules of inference can be used in practice to compute the value of *covers* [15].

The space of hypotheses that must be searched during learning is structured as a partial order through a generality relation between clauses, called θ-subsumption, as follows.

Definition 2.2.10 (Substitution) *A* substitution $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$ *is a function from variables to terms. The application $F\theta$ of a substitution θ to a well-formed formula F is obtained by replacing all occurrences of each variable X_i in F by the term $_i$.*

Definition 2.2.11 (Subsumptions) *A program clause c θ-subsumes another program clause c' if there exists a substitution θ such that $c\theta \subseteq c'$. Two clauses c, c' are θ-subsumption equivalent if they both θ-subsume each other. A clause c is reduced if it is not θ-subsumption equivalent to any proper subset of itself.*

θ-subsumption embodies the syntactic equivalent of the notion of generality: if c θ-subsumes c' then it is at least as general than c', written $c \leq c'$. The latter operator is a partial ordering, and as such turns the set of reduced clauses into a lattice. This means that any two clauses c and c' have a least upper bound and a greatest lower bound, unique up to variable renaming.

Definition 2.2.12 (Least general generalization) *The* least general generalization *(lgg) of two reduced clauses c and c', denoted by $lgg(c, c')$, is the least upper bound of c and c' in the θ-subsumption lattice.*

The definition of *lgg* provides the basis for the two basic ILP hypothesis learning techniques we will be concerned with here: *bottom-up* learning, which involves building least general generalizations from the dataset (and the background knowledge), and *top-down* learning, which defines the search of refinement graphs. Despite differing on the details, all current ILP methods belong to one of these two classes. Among them there are Golem [156], Progol [157], FOIL [158] and Aleph [159].

Bottom-up methods search the hypothesis space in a bottom-up fashion, starting from the most specific clause allowed by the language bias that cover a given example, and then generalize it until it can not be further generalized without covering a negative example. The generalization is carried out using a *generalization operator*

$$\rho(c) := \{c' \mid c' \in \mathcal{L}, c' < c\}$$

which can perform two types of operations: (i) apply an inverse substitution to the clause, i.e.substitute a constant with a variable, and (ii) remove a literal from the body of the clause (which necessarily makes the clause more general, given the DNF form.)

Top-down methods work in the opposite direction, starting from an empty clause (which entails all examples) and gradually specializing it. The specialization is carried out using a *refinement operator*

$$\rho(c) := \{c' \mid c' \in \mathcal{L}, c < c'\}$$

and typically compute only the set of most general specialization of a clause under θ-subsumption. The two basic syntactic operations are (i) apply a substitution to the clause, thus introducing a constant in place of a variable, and (ii) add a literal to the body of the clause.

ILP is very versatile, and has been applied successfully to a variety of settings. Focusing on the biological domain, ILP has been employed for classical computational tasks such as predicting sequence-based homology and gene/protein function [160], finding regularities in microarray data [161], modeling protein–ligand [162] and protein–protein interactions [163], inferring signal transduction pathways [74], discoverying pharmacophores [164], analyzing the inhibition of enzymes in metabolic networks [165], and to drug design [166, 167].

The biggest advantages of ILP over other classes of methods are that it naturally deals with relational data — any learning problem that can be expressed in FOL is a candidate for ILP. Another advantage is that ILP explicitly enables the user to include *background knowledge* B, separate from

the data, in the learning problem: B allows to restrict the space of hypotheses, reducing the amount of data required for learning and providing additional cues to reduce overfitting. ILP methods also allow to learn concepts from one-class data alone (i.e. from positive examples only citemuggleton1997learning), which is an essential capability in many domains. Finally, the learned concept (logic program) is not only predictive, but may also serve as an intensional *description* of the data. The concept consists of few rules, FOL sentences that are close to natural language and can be readily understood by domain experts.

Alas, the bottom-up and top-down search strategies are indeed *heuristic*. Since the hypothesis space is exponentially large and highly non-convex, learning is not guaranteed to find the *globally* optimal hypothesis. This limitation affects all ILP systems. To alleviate this issue, the language bias can be used as a mechanism to control the quality of the learned concept by restricting and biasing the search. Additionally, many ILP methods can revise the learned concept as a post-processing pass to prune irrelevant (too-generic or too-specific) clauses.

Ignoring these complexity issues, Inductive Logic Programming, due to its logical background, is afflicted by two serious limitations. First, it provides no sound mechanism for handling noise and uncertainty. To the best of our knowledge, all ILP methods published to date rely on heuristics to preprocess the data prior to learning. Second, ILP is not designed for dealing with similarities and distances, a fundamental quantity in many learning tasks. In general, logic is not well fit for representing numerical attributes, nor arithmetic. The issue is even more dire for rational values, which require quantization and a consequent loss of precision. These two issues prevent ILP from being immediately applicable to many central learning tasks.

2.3 Statistical-Relational Learning

As already anticipated in the previous Chapter, the field of Statistical Relational Learning (SRL), sometimes also called Probabilistic Logic Programming, was born out the need for methods that could deal with *relational* data in a probabilistically sound way. The methods developed in SRL often represent generalizations of either kernel methods, probabilistic graphical models, or relational learning techniques. Given their very broad and diverse background, methods in SRL are very varied, and the relation between them has not been explored entirely. This state of affairs is often referred to as the SRL "alphabet soup" [168, 169]. In this section we will not attempt an exhaustive enumeration of the plethora of methods SRL has to offer, but rather overview

some of the most peculiar ones and hint at their properties. For more details on the subject, please refer to any of the following monographs [1, 170].

Just like in ILP, in SRL the data is given in relational form. Objects are composed of Boolean variables, i.e. predicates, sometimes associated with a prior probability. As a general rule, what distinguishes SRL methods from other statistical or probabilistic structured output methods — such as structured output SVMs and probabilistic graphical models — is the fact that they encode a First-Order model, rather than a propositional one: rather than describing the relations between individual variables directly, FOL is used to define rules, or constraints, or logic programs, which indirectly define the structure of the model. How this is done depends on the particular model at hand.

We will distinguish between three classes of models, namely those that define a probability distribution over *possible worlds*, or Herbrand interpretations; those that procedurally define a probabilistic semantics according to a logic-program or equivalent formalism; and those that do not explicitly define a probability distribution, but still leverage statistical induction.

Models in the first class include Markov Logic Networks (MLN) [171], where the probability of an interpretation — or truth assignment to all predicates — is defined by how many *weighted* FOL constraints it satisfied. We describe MLNs more thoroughly in Chapter 3.

Models in the second class include Statistical Logic Programs [172], PRISM [173] and Church [174], where a Prolog- or Lisp-like logical language is used to define a logic program, whose derivation trees then define the probability of the outputs. We will not further discuss this class of models, for simplicity.

In the last class there are models like nFOIL and kFOIL, which employ a (dynamic) propositionalization technique to flatten a First-Order logic program into a propositional representation manageable through canonical statistical learning methods. Semantic Based Regularization (SBR) [7] also falls in this group. SBR is based on a multi-task kernel learning paradigm, where each predicate is modeled as a kernel classifier; the learning tasks are constrained by a background knowledge of First Order rules. We will give a more thorough explanation of SBR in Chapter 4, and of kFOIL in Chapter 5.

All models in SRL provide great flexibility and expressive power, but those come at the cost of computational complexity. Methods that derive from probabilistic graphical models inherit the same type of intractable inference; furthermore most other probabilistic models require sampling to compute output probabilities. This is to be expected, because relational learning problems are distinctly more complex than propositional ones.

Chapter 3

Joint Refinement of Heterogeneous Predictions

3.1 Motivation

Automatic assessment of protein features from amino acid sequence is a fundamental problem in bioinformatics. Reliable methods for inferring features such as secondary structure, functional residues, subcellular localization, among others, are a first step towards elucidating the function of newly sequenced proteins, and provide a complement and a reasonable alternative to difficult, expensive and time-consuming experiments. A wealth of predictors have been developed in the last thirty years for inferring many diverse types of features, see e.g. Juncker *et al.* [175] for a review.

A key observation, often used to improve the prediction performance, is that several protein features are strongly correlated, i.e. they impose constraints on each other. For instance, information about solvent accessibility of a residue can help to establish whether the residue has a functional role in binding other proteins or substrates [176], whether it affects the structural stability of the chain [86], whether it is susceptible to mutations conferring resistance to drugs [177], whether it occurs within a flexible or disordered segment [178], *etc.* There are several other examples in the literature.

Researchers have often exploited this observation by developing predictors that accept correlated features as additional inputs. This way, the output is conditioned on the known value of the input features, thus reducing the possible inconsistencies. It is often the case that the additional input features are themselves predicted. Highly complex prediction tasks like 3D protein structure prediction from sequence are typically addressed by splitting the problem into simpler subproblems (e.g. surface accessibility, secondary struc-

ture), whose predictions are integrated to produce the final output. Following this practice, multiple heterogeneous predictors have been integrated into suites (see e.g. Distill [22], SPACE [179] and PredictProtein [23]) providing predictions for a large set of protein features, from subcellular localization to secondary and tertiary structure to intrinsic disorder.

However, existing prediction architectures (with a few specific exceptions, e.g. [180] and [10]) are limited in that the output feature can't influence a possibly mis-predicted input feature. In other words, while feature relations establish a set of *mutual* constraints, all of which should simultaneously hold, current predictors are inherently *one-way*.

Motivated by this observation, we propose a novel framework for dealing with the integration and mutual improvement of correlated predicted features. The idea is to explicitly leverage all constraints, while accounting for the fact that both the inputs, i.e. the *raw* predictions, and the constraints themselves are not necessarily correct. The refinement is carried out by a probabilistic-logical consistency layer, which takes the raw predictions as inputs and a set of weighted rules encoding the biological constraints relating the features. To implement the refiner, we use Markov Logic Networks (MLN) [149], a statistical-relational learning method able to perform statistical inference on first-order logic objects. Markov logic allows to easily define complex, rich first-order constraints, while the embedded probabilistic inference engine is able to seamlessly deal with potentially erroneous data and soft rules. We rely on an adaptation of MLN allowing to include grounding-specific weights (grounding specific Markov Logic Networks) [3], i.e. weights attached to specific instances of rules, corresponding in our setting to the raw predictions. The resulting refining layer is able to improve the raw predictions by removing inconsistencies and constraint violations.

Our method is very general. It is designed to be applicable, in principle, to any heterogeneous set of predictors, abstracting away from their differences (inference method, training dataset, performance metrics), without requiring any changes to the predictors themselves. The sole requirement is that the predictions be assigned a confidence or reliability score to drive the refinement process.

As an example application, we show how to apply our approach to the joint refinement of three highly related features predicted by the PredictProtein Suite [23]. The target features are subcellular localization, generated with Loctree [4]; disulfide bonding state, with Disulfind [5]; and metal bonding state, with MetalDetector [6]. We propose a few simple, easy to interpret rules, which represent biologically motivated constraints expressing the expected interactions between subcellular localization, disulfide and metal bonds.

The target features play a fundamental role in studying protein structure and function, and are correlated in a non-trivial manner. Most biological processes can only occur in predetermined compartments or organelles within the cell, making subcellular localization predictions an important factor for determining the biological function of uncharacterized proteins [4]; furthermore, co-localization is a necessary prerequisite for the occurrence of physical interactions between binding partners [72], to the point that lack thereof is a common mean to identify and remove spurious links from experimentally determined protein-protein interaction networks. Disulfide bridges are the result of a post-translational modification consisting in the formation of a covalent bond between distinct cysteines either in the same or in different chains [40]. The geometry of disulfide bonds is fundamental for the stabilization of the folding process and the final three-dimensional structure by fixing the configuration of local clusters of hydrophobic residues; incorrect bond formation can lead to misfolding [181]. Furthermore, specific cleavage of disulfide bonds directly controls the function of certain soluble and cell-membrane proteins [182]. Finally, metal ions provide key catalytic, regulatory or structural features of proteins; about 50% of all proteins are estimated to be metalloproteins [183], intervening in many aspects of of the cell life.

Subcellular localization and disulfide bonding state are strongly correlated: a reducing subcellular environment makes it less likely for the protein to form disulfide bridges [184]. At the two extremes we find the cytosol, which is clearly reducing, and the extra-cellular environment for secreted proteins, which is oxydizing and does not hinder disulfide bonds, with the other compartments (nucleus, mitochondrion, *etc.*) exhibiting milder behaviors. Similarly, due to physicochemical and packing constraints, it is unlikely for a cysteine to link both another cysteine (or more than one) and a ligand; with few exceptions, cysteines are involved in at most one of these bonds [6].

This is the kind of prior knowledge we will use to carry out the refinement procedure. We note that all these constraints are not *hard*: they hold for a majority of proteins, but there are exceptions [184]. In the following, we will show that that different predictors offer complementary advantages, and how our method is able to integrate them using non-trivial constraints, resulting in an overall improvement of prediction accuracy and consistency.

3.1.1 Overview of the Proposed Method

In this chapter we propose a framework to jointly refine existing predictions according to known biological constraints. The goal is to produce novel, refined predictions from the existing ones, so as to minimize the inconsistencies,

Figure 3.1: Diagram of the refinement pipeline.

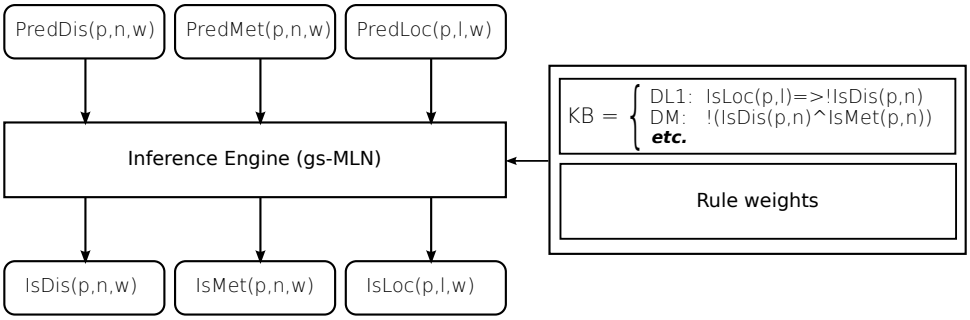


Table 3.1: List of predicates used in the MLN Refiner.

Predicate	**Meaning**
`PredLoc(p,l,w)`	Protein `p` is predicted in compartment `l` with confidence `w`
`PredDis(p,n,w)`	Cysteine at position `n` is predicted disulf. bound with confidence `w`
`PredMet(p,n,w)`	Cysteine at position `n` is predicted metal bound with confidence `w`
`IsLoc(p,l)`	Protein `p` is in compartment `l`
`IsDis(p,n)`	Cysteine at position `n` is disulf. bound
`IsMet(p,n)`	Cysteine at position `n` is metal bound
`ProxyLoc(p,l)`	Proxy predicate to account for estimated Loctree performance
`ProxyDis(p,n)`	Proxy predicate to account for estimated Disulfind performance
`ProxyMet(p,n)`	Proxy predicate to account for estimated MetalDetector performance

in a way that requires minimal training and no changes to the underlying predictors. The proposed system takes the raw predictions, which are assumed to be associated with a confidence score, and passes them through a probabilistic-logical consistency layer. The latter is composed of two parts: a knowledge base (KB) of biological constraints relating the features to be refined, encoded as weighted first-order logic formulae, which acts as an input to the second part of the method; and a probabilistic-logical inference engine, implemented by a grounding-specific Markov Logic Network (gs-MLN) [6]. For a graphical depiction of the proposed method see Figure 3.1.

An example will help to elucidate the refinement pipeline. For simplicity, let's assume that we are interested in refining only two features: subcellular localization and disulfide bonding state. The first step is to employ two arbitrary predictors to generate the raw predictions for a given protein `P`. Note that disulfide bonding state is a per-cysteine binary prediction, while subcellular localization is a per-protein n-ary prediction; both have an associated reliability score, which can be any real number. For a complete list of predicates used in this chapter, see Table 3.1.

Table 3.2: Knowledge Base used in the MLN refiner.

Name	Weight	Rule	Description
I1	per-protein	PredLoc(p,l,w) $\wedge$ IsLoc(p,l)	Input rule for sub. loc.
I2	per-cysteine	PredDis(p,n,w) $\wedge$ IsDis(p,n)	Input rule for dis. bonding state
I3	per-cysteine	PredMet(p,n,w) $\wedge$ IsMet(p,n)	Input rule for metal bonding state
I1P	per-protein	PredLoc(p,l,w) $\wedge$ ProxyLoc(p,l)	Input rule for proxy sub. loc.
I2P	per-cysteine	PredDis(p,n,w) $\wedge$ ProxyDis(p,n)	Input rule for proxy dis. bonding state
I3P	per-cysteine	PredMet(p,n,w) $\wedge$ ProxyMet(p,n)	Input rule for proxy metal bonding state
PX1	from data	ProxyLoc(p,l) $\Leftrightarrow$ IsLoc(p,l)	Proxy rule for sub. loc.
PX2	from data	ProxyDis(p,n) $\Leftrightarrow$ IsDis(p,n)	Proxy rule for dis. bonding state
PX3	from data	ProxyMet(p,n) $\Leftrightarrow$ IsMet(p,n)	Proxy rule for metal bonding state
DL1	from data	IsLoc(p,l) $\Rightarrow$!IsDis(p,n)	Compartment l *hinders* the formation of dis. bonds
DL2	from data	IsLoc(p,l) $\Rightarrow$ IsDis(p,n)	Compartment l *favors* the formation of dis. bonds
DM	from data	!(IsDis(p,n) $\wedge$ IsMet(p,n))	Half-cysteines can't bind metal atoms
L1	∞	$\bigvee_{l}$ IsLoc(p,l)	A protein must belong to at least one compartment
L2	∞	$\forall$l1 IsLoc(p,l1) $\wedge \bigwedge_{l2}$!IsLoc(p,l2)	A protein must belong to at most one compartment

Let's assume that the predictions are as follows:

```
PredLoc(P,Nuc,0.1)   PredLoc(P,Cyt,1.2)
!PredLoc(P,Mit,0.8)  PredLoc(P,Ext,1.0)

PredDis(P,20,0.8)    PredDis(P,26,0.6)
```

where "!" stands for logical negation. The first four predicates encode the fact that protein P is predicted to reside in the nucleus with confidence 0.1, in the cytosol with confidence 1.2, *etc.* The remaining three predicates encode the predicted bonding state of three cysteines at positions 11, 20 and 26: the first cysteine is free with confidence 0.2, the remaining two are bound with confidence 0.8 and 0.6, respectively. In this particular example, the protein is assigned conflicting predictions, as the cytosolic environment is known to hinder the formation of disulfide bridges. We expect one of them to be wrong.

Given the above logical description, our goal is to infer a new set of *refined* predictions, encoded as the predicates IsLoc(p,l) and IsDis(p,n). To perform the refinement, we establish a set of logical rules describing the constraints we want to be enforced, and feed it to the inference engine. For a list of rules, see Table 3.2.

First of all, we need to express the fact that the raw predictions should act as the primary source of information for the refined predictions. We accomplish this task using the input rules I1 and I2. These rules encode how the refined prediction predicates IsDis and IsLoc depend primarily on the raw predicates PredDis and PredLoc. The weight w is computed from the estimated reliability output by the predictor, and (roughly) determines how likely the refined predictions will resemble the raw ones.

Next we need to express the fact that a protein must belong to at least

one cellular compartment, using rule `L1`, and, as normally assumed when performing subcellular localization prediction, that it can not belong to more than one, using rule `L2`. In this example, and in the rest of the chapter, we will restrict the possible localizations to the nucleus, the cytosol, the mitochondrion, and the extracellular space. The two above rules are assigned an infinite weight, meaning that they will hold with certainty in the refined predictions.

The last two rules used in this example are `DL1` and `DL2`, which express the fact that the cytosol, mitochondrion and nucleus tend to hinder the formation of disulfide bridges, while the extracellular space does not. In this case, the weights associated to the rules are inferred from the training set, and reflect how much the rules hold in the data itself.

Once we specify the raw predictions and knowledge base, we feed them to the gs-MLN. The gs-MLN is then used to infer the set of refined predictions, that is, the `IsLoc` and `IsDis` predicates. The gs-MLN allows to query for the set of predictions that is both most similar to the raw predictions, and at the same time violates the constraints the least, taking in account the confidences over the raw predictions and the constraints themselves. See the Methods section for details on how the computation is performed. In this example, the result of the computation is the following: `IsLoc(P,Ext)`, `IsDis(P,11)`, `IsDis(P,20)`, `IsDis(P,26)`. The protein is assigned to the second most likely subcellular localization, "extracellular", and the cysteine which was predicted as free with a low confidence is changed to disulfide bonded.

It is easy to see that this framework allows to express very complicated rules between an arbitrary number of features, without particular restrictions on their type (binary, multi-label) and at different levels of detail (per-residue or per-protein). Furthermore, this approach minimizes the impact of overfitting: there is only one learned weight for each rule, and very few rules. To assess the performance of our refiner, we experiment with improving subcellular localization together with disulfide bonding state and metal bonding state. The knowledge base used for localization and disulfide bridges was introduced in this section. As for metals, the information is input using rule `I3`, and we model the interaction with disulfide bonds through rule `DM`, which states that the two types of bonds are mutually exclusive.

3.1.2 Related work

There is a vast body of work dedicated to the issue of information integration, and in particular to the exploitation of correlated protein features. In many cases, the proposed methods are limited to augmenting the inputs us-

ing correlated features (either true or predicted) as additional hints to the predictors. In this setting, a work closely related to ours is [185], in which Savojardo and colleagues propose a prediction method for disulfide bridges that explicitly leverages predicted subcellular localization [186]. As in the other cases, the authors implement a one-way approach, in which a predicted feature (localization) is employed to improve a related one (disulfide bonding state). The protein prediction suites briefly mentioned above (Distill [22], SPACE [179] and PredictProtein [23]) provide another clear example of one-way architectures. Prediction suites are built by stacking multiple predictors on top of each other, with each layer making use of the predictions computed by the lower parts of the stack. In this case, the main goal is the computation of higher-level features from simpler ones. Note however that the issue of two-way consistency is ignored: these architecture do not back-propagate the outputs of the upper layers to the bottom ones. On the other hand, our approach allows to jointly improve *all* predictions by enforcing consistency in the refined outputs.

Another popular way to carry out the prediction of correlated features is multi-task learning. In this setting, one models each prediction task as a separate problem and trains all the predictors jointly. The main benefit comes from allowing information to be shared between the predictors during the training and inference stages. These methods can be grouped in two categories: iterative and collective.

Iterative methods exploit correlated predictions by re-using them as inputs to the algorithm, and iterating the training procedure until a stopping criterion is met. This approach can be found in, e.g. Yip *et al.* [10], which proposes a method to jointly predict protein-, domain-, and residue-level interactions between distinct proteins. Their proposal involves modeling the propensity of each protein, domain and residue to interact with other objects at the same level as a distinct regression task. After each iteration of the training/inference procedure, the most confident predictions at one level are propagated as additional training samples at the following level. This simple mechanism allows for information to bi-directionally flow between different tasks and levels. Another very relevant work is [180], in which Maes *et al.* jointly predict the state of five sequential protein features: secondary structure (in 3 and 8 states), solvent accessibility, disorder and structural alphabet. Also in this case, distinct predictors are run iteratively using the outputs at the previous time slice as additional inputs. Collective methods instead focus on building combinations of classifiers, e.g. neural network ensembles, using shared information in a single training iteration. As an example, [187] describes how to maximize the diversity between distinct neural networks with the aim of improving the overall accuracy. However most applications

in biology focus on building ensembles of predictors for the *same* task, as is the case in Pollastri *et al.* [188] for secondary structure.

The main differences with our method are the following: (a) There exist a number of independently developed predictors for a plethora of correlated features. It would be clearly beneficial to refine their predictions in some way. Our goal is to be able to integrate them without requiring any change to the predictors themselves. The latter operation may be, in practice, infeasible, either because the source is unavailable, or because the cost of retraining after every change is unacceptably high. All of the methods presented here are designed for computing predictions from the ground up; our method is instead designed for this specific scenario. (b) Our method allows one to control the refinement process by including prior knowledge about the biological relationships affecting the features of interest; furthermore the language used to encode the knowledge base, first-order logic, is well defined and flexible. The other methods are more limited: any prior knowledge must be embedded implicitly in the learning algorithm itself. (c) The weights used by our algorithm are few, simple statistics of the data, and do not require any complex training. On the other hand, all the methods presented here rely on a training procedure, and have a higher risk of incurring in overfitting issues.

3.2 Results and Discussion

3.2.1 Data Preparation

We assessed the performance of our framework on a representative subset of the Protein Data Bank [189], the 2010/06/16 release of PDBselect [190]. The full dataset includes 4,246 unique protein chains with less than 25% mutual sequence similarity.

Focusing only on proteins containing cysteines, we extracted the true disulfide bonding state using the DSSP software [38], and the true metal bonding state from the PDB structures using a contact distance threshold of 3 Å. Metals considered in this experiment are the same used for training MetalDetector, a total of 33 unique metal atoms and 75 molecular metals. See Passerini *et al.* [191] for more details.

Subcellular localization was recovered using the annotations in DBSubLoc [192] and UniProt [193]; we translated between PDB and UniProt IDs using the chain-level mapping described by Martin [194], dropping all proteins that could not be mapped. To increase the dataset coverage, we kept all those proteins whose true localization did not belong to any of the classes predicted by Loctree (which for animal proteins amount to cytosol, mitochondrion,

nucleus and extracellular – secreted), was ambiguous or missing, and marked their localization annotation as "missing". Loctree is also able to predict proteins in a fifth, composite class, termed "organelle", which includes the endoplasmic reticulum, Golgi apparatus, peroxysome, lysosome, and vacuole. The chemical environment within these organelles can be vastly different, so we opted for removing them from the dataset, for simplicity.

Subcellular localization prediction requires different prediction methods for each kingdom. The preprocessing resulted in a total of 1184 animal proteins, and a statistically insignificant amount of plant and bacterial proteins; we discarded the latter two. Of the remaining proteins, 526 are annotated with a valid subcellular localization (i.e. not "missing"). The data includes 5275 cysteines, of which 2456 (46.6%) are half cysteines (i.e. involved in a disulfide bridge) and 458 (8.7%) bind metal atoms. We also have two half cysteines that bind a metal (in protein 2K4D, chain A); we include them in the dataset as-is.

3.2.2 Evaluation Procedure

Each experiment was evaluated using a standard 10-fold cross-validation procedure. For each fold, we computed the rule weights over the training set, and refined the remaining protein chains using those weights. The rule weights are defined as the log-odds of the probability that a given rule holds in the data, that is, if the estimated prediction reliability output by the predictor is r, the weight is defined as $w = \log(r/(1-r))$. Given the weights, we refine all the raw features of proteins in the test set. If the subcellular localization for a certain protein is marked as "missing", we use the predicted localization to perform the refinement. In this case, the refined localization is not used for computing the localization performance, and only the disulfide and metal bond refinements contribute to the fold results, in a semi-supervised fashion.

For binary classification (i.e. disulfide and metal bonding state prediction) let us denote by T_p, T_n, F_p and F_n the number of true positives, true negatives, false positives, and false negatives, respectively, and N the total number of instances (cysteines). We evaluate the performance of our refiner with the following standard measures:

$$Q = \frac{T_p + T_n}{N} \tag{3.1}$$

$$P = \frac{T_p}{T_p + F_p} \tag{3.2}$$

$$R = \frac{T_p}{T_p + F_n} \tag{3.3}$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{3.4}$$

The accuracy Q, precision P and recall R are standard performance metrics. The F_1 score is the harmonic mean of precision and recall, and is useful as an estimate balancing the contribution of the two complementary measures. We report the average and standard deviation of all above measures taken over all folds of the cross-validation.

For multi-class classification (subcellular localization) we compute the confusion matrix M over all classes, where each element M_{ij} counts the number of instances whose true class is i and were predicted in class j. The more instances lie on the diagonal of the confusion matrix, the better the predictor.

We note that, in general, it is difficult to guarantee that our test set does not overlap with the training set of the individual raw predictors. This may result in an artificial overestimate of the performance of the raw predictors. However, training in our model consists in estimating the rule weights from the raw predictions themselves. As a consequence, the results of our refiner may be underestimated when compared with the inflated baseline performance. We also note that, since our model requires estimating very few parameters, i.e. one weight per rule, it is less susceptible to overfitting than methods having many parameters which rely on a full-blown training procedure.

3.2.3 Raw Predictions

We generate the predictions for subcellular localization, disulfide bridges, metal bonds and solvent accessibility using the respective predictors. All predictors were installed locally, using the packages available from the PredictProtein Debian package repository[1], and configured to use the default parameters. For all protein chains predicted in the "organelle" class, we marked the prediction as "missing", for the reasons mentioned above.

For Disulfind and MetalDetector, we converted the per-cysteine weighted binary predictions into two binary predicates for each cysteine, `PredDis/3` and `PredMet/3`, using as prediction confidence `w` the SVM margin. For Loctree, we output four `PredLoc/3` predicates for each protein, one for each possible subcellular localization, and computed the confidence by using a continuous version of the Loctree-provided output-to-confidence mapping.

[1] https://www.rostlab.org/owiki/index.php/Packages

The raw predictor performance can be found alongside with the refiner performance in Tables 3.3 to 3.6.

3.2.4 Alternative Refinement Pipelines

In order to assess the performance of our method, we carried out comparative experiments using two alternative refinement architectures. Both architectures are based on state-of-the-art sequential prediction methods, namely Hidden Markov Support Vector Machines (HMSVM) [195] and Bidirectional Recurrent Neural Networks (BRNN) [196]. Both methods can naturally perform classification over sequences, and have been successfully applied to several biological prediction tasks.

The alternative architectures are framed as follows. The predictors are trained to learn a mapping between raw predictions and the ground truth, using the same kind of pre-processing as the MLN refiner. Cysteines belonging to a protein chain form a single example, and all cysteines in an example are refined concurrently. The input consists of all three raw predictions in both cases.

The two methods were chosen as to validate the behavior of more standard refinement pipelines relying on both hard and soft constraints. In the case of HMSVMs, the model outputs a single label for each residue: a cysteine can be either free, bound to another cysteine, or bound to a metal. This encoding acts as a hard constraint on the mutual exclusivity between the two labels. In the case of BRNNs, each cysteine is modeled by two independent outputs, so that all four configurations (free, disulfide bound, metal bound, or both) are possible. The BRNN is given the freedom to learn the (soft) mutual exclusivity constraint between the two features from the data itself.

Pure sequential prediction methods, like HMSVMs, are at the same specialized for, and limited to, refining sequential features, in our case disulfide and metal bonding state. Therefore, we can't use the HMSVM pipeline for localization refinement. As a result, the alternative pipeline is faced with a reduced, and easier, computational task. While BRNN are also restricted to sequential features, more general recursive neural networks [197] can in principle model arbitrary network topologies. However, they cannot explicitly incorporate constraints between the outputs, which is crucial in order to gain mutual improvement between subcellular localization and bonding state predictions. As experimental results will show, these alternative approaches already fail to jointly improve sequential labeling tasks.

We performed a 10-fold inner cross-validation to estimate the model hyperparameters (regularization tradeoff for the HMSVM, learning rate for the neural network), using the same fold splits as the main experiment. The

results can be found in Table 3.3 through 3.6.

3.2.5 True Subcellular Localization

As a first experiment, we evaluate the effects of using the true subcellular localization to refine the remaining predictions, i.e. we supply the refiner with the correct `IsLoc` directly, while querying the `IsDis` and `IsMet` predicates. The experiment represents the ideal case of a perfect subcellular localization predictor, and we can afford to unconditionally trust its output.

The experiment is split in four parts of increasing complexity.

- In the 'Dis. + Met.' case we refine both `IsDis` and `IsMet` from the respective raw predictions, using only the `DM` rule (see Table 3.2) to coordinate disulfide and metal bonding states; the localization in this case is ignored. The experiment is designed to evaluate wheter combining only disulfide and metal predictions is actually useful in our dataset.

- In the 'Dis. + Loc.' case we refine `IsDis` from the raw disulfide predictions and the true localization, using the `DL1` and `DL2` rules.

- In the 'Dis. + Met. + Loc.' case we refine `IsDis` and `IsMet` making the refined disulfide bonding state interact with metals (using the `DM` rule), solvent accessibility (with the `DA` rule), and subcellular localization (with `DL1` and `DL2`.)

The results can be found in Table 3.3.

Three trends are apparent in the results. First of all, we find subcellular localization to have a very strong influence on disulfide bonding state, as expected. In particular, in the 'Dis. + Loc.' case, which includes no metal predictions, the accuracy and F_1 measure improves from 0.804 and 0.811 (raw) to 0.857 and 0.856 (refined), respectively. The change comes mainly from an increase in precision: the true subcellular localization helps reducing the number of false positives.

The interaction between metals and disulfide bonds is not as clear cut: in the 'Dis. + Met.' case, which includes no subcellular localization, the refined disulfide predictions slightly improve, in terms of F_1 measure, while the metal predictions slightly worsen. The latter case is mainly due to the drop in recall, from 0.827 to 0.739. This is to be expected, as the natural scarcity of metal residues makes the metal prediction task harder (as can be seen observing the differential behavior of accuracy and F_1 measure). As a consequence the confidence output by MetalDetector is lower than the confidence output by Disulfind. In other words, in the case of conflicting raw predictions, the disulfide predictions usually dominate the metal predictions.

Table 3.3: Results for True Sub. Loc.

Disulfide Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.804 ± 0.03	0.720 ± 0.06	0.917 ± 0.04	0.811 ± 0.04
Dis. + Met.	0.832 ± 0.04	0.767 ± 0.05	0.913 ± 0.04	0.833 ± 0.04
Dis. + Loc.	0.857 ± 0.03	0.801 ± 0.04	0.921 ± 0.03	0.856 ± 0.03
Dis. + Met. + Loc.	0.867 ± 0.03	0.819 ± 0.04	0.919 ± 0.03	0.865 ± 0.03
HMSVM	0.874 ± 0.03	0.884 ± 0.06	0.851 ± 0.03	0.866 ± 0.03
BRNN	0.892 ± 0.02	0.900 ± 0.03	0.863 ± 0.05	0.880 ± 0.03
Metal Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.952 ± 0.02	0.686 ± 0.09	0.827 ± 0.10	0.747 ± 0.09
Dis. + Met.	0.950 ± 0.02	0.711 ± 0.09	0.739 ± 0.14	0.713 ± 0.09
Dis. + Loc.	–	–	–	–
Dis. + Met. + Loc.	0.952 ± 0.02	0.709 ± 0.08	0.783 ± 0.11	0.736 ± 0.07
HMSVM	0.950 ± 0.02	0.741 ± 0.12	0.697 ± 0.08	0.711 ± 0.07
BRNN	0.948 ± 0.02	0.683 ± 0.09	0.763 ± 0.11	0.715 ± 0.07

Finally, in 'Dis. + Met. + Loc.' case, both disulfide and metal bonds improve using the true subcellular localization compared to the above settings. In particular, metal ligand prediction, while still slightly worse than the baseline (again, due to class unbalance, as mentioned above) sees a clear gain in recall (from 0.739 in the 'Dis. + Met.' case to 0.783). This is an effect of using localization: removing false disulfide positives leads to less spurious conflicts with the metals.

The two alternative pipelines behave similarly. They both manage to beat the Markov Logic Network on the easier of the two tasks, disulfide refinement, while performing worse on the metals. We note that the HMSVM and BRNN, contrary to our method, both have a chance to rebalance the raw metal predictions with respect to the disulfide predictions during the training stage, learning a distinct bias/weight for the inputs. Nevertheless, they still fail to improve upon our refined metals.

3.2.6 Predicted Subcellular Localization

This experiment is identical to the previous one, except we use predicted subcellular localization in place of the true one. Similarly to the previous section, we consider three sub-cases. In the 'Dis. + Loc.' case, we refine localization and disulfide bonding state, while in the 'Dis. + Met. + Loc.'

case we refine all three predicted features together. The results can be found in Table 3.4. The 'Dis. + Met.' case is reported as well for ease of comparison.

Here we can see how our architecture can really help with the mutual integration of protein features. In general, we notice that refined disulfide bonds are enhanced by the integration of localization, even if less so than in the previous experiment. At the same time, localization also benefits by the interaction with disulfide bonds, as can be seen in the 'Dis. + Loc.' case. The biggest gain is obtained for the ExtraCellular and Nucleus classes, which are also the most numerous classes in the dataset: several protein chains are moved back to their correct class. The introduction of metals improves directly disulfide bonds and indirectly localization, even though its effect is relatively minor.

On the downside, refined metal predictions worsen in all cases. This is due, again, to the unbalance of the small number of metal binding residues found in the data, and to the difference between the confidences output by Disulfind and MetalDetector.

Surprisingly, the alternative pipelines are not as affected by the worsening of the localization information: their performance is on par as with the true localization. This is in part explained by the simpler task the alternative methods carry out, as it does not involve refinement of the raw localization itself. It turns out that using predicted localization itself, the alternative methods manage to perform better than us also for metal refinement. In the following, we will show an improvement to our pipeline to address this issue.

3.2.7 Predicted Subcellular Localization with Predictor Reliability

The previous experiment shows that our refiner performs suboptimally on the metal refinement task due to class unbalance. A common way to alleviate this issue is to re-weight the classes according to some criterion. In our case, the positive metal residues are dominated by the negative ones, making the overall accuracy of MetalDetector higher than that of Disulfind. Our method naturally supports the re-weighting of predictors with different accuracy: the weight assigned to a `Pred` predicate can be strengthened or weakened depending on our estimate of the predictor accuracy.

To implement this strategy, we add an intermediate *proxy* predicate, weighted according to the actual predictor performance over the training set. The proxy predicate mediates the interaction between the raw prediction (the `Pred` predicate) and the refined prediction (the `Is` predicate). The actual proxy predicates are `ProxyLoc`, `ProxyDis` and `ProxyMet`, used by rules

Table 3.4: Results for Predicted Sub. Loc.

Disulfide Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.804 ± 0.03	0.720 ± 0.06	0.917 ± 0.04	0.811 ± 0.04
Dis. + Met.	0.832 ± 0.04	0.767 ± 0.05	0.913 ± 0.04	0.833 ± 0.04
Dis. + Loc.	0.809 ± 0.03	0.732 ± 0.06	0.923 ± 0.04	0.815 ± 0.04
Dis. + Met. + Loc.	0.843 ± 0.03	0.779 ± 0.04	0.919 ± 0.04	0.843 ± 0.03
HMSVM	0.882 ± 0.03	0.890 ± 0.05	0.856 ± 0.04	0.872 ± 0.04
BRNN	0.884 ± 0.03	0.895 ± 0.03	0.847 ± 0.05	0.870 ± 0.03
Metal Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.952 ± 0.02	0.686 ± 0.09	0.827 ± 0.10	0.747 ± 0.09
Dis. + Met.	0.950 ± 0.02	0.711 ± 0.09	0.739 ± 0.14	0.713 ± 0.09
Dis. + Loc.	–	–	–	–
Dis. + Met. + Loc.	0.949 ± 0.02	0.707 ± 0.09	0.731 ± 0.14	0.705 ± 0.08
HMSVM	0.952 ± 0.02	0.755 ± 0.07	0.707 ± 0.09	0.725 ± 0.06
BRNN	0.950 ± 0.02	0.694 ± 0.09	0.768 ± 0.11	0.723 ± 0.08

Subcellular Localization				
Raw predictions				
	Cytosol	ExtraCell.	Mitoch.	Nucleus
Cytosol	14	11	2	5
ExtraCell.	17	206	2	29
Mitoch.	12	8	6	8
Nucleus	46	67	15	78
Dis. + Loc.				
Cytosol	15	10	2	5
ExtraCell.	18	223	2	11
Mitoch.	12	223	2	8
Nucleus	48	55	16	87
Dis. + Met. + Loc.				
Cytosol	15	9	2	6
ExtraCell.	18	223	2	11
Mitoch.	12	6	8	8
Nucleus	48	44	21	93

Table 3.5: Results for True Sub. Loc. with Proxy.

Disulfide Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.804 ± 0.03	0.720 ± 0.06	0.917 ± 0.04	0.811 ± 0.04
Dis. + Met.	0.838 ± 0.03	0.776 ± 0.04	0.912 ± 0.04	0.838 ± 0.04
Dis. + Loc.	0.853 ± 0.03	0.796 ± 0.05	0.921 ± 0.03	0.853 ± 0.03
Dis. + Met. + Loc.	0.865 ± 0.03	0.817 ± 0.04	0.917 ± 0.03	0.863 ± 0.03
HMSVM	0.874 ± 0.03	0.884 ± 0.06	0.851 ± 0.03	0.866 ± 0.03
BRNN	0.892 ± 0.02	0.900 ± 0.03	0.863 ± 0.05	0.880 ± 0.03
Metal Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.952 ± 0.02	0.686 ± 0.09	0.827 ± 0.10	0.747 ± 0.09
Dis. + Met.	0.952 ± 0.02	0.696 ± 0.08	0.795 ± 0.11	0.739 ± 0.08
Dis. + Loc.	–	–	–	–
Dis. + Met. + Loc.	0.952 ± 0.02	0.695 ± 0.08	0.807 ± 0.10	0.743 ± 0.08
HMSVM	0.950 ± 0.02	0.741 ± 0.12	0.697 ± 0.08	0.711 ± 0.07
BRNN	0.948 ± 0.02	0.683 ± 0.09	0.763 ± 0.11	0.715 ± 0.07

`I1P` to `I3P`, and `PX1` to `PX3`. See Tables 3.1 and 3.2 for the details. The results can be found in Table 3.6. For completeness, we also include the proxy results for true subcellular localization in Table 3.5.

The proxy helps the MLN refiner: the refined metal predictions are on-par with the raw ones, while at the same time improving the disulfide bonds. The effects are especially clear when comparing the 'Dis. + Met.' cases of Tables 3.3 (true localization, no proxy) and 3.5 (true localization, with proxy), with F_1 scores changing from 0.833 and 0.713 for bridges and metals, respectively, to 0.838 and 0.739. We note that our method is the only one able to recover the same performance as MetalDetector while also improving the other two refined features. On the contrary, the alternative pipelines tend to favor one task (disulfide bridges) over the other, and fail in all cases to replicate the baseline performance.

The down-side is that localization refinement is slightly worse: the raw Nucleus predictions are less accurate than the Cytosol ones, leading to the Cytosol being assigned a higher proxy weight. Since both compartments prevent disulfide bonds, the MLN refiner tends to assign chains with no half cysteines to the latter.

Table 3.6: Results for Predicted Sub. Loc. with Proxy.

Disulfide Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.804 ± 0.03	0.720 ± 0.06	0.917 ± 0.04	0.811 ± 0.04
Dis. + Met.	0.838 ± 0.03	0.776 ± 0.04	0.912 ± 0.04	0.838 ± 0.04
Dis. + Loc.	0.803 ± 0.03	0.727 ± 0.05	0.922 ± 0.04	0.811 ± 0.04
Dis. + Met. + Loc.	0.846 ± 0.03	0.784 ± 0.04	0.918 ± 0.04	0.845 ± 0.04
HMSVM	0.882 ± 0.03	0.890 ± 0.05	0.856 ± 0.04	0.872 ± 0.04
BRNN	0.884 ± 0.03	0.895 ± 0.03	0.847 ± 0.05	0.870 ± 0.03
Metal Bonding State				
Experiment	Q	P	R	F_1
Raw predictions	0.952 ± 0.02	0.686 ± 0.09	0.827 ± 0.10	0.747 ± 0.09
Dis. + Met.	0.952 ± 0.02	0.696 ± 0.08	0.795 ± 0.11	0.739 ± 0.08
Dis. + Loc.	–	–	–	–
Dis. + Met. + Loc.	0.952 ± 0.02	0.706 ± 0.08	0.782 ± 0.10	0.735 ± 0.06
HMSVM	0.952 ± 0.02	0.755 ± 0.07	0.707 ± 0.09	0.725 ± 0.06
BRNN	0.950 ± 0.02	0.694 ± 0.09	0.768 ± 0.11	0.723 ± 0.08

Subcellular Localization				
Raw predictions				
	Cytosol	ExtraCell.	Mitoch.	Nucleus
Cytosol	14	11	2	5
ExtraCell.	17	206	2	29
Mitoch.	12	8	6	8
Nucleus	46	67	15	78
Dis. + Loc.				
Cytosol	13	13	2	4
ExtraCell.	22	224	0	8
Mitoch.	12	6	8	8
Nucleus	61	56	14	75
Dis. + Met. + Loc.				
Cytosol	14	12	2	4
ExtraCell.	22	224	0	8
Mitoch.	12	6	8	8
Nucleus	67	41	17	81

3.2.8 Conclusions

In this chapter we introduced a novel framework for the joint integration and refinement of multiple related protein features. The method works by resolving conflicts with respect to a set of user-provided, biologically motivated constraints relating the various features. The underlying inference engine, implemented as a grounding-specific Markov Logic Network [3], allows to perform probabilistic reasoning over rich first-order logic rules. The designer has complete control over the refinement procedure, while the inference engine accounts for potential data noise and rule fallacy.

As an example, we demonstrate the usefulness of our framework on three distinct predicted features: subcellular localization, disulfide bonding state, metal bonding state. Our refiner is able to improve the predictions by removing violations to the constraints, leading to more consistent results. In particular, we found that subcellular localization plays a central role in determining the state of potential disulfide bridges, confirming the observations of Savojardo *et al.* [185]. Our method however also allows to improve subcellular localization in the process, helping to discriminate between chains residing in reducing and oxydizing cellular compartments, especially nuclear and secreted chains. We also found that disulfide predictions benefit from metal bonding information, although to a lesser extent, especially when used in conjunction with localization predictions. On the other hand metals, which are in direct competition with the more abundant disulfide bonds, are harder to refine. We presented a simple and natural re-weighting strategy to alleviate this issue. The task would be further helped by better localization predictions, which tend to improve the distribution of disulfide bridges, as shown by the experiments with true subcellular localization.

We compared our refinement pipeline with two alternatives based on state-of-the-art sequential prediction methods, Hidden Markov Support Vector Machines and Bidirectional Recursive Neural Networks. These methods have two fundamental advantages: they are run through a full-blown training procedure, and are only asked to refine the two sequential features, a task for which they are highly specialized. However, the results show that they tend to favor the easier task (disulfide bridges) over the other, struggling to achieve the same results of the baseline on the harder task (metals). On the contrary, our method is more general, and does not favor one task at the expense of the others.

Our framework is designed to be very general, with the goal of refining arbitrary sets of existing predictors for correlated features, such as Distill [22] and PredictProtein [23], for which re-training is difficult or infeasible. As a consequence, our framework does not require any change to the underlying

predictors themselves, only requiring that they provide an estimated reliability for their predictions.

3.3 Methods

3.3.1 Predictors

Disulfind [5] is a web server for the prediction of disulfide bonding state and binding geometry from sequence alone. Like other tools for the same problem, Disulfind splits the task in two simpler sub-problems as follows. First an SVM binary classifier is employed to independently infer the bonding state of each cysteine in the input chain. The SVM is provided with both local and global information. Local information includes a window of position-specific conservations derived from multiple alignment, centered around each target cysteine. Global information represent global features of the whole chain, such as length, amino acid composition, and average cysteine conservation. Then a bidirectional recursive neural network (BRNN) is used to collectively refine the possibly incorrect SVM predictions, assigning a revised binding probability to each cysteine. Finally, the predictions are post-processed with a simple finite-state automaton to enforce an even number of positive disulfide bonds. For the technical details, see Vullo *et al.* [198].

MetalDetector [191] is a metal bonding state classifier, whose architecture is very similar to Disulfind. It is split in two stages, an SVM classifier for local, independent per-residue

Loctree [4] is a multiclass subcellular localization predictor based on a binary decision tree of SVM nodes. The topology of the tree mimics the biological structure of the cellular protein sorting system. It is designed to predict the subcellular localization of proteins given only their sequence, and uses multiple input features: a multiple alignment step is performed against a local, reduced redundancy database of UniProt proteins, and makes use of a stripped, specially tailored version of Gene Ontology vocabulary terms to improve its performance. It also uses psort 3.0 [199]. The predictor incorporates three distinct topologies, one for each of the considered kingdoms: prokaryotes, eukariotic plants (viridiplantae), eukariotic non-plants (metazoa).

3.3.2 Markov Logic Networks

A Markov Logic network (MLN) [171, 149], is a Statistical Relational Learning method (see Chapter 2 to define a probability distribution over all *possible*

worlds (truth assignments) of a set of formulae allowing to perform reasoning over possibly wrong or conflicting facts.

A MLN consists of a *finite* domain of objects (constants) $\mathtt{C}$ and a knowledge base $\mathtt{KB}$ of logical rules. Each formula $\mathtt{F}_i$ in $\mathtt{KB}$ is associated a real-valued weight w_i, representing the confidence we have in that rule. Weights close to zero mean that the formula is very uncertain, while larger weights mean that it is likely to hold (if positive) or not (if negative). Contrarily to pure FOL, in Markov Logic the formulae in the $\mathtt{KB}$ are explicitly fallible; as a consequence, Markov Logic admits interpretations that don't satisfy all the constraints.

Instantiating all the formulae in $\mathtt{KB}$ using all possible combinations of constants in $\mathtt{C}$ leads to a *grounding* of the knowledge base. As an example, if $\mathtt{C}$ consists of three objects, a protein $\mathtt{P}$ and two cysteines at position $\mathtt{4}$ and $\mathtt{19}$, and the knowledge base consists of the formula $\mathtt{DM = !(IsDis(p,n) \wedge IsMet(p,n))}$, then the grounding will be the set of ground formulae: $\{\mathtt{DM(P,4)}, \mathtt{DM(P,19)}\}$. A *possible world* is a truth assignment of the grounding of $\mathtt{KB}$. Markov Logic defines a way to assign to each possible world a probability, determined by the weight of the formulas that it satisfies.

A MLN defines a joint probability distribution over the set of interpretations (i.e. truth assignments) of the grounding of $\mathtt{KB}$. In the previous example, if the formula $\mathtt{DM}$ has a positive weight, then the assignment $\mathtt{DM(P,4)} \wedge \mathtt{DM(P,19)}$ will be the most likely, while $\mathtt{!DM(P,4)} \wedge \mathtt{!DM(P,19)}$ will be the least likely, with the other possible worlds standing in between. In addition, if an assignment satisfies a formula with a negative weight, it becomes less likely.

Given a set of ground atoms $\mathtt{x}$ of known state, and a set of atoms $\mathtt{y}$ whose state we want to determine, we can define the conditional distribution generated by a MLN as follows:

$$p(\mathtt{y} \,|\, \mathtt{x}\,;\, w) = \frac{1}{Z(\mathtt{x})} \exp \sum_{\mathtt{F}_i \in \mathtt{KB}} w_i n_i(\mathtt{x}, \mathtt{y})$$

Here $n_i(\mathtt{x}, \mathtt{y})$ counts how many times the formula $\mathtt{F}_i$ is satisfied by groundings of world $(\mathtt{x}, \mathtt{y})$, and $Z(\mathtt{x})$ is a normalization term. In other words, the above formula says that the probability of $\mathtt{y}$ being in a given state is proportional to the weighted number of formulae in $\mathtt{KB}$ that the interpretation $(\mathtt{x}, \mathtt{y})$ satisfies. We can query a MLN for the most likely state of the unknown predicates $\mathtt{y}$ from the known facts $\mathtt{x}$ by taking the truth assignment of $\mathtt{y}$ that maximizes the above conditional probability.

Maximum a-posteriori (MAP) inference, i.e. the task of finding the most likely interpretation for query predicates $\mathtt{y}$ given a set of evidence predicates $\mathtt{x}$, involves finding the interpretation of the variables $\mathtt{y}$ that maximizes the

weighted sum of the constraints. Even though MLNs can be defined over non-finite FOL [200], in practice the dataset and knowledge bases are always finite. Consequently, MAP inference is equivalent to a MAX-SAT problem over the grounded network. As we have already seen in Chapter 2, MAX-SAT is NP-complete, rendering exact inference intractable. There are however a number of heuristic methods [201] which can quickly find reasonable approximate solutions. Given a MAX-SAT solution, estimating the probabilities of the output can be performed approximately by Gibbs sampling [149].

A very promising alternative to brute-force MAX-SAT is *lifted inference*, which has recently gained momentum in the SRL field. Lifted inference exploits the symmetries imposed by the high-level relational description of the model (in the MLN case, the formulae) by working on equivalence classes of objects rather than on individual instances. Lifted inference has been developed for MLNs in, e.g. [202, 203, 204]. For a short introduction to lifted inference for probabilistic and statistical-relational models, see [150].

Markov Logic can be trained either as a generative model [149] with a canonical log-likelihood maximization procedure (see Chapter 2) or as a discriminative model. Discriminative techniques has attained by either maximizing the *conditional* log-likelihood [149], using a voted perceptron analogue [205], and by max-margin methods [206].

An issue with standard Markov Logic is that distinct groundings of the same formula $\mathtt{F}_i$ are assigned the same weight $\mathtt{w}_i$. This is not the case for our raw predictions, which are specific for each protein (e.g. subcellular localization) or each residue within a protein (e.g. metal or disulfide bonding state).

To overcome this issue, we make use of grounding-specific Markov Logic Networks (gs-MLN), introduced in Lippi *et al.* [3], an extension that adds the ability of specifying per-grounding weights. The idea is to substitute the fixed per-formula weight $\mathtt{w}$ with a new function ω that depends on the particular grounding. The conditional distribution is modified to be of the form:

$$p(y|x;\theta) = \frac{1}{Z(x)} \exp \sum_{\mathtt{F}_i \in \mathtt{KB}} \sum_{\mathtt{g} \in \mathtt{G}(\mathtt{F}_i)} \omega(\mathtt{g}, \theta_i) n_{ij}(x, y)$$

Here the variable $\mathtt{g}$ ranges over all satisfied groundings of formula $\mathtt{F}_i$, and the function ω evaluates the weight of the given grounding $\mathtt{g}$ according to a set of per-formula parameters θ_i.

Chapter 4

Multi-level Protein Interaction Prediction

4.1 Background

Physical interactions between proteins are the workhorse of cell life and development [64], and play an extremely important role both in the mechanisms of disease [65] and in the design of new drugs [66]. In recent years, there has been enormous interest in reverse engineering the protein–protein interaction (PPI) networks of several species, particularly due to the availability of high-throughput experimental techniques, leading to an abundance of large databases on all aspects of PPIs [80].

Notwithstanding the increased availability of interaction data, the natural question of whether two arbitrary proteins interact, and why, is still open. The growing literature on protein interaction prediction [79, 80, 81] is symptomatic of the gap separating the amount of available data and the effective size of the interaction network [82]. The present chapter is a contribution towards filling this gap.

Our work is based on the observation that physical interactions can be viewed at three levels of detail. At a higher level, two proteins interact to perform some function within a biological pathway (e.g. metabolism, signaling, regulation, *etc.*) [67]. At a lower level, the same interaction occurs between a pair of specific *domains* appearing in the proteins; the types of the domains involved characterize the functional semantics of the interaction [70]. At the lowest level, the interaction is instantiated by the binding of a pair of protein *interfaces*, patches of solvent accessible residues with compatible shapes and chemical properties [71]. The low-level features of the binding sites determine whether the interaction is transient or permanent, whether two proteins com-

pete for interaction with a third one, *etc.* Figure 4.1 illustrates the multi-level mechanisms with an example taken from the PDB.

Despite the significance of low-level details in elucidating the mechanics of protein–protein interactions, most of the current experimental data comes from high-throughput screening techniques, such as yeast two-hybrid (Y2H) assays [207]. These techniques do *not* provide information on domain- or residue-level interactions, which require solving the three-dimensional structure of each protein-protein complex, an expensive and time consuming task addressed by X-Ray crystallography, NMR, or electron microscopy techniques [208]. As a consequence, protein–protein interaction data is under-characterized at the domain and residue levels: the current databases are relatively lacking when compared to the magnitude of the existing body of data about protein-level interactions [76]. At the time of writing, the PDB hosts 84,418 structures, but merely 4,210 resolved complexes (according to `http://www.rcsb.org/pdb/statistics/holdings.do`, retrieved on 2013/06/20). The latter cover only a tiny fraction of the interactions stored in databases such as BioGRID and MIPS.

From a purely biological perspective, predictions at different levels have several important applications. The network topology and individual features of protein interactions are an essential component of a wide range of biological tasks: inferring protein function [61] and localization [72], reconstructing signal and metabolic pathways [73], discovering candidate targets for drug development [65]. Finer granularity predictions at the domain level allow to discover affinities between domain types that can be carried over to other proteins [74, 75]; domain–domain networks have also been assessed as being typically more reliable than their protein counterparts [76]. Finally, residue-level predictions, i.e., interface recognition, enable the detailed study

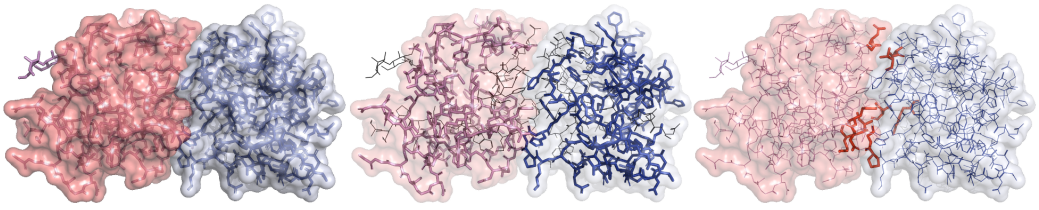


Figure 4.1: Two bound proteins and their interacting domains and residues, captured in PDB complex 4IOP. The proteins are a Killer cell lectin-like receptor (in violet) and its partner, a C-type lectin domain protein (in blue). Left: interaction as visible from the contact surface. Center: the two C-type lectin domains instantiating the interaction. Right: effectively interacting residues in red.

of the principles of protein interactions, and are crucial for tasks such as rational drug design [66], metabolic reconstruction and engineering [77], and identification of hot-spots [78] in the absence of structure information.

Given the usefulness of knowing the details of protein–protein interactions at diverse levels of detail, in this chapter we address the problem of collectively predicting the binding state of all proteins, domains, and residues in a network. We call this task the *multi-level protein interaction prediction* problem (MLPIP for short).

From a computational point of view, the most important feature of the multi-level prediction problem is its inherently relational nature. Proteins, domains and residues are organized in a hierarchy, which dictates constraints on the binding state of pairs of objects at the different levels, as follows. On the one hand, whenever two proteins are bound, at least two of their domains must also be bound, and, similarly, there must be residues in the two domains that form an interface. On the other hand, if no residues of the two proteins interact, neither do their domains, nor the proteins themselves. In other words, predictions at different levels must be *consistent*.

In this chapter we cast the multi-level prediction problem as a statistical-relational learning task, leveraging the latest developments in the field. Our prediction method is based on Semantic Based Regularization [7], an elegant semi-supervised prediction framework that caters both the effectiveness of kernel machines and the expressivity of First-Order Logic (FOL). The constraints described above are encoded as FOL rules, which are used to enforce consistent predictions at all levels of the interaction hierarchy. By computing multi-level predictions, our method can not only infer which protein pairs are likely to interact, but also provide details about how the interactions take place. Our empirical evaluation shows the effectiveness of this constraint-based approach in boosting predictive performance, achieving substantial improvements over both an unconstrained baseline and the only existing alternative MLPIP method [10].

4.1.1 Problem definition

PPI networks are most naturally formalized as graphs, where nodes represent proteins and edges represent interactions. Given a set of features describing the properties of the proteins in the network (e.g. primary structure, localization, tertiary structure —when available—, *etc.*), inferring the PPI network structure amounts to determining those pairs of proteins that are likely to interact. This task is often cast as a pairwise classification problem, where a binary classifier takes as input a pair of proteins (or rather their feature-based representations) and predicts whether they interact or not. Standard

binary classification methods, such as Support Vector Machines [14], can be used to implement the pairwise classifier. In this setting, the interaction depends only on the features of the two incident nodes, and is independent of all other nodes. Interactions between domains or residues can be predicted similarly.

The most straightforward way to address the MLPIP problem is to cast the three interaction prediction problems, for proteins, domains and residues respectively, as independent pairwise classification tasks. However, as previously discussed, these problems are clearly strongly related: two proteins interact via one or more domains, which in turn contain patches of residues that constitute the interaction surface. Ignoring these relationships can lead to heavily suboptimal, inconsistent predictions, where, e.g. two proteins are predicted to interact but none of their domains are predicted to be involved in this interaction. Making these relationships explicit and forcing predictors to satisfy consistency constraints is the key contribution of this work. In the machine learning community, this kind of scenario characterized by multiple related prediction tasks is usually cast as a statistical-relational learning problem [1, 2], where the goal is to *collectively* classify the state of all objects of interest, taking into account the relations existing between them. The solution we adopt is grounded in this learning framework.

4.1.2 Overview of the proposed method

In this chapter we propose solving the multi-level prediction problem adapting a state-of-the-art statistical-relational learning framework, namely Semantic Based Regularization (SBR) [7]. SBR ties multiple learning tasks, which are themselves addressed by kernel machines, using constraints expressing First-Order Logic knowledge. In the following we give an overview of the SBR framework, also pictured in Figure 4.2; see Methods for further details.

Let $\mathcal{X}$ be a set of objects. In most scenarios, objects are *typed*, so that objects of the same type can be considered as belonging to the same group. In our setting, object types are proteins, domains and residues, with corresponding sets $\mathcal{X}_P, \mathcal{X}_D$ and $\mathcal{X}_R$ respectively. Predicates represent properties of objects or relationships between them. Depending on the scenario, some predicates are always known (called *given* predicates), some other are known only for a subset of the objects, and their value should be predicted when unknown (*query* or *target* predicates). The $\texttt{parendpd}(\texttt{p},\texttt{d})$ predicate, for instance, specifies that domain $\texttt{d} \in \mathcal{X}_D$ is part of protein $\texttt{p} \in \mathcal{X}_P$, i.e. the predicate is true for all $(\texttt{p},\texttt{d})$ pairs for which $\texttt{d}$ is a domain of $\texttt{p}$, and false otherwise. The value of this predicate is known for all objects in our domain;

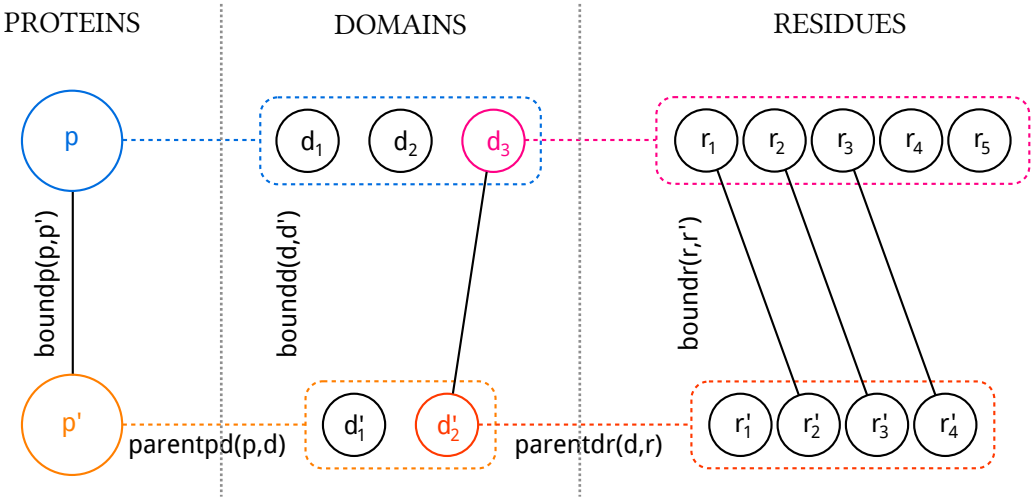


Figure 4.2: Visualization of the proposed method applied to a pair of proteins $\mathtt{p}$ and $\mathtt{p}'$ and their parts. Circles represent proteins, domains and residues. Dotted lines indicate a parent-child relationship between objects, representing the `parentpd` and `parentdr` predicates. Solid lines link pairs of bound objects, i.e. objects for which the `boundp`, `boundd` or `boundr` predicates are true.

note that there indeed are many proteins whose domains are unknown, but in this case there is no corresponding domain object in our data). The $\mathtt{boundp}(\mathtt{p}, \mathtt{p}')$ predicate specifies whether two proteins $\mathtt{p}$ and $\mathtt{p}'$ are interacting. This is one of the target predicates, whose truth value should be predicted for novel protein-protein pairs. Similar predicates are defined for domain and residue level bindings. Target predicates are modelled as binary classifiers, i.e. functions trained to predict the truth value of the predicate. Relationships between predicates can be introduced in order to enforce constraints known to hold in the domain. SBR allows to exploit the full power of First-Order Logic in doing this. As a matter of example, the notion that two interacting proteins should have at least one interacting domain can be modelled as (see Methods for details on First-Order Logic notation):

$$\begin{aligned} \forall\,(\mathtt{p},\mathtt{p}')\mathtt{boundp}(\mathtt{p},\mathtt{p}') \;\Rightarrow\; & \exists\,(\mathtt{d},\mathtt{d}')\;\mathtt{boundd}(\mathtt{d},'\mathtt{d})\;\wedge \\ & \mathtt{parentpd}(\mathtt{p},\mathtt{d})\;\wedge \\ & \mathtt{parentpd}(\mathtt{p}',\mathtt{d}') \end{aligned}$$

Each binary classifier is implemented in the SBR framework as a kernel machine [105]. The key component of kernel machines is the kernel function, which measures the similarity between objects in terms of their representations. A protein, for instance, can be represented as the sequence of its

residues, plus additional information as its subcellular localization and/or its phylogenetic profile. Having the same subcellular localization, for instance, should increase the similarity between two proteins, as having a similar amino acid composition. Designing appropriate kernels is a crucial component of a successful predictor. A kernel machine is a function which predicts a certain property of an object x in terms of a weighted sum of similarities to other objects for which the property is known, i.e.:

$$f(x) = \sum_i w_i K(x, x_i)$$

A kernel machine could for instance predict whether a protein is an enzyme or not (binary classification), in terms of weighted similarity to other proteins. Being similar to an enzyme x_i will drive the prediction towards the positive (enzyme) class (positive weight w_i), while being similar to a non-enzyme x_j will drive the prediction towards the opposite class (negative weight w_j).

In the interaction prediction setting, target predicates actually predict properties of *pairs* of objects (proteins, domains or residues). We thus employ a pairwise kernel machine classifier to model the target predicate:

$$f(x, x') = \sum_i w_i K((x, x'), (x_i, x_i'))$$

Here the kernel function measures the similarity between two pairs of objects, so that, e.g. two proteins will be predicted as interacting if they are similar to protein pairs which are known to interact, and dissimilar from pairs known to not interact.

Given a kernel between objects $K(x, x')$, it is possible to construct a pairwise kernel by means of a the following transformation [209]:

$$\begin{aligned} K((x_i, x_j), (x_k, x_l)) = {} & K(x_i, x_k) \cdot K(x_i, x_l) + \\ & K(x_j, x_k) \cdot K(x_j, x_l) \end{aligned} \tag{4.1}$$

This transformation guarantees that, if the input function K is a valid kernel, so is the resulting pairwise function.

As already explained, in SBR each target predicate is implemented as a kernel machine, and the state of a predicate for an uncharacterized pair of proteins can be inferred by querying the machine. Positive predictions correspond to `true` predicates, i.e. bound protein pairs, and negative predictions to `false` ones. The confidence of the kernel machine, also called *margin*, embodies the confidence in the state of the predicate, that is, how strongly two proteins are believed to interact (or not). Given the output of the kernel machines for all target predicates, SBR uses the First-Order Logic

rules to condition the state of the correlated predicates. It does so by first translating the FOL rules into continuous constraints, which we discuss more thoroughly in Methods. The variables coming into play into the continuous constraints are the confidences of all target predicates (and the state of all given predicates) appearing in the equivalent FOL constraint. The amount of violation is reflected by the value of the continuous constraints: if the predicted predicates satisfy a FOL rule, the corresponding constraint will have a value equal to 1; on the other hand, the closer the constraint value to zero, the more the FOL rule is violated.

SBR computes a solution to the inference problem, i.e. deciding the truth value of all target predicates, that maximizes both the confidence of individual predicates and the amount of satisfaction of all constraints. Informally, the optimal assignment to all predicates, i.e. the binding state of protein, domain and residue pairs, y^*, is a solution to the following optimization problem:

$$y^* = \arg\max_y \text{consist}(y, f) + \text{consist}(y, KB)$$

where the first term accounts for consistency between inferred truth values and confidence of the individual predictions, and the second incorporates information on the degree of satisfaction of the constraints build from the FOL knowledge. Contrarily to standard kernel methods, this optimization problem is non-convex. This is commonly the case for complex statistical-relational learning tasks [1], and implies that we are restricted to finding local optima. We will see in the Results section that, in practice, this does not compromise the quality of the solutions.

SBR is a semi-supervised method [99], meaning that the set of target proteins is given beforehand and can be exploited during the learning stage to fine-tune the model. Semi-supervised learning is known to enhance the prediction ability when appropriately used [210], and can be applied very naturally to PPI prediction, as the full set of proteins is always known.

To summarize, at each level the state of an uncharacterized pair of objects, e.g. proteins p and p', is mainly inferred by the similarity of the pair (p, p') to other pairs that are known to interact or not, through the pairwise kernel function K and the learned weights $\boldsymbol{w}$. Thus the kernel allows to propagate information *horizontally* within the same level. At the same time, the FOL constraints allow to propagate information *vertically* between the levels, by keeping the interaction pattern along the protein-domain-residue hierarchy consistent.

Table 4.1: List of SBR predicates.

target predicates	
boundp(p, p′)	true iff the protein pair (p, p′) is bound
boundd(d, d′)	true iff the domain pair (d, d′) is bound
boundr(r, r′)	true iff the residue pair (r, r′) is bound
given predicates	
parendpd(p, d)	true iff protein p is parent of domain d
parenddr(d, r)	true iff domain d is parent of residue r
parendpr(p, r)	true iff protein p is parent of residue r
hasdom(p)	true iff protein p has at least one domain
hasres(d)	true iff domain d has at least one residue

4.1.3 Modeling multi-level interactions

As already explained, we use two distinct kinds of predicates: given predicates and target predicates. Given predicates encode *a priori* knowledge about the problem, in our case the structure of the multi-level object hierarchy. In particular, given a protein p and a domain d, the parendpd(p, d) predicate is true if and only if domain d occurs in protein p; the parentdr predicate is the analogous for domains and residues. This simple representation suffices to encode the whole protein–domain–residue hierarchy. To simplify the notation, we also introduce the hasdom(p) predicate to encode the fact that protein p has at least one domain. More formally:

$$\texttt{hasdom}(\texttt{p}) := \exists\ \texttt{d}\ \texttt{parendpd}(\texttt{p}, \texttt{d})$$

The hasdom predicate can be computed directly by SBR using the above definition; we instead pre-compute its value for all protein pairs for run-time efficiency.

The boundp(p, p′) *target* predicate models the binding state of two distinct proteins. Its state is known for certain protein pairs, i.e. those in the training set, and our goal is to predict its state on the remaining ones. The boundd(d, d′) predicate plays the same role for domains. For a complete list of predicates, see Table 4.1.

In what follows we describe how to design inter-level FOL constraints to properly enforce consistency between predictions at different levels. We focus on modeling the constraints tying proteins and domains; it is easy to see that the ones between domains and residues can be modelled similarly (with one peculiar exception that will be pointed out later). Table 4.2 reports the complete list of rules.

Table 4.2: List of SBR rules.

Name	Definition
P→D	$\forall\,(\mathtt{p},\mathtt{p}')\ \mathtt{hasdom}(\mathtt{p})\ \wedge\ \mathtt{hasdom}(\mathtt{p}')\ \Rightarrow$ $\mathtt{boundp}(\mathtt{p},\mathtt{p}')\ \Rightarrow\ \exists\,(\mathtt{d},\mathtt{d}')\ \mathtt{boundd}(\mathtt{d},\mathtt{d}')\ \wedge\ \mathtt{parendpd}(\mathtt{p},\mathtt{d})\ \wedge\ \mathtt{parendpd}(\mathtt{p}',\mathtt{d}')$
D→P	$\forall\,(\mathtt{p},\mathtt{p}')\ \exists\,(\mathtt{d},\mathtt{d}')$ $\mathtt{boundd}(\mathtt{d},\mathtt{d}')\ \wedge\ \mathtt{parendpd}(\mathtt{p},\mathtt{d})\ \wedge\ \mathtt{parendpd}(\mathtt{p}',\mathtt{d}')\ \Rightarrow\ \mathtt{boundp}(\mathtt{p},\mathtt{p}')$
D→R	$\forall\,(\mathtt{d},\mathtt{d}')\ \mathtt{hasres}(\mathtt{d})\ \wedge\ \mathtt{hasres}(\mathtt{d}')\ \Rightarrow$ $\mathtt{boundd}(\mathtt{d},\mathtt{d}')\ \Rightarrow\ \exists_n\,(\mathtt{r},\mathtt{r}')\ \mathtt{boundr}(\mathtt{r},\mathtt{r}')\ \wedge\ \mathtt{parenddr}(\mathtt{d},\mathtt{r})\ \wedge\ \mathtt{parenddr}(\mathtt{d}',\mathtt{r}')$
R→D	$\forall\,(\mathtt{d},\mathtt{d}')\ \exists\,(\mathtt{r},\mathtt{r}')$ $\mathtt{boundr}(\mathtt{r},\mathtt{r}')\ \wedge\ \mathtt{parenddr}(\mathtt{d},\mathtt{r})\ \wedge\ \mathtt{parenddr}(\mathtt{d}',\mathtt{r}')\ \Rightarrow\ \mathtt{boundd}(\mathtt{d},\mathtt{d}')$
P→R	same as D→R, with proteins in place of domains
R→P	same as R→D, with proteins in place of domains

Inter-level constraints can be seen as propagating information from the upper layer to the lower one and in the opposite direction. To model this mechanism, we use two distinct constraints: the P→D rule and the D→P rule. A simplified version of the P→D rule is:

$$\begin{aligned}\forall\,(\mathtt{p},\mathtt{p}')\ \mathtt{boundp}(\mathtt{p},\mathtt{p}')\ \ \Rightarrow\ \ & \exists(\mathtt{d},\mathtt{d}')\ \mathtt{boundd}(\mathtt{d},\mathtt{d}')\wedge\\ & \mathtt{parendpd}(\mathtt{p},\mathtt{d})\ \wedge\\ & \mathtt{parendpd}(\mathtt{p}',\mathtt{d}')\end{aligned}$$

Intuitively, the rule means that whenever two proteins are bound (and therefore the left-hand side (LHS) of the implication is true) then there must be *at least one* pair of child domains that are bound (the right-hand side (RHS) is true). In classical First-Order Logic the rule would require that, whenever none of the child domains is bound (the RHS is false), then the parent proteins must not be bound (the LHS is false).

Note that, in the above formulation, the rule is applied indiscriminately to *all* protein pairs, even to those that have no known child domains in the considered dataset. Therefore, the rule can be reformulated in order to enforce it only for those protein pairs that do in fact have child domains, using the hasdom predicate, as follows:

$$\begin{aligned}\forall\,(\mathtt{p},\mathtt{p}')\ \ \mathtt{hasdom}(\mathtt{p})\ \ \wedge\ \ & \mathtt{hasdom}(\mathtt{p}')\Rightarrow\\ \Big(\mathtt{boundp}(\mathtt{p},\mathtt{p}')\ \ \Rightarrow\ \ & \exists(\mathtt{d},\mathtt{d}')\ \mathtt{boundd}(\mathtt{d},\mathtt{d}')\wedge\\ & \mathtt{parendpd}(\mathtt{p},\mathtt{d})\wedge\\ & \mathtt{parendpd}(\mathtt{p}',\mathtt{d}')\Big)\end{aligned}$$

This is the complete P→D rule. The left-hand side is always false for proteins without domains, making the rule always satisfied in this case (effectively

disabling the effect of the rule on the learning process). We define the complementary D→P rule as follows:

$$\begin{aligned}\forall(\mathtt{p},\mathtt{p}') \quad & \Big(\exists(\mathtt{d},\mathtt{d}')\ \mathtt{boundd}(\mathtt{d},\mathtt{d}')\ \wedge \\ & \mathtt{parendpd}(\mathtt{p},\mathtt{d})\ \wedge \\ & \mathtt{parendpd}(\mathtt{p}',\mathtt{d}') \\ & \Rightarrow\ \mathtt{boundp}(\mathtt{p},\mathtt{p}')\Big)\end{aligned}$$

This rule is applied to all protein pairs, demanding that if there is a pair of bound children domains then the proteins must be bound too, and vice versa that if the parent proteins are unbound so are the domains. The P→D and D→P rules could be merged into a single equivalent rule using the double implication (⇔). However, the rules have been considered separately to keep their effects on the results separated and easier to analyze.

To simulate the unidirectional information propagation between levels, as done by Yip *et al.* [10] (see Related Work), we modified how SBR converts logic implications by using the t-norm residuum, which states that a logic implication is true if the RHS is at least as true as the LHS. This modification also removes a bias in the translation of the implication that was affecting the original formulation of SBR, whose effect is to often move the LHS toward the false value. See Methods for details.

The constraints for domains and residues can be similarly defined with one important exception. The P→D rule described above (correctly) requires *at least one* domain couple to be bound for each interacting protein pair. However, when two domains are bound, the interaction interface involves more than one residue pair: for instance, binding sites collected in the protein–protein docking benchmark version 3.0 [211] consist of 25 residues on average [212]. We integrate this observation in the D→R rule using the n-existential operator $\exists_n$ in place of the regular existential (see Table 4.2 for the complete formulation), so that whenever two domains are bound, *at least* n pairs of their residues must be bound. Since interfaces in the employed dataset are typically 5 residues long, $n = 5$ has been used in the experiments. Our results demonstrate that this seemingly small modification has a rather extensive impact on the prediction of domain and residue level interactions.

4.1.4 Related work

In this section we briefly summarize previous PPI interaction prediction approaches using methods that are most closely related to the present chapter: kernel methods, semi-supervised methods, and logic-based methods. For a

broader exposition of interaction prediction methods, please refer to one of the several surveys on the subject [213, 80, 81, 70].

The earliest attempt to employ kernel methods [105] for PPI prediction is the work of Bock *et al.* [214], which casts interaction prediction as pairwise classification, using amino-acid composition and physico-chemical properties alone. Ben-Hur *et al.* [209] extended previous work by applying pairwise kernels and combining multiple data sources (primary sequence, Pfam domains, Gene Ontology annotations and interactions between orthologues). Successive works focused primarily into aggregating more diverse sources, including phylogenetic profiles, genetic interactions, and subcellular localization and function [81]. Kernel machines have also been applied to the prediction of binding sites from sequence, as resumed in [71]. The appeal of supervised kernel methods is that they provide a proved and theoretically grounded set of techniques that can easily integrate various information sources, and can naturally handle noise in the data. However, they have two inherent limitations: (i) the binding state of two proteins is inferred independently from the state of all other proteins, and (ii) due to their supervised nature, they do not take advantage of unsupervised data, which is very abundant in the biological network setting.

Semi-supervised learning (SSL) techniques [210, 99] attempt to solve these issues. In the SSL setting the set of target proteins is known in advance, meaning that the learning algorithm has access to their distribution in feature space. This way the inference task can be simplified by introducing unsupervised constraints that assign the same label to proteins that are, e.g., close enough in feature space, or linked in the interaction network, actuating a form of information propagation. There are several works in the PPI literature that embed the known network topology using SSL constraints. Qi *et al.* [215] employ SSL methods to the special case of viral-host protein interactions, where supervised examples are extremely scarce. Using similar methods, You *et al.* [216] attempt to detect spurious interactions in a known network by projecting it on a low-dimensional manifold. Other studies [217, 218] applied SSL techniques to the closely related problems of gene–protein and drug–protein interaction prediction. Despite the ability of SSL to integrate topology information, no study so far has applied it to highly relational problems such as the MLPIP.

An alternative strategy for interaction prediction is Inductive Logic Programming (ILP) [219], a group of logic-based formalisms that extract rules explaining the likely underlying causes of interactions. ILP methods were studied in the work of Tran *et al.* [163] using a large number of features: SWISS-PROT keywords and enzyme properties, Gene Ontology functional annotations, gene expression, cell cycle and subcellular localization. Further

advances in this direction, with a special focus on using domain information, can be found in [74, 75]. The advantage of ILP methods over purely statistical methods is that they are inherently able to deal with relational information, making them ideal candidates for solving the MLPIP problem. Alas, contrary to kernel methods, they tend to be very susceptible to noise, which is a very prominent feature of interaction dataset, and are less effective in exploiting complex feature representations, e.g. involving highly non-linear interactions between continuous features.

Recently, some works highlighted the importance of the multi-level nature of protein–protein interactions. Gonzalez *et al.* [220] propose a method to infer the residue contact matrix from a known set of protein interactions using SVMs; on the contrary, our goal is to predict the interactions concurrently at all levels of the hierarchy. Another study [76] highlights the relevance of domain-level interactions, and the unfortunate lack of details thereof, and formulates a method to reinterpret a known PPI network in terms of its constituent domain interactions; our has a different focus and a more general scope.

Most relevant to this chapter is the work of Yip *et al.* [10], where the authors propose a procedure to solve the MLPIP problem based on a mixture of different techniques. The idea is to decompose the problem as a sequence of three prediction tasks, which are solved iteratively. Given an arbitrary order of the three levels (e.g. proteins first, then domains, then residues), their procedure involves computing putative interactions in the first level (in this case proteins), then using the most confident predictions as novel training examples at the following level (i.e., domains). The procedure is repeated until a termination criterion is met.

Intra-level predictions are obtained with Support Vector Regression (SVR) [118]. In particular, each object has an associated SVR machine that models its propensity to bind any other object in the same level. The extrapolated values act as confidences for the predictions themselves. The mechanism for translating the most confident predictions at one level into training examples for the next level depends on the relative position of the two levels in the hierarchy. Downward propagation (e.g. from proteins to domains) simply associates to each novel example the same confidence as the parent prediction: in other words, if two proteins are predicted as bound with high confidence, all their domains will be considered bound with the same confidence. Upward propagation (e.g. from domains to proteins) is a bit more involved: the confidence assigned to the example is a noisy-OR combination of confidences for all the involved child objects (domains).

While this method has been shown to work reasonably well, it is afflicted by several flaws. First of all, while the iterative procedure is grounded in co-

training [221], the specific choice of components is not as theoretically sound. For instance, the authors apply regression techniques on a classification task, which may lead to sub-optimal results. The inter-level example propagation mechanisms are *ad hoc*, do not exploit all the information at each level (only the most confident predictions are propagated), and are designed to merely propagate information between levels, not to enforce consistency on the predictions. In particular, the downward propagation rule is rather arbitrary: it is not clear why *all* domains of bound proteins should be themselves bound with the same confidence. Finally, these rules, which are intimately tied to the specific implementation, are not defined using a formal language, and are therefore difficult to extend. For instance, it would be difficult to implement in said framework something similar to an n-existential propagation rule, which is extremely useful for dealing with residue interactions.

Semantic Based Regularization seems to have many obvious advantages in this context. A first advantage is that it decouples the implementation of the functions from how consistency among levels is defined. Indeed, consistency is implemented via a set of constraints, which are applied over the output of the predictors. However, there is no limitation in which kind of predictors are used. For example, we used kernel machines as basic machinery for implementing the predictor, where different state-of-the-art kernels can be used at the single levels, while still be able to define a single optimization problem.

Furthermore, SBR allows to natively propagate the predictions of one level to the other levels. Since the predictions and not the supervisions are propagated, SBR accuracy can get advantage of the abundant unsupervised data. The availability of an efficient implementation of the n-existential quantifier is also a crucial advantage: if two proteins or domains are interacting, a small set of residues must be interacting as well. SBR does not simply propagate a generic prior to all the residues for a protein or domain, which could decrease accuracy of the reductions for the negative supervisions. SBR instead performs a search process in order to select a subset of residue candidates, where to enforce the interaction. As shown in the experimental results, this greatly improves residue prediction accuracy. Finally, the circular dependencies that make learning difficult are dealt in the context of a general and well defined framework, which implements various heuristics to make training effective.

4.2 Results and Discussion

4.2.1 Dataset

In this work we use the dataset of Yip *et al.* [10], described here for completeness. The dataset represents proteins, domains and residues using features gathered from a variety of different sources:

- Protein features include phylogenetic profiles derived from COG, subcellular localization, cell cycle and environmental response gene expression; protein-pair features were extracted from Y2H and TAP-MS data. The gold standard of positive interactions was constructed by aggregating experimentally verified or structurally determined interactions taken from MIPS, DIP, and iPfam.

- At the domain level, the dataset includes both features for domain families and for domain instances based on frequencies of domains within one or more species and phylogenetic correlations of Pfam alignments. The gold standard of positive interactions was built from 3D structures of complexed proteins taken from PDB.

- Residue features consist of sequence-based properties, namely charge complementarity, Psi-Blast [222] profiles, predicted secondary structure, and predicted solvent accessibility.

Kernels computed from the individual features were combined additively into a single kernel function for each level, and then transformed into pairwise kernels using Equation (4.1); the resulting functions were used as inputs to SBR.

This procedure yields a dataset of 1681 proteins, 2389 domains, and 3035 residues, with a gold standard of 3201 positive (interacting) protein pairs, 422 domain pairs, and 2000 residue pairs. Since interaction experiments can not determine which pairs do *not* interact, the gold standard of negative pairs is built by randomly sampling, at each level, a number of pairs that are not known to interact (*i.e.* not positive). This is a common approach to negative labeling in the PPI prediction literature [223]. To keep the dataset balanced, the number of sampled negative pairs is identical to the number of objects in the gold standard of positives. For more details on the dataset preparation, please refer to [10].

Turning our attention to the resulting dataset, we note that most of the supervision is located at the protein level: out of all possible interactions between pairs of proteins, which are $\frac{1}{2}(1681 \times 1680)$, 0.226% are known (either

positive or negative). On the contrary, the other levels hold much less information: only 0.042% of all possible residue pairs, and 0.014% of all possible domain pairs, are in the dataset. As briefly mentioned above, this is due to the different requirements for experimentally determining interactions at the three levels.

4.2.2 Evaluation procedure

In this work we compare our method to that of Yip *et al.* [10], where the authors evaluated their method using a 10-fold inner cross-validation procedure. To keep the comparison completely fair, we repeated said procedure with SBR, reusing the very same train/test splits. Since correlated objects, e.g. a protein and its domains/residues, share information, the folds were structured as to avoid such information to leak between train and test folds: this was achieved by keeping correlated objects in the same fold.

SBR has two scalar hyper-parameters that control the contribution of various parts of the objective function: λ_c is the weight associated to the constraints (how much the current solution is consistent with respect to the rules) and λ_r, which controls the model complexity (see the Methods section for more details). The λ_r parameter was optimized on the first fold by training the model without the logic rules and it was then kept fixed for all the folds of the k-fold cross-validation. The resulting value is $\lambda_r = 0.1$. The λ_c parameter has not been optimized and kept fixed at $\lambda_c = 1$. Please note that further significant gains for the proposed method could be achieved by fine-tuning this meta-parameter. However, since the dataset from Yip *et al.* does not include a validation split, no sound way to optimize this parameter was possible without looking at the test set, or redefining the splits (making difficult to compare against Yip's reported results). Therefore, we decided to not perform any tuning for this meta-parameter.

In this work we use the same performance measure as our competitors, the Area Under the Receiver Operating Characteristic Curve (AUCROC, or AUC for short). The AUC measures the ability of to correctly discriminate between positives and negatives, or alternatively, the ability to rank positives above the negatives. It is independent of any classification threshold, and thus particularly fit in scenarios of heavy class unbalance. We computed the average AUC of our method and that of our competitor over all folds of the cross-validation; the results can be found in Table 4.3.

Table 4.3: Area under the ROC curve values attained by Yip *et al.* [10], SBR, and SBR-$\exists_n$ (SBR equipped with the n-existential quantifier).

	Independent	Unidirectional			Bidirectional			Full
Level		P→D	D→R	P→R	P↔D	D↔R	P↔R	
Results for Yip *et al.* [10]								
Proteins	0.715				0.720		0.723	0.726
Domains	0.521	0.585			0.701	0.680		0.699
Residues	0.568		0.513	0.530		0.618	0.658	0.736
Results for SBR								
Proteins	0.811				0.823		0.822	0.822
Domains	0.605	0.820			0.832	0.890		0.946
Residues	0.591		0.668	0.673		0.681	0.675	0.689
Results for SBR-$\exists_n$								
Proteins	0.811				0.823		0.822	0.823
Domains	0.605	0.820			0.832	0.888		0.951
Residues	0.591		0.745	0.760		0.777	0.772	0.797

4.2.3 Results

To evaluate the effects of the constraints on the performances of SBR, we performed three independent experiments using rules of increasing complexity. This setup follows closely that of Yip *et al.* [10].

Independent Levels

As a baseline, we estimate the performance of our method when constraints are ignored. This is equivalent to the method of Yip *et al.* when no information flow between level is allowed. The results can be found in the "Independent" column of Table 4.3.

In absence of constraints SBR reduces to standard ℓ_2-regularized SVM classification: learning and inference become convex problems, and the method computes the globally optimal solution. Thus, the only differences between our method and the competitor are: (i) using classification versus regression, and (ii) using pairwise classification, instead of training a single model for each entity (protein, domain, residues) predicting its interactions. These differences alone produce a significant boost in performance for proteins and domains, amounting to about +0.1 AUC each, while the improvement for residues is less marked, only +0.01.

Unidirectional Constraints

In the second experiment, we evaluate the effect of introducing unidirectional constraints between pairs of levels. In the P→D case only the P→D rule is active, meaning that bound protein pairs enforce positive domain pairs and negative domain pairs enforce negative protein pairs. The D→R and P→R cases are defined similarly. In all three cases, the level not appearing in the rule (e.g. the residue level in the P→D case) is predicted independently. This setup makes it easy to study the effects of propagating information from one level to the other without interferences. The results can be found in the "Unidirectional" column of Table 4.3. In the same column we also show the results for Yip *et al.* for the unidirectional flow setting, where examples are propagated from one level to the next but not vice versa. However, since the competitor's algorithm is iterative, information about lower levels can indeed affect the upper levels in successive iterations.

The results show that introducing unidirectional constraints in SBR improves the predictions in all cases. In particular, using (predicted and known) protein interactions helps inferring correct domain and residue interactions: the former improve by about +0.2 AUC (P→D case) and the latter by +0.08 (P→R case). Residues also benefit from domain-level information, with a 0.07 increase in performance (D→R case). Interestingly, proteins tend to help residue predictions slightly more than domains, despite the indirection between the two levels; this is likely an effect of the larger percentage of supervised pairs available.

Compared to SBR, the method of Yip *et al.* does not benefit as much from unidirectional information flow. The effect of protein-level information on the domains is marginal (+0.06 AUC for P→D), and residue predictions are in fact worse than in the independent case (−0.05 and −0.04 AUC in the D→R and P→R cases, respectively).

Bidirectional Constraints

In the third experiment we study the impact of using bidirectional constraints between pairs of levels; the level not appearing in the rules is predicted independently, as above. In the P↔D case, both the P→D and D→P rules are active, meaning that the protein and domain levels are enforced to be fully consistent; the P↔R and D↔R cases are defined analogously. This experiment is comparable to the bidirectional flow setting of Yip *et al.*. The results can be found in the "Bidirectional" column of Table 4.3.

We observe that the new constraints have a positive effect on predictions at all three levels: proteins change from 0.811 AUC to 0.823, domains from

0.820 to 0.832 and residues from 0.673 to 0.681. The change is not as marked as between the independent and unidirectional experiments. In particular, domains see the largest increase in performance (+0.02 AUC), while proteins and residues are less affected. The result is unsurprising for proteins, which hold most of the supervision and are thus (i) more likely to be predicted correctly in the independent setting, and (ii) less likely to be assisted from hints coming from the other, less supervised levels.

As for the method of Yip *et al.*, the bidirectional flow mostly affects the domain and residue levels, whose improvement is +0.2 AUC and +0.05, respectively; the change for protein interactions is negligible. Regardless of the relative performance increase, SBR is able to largely outperform the competitor in all three cases.

We note that the fact that all three cases (P↔D, P↔R and D↔R) improve over both the independent and the unidirectional experiments shows that not only the bidirectional constraints are in fact sound, but also that, despite the increased computational complexity, SBR is still able to exploit them appropriately.

All Constraints

In the final experiment we activate the P→D, D→P, D→R and R→D rules, as defined in Table 4.2, making all levels interact. This is the most complex setting, and produces fully consistent predictions through the hierarchy. It is comparable to the "PDR" bidirectional setting of Yip *et al.*. The AUC scores can be found in column "Full" of Table 4.3.

In this experiment the P→R and R→P constraints are not used. Direct information flow between proteins and residues is not needed, because it would be redundant: from a formal logic point of view, this corresponds to the observation that the logic rule expressing protein to residue consistency is implied by other consistency rules. Indeed, we have experimentally verified that adding this propagation flow does not significantly affects the results.

In this experiment, protein predictions are stable with respect to the previous experiments, confirming the intuition that the abundance of supervision at this level makes it less likely to benefit from predictions at the other ones. On the contrary, domains see a large performance upgrade, from 0.890 to 0.946 AUC, when made to interact with both proteins and residues. The improvement for residues is instead marginal.

The results for Yip *et al.* are mixed, with proteins and domains faring almost identically to the previous experiment, and residues improving by a large amount (about +0.08 AUC) over the bidirectional P↔R case. This result stands in contrast with that of SBR, and is the only case in which

the method of Yip and colleagues works better than SBR. The issue lies within our formulation of the D→R rule: whenever two domains are bound, the rule is satisfied when at least one residue pair is bound. As already mentioned above, this is not realistic: protein interfaces span more than two residues, typically five or more. We therefore extended SBR to support the n-existential quantifier, which allows to reformulate the D→R rule to take this observation into account (see the Methods section for more details on the n-existential quantifier). The new D→R rule, shown in Table 4.2, requires for each pair of bound domains at least $n = 5$ residues to be bound. We chose the constant $n = 5$ to be both realistic and, since the computational cost increases with n, small enough to be easily computable. We applied the same modification to the P→R rule.

The complete results for the resulting method, termed SBR-$\exists_n$, can be found at the bottom of Table 4.3. When comparing to standard SBR, i.e., without the n-existential, we see that the performance of residues improves drastically in all cases (unidirectional, bidirectional, and with all constraints activated), allowing SBR-$\exists_n$to always outperform the method of Yip *et al.* by a significant margin also on the residue interactions. As a side effect of the better residue predictions, thanks to the D→R and R→D rules domains also improve in the all-constraints experiment. In particular, in the "Full" experiment the AUC improvement of SBR-$\exists_n$over Yip *et al.* is +0.1 AUC for proteins and +0.06 for residues, with domains seeing a rather impressive improvement of +0.25.

Finally, these results highlight the ability of SBR to enforce constraints even with highly complex combinations of rules, allowing the modeler to fully exploit the flexibility and performance improvement offered by non-standard FOL extensions like the n-existential operator.

4.2.4 Discussion

The results presented in the previous section offer a clear perspective on the advantages of the proposed method. By employing appropriate classification techniques and training a single global pairwise model per level, rather than relying on the less than optimal local (per-object) regression models of Yip *et al.*, a considerable improvement was achieved even in the unconstrained experiment. Furthermore, when enforcing consistency among the protein, domain and residue levels and using the n-existential quantifier, the experimental results are significantly better than both the unconstrained baseline and the corresponding results of Yip and colleagues, at all levels and in all experimental settings.

It is worth noting that SBR performance improves monotonically with the

increase of constraint complexity in the reported experiments. This result is far from obvious, and confirms both that the biologically-motivated knowledge base is useful, and that SBR is able to effectively apply it. In contrast, the competitor's method does not always improve in a similar manner.

In general, the performance gain brought forth by inter-level propagation is not homogeneously distributed between the three levels. We register a large improvement for domains and residues, especially when SBR is used in conjunction with the n-existential quantifier. Proteins are less affected by consistency enforcement, most likely due to the availability of more supervised examples.

We note that the FOL rules have a twofold effect. Firstly, they propagate information between the levels, enabling predicted interactions at one level to help inferring correct interactions at the other two levels. This is especially clear in the "Full" experiment with the n-existential quantifier: in this case, better residue level predictions increase the overall quality of domain predictions as well. Secondly, the rules also guarantee that the predictions are consistent along the object hierarchy. hierarchy.

4.3 Conclusions

In this work we described the multi-level protein interaction prediction (MLPIP) problem, which requires to establish the binding state of all uncharacterized pairs of proteins, domains and residues. Contrary to standard protein–protein interaction prediction, the MLPIP problem offers many advantages and opens up new challenges. The primary contribution of this chapter is the extension and application to the MLPIP task of a state-of-the-art statistical relational learning technique called Semantic Based Regularization.

SBR is a flexible framework to inject domain knowledge into kernel machines. In this chapter SBR has been used to tie together protein, domain and residue interaction predictions tasks. In particular, the domain knowledge expresses that two proteins interact if and only if there is an interaction between at least one pair of domains of the proteins. Similarly two domains can interact if and only if there are at least some residues interacting. While these tasks could be learned separately, tying them together has multiple advantages. First the predictions will be consistent and more accurate, as the predictions at one level will help the predictions at the other levels. Secondly, the domain knowledge can be enforced also on the unsupervised data (proteins, domains and residues for which interactions are unknown). Unsupervised data is typically abundant in protein interaction prediction tasks but often neglected. This methodology allows to powerfully leverage it, sig-

nificantly improving the prediction accuracy.

While other work in the literature has exploited the possibility of tying the predictions at multiple levels, the presented methodology employs a more principled inference process among the levels, where the domain knowledge can be exactly represented and precisely enforced. The experimental results confirm the theoretical advantages by showing significant improvements in domain and residue interaction prediction accuracy both with respect to approaches performing independent predictions and the only previous approach attempting at linking the prediction tasks.

Given the flexibility offered by SBR, the proposed method can be extended in several ways. The simplest extension involves engineering a more refined rule set, for instance by introducing (soft) constraints between the binding state of consecutive residues, which are likely to share the same state. More ambitious goals, requiring a redesign of the experimental dataset, include encoding selected information sources, such as domain types, subcellular co-localization and Gene Ontology annotations, as First-Order Logic constraints rather than with kernels, to better leverage their relational nature.

Chapter 5

Predicting Drug-Resistant Mutants

5.1 Background

HIV is a pandemic cause of lethal pathologies in more than 33 million people. Its horizontal transmission trough mucosae is difficult to control and treat because the virus has a high virulence and it infects several type of immune surveillance cells, such as those characterized by CD4 receptor (CD4+ cells). The major problem in treating the human virus infection is the drug selectivity since the virus penetrates in the cell where it releases its genetic material to replicate itself by using the cell mechanisms. A drug target is the replicating apparatus of the cell. HIV antiviral molecules will be directed against several cells such as macrophages or lymphocytes T to interfere with viral replication. The HIV releases a single-strand RNA particle, a reverse transcriptase and an integrase into the cell cytoplasm. Quickly the RNA molecule is retro-transcribed in a DNA double strand molecule, which is integrated into the host genome. The integration events induce a cellular response, which begins with the transcription of the Tat gene by the RNA polymerase II. Tat is a well-known protein responsible for the HIV activation since it recruits some cytoplasm host proteins involved in the expression of viral genes. Remarkably, HIV can establish a life-long latent infection by suppressing its transcription, thus making ineffective the large part of antiviral drugs aimed at controlling the viral replication. However replicating viruses adopt several drug resistance strategies, for instance, HIV induces amino acid mutations reducing the efficacy of the pharmaceutical compounds. The present work is aimed at gaining knowledge on mutations that may occur into the viral RNA transcriptase [224]. This is an important target to develop

antiretroviral medicines and different types of molecules have been found active: the Nucleoside Reverse Transcriptase Inhibitors (NRTI) and Non NRTI (NNRTI). Although RNA RT inhibitors are active, the HIV virus is capable of quickly changing the RNA RT encoding sequence thus acquiring drug resistance. The antiviral therapy is based on the use of cocktails of molecules including new RNA RT inhibitors. A computational approach to predict possible mutation sites and their sensibility to drug is thus an important tool in drug discovery for the antiretroviral therapy.

Computational methods can assist here by exploring the space of potential virus mutants, providing potential avenues for anticipatory drugs [225]. To achieve such a goal, one first needs to understand what kind of mutants may lead to resistance. A general engineering technique for building artificial mutants is referred to as *rational design* [226]. The technique consists in modifying existing proteins by site directed mutagenesis. It relies on a deep domain knowledge in order to identify candidate mutations that may affect protein structure or function. The process typically involves extensive trial-and-error experiments and is also aimed at improving the understanding mechanisms of a protein behavior.

In this work we report on our initial investigation to develop an artificial system mimicking the rational design process. We consider two increasingly complex learning settings and corresponding learning techniques. In the first one we rely on a training set made of single amino acid mutations known to confer resistance to a certain class of inhibitors (we will refer to this as mutation-based learning). An Inductive Logic Programming (ILP) learner [227] is trained for each class of inhibitors in order to extract general rules describing mutations conferring resistance to the drug class. The learned rules are then used to infer novel mutations which may induce similar resistance. In the second setting we learn directly from mutants (comprising of up to 51 amino acid mutations) that have been experimentally tested for their resistance to the same classes of inhibitors (we will refer to this as mutant-based learning). This second setting is actually the most common situation, in which one is presented with a number of mutants together with some evidence of their susceptibility to certain treatments, but no clear information on which mutation is responsible for their behaviour. In this setting we employ a statistical relational learning approach [228] capable of learning weighted combinations of relational rules discriminating between groups of instances, drug-resistant vs drug-susceptible mutants in our case. The learned model is then used to generate novel mutants together to a score indicating their predicted resistance.

Many machine learning methods have been applied in the past to mutation data for predicting single amino acid mutations on protein stability

changes [88] and the effect of mutations on the protein function [89, 90] or drug susceptibility [96]. To the best of our knowledge this is the first attempt to learn relational features of mutations affecting a protein behavior and use them for generating novel relevant mutations. Furthermore, even if we focus on single amino acid mutations in our experimental evaluation, our approach can be quite easily extended to multiple point mutations, and we are actively working in this direction. Note that the other approaches first generate all potential mutations and then decide which of them leads to resistance, resulting in a combinatorial explosion if trying to predict multiple mutations, whereas we constraint the search space generating the predicted relevant ones directly.

We report an experimental evaluation focused on HIV RT. RT is a well-studied protein: a large number of mutants have been shown to resist to one or more drugs and databases exist that collect those data from different sources and make them available for further analyses [229]. We tested the ability of our approach to generate drug-resistant amino acid mutations for NRTI and NNRTI. Our results show statistically significant improvements for both drug classes over the baseline results obtained through a random generator. A preliminary version of this work was presented in [230]. We extend those findings by substantially augmenting the background knowledge and introducing the statistical relational learner which allows for much more refined mutant scores and improved results.

The approach can be in general applied in mutation studies aimed at understanding protein function. By searching for residues most likely to have a functional role in an active site, the approach can for instance be used in the engineering of enzyme mutants with an improved activity for a certain substrate.

5.2 Results

5.2.1 Datasets

We applied our approach to predict HIV RT mutations conferring resistance to two classes of inhibitors: NRTI and NNRTI. The two classes of inhibitors differ in the targeted sites and rely on quite different mechanisms [231, 232]. NNRTI inhibit the reverse transcriptase by binding to the enzyme active site, therefore directly interfering with the enzyme function. NRTI are instead incorporated into the newly synthesized viral DNA for preventing its elongation.

We collected datasets for both mutation-based and mutant-based learn-

ing. The former (Dataset 1) is a dataset of amino acid mutations derived from the Los Alamos National Laboratories (LANL) HIV resistance database[1] by Richter et al. [233], who used it to mine relational rules among mutations. It consists of 95 amino acid mutations labeled as resistant to NRTI and 56 labeled as resistant to NNRTI, over a set of 581 observed mutations. (Dataset 2) HIV RT mutation data from the Stanford University HIV Drug Resistance Database. The database provides a dataset of selected mutants of HIV RT with results of susceptibility studies to various drugs, and was previously employed [96] for predicting drug resistance of novel (given) mutants[2]. It is composed of 838 different mutants annotated with susceptibility levels (low, medium and high) to drugs belonging to the NRTI (639 mutants) and NNRTI (747 mutants) drug classes. We considered a setting aimed at identifying amino acid mutations conferring high susceptibility (with respect to medium or low), and considered a mutant as highly susceptible to a drug class if it was annotated as being highly susceptible to at least one drug from that class.

5.2.2 Learning in first order logic

Our aim is to learn a first-order logic hypothesis for a target concept, i.e. mutation conferring resistance to a certain drug, and use it to infer novel mutations consistent with such hypothesis. We rely on definite clauses which are the basis of the Prolog programming language. A definite clause is an expression of the form `h` $\leftarrow$ `b1 AND ... AND bn`, where `h` and the `bi` are atomic literals. Atomic literals are expressions of the form `p(t1, ..., tn)` where `p/n` is a predicate symbol of arity `n` and the `ti` are terms, either constants (denoted by lower case) or variables (denoted by upper case) in our experiments. The atomic literal `h` is also called the head of the clause, typically the target predicate, and `b1 AND ... AND bn` its body. Intuitively, a clause represents that the head `h` will hold whenever the body `b1 AND ... AND bn` holds. For instance, a simple hypothesis like `res_against(A,nnrti)` $\leftarrow$ `mutation(A,C) AND close_to_site(C)` would indicate that a mutation `C` in the proximity of a binding site confers to mutant `A` resistance against a `nnrti`. Learning in this setting consists of searching for a set of definite clauses $H = \{c_i, ..., c_m\}$ covering all or most positive examples, and none or few negative ones if available. First-order clauses can thus be interpreted as relational features that characterize the target concept. The main advantage of these logic-based approaches with respect to other machine learning tech-

[1] http://www.hiv.lanl.gov/content/sequence/RESDB/

[2] downloadable at http://hivdb.stanford.edu/cgi-bin/GenoPhenoDS.cgi

niques is the expressivity and interpretability of the learned models. Models can be readily interpreted by human experts and provide direct explanations for the predictions. On the other hand, purely logic-based approaches fail to incorporate uncertainty in the hypotheses they produce, and different degrees of importance of the clauses of which hypotheses are made. Statistical relational learning [1, 170] techniques aim at filling this gap by combining statistics and expressive representational languages in developing predictive models. A simple and effective solution consists of learning a weighted combination of clauses, where clauses and their weights are jointly learned in trying to model the concept of interest.

5.2.3 Background knowledge

We built a relational knowledge base for the problem domain. In Table 5.1 we summarize the predicates that we included as a background knowledge. We represented the amino acids of the reference wild type (consensus sequence) with their positions in the primary sequence (`aa/2`) and the specific mutations characterizing them (`mut/4`). Target predicates were encoded as resistance of the mutation or mutant to a certain drug (`res_against/2`).

Note that this encoding considers mutations at the amino acid rather than nucleotide level, i.e. a single amino acid mutation can involve up to three nucleotide changes. Focusing on single nucleotide changes would have drastically expanded the space of possible mutations. We thus kept the focus on amino acid mutations but we included the cost (in terms of nucleotide changes) of a certain amino acid mutation to further refine our search procedure as explained in the following.

This additional background knowledge was included in order to highlight characteristics of residues and the mutations:

`typeaa/2` indicates the type of the natural amino acids according to the Venn diagram grouping based on the amino acids properties proposed in [234]. For example, a serine is a tiny and polar amino acid.

`color/2` indicates the type of the natural amino acids according to the coloring proposed in [235]. For example the magenta class includes basic amino acids as lysine and arginine while the blue class includes acidic amino acids as aspartic and glutamic acids. These groups of amino acids do not overlap as in the previous case.

`same_type_aa/3` indicates whether two residues belong to the same type `T`, i.e. a change from one residue to the other conserves the type of the amino acid.

Figure 5.1: Summary of the background knowledge facts and rules. MutID is a mutation or a mutant identifier depending on the type of the learning problem.

Background Knowledge Predicates	
`aa(Pos,AA)`	indicates a residue in the wild type sequence
`mut(MutID,AA,Pos,AA1)`	indicates a mutation: mutation or mutant identifier, position and amino acids involved, before and after the substitution
`res_against(MutID,Drug)`	indicates whether a mutation or mutant is resistant to a certain drug
`color(Color,AA)`	indicates the coloring group of a natural amino acid
`typeaa(T,AA)`	indicates the type (e.g. aliphafatic, charged, aromatic, polar) of a natural amino acid
`same_color_type(R1,R2)`	indicates whether two residues belong to the same coloring group
`same_typeaa(R1,R2,T)`	indicates whether two residues are of the same type `T`
`same_color_type_mut(MutID, Pos)`	indicates a mutation to a residue of the same coloring group
`different_color_type_mut(MutID, Pos)`	indicates a mutation changing the coloring group of the residue
`same_type_mut_t(MutID, Pos, T)`	indicates a mutation to a residue of the same type `T`
`different_type_mut_t(MutID, Pos)`	indicates a mutation changing the type of the residue
`aamutations(Pos,R1,R2,Num)`	indicates whether a given mutation requires at least a single, double, or triple nucleotide substitution
`close_to_site(Pos)`	indicates whether a specific position is close to a binding or active site if any
`location(L,Pos)`	indicates in which fragment of the primary sequence the amino acid is located
`conservation(Pos,ConsClass)`	indicates whether a position is highly conserved or not
`in_ss(SS,N,Pos)`	indicates whether a mutation occurs within the `N`th secondary structure element of a given type
`in_motif(Pos,Motif)`	indicates whether a mutation occurs within a known sequence motif
`catalytic_propensity(AA,CP)`	indicates whether an amino acid has a high, medium or low catalytic propensity
`mutated_residue_cp(Rw,Pos,Rm,CPold,CPnew)`	indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high)

same_color_type/2 indicates whether two residues belong to the same coloring group, i.e. a change from one residue to the other conserves the coloring group of the amino acid.

same_type_mut_t/3 indicates that a residue substitution at a certain position does not modify the amino acid type `T` with respect to the wild type. For example mutation i123v conserves the `aliphatic` amino acid type while mutation i123d does not (i.e. `different_type_mut_t/3` holds for it).

same_color_type_mut/2 indicates that a residue substitution at a certain position does not modify the amino acid coloring group with respect to the wild type. For example mutation d123e conserves the `blue` amino acid group while mutation d123a does not (i.e. `different_color_type_mut/2` holds for it).

aamutations/4 indicates whether a given amino acid mutation can be triggered by a single, double, or triple nucleotide substitution. For instance to change an alanine `a` into an aspartic acid `d` a single nucletotide substitution can be sufficient as in the case `a: gct` $\rightarrow$ `d: gat`.

Other background knowledge facts and rules were added in order to express structural relations along the primary sequence, secondary structure, and catalytic propensity of the involved residues:

close_to_site/1 indicates whether a specific position is less than 5 positions away from a residue belonging to a binding or active site. In our specific case, the background theory incorporates knowledge about a metal binding site and a heterodimerization site.

location/2 indicates in which fragment of the primary sequence the amino acid is located. Locations are numbered from 0 by dividing the sequence into fragments of 10 amino acid lenght.

conservation/2 indicates whether a position is highly conserved or not. Conservation is defined in terms of positional variation (entropy) among a curated multiple-alignment of reverse transcriptase sequences, taken from the LANL HIV resistance database[3].

in_ss/3 indicates whether a mutation occurs within a known secondary structure element. We encoded position specific knowledge for the four

[3]http://www.hiv.lanl.gov/

secondary structure classes: helix, strand, turn, and coil. This information was derived from the 3D model of the RT structure by using the DSSP program [38].

`in_motif/2` indicates whether a mutation occurs within a known sequence motif. Our background theory includes information about PROSITE [236] and Pfam motifs [237].

`catalytic_propensity/2` indicates whether an amino acid has a high, medium or low catalytic propensity according to [238].

`mutated_residue_cp/5` indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high).

5.3 Methods

5.3.1 Homology Modeling

Two models of the RT tertiary structure have been produced from the consensus sequence on which each dataset (described in Results) is based. Each 3D model has been produced by using the modelling package Modeller 9.10 [239], starting from the template structures of two close homologues in the unbound form, which are available in the Protein Data Bank (PDB)[240]: 1DLO and 1HMV.

1: **input:** *background knowledge* $\mathcal{B}$*, learned model* H
2: **output:** *rank of the most relevant mutations* $\mathcal{R}$
3: **procedure** GenerateMutations($\mathcal{B}, H$)
4: Initialize $\mathcal{M} \leftarrow \varnothing$
5: $A \leftarrow$ find all assignments a that satisfy at least one clause $c_i \in H$
6: **for** $a \in A$ **do**
7: $m \leftarrow$ mutation corresponding to assignment a
8: $score \leftarrow S_H(m)$ ▷ score m according to model H
9: $\mathcal{M} \leftarrow \mathcal{M} \cup \{(m, score)\}$
10: **end for**
11: $\mathcal{R} \leftarrow$ RankMutations($\mathcal{M}$) ▷ rank relevant mutations
12: **return** $\mathcal{R}$
13: **end procedure**

Figure 5.2: Mutation generation algorithm.

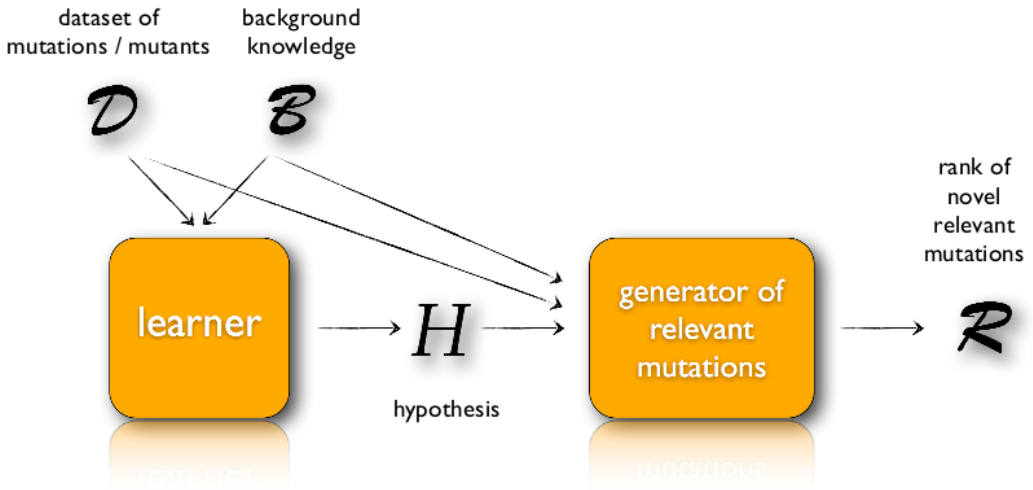


Figure 5.3: Schema of the mutation engineering algorithm.

5.3.2 Algorithm overview

The proposed approach is sketched in Figure 5.1.

Step 1: Learning phase

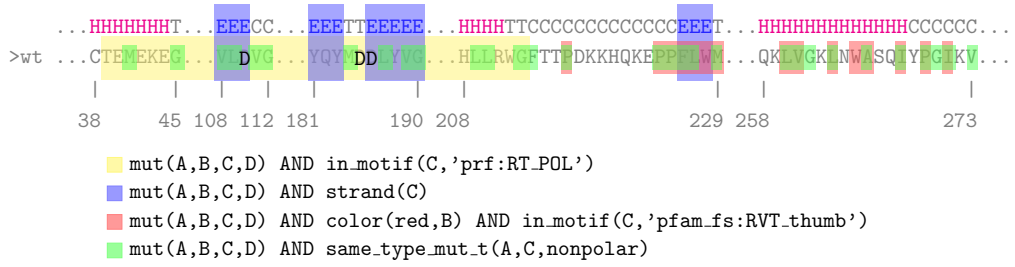
The first step is the learning phase. A learner is fed with a logical representation of the data $\mathcal{D}$ and of the domain knowledge $\mathcal{B}$ to be incorporated, and it returns a first-order logical hypothesis H for the concept of mutation conferring resistance to a certain class of inhibitors.

In this context there are two suitable ways to learn the target concept, depending on the type of input data and their labeling:

a) the one-class classification setting, learning a model from positive instances only. This is the approach we employ for Dataset 1: positive examples are mutations for which experimental evidence is available that shows resistance to a drug, but no safe claim can be made on non-annotated mutations.

b) the binary classification setting, learning to discriminate between positive and negative instances. This setting is appropriate for Dataset 2: positive examples are in our experiments mutants labelled as highly susceptible to the drug class, negative examples are those with medium or low susceptibility.

Figure 5.4: An example of hypothesis, learned by Aleph on Dataset 1, for the NNRTI task with highlighted amino acid positions d by the hypothesis clauses.



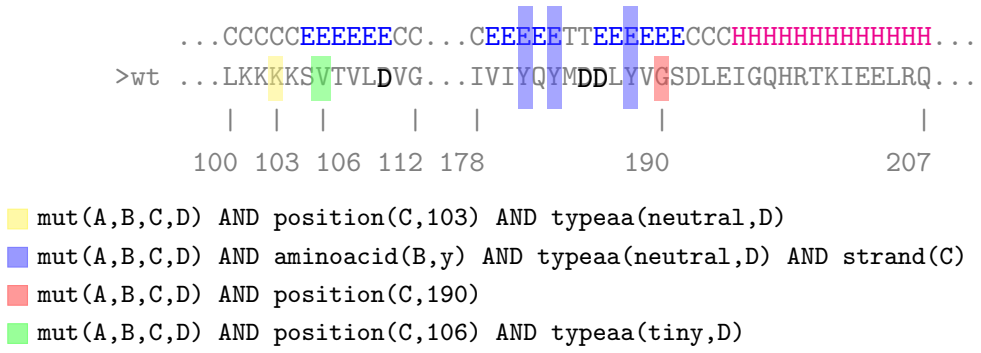
In the one-class classification case we employ the Aleph (A Learning Engine for Proposing Hypotheses) ILP system[4], which learns first order logic hypotheses in a bottom-up fashion. It incrementally builds a hypothesis trying to cover all positive examples. The hypothesis search is guided by a Bayesian evaluation function, described in [241], scoring candidate solutions according to an estimate of the Bayes' posterior probability that allows to tradeoff hypothesis size and generality. Aleph adds clauses to the hypothesis based on their coverage of training examples. Given a learned model, the first clauses are those covering most training examples and thus usually the most representative of the underlying concept.

In Figure 5.2 we show a simple example of learned hypothesis covering a set of training mutations from Dataset 1. The learned hypothesis models the ability of a mutation to confer resistance to NNRTI and is composed of four first-order clauses, each one covering different sets of mutations of the wild type as highlighted in colors: yellow for the first clause, blue for the second, red for the third, and green for the fourth one. Some mutations are covered by more than one clause as shown by the color overlaps. For instance, a mutation of the glycine in position 190 satisfies three clauses: the first, the second and the fourth. On top of the RT consensus sequence we also report the corrisponding secondary structure annotation, by highlighing in magenta the helices and in blue the β-strands. The PROSITE and Pfam motifs `prf:RT_POL` and `pfam_fs:RVT_thumb` appearing in the clauses identify specific regions along the RT sequence. Bold letters in the picture indicate residues involved in the RT metal binding site (D110, D185 and D186).

In the binary classification case, we employ kFOIL [228], a statistical relational approach which learns a weighted combination of clauses discriminat-

[4]http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/aleph.html

Figure 5.5: An example of hypothesis, learned by kFOIL on Dataset 2, for the NNRTI task with highlighted amino acid positions covered by the hypothesis clauses.



ing positive from negative instances. kFOIL is a kernel-based approach [242], capable of learning hypotheses made of complex non-linear combinations of clauses. For the sake of interpretability we limit ourselves to second degree polynomial kernels, where the predictive model is a combination of conjuctions of up to two clauses.

In Figure 5.3 we show an example of (part of) a learned hypothesis. As in the above example, the model is composed of four first-order clauses, each contributing to the characterization of NNRTI resistance mutations. Three of the four clauses specify positions 103, 106 and 190 directly as likely targets for resistance conferring mutations. The second clause, which is position aspecific, specifies mutations of thyrosines occurring within a strand, where the mutation is a non-charged aminoacid. The conjunction of any of the position aspecific with any of the position specific further details the features that likely resistant mutations are expected to exhibit.

Step 2: Generative phase

The second step of our approach is the generative phase, in which the learned hypothesis is employed to find novel mutations that can confer drug resistance to an RT mutant. A set of candidate mutations can be generated by using the Prolog inference engine starting from the rules in the learned model. The rules are actually constraints on the characteristics that a mutation of the wild type should have in order to confer resistance to a certain inhibitor, according to the learned hypothesis.

Algorithm 5.2 details the mutation generation procedure. We assume, for simplicity, to have a model H for a single drug class. The procedure works by

querying the Prolog inference engine for all possible variable assignments that satisfy the hypothesis clauses, each representing a mutation by its position and the amino acid replacing the wildtype residue. The set of mutations generated by the model is ranked according to a scoring function S_H before being returned by the algorithm. When using Aleph, we define S_H as the number of clauses in H that a candidate mutation m satisfies. When using kFOIL, S_H is the value of the weighted combination of the satisfied clauses. The latter case allows a much more refined scoring, as will be showed in the experimental evaluation.

Consider the example model in Figure 5.2. Among the mutations generated using the model are all those changing the glycine in position 190 in a non polar amino acid: 190P, 190A, 190F, 190I, 190L, 190V, 190M. Here 190P indicates a change of the wild type amino acid at position 190 into a proline. Each of these mutations satisfies the first, the second and the fourth clause, receiving a score of three. Note that mutation 190A is part of the known NNRTI surveillance mutations (see [243]).

As for the model in Figure 5.3, the position specific rules all identify known surveillance mutations: 103N, 103S, 106M, 106A, 190A, and 190S. Clause two affects position 181, a thyrosine occurring within a strand, and corresponds to surveillance mutations 181C, 181I, 181V.

5.3.3 Learning from mutations

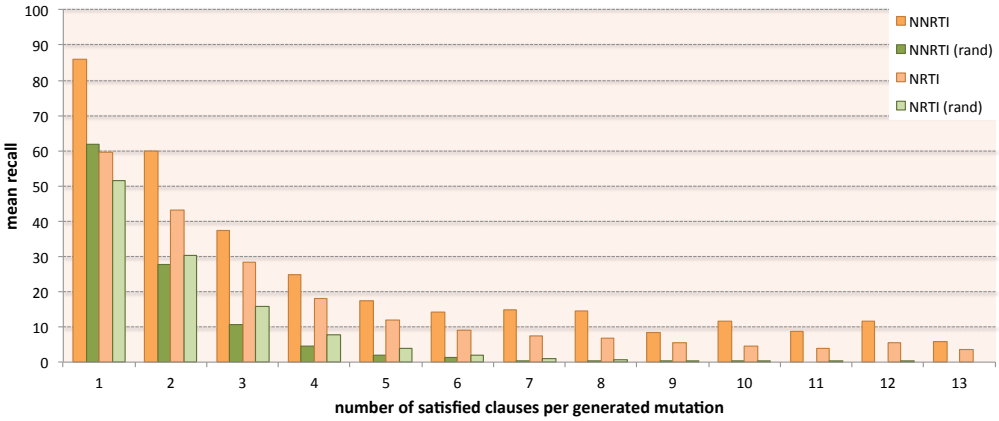
We first learn general rules characterizing known resistance mutations (from Dataset 1) to be used for predicting novel candidate ones.

We divided the dataset of mutations into a training and a test set (70/30) in a stratified way, which means by preserving, both in the train and test set, the proportion of examples belonging to one of the two drug classes. This produces a training set of 106 mutations and a test set of 45 ones.

We trained the ILP learner on the training set and we evaluated on the test set the set of mutations generated using the learned model. The evaluation procedure takes the set of generated mutations and computes its enrichment in test mutations. We compare the recall of the approach, i.e. the fraction of test mutations generated by the model, with the recall of a baseline algorithm that chooses at random a set (of the same cardinality) of possible mutations among all legal ones.

We computed 30 random 70/30 train/test splits and performed 30 runs of our algorithm on each split (Aleph has a random component generating the seed for the hypothesis search). Figure 5.4 reports results averaged over all runs for both NNRTI and NRTI tasks. In this setting, the average size of the learned hypotheses for NNRTI and NRTI are 10 and 14 rules respectively.

Figure 5.6: Mean recall of the generated mutations on the resistance test set mutations from Dataset 1 by varying the number of satisfied clauses. The mean recall values in orange refer to the proposed generative algorithm. The mean recall values in green refer to a random generator of mutations.



The figure shows the mean recall on the test set when increasing the score threshold for accepting a mutation, i.e. the number of clauses a mutation must satisfy in order to be accepted. The results of the random baseline consider the same number of mutations selected by the method for each threshold. The recall trend is shown in orange for our approach and in green for the random generator for both classes of inhibitors. Recall differences are statistically significant according to a paired Wilcoxon test (α=0.01).

We finally learned a model on the whole dataset in order to generate a single set of mutations for further inspection. We report five examples of novel mutations with the highest score for each one of the tasks: S105Y, S105T, S105N, S105G, S105C for NNRTI and 50A, 63A, 63M, 159L, 195V for NRTI.

For NNRTI, known resistance mutations are found in positions 103 and 106, possibly explaining the high score of mutations at position 105. In [244], the authors found a set of novel mutations conferring resistance to efavirenz and nevirapine, which are NNRTI. Our mutation generation algorithm partially confirms their findings. Apart from mutation 138Q, not generated by our model, all other mutations have been generated, with 90I satisfying two out of five clauses and 101H, 196R, and 28K satisfying one.

Table 5.2 reports the most commonly learned clauses for both NNRTI and NRTI classification tasks. The rules for NNRTI resistance give relevance to mutations in β-strands, while for NRTI, mutations on turns and coils seem to be more relevant. It is also evident that the most susceptible region for

Table 5.1: List of the ten most frequent rules learned on Dataset 1, sorted by average number of models they appear in.

# models	learned clause
	NNRTI
21.8	`mut(A,B,C,D) AND strand(C)`
20.5	`mut(A,B,C,D) AND location(11,C)`
17.1	`mut(A,B,C,D) AND strand(C) AND in_motif(C,'prf:RT_POL')`
9.9	`mut(A,B,C,D) AND in_motif(C,'pfam_fs:RVT_1')`
9.4	`mut(A,B,C,D) AND same_type_mut_t(A,C,neutral) AND strand(C)`
7.9	`mut(A,B,C,D) AND color(red,D) AND in_motif(C,'prf:RT_POL')`
7.3	`mut(A,B,C,D) AND same_type_mut_t(A,C,nonpolar)`
6.8	`mut(A,B,C,D) AND in_motif(C,'prf:RT_POL')`
6.1	`mut(A,B,C,D) AND color(red,B)`
5.9	`mut(A,y,C,D)`
	NRTI
25.2	`mut(A,B,C,D) AND location(7,C)`
18.8	`mut(A,B,C,D) AND in_motif(C,'prf:RT_POL')`
16.1	`mut(A,B,C,D) AND turn(C) AND in_motif(C,'prf:RT_POL')`
11.3	`mut(A,B,C,D) AND coil(C) AND conservation(C, high)`
11.1	`mut(A,B,C,D) AND conservation(C, high)`
11	`mut(A,B,C,D) AND same_color_type_mut(A,B) AND in_motif(B,'prf:RT_POL')`
8.7	`mut(A,B,C,D) AND same_color_type_mut(A,B)`
7.3	`mut(A,B,C,D) AND in_motif(C,'pfam_fs:RVT_1')`
7.3	`mut(A,B,C,D) AND color(red,B) AND in_motif(C,'prf:RT_POL')`

developing resistance to these inhibitors is the region between positions 54 and 234 along the primary sequence, corresponding to the motif `prf:RT POL`. In addition, for the resistance to NNRTI the region between positions 98 and 107 is more relevant, while for NRTI it is the region between positions 64 and 71 (see the `location` predicate).

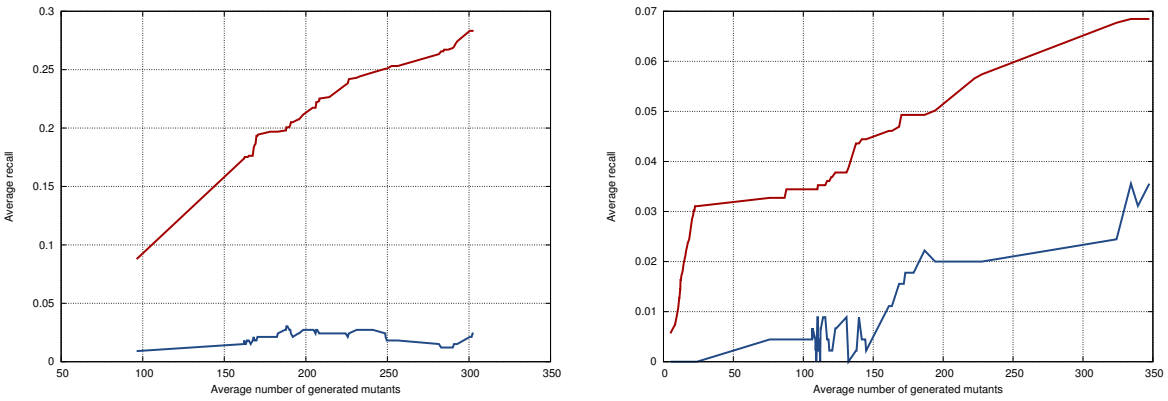
5.3.4 Learning from mutants

The next set of experiments is focused on learning mutations from mutant data (Dataset 2). Learned models are still limited to single amino acid mutations, and so are novel mutants generated by the system.

We randomly assigned the mutants in Dataset 2 to 30 train/test set splits, by avoiding having mutants containing the same resistance mutation (according to the labelling used in Dataset 1) in both training and test sets. For each of the 30 splits, we evaluated the recall of the generated mutations on the known resistance mutations (from Dataset 1), by first removing all the mutations that were also present in the training set. Comparison is again made on a baseline algorithm generating random legal mutations.

Results averaged on the 30 random splits are reported in Figure 5.5. The curve shows the average recall of the generated mutations while varying the

Figure 5.7: Mean recall of the generated mutations on the resistance test set mutations from Dataset 2 by varying the threshold on the prediction confidence, and the corresponding average number of overall generated mutations (i.e., not necessarily in the test set), in red. The blue line refers to the random generator of mutations.



(a) NNRTI (b) NRTI

threshold over their confidence, and the corresponding number of overall generated mutations. For NNRTI, we can see that we obtain an average recall of 25% while generating only 250 mutants, and can reach up to 27% with about 300 generated mutants. In both cases the results are statistically significantly higher than those achieved by a random generator (paired Wilcoxon test, α=0.01).

The hypothesis for the resistance to NNRTI identifies more than half (12 out of 18) of the known resistance survaillance mutations reported in [243]: 103N, 103S, 106A, 181C, 181I, 181V, 188L, 188C, 190A, 190S, 190E, all with very high confidence. The model also predicts other not previously reported mutations as being resistant with high confidence, for instance 183F and 232A, very close to known surveillance mutations 181C and 230L.

Also in the case of NRTI the generative algorithm suggests most (32 of 34) known survaillance mutations reported in [243]: all of them except those targeting position 69 (including an insertion).

As previously stated, in our experiments we constrain the learner to weighted combinations of conjunctions of up to two clauses. Note that two clauses with distinct `position` predicates cannot be simultaneously satisfied by the same mutation. Conjuctions of two clauses will thus typically involve one position-specific and one position-aspecific clause.

Table 5.3 lists the most frequently learned clauses in the 30 distinct mod-

Table 5.2: List of the ten most frequent learned rules for Dataset 2, sorted by number of models they appear in. The table also includes the clause `position(C,X)`, which is present in all models for different values of `X`.

# models	learned clause
	NNRTI
all	`mut(A,B,C,D) AND position(C,X)`
9	`mut(A,B,C,D) AND position(C,103) AND typeaa(neutral,D)`
6	`mut(A,B,C,D) AND position(C,106) AND typeaa(tiny,D)`
6	`mut(A,y,C,D) AND typeaa(neutral,D) AND strand(C)`
6	`mut(A,y,C,D) AND strand(C)`
5	`mut(A,B,C,a) AND position(C,106)`
5	`mut(A,y,C,D) AND typeaa(neutral,D)`
4	`mut(A,B,C,D) AND position(C,90) AND correlated_mut(A,C,E)`
4	`mut(A,B,C,D) AND position(C,143) AND same_type_aa(D,B,polar)`
3	`mut(A,B,C,D) AND typeaa(aromatic,B) AND strand(C) AND AND typeaa(neutral,D)`
	NRTI
all	`mut(A,B,C,D) AND position(C,X)`
17	`mut(A,m,C,D) AND same_type_aa(B,D,nonpolar)`
13	`mut(A,m,C,D) AND highconservation(C)`
12	`mut(A,w,C,D)`
9	`mut(A,m,C,D) AND inMotif(C,pfam_ls:RVT_1)`
9	`mut(A,m,C,D)`
9	`mut(A,p,C,D)`
6	`mut(A,B,C,D) AND position(C,165) AND correlated_mut(A,C,E)`
6	`mut(A,B,C,D) AND position(C,188) AND correlated_mut(A,C,E)`
6	`mut(A,m,C,D) AND inMotif(C,prf:RT_POL)`
6	`mut(A,m,C,D) AND inMotif(C,pfam_fs:RVT_1)`

els learned during the cross-validation procedure.

It is easy to see that the most frequent clauses tend to favor mutations to positions 103, 106, and 143 for NNRTI resistance and 165 and 188 for NRTI resistance, among other less frequent positions. The clauses also specify properties of the mutations occurring at these positions. On the one hand, NNRTI resistant mutations are predicted to have a strong preference for strand residues (with `strand` occurring three times in Table 5.3) and for non-charged mutations. On the other hand, NRTI resistant mutations are predicted to occur within PROSITE motif `RT_POL` and Pfam motif `RVT_1`; mutations to highly conserved methionine positions are also predicted to confer resistance, as confirmed by surveillance mutation 184V.

5.4 Discussion and Future Work

The results shown in the previous section are a promising starting point to generalize our approach to more complex settings. We showed that the ap-

proach scales from few hundreds of mutations as learning examples to almost a thousand of complete mutants. Moreover the learned hypotheses significantly constrain the space of all possible single amino acid mutations to be considered, paving the way to the expansion of the method to multi-site mutant generation. This represents a clear advantage over alternative existing machine learning approaches, which would require the preliminary generation of all possible mutants for their evaluation. Restricting to RT mutants with two mutated amino acids, this would imply testing more than a hundred million candidate mutants. At the same time our statistical relational learning approach cannot attain the same accuracy levels of a sophisticated technique modelling for instance the three dimensional rearrangements of the resulting mutant. We plan to combine the respective advantages of the two approaches by using our statistical relational model as a pre-filtering stage, producing candidate mutants to be further analysed by complex modelling techniques and additional tools evaluating, for instance, a mutant stability. An additional direction to refine our predictions consists of jointly learning models of resistance to different drugs (e.g. NNRTI and NRTI), possibly further refining the joint models on a per-class basis. On a predictive (rather than generative) task, this was shown [245] to provide improvements over learning distinct per-drug models.

Our approach is not restricted to learning drug-resistance mutations in viruses. More generally, it can be applied to learn mutants having certain properties of interest, e.g. improved or more specific activity of an enzyme with respect to a substrate, in a full protein engineering fashion.

Chapter 6

Conclusions

In this thesis we presented three applications of Statistical Relational Learning methods to old and novel problems in computational biology, which highlight how SRL can indeed be very effective in dealing with a range of different problems.

In Chapter 3 we showed how to *jointly* improve the outputs of multiple correlated predictors of protein features with a probabilistic-logical consistency layer. Given a set of correlated predictions and a set of weighted First-Order Logic rules that represent biologically motivated constraints between the predicted features, the proposed consistency layer, renders the predictions consistent by removing rule violations. We have presented promising experimental results for the task of refining subcellular localization, disulfide bonding state, and metal bonding state. We stress the fact that the consistency layer is fully general, and could in principle be applied to an array of arbitrary heterogeneous predictors; moreover, it does not require any change to the underlying software.

We are currently planning to extend this approach in two ways. First of all, we want to include more features into the refinement process, in order to increase the chance of improving the overall consistency. The proposed framework is indeed designed to be as modular as possible: adding a new predictor is just a matter of encoding the appropriate FOL rules. Ideally, we would apply the refiner to one of the large scale prediction hierarchies available, such as Distill [22], SPACE [179] and PredictProtein [23].

Clearly, broadening the scope of the refiner to more features assumes that gs-MLNs [3] can scale up easily, which has not yet been proven. However, there are at least two promising alternatives to gs-MLNs, with roughly the same expressive power, which are actively supported and may be worth considering: Semantic Based Regularization [7] (or rather a supervised version thereof), which we illustrated in Chapter 4, and Probabilistic Similarity Logic

(PSL) [246]. Both methods leverage t-norms to translate FOL formulae into continuous analogues, and include support for similarity measures (kernels) between relational entities. In both cases training requires solving complex optimization tasks, but their authors have already provided efficient computational procedures that have proven effective in practice. Even though not strictly required, as shown in Chapter 3, effective weight learning may improve the performance of the refiner.

In Chapter 4 we proposed a solution to the multi-level protein–protein interaction (PPI) prediction problem, i.e. the problem of inferring the binding state of all proteins, domain and residue pairs in an interaction network. Given the hierarchical nature of interactions (*proteins* bind by means of specific *domains*, which in turn form interfaces through patches of *residues*), predictions at different levels are indeed correlated. We proposed a Statistical Relational Learning method, based on Semantic Based Regularization (SBR) [7], that collectively infers the binding state of all object pairs. SBR is a multi-task learning method based on kernel machines, that employs First Order Logic constraints to tie the learning tasks together. Our experimental evaluation stressed how SBR substantially outperforms the baseline in several experimental settings, indicating that exploiting the hierarchical nature of PPIs can indeed lead to more accurate, and more consistent, predictions.

Given the success of SBR on the multi-level PPI task, we plan to further investigate its applicability. Specifically, the idea is to augment the current experiment with protein function prediction, i.e. prediction of Gene Ontology terms [20], including subcellular localization information. Protein function labels are hierarchical, arranged as nodes in a rooted tree representing a general-to-specific relation. Predictions therefore consist of multiple, interdependent labels, one for each node in the tree, a fact that can be translated into FOL by requiring that, if a node u is predicted true, so must be all nodes on the path from u to the root. We plan to integrate the interaction and function prediction tasks by introducing FOL constraints between their outputs, according to these observations: i) interacting proteins often share the same function; ii) proteins residing in different sub-cellular locations are not likely to interact. Moreover, there is a strong correlation between localization and function (e.g. proteins that manipulate the DNA are likely to reside in the nucleus), and function and interactions (e.g. certain enzymes lose their catalytic ability when bound). One major challenge is that some proteins may exhibit multiple functions, and may reside in multiple cellular components. We plan to explicitly take this ambiguity into account by using soft (non deterministic) constraints. The resulting method would be able to collectively predict function, localization and interactions for a whole set of proteins, and would guarantee the outputs to be consistent as a whole.

In a second step, we plan to extend the above experiment to include signal transduction pathway information. Biological pathways are typically mined from experimentally validated, functionally annotated PINs, in a semi-manual fashion. Pathways can be exploited in two manners. On one hand, since most organisms and cell types rely on the same basic biological functions, they also share the same fundamental pathways. Given a database of known pathways, we can constrain the predicted interaction/function network to include/exclude said pathways. This is indeed not an easy task, as it requires to develop a way to perform (soft) labeled sub-graph matching. The problem may be reduced by extracting small common motifs from the pathways, which capture the most frequent function-interaction patterns, and encoding them as weighted FOL rules. Of course, there are still open problems with this approach.

Finally, in Chapter 5 we proposed a simple statistical relational algorithm for mining relational patterns characterizing mutations conferring drug resistance to viral proteins. The input consists of mutation data labeled with drug-resistance information, either as sets of *mutations* conferring resistance to a certain drug, or as sets of *mutants* with information on their susceptibility to the drug. The learned rules can be used to generate a set of potentially resistant mutants. Promising results were obtained in generating resistant mutations for both nucleoside and non-nucleoside HIV reverse transcriptase inhibitors.

As already mentioned, our approach can be generalized quite easily to learning mutants characterized by more complex rules correlating multiple mutations, i.e. multi-point mutations. This is a very interesting topic, as current methods are limited to single-point mutations. Research on this extension, however, is hampered by the lack of exhaustive labeled mutation datasets.

Bibliography

[1] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[2] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*, 2008.

[3] M. Lippi and P. Frasconi. Prediction of protein β-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, 2009.

[4] R. Nair and B. Rost. Mimicking cellular sorting improves prediction of subcellular localization. *Journal of molecular biology*, 348(1):85–100, 2005.

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

[6] M. Lippi, A. Passerini, M. Punta, B. Rost, and P. Frasconi. Metaldetector: a web server for predicting metal-binding sites and disulfide bridges in proteins from sequence. *Bioinformatics*, 24(18):2094–2095, 2008.

[7] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine learning*, 86(1):57–88, 2012.

[8] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[9] S. Džeroski and I. Bratko. Handling noise in inductive logic programming. In S. Muggleton, editor, *ILP92*, Report ICOT TM-1182, 1992.

[10] K.Y. Yip, P.M. Kim, D. McDermott, and M. Gerstein. Multi-level learning: improving the prediction of protein, domain and residue interactions by allowing information flow between levels. *BMC bioinformatics*, 10(1):241, 2009.

[11] Pierre Baldi et al. *Bioinformatics: the machine learning approach*. The MIT Press, 2001.

[12] Bernhard Schèolkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel methods in computational biology*. The MIT press, 2004.

[13] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Nada Lavrac and Saso Dzeroski. *Inductive logic programming*. E. Horwood, 1994.

[16] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[17] Kevin P Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[18] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[19] Andrew R Webb. *Statistical pattern recognition*. Wiley. com, 2003.

[20] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[21] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[22] D. Baú, A. Martin, C. Mooney, A. Vullo, I. Walsh, and G. Pollastri. Distill: a suite of web servers for the prediction of one-, two-and three-dimensional structural features of proteins. *BMC bioinformatics*, 7(1):402, 2006.

[23] B. Rost, G. Yachdav, and J. Liu. The predictprotein server. *Nucleic acids research*, 31(13):3300–3304, 2003.

[24] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, and Keith Roberts. *Molecular biology of the cell 4th edition*. National Center for Biotechnology InformationÕs Bookshelf, 2002.

[25] Harvey Lodish, Arnold Berk, Chris A. Kaiser, Monty Krieger, Hidde Ploegh Anthony Bretscher and, Angelika Amon, and Matthew P. Scott. *Molecular Cell Biology*. 2012.

[26] William Cohen. *A computer scientist's guide to cell biology: a travelogue from a stranger in a strange land*. Springer, 2007.

[27] David R Bentley, Shankar Balasubramanian, Harold P Swerdlow, Geoffrey P Smith, John Milton, Clive G Brown, Kevin P Hall, Dirk J Evers, Colin L Barnes, Helen R Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218):53–59, 2008.

[28] Christian B Anfinsen. Studies on the principles that govern the folding of protein chains. *Science*, 181:223–30, 1973.

[29] Jeremy M Berg, John L Tymoczko, and Lubert Stryer. Biochemistry. 2002.

[30] LD Hall. Nuclear magnetic resonance. *Advances in Carbohydrate Chemistry*, 19:51–93, 1964.

[31] MS Smyth and JHJ Martin. x ray crystallography. *Molecular Pathology*, 53(1):8–14, 2000.

[32] Hl E Huxley. Electron microscope studies on the structure of natural and synthetic protein filaments from striated muscle. *Journal of molecular biology*, 7(3):281–IN15, 1963.

[33] John-Marc Chandonia and Steven E Brenner. The impact of structural genomics: expectations and outcomes. *Science*, 311(5759):347–351, 2006.

[34] F Ulrich Hartl and Manajit Hayer-Hartl. Converging concepts of protein folding in vitro and in vivo. *Nature structural & molecular biology*, 16(6):574–581, 2009.

[35] Mary-Jane Gething and Joseph Sambrook. Protein folding in the cell. 1992.

[36] Jose Nelson Onuchic and Peter G Wolynes. Theory of protein folding. *Current opinion in structural biology*, 14(1):70–75, 2004.

[37] A Szilágyi, J Kardos, S Osváth, L Barna, and P Závodszky. Protein folding. In *Handbook of Neurochemistry and Molecular Neurobiology*, pages 303–343. Springer, 2007.

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

[39] George A Khoury, Richard C Baliban, and Christodoulos A Floudas. Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. *Scientific reports*, 1, 2011.

[40] W.J. Wedemeyer, E. Welker, M. Narayan, and H.A. Scheraga. Disulfide bonds and protein folding. *Biochemistry*, 39(15):4207–4216, 2000.

[41] Chunaram Choudhary, Chanchal Kumar, Florian Gnad, Michael L Nielsen, Michael Rehman, Tobias C Walther, Jesper V Olsen, and Matthias Mann. Lysine acetylation targets protein complexes and coregulates major cellular functions. *Science*, 325(5942):834–840, 2009.

[42] Christopher M Dobson. Protein folding and misfolding. *Nature*, 426(6968):884–890, 2003.

[43] Kresten Lindorff-Larsen, Peter Rogen, Emanuele Paci, Michele Vendruscolo, and Christopher M Dobson. Protein folding and the organization of the protein topology universe. *Trends in biochemical sciences*, 30(1):13–19, 2005.

[44] AV Finkelstein and OV Galzitskaya. Physics of protein folding. *Physics of Life Reviews*, 1(1):23–56, 2004.

[45] Ken A Dill, S Banu Ozkan, M Scott Shell, and Thomas R Weikl. The protein folding problem. *Annual review of biophysics*, 37:289, 2008.

[46] Walter Pirovano and Jaap Heringa. Protein secondary structure prediction. In *Data Mining Techniques for the Life Sciences*, pages 327–348. Springer, 2010.

[47] Anders Krogh, BjoÈrn Larsson, Gunnar Von Heijne, and Erik LL Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of molecular biology*, 305(3):567–580, 2001.

[48] Craig T Porter, Gail J Bartlett, and Janet M Thornton. The catalytic site atlas: a resource of catalytic sites and residues identified in enzymes using structural data. *Nucleic acids research*, 32(suppl 1):D129–D133, 2004.

[49] Mireia Garcia-Viloca, Jiali Gao, Martin Karplus, and Donald G Truhlar. How enzymes work: analysis by modern rate theory and computer simulations. *Science*, 303(5655):186–195, 2004.

[50] Julian Mintseris and Zhiping Weng. Structure, function, and evolution of transient and obligate protein–protein interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(31):10930–10935, 2005.

[51] Yanay Ofran and Burkhard Rost. Analysing six types of protein–protein interfaces. *Journal of molecular biology*, 325(2):377–387, 2003.

[52] James A Wells and Christopher L McClendon. Reaching for high-hanging fruit in drug discovery at protein–protein interfaces. *Nature*, 450(7172):1001–1009, 2007.

[53] Irina S Moreira, Pedro A Fernandes, and Maria J Ramos. Hot spotsâĂŤa review of the protein–protein interface determinant amino-acid residues. *Proteins: Structure, Function, and Bioinformatics*, 68(4):803–812, 2007.

[54] Hans Frauenfelder. New looks at protein motions. *Nature*, 338:623–624, 1989.

[55] Jörg Gsponer and M Madan Babu. The rules of disorder or why disorder rules. *Progress in biophysics and molecular biology*, 99(2):94–103, 2009.

[56] Recep Colak, TaeHyung Kim, Magali Michaut, Mark Sun, Manuel Irimia, Jeremy Bellay, Chad L Myers, Benjamin J Blencowe, and

Philip M Kim. Distinct types of disorder in the human proteome: functional implications for alternative splicing. *PLoS computational biology*, 9(4):e1003030, 2013.

[57] Yongqi Huang and Zhirong Liu. Smoothing molecular interactions: The âĂIJkinetic bufferâĂİ effect of intrinsically disordered proteins. *Proteins: Structure, Function, and Bioinformatics*, 78(16):3251–3259, 2010.

[58] Steven J Metallo. Intrinsically disordered proteins are potential drug targets. *Current opinion in chemical biology*, 14(4):481–488, 2010.

[59] Vladimir Uversky, Christopher Oldfield, Uros Midic, Hongbo Xie, Bin Xue, Slobodan Vucetic, Lilia Iakoucheva, Zoran Obradovic, and A Keith Dunker. Unfoldomics of human diseases: linking protein intrinsic disorder with diseases. *BMC genomics*, 10(Suppl 1):S7, 2009.

[60] Calvin Yu-Chian Chen and Weng Ieong Tou. How to design a drug for the disordered proteins? *DRUG DISCOVERY TODAY*, 18(19-20):910–915, 2013.

[61] Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, et al. A large-scale evaluation of computational protein function prediction. *Nature methods*, 2013.

[62] Robert Rentzsch and Christine A Orengo. Protein function prediction–the power of multiplicity. *Trends in biotechnology*, 27(4):210–219, 2009.

[63] Tobias Hamp, Rebecca Kassner, Stefan Seemayer, Esmeralda Vicedo, Christian Schaefer, Dominik Achten, Florian Auer, Ariane Boehm, Tatjana Braun, Maximilian Hecht, et al. Homology-based inference sets the bar high for protein function prediction. *BMC bioinformatics*, 14(Suppl 3):S7, 2013.

[64] Ozlem Keskin, Attila Gursoy, Buyong Ma, Ruth Nussinov, et al. Principles of protein-protein interactions: what are the preferred ways for proteins to interact? *Chemical reviews*, 108(4):1225–1244, 2008.

[65] Andrew L Hopkins. Network pharmacology: the next paradigm in drug discovery. *Nature chemical biology*, 4(11):682–690, 2008.

[66] Peter Csermely, Tamás Korcsmáros, Huba JM Kiss, Gábor London, and Ruth Nussinov. Structure and dynamics of molecular networks:

A novel paradigm of drug discovery. *Pharmacology & Therapeutics*, 138(3):333–408, 2013.

[67] Michael P Cary, Gary D Bader, and Chris Sander. Pathway information for systems biology. *FEBS letters*, 579(8):1815–1820, 2005.

[68] Simon C Lovell and David L Robertson. An integrated view of molecular coevolution in protein–protein interactions. *Molecular biology and evolution*, 27(11):2567–2575, 2010.

[69] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[70] Benjamin A Shoemaker and Anna R Panchenko. Deciphering protein–protein interactions. part ii. computational methods to predict protein and domain interaction partners. *PLoS computational biology*, 3(4):e43, 2007.

[71] Iakes Ezkurdia, Lisa Bartoli, Piero Fariselli, Rita Casadio, Alfonso Valencia, and Michael L Tress. Progress and challenges in predicting protein–protein interaction sites. *Briefings in bioinformatics*, 10(3):233–246, 2009.

[72] Jonathan Q Jiang and Maoying Wu. Predicting multiplex subcellular localization of proteins using protein-protein interaction network: a comparative study. *BMC bioinformatics*, 13(Suppl 10):S20, 2012.

[73] Xing-Ming Zhao, Rui-Sheng Wang, Luonan Chen, and Kazuyuki Aihara. Uncovering signal transduction networks from high-throughput data by integer linear programming. *Nucleic acids research*, 36(9):e48–e48, 2008.

[74] Thanh Phuong Nguyen, Tu Bao Ho, et al. Discovering signal transduction networks using signaling domain-domain interactions. *Genome Informatics*, 17(2):35–45, 2006.

[75] Thanh-Phuong Nguyen and Tu-Bao Ho. An integrative domain-based approach to predicting protein–protein interactions. *Journal of Bioinformatics and Computational Biology*, 6(06):1115–1132, 2008.

[76] Vesna Memiševič, Anders Wallqvist, and Jaques Reifman. Reconstituting protein interaction networks using parameter-dependent domain-domain interactions. *BMC Bioinformatics*, 14:154, 2013.

[77] Esa Pitkänen, Juho Rousu, and Esko Ukkonen. Computational methods for metabolic reconstruction. *Current opinion in biotechnology*, 21(1):70–77, 2010.

[78] Nurcan Tuncbag, Attila Gursoy, and Ozlem Keskin. Identification of computational hot spots in protein interfaces: combining solvent accessibility and inter-residue potentials improves the accuracy. *Bioinformatics*, 25(12):1513–1520, 2009.

[79] Anna CF Lewis, Ramazan Saeed, and Charlotte M Deane. Predicting protein–protein interactions in the context of protein evolution. *Molecular BioSystems*, 6(1):55–64, 2010.

[80] Nurcan Tuncbag, Gozde Kar, Ozlem Keskin, Attila Gursoy, and Ruth Nussinov. A survey of available tools and web servers for analysis of protein–protein interactions and interfaces. *Briefings in Bioinformatics*, 10(3):217–232, 2009.

[81] Lucy Skrabanek, Harpreet K Saini, Gary D Bader, and Anton J Enright. Computational prediction of protein–protein interactions. *Molecular biotechnology*, 38(1):1–17, 2008.

[82] Mark N Wass, Alessia David, and Michael JE Sternberg. Challenges for the prediction of macromolecular interactions. *Current opinion in structural biology*, 21(3):382–390, 2011.

[83] Andrew Chatr-aryamontri, Bobby-Joe Breitkreutz, Sven Heinicke, Lorrie Boucher, Andrew Winter, Chris Stark, Julie Nixon, Lindsay Ramage, Nadine Kolas, Lara OâĂŹDonnell, et al. The biogrid interaction database: 2013 update. *Nucleic acids research*, 41(D1):D816–D823, 2013.

[84] Philipp Pagel, Stefan Kovac, Matthias Oesterheld, Barbara Brauner, Irmtraud Dunger-Kaltenbach, Goar Frishman, Corinna Montrone, Pekka Mark, Volker Stümpflen, Hans-Werner Mewes, et al. The mips mammalian protein–protein interaction database. *Bioinformatics*, 21(6):832–834, 2005.

[85] Richard Durbin. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[86] Yana Bromberg and Burkhard Rost. Snap: predict effect of non-synonymous polymorphisms on function. *Nucleic acids research*, 35(11):3823–3835, 2007.

[87] Andreas Wagner. Neutralism and selectionism: a network-based reconciliation. *Nature Reviews Genetics*, 9(12):965–974, 2008.

[88] Emidio Capriotti, Piero Fariselli, Ivan Rossi, and Rita Casadio. A three-state prediction of single point mutations on protein stability changes. *BMC bioinformatics*, 9 Suppl 2:S6, 2008.

[89] Chris J Needham, James R Bradford, Andrew J Bulpitt, Matthew a Care, and David R Westhead. Predicting the effect of missense mutations on protein function: analysis with Bayesian networks. *BMC bioinformatics*, 7:405, 2006.

[90] Yana Bromberg and Burkhard Rost. SNAP: predict effect of non-synonymous polymorphisms on function. *Nucleic acids research*, 35(11):3823–35, January 2007.

[91] Johan A Grahnen, Priyanka Nandakumar, Jan Kubelka, and David A Liberles. Biophysical and structural considerations for protein sequence evolution. *BMC evolutionary biology*, 11(1):361, 2011.

[92] David N Cooper and Michael Krawczak. Human gene mutation database. *Human genetics*, 98(5):629–629, 1996.

[93] Vasily Ramensky, Peer Bork, and Shamil Sunyaev. Human non-synonymous snps: server and survey. *Nucleic acids research*, 30(17):3894–3900, 2002.

[94] Cédric Debès, Minglei Wang, Gustavo Caetano-Anollés, and Frauke Gräter. Evolutionary optimization of protein folding. *PLoS computational biology*, 9(1):e1002861, 2013.

[95] Rafael Sanjuán, Miguel R Nebot, Nicola Chirico, Louis M Mansky, and Robert Belshaw. Viral mutation rates. *Journal of virology*, 84(19):9733–9748, 2010.

[96] Soo-Yon Rhee, Jonathan Taylor, Gauhar Wadhera, Asa Ben-Hur, Douglas L Brutlag, and Robert W Shafer. Genotypic predictors of human immunodeficiency virus type 1 drug resistance. *Proceedings of the National Academy of Sciences of the United States of America*, 103(46):17355–60, November 2006.

[97] Richard O Duda, Peter E Hart, and David G Stork. Unsupervised learning and clustering. *Pattern classification*, page 571, 2001.

[98] Marco Wiering and Martijn van Otterlo. *Reinforcement Learning: State-of-the-art*, volume 12. Springer, 2012.

[99] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[100] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[101] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81, 1997.

[102] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

[103] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

[104] Shigeo Abe. *Support vector machines for pattern classification*. Springer, 2010.

[105] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[106] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels*. The MIT Press, 2002.

[107] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[108] Thorsten Joachims. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 19(4), 1999.

[109] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[110] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[111] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[112] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[113] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels*. 2002.

[114] John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[115] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[116] Thomas Gärtner. *Kernels for structured data*, volume 72. World Scientific, 2008.

[117] Uwe Dick and Kristian Kersting. Fisher kernels for relational data. In *Machine Learning: ECML 2006*, pages 114–125. Springer, 2006.

[118] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[119] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[120] Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, pages 61–72, 1999.

[121] Lorenzo Bruzzone, Mingmin Chi, and Mattia Marconcini. A novel transductive svm for semisupervised classification of remote-sensing images. *Geoscience and Remote Sensing, IEEE Transactions on*, 44(11):3363–3373, 2006.

[122] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[123] Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu. Predicting structured objects with support vector machines. *Communications of the ACM*, 52(11):97–104, 2009.

[124] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[125] Kristiaan Pelckmans, Jos De Brabanter, Johan AK Suykens, and Bart De Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5):684–692, 2005.

[126] Daphne Kollar and Nir Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[127] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[128] Daphne Koller, Nir Friedman, Lise Getoor, and Ben Taskar. Graphical models in a nutshell. *STATISTICAL RELATIONAL LEARNING*, page 13, 2007.

[129] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[130] Kiyoshi Asai, Satoru Hayamizu, and Ken'ichi Handa. Prediction of protein secondary structure by the hidden markov model. *Computer applications in the biosciences: CABIOS*, 9(2):141–146, 1993.

[131] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

[132] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014, 2008.

[133] Kevin P Murphy. Dynamic bayesian networks. *Probabilistic Graphical Models, M. Jordan*, 2002.

[134] Charles Kervrann and Fabrice Heitz. A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics. *Image Processing, IEEE Transactions on*, 4(6):856–862, 1995.

[135] Stan Z Li. *Markov random field modeling in computer vision*. Springer-Verlag New York, Inc., 1995.

[136] Fei Sha and Fernando Pereira. 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*, pages 134–141. Association for Computational Linguistics, 2003.

[137] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. 2007.

[138] Michal Rosen-Zvi, Michael I Jordan, and Alan Yuille. The dlr hierarchy of approximate inference. *arXiv preprint arXiv:1207.1417*, 2012.

[139] Tom Heskes et al. Stable fixed points of loopy belief propagation are minima of the bethe free energy. *Advances in neural information processing systems*, 15:359–366, 2003.

[140] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *Information Theory, IEEE Transactions on*, 51(11):3697–3717, 2005.

[141] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[142] Yee Whye Teh and Michael I Jordan. Hierarchical bayesian nonparametric models with applications. *Bayesian Nonparametrics: Principles and Practice*, pages 158–207, 2010.

[143] Matthew J Beal, Zoubin Ghahramani, and Carl E Rasmussen. The infinite hidden markov model. In *Advances in neural information processing systems*, pages 577–584, 2001.

[144] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[145] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[146] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. *STATISTICAL RELATIONAL LEARNING*, page 339, 2007.

[147] Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *AAAI*, volume 8, pages 1106–1111, 2008.

[148] BTCGD Roller. Max-margin markov networks. *Proc. Advances in Neural Information Processing Systems, MIT Press*, page 25, 2004.

[149] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

[150] Kristian Kersting. Lifted probabilistic inference. In *ECAI*, pages 33–38, 2012.

[151] Hendrik Blockeel. Statistical relational learning. In *Handbook on Neural Information Processing*, pages 241–281. Springer, 2013.

[152] David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, volume 2, pages 259–266, 2002.

[153] Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter Flach, Katsumi Inoue, and Ashwin Srinivasan. Ilp turns 20. *Machine Learning*, 86(1):3–23, 2012.

[154] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[155] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[156] Stephen Muggleton and Cao Feng. Efficient induction of logic programs. *Inductive logic programming*, 38:281–298, 1992.

[157] Stephen Muggleton. Inverse entailment and progol. *New generation computing*, 13(3-4):245–286, 1995.

[158] J. Ross Quinlan and R. Mike Cameron-Jones. Induction of logic programs: Foil and related systems. *New Generation Computing*, 13(3-4):287–312, 1995.

[159] The aleph manual.

[160] Ross D King. Applying inductive logic programming to predicting gene function. *AI Magazine*, 25(1):57, 2004.

[161] Einar Ryeng and Bjorn Kåre Alsberg. Microarray data classification using inductive logic programming and gene ontology background information. *Journal of Chemometrics*, 24(5):231–240, 2010.

[162] Jose A Santos, Houssam Nassif, David Page, Stephen Muggleton, and Michael E Sternberg. Automated identification of protein-ligand interaction features using inductive logic programming: a hexose binding case study. *BMC bioinformatics*, 13(1):162, 2012.

[163] Tuan Nam Tran, Kenji Satou, and Tu Bao Ho. Using inductive logic programming for predicting protein-protein interactions from multiple genomic data. In *Knowledge Discovery in Databases: PKDD 2005*, pages 321–330. 2005.

[164] Paul Finn, Stephen Muggleton, David Page, and Ashwin Srinivasan. Pharmacophore discovery using the inductive logic programming system progol. *Machine Learning*, 30(2-3):241–270, 1998.

[165] Alireza Tamaddoni-Nezhad, Raphael Chaleil, Antonis Kakas, and Stephen Muggleton. Application of abductive ilp to learning metabolic network inhibition from temporal data. *Machine Learning*, 64(1-3):209–230, 2006.

[166] Ross D King, Stephen Muggleton, Richard A Lewis, and MJ Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the national academy of sciences*, 89(23):11322–11326, 1992.

[167] Kazuhisa Tsunoyama, Ata Amini, Michael JE Sternberg, and Stephen H Muggleton. Scaffold hopping in drug discovery using inductive logic programming. *Journal of chemical information and modeling*, 48(5):949–957, 2008.

[168] Thomas G Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.

[169] Luc De Raedt, Bart Demoen, Daan Fierens, Bernd Gutmann, Gerda Janssens, Angelika Kimmig, Niels Landwehr, Theofrastos Mantadelis, Wannes Meert, Ricardo Rocha, et al. Towards digesting the alphabet-soup of statistical relational learning.

[170] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*. Springer, 2008.

[171] Pedro Domingos, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. Unifying logical and statistical ai. In *AAAI*, volume 6, pages 2–7, 2006.

[172] Stephen Muggleton. Learning structure and parameters of stochastic logic programs. In *Inductive Logic Programming*, pages 198–206. Springer, 2003.

[173] Taisuke Sato and Yoshitaka Kameya. Prism: a language for symbolic-statistical modeling. In *IJCAI*, volume 97, pages 1330–1339. Citeseer, 1997.

[174] Noah Goodman, Vikash Mansinghka, Daniel Roy, Keith Bonawitz, and Daniel Tarlow. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*, 2012.

[175] A.S. Juncker, L.J. Jensen, A. Pierleoni, A. Bernsel, M.L. Tress, P. Bork, G. Von Heijne, A. Valencia, C.A. Ouzounis, R. Casadio, et al. Sequence-based feature prediction and annotation of proteins. *Genome Biol*, 10(2):206, 2009.

[176] M.D. Toscano, K.J. Woycechowsky, and D. Hilvert. Minimalist active-site redesign: Teaching old enzymes new tricks. *Angewandte Chemie International Edition*, 46(18):3212–3236, 2007.

[177] R.M. Bush et al. Predicting adaptive evolution. *Nature Reviews Genetics*, 2(5):387–391, 2001.

[178] Predrag Radivojac, Zoran Obradovic, David K Smith, Guang Zhu, Slobodan Vucetic, Celeste J Brown, J David Lawson, and A Keith Dunker. Protein flexibility and intrinsic disorder. *Protein Science*, 13(1):71–80, 2004.

[179] V. Sobolev, E. Eyal, S. Gerzon, V. Potapov, M. Babor, J. Prilusky, and M. Edelman. Space: a suite of tools for protein structure prediction and analysis based on complementarity and environment. *Nucleic acids research*, 33(suppl 2):W39–W43, 2005.

[180] Francis Maes, Julien Becker, and Louis Wehenkel. Iterative multi-task sequence labeling for predicting structural properties of proteins. *ESANN 2011*, 2011.

[181] Carolyn S Sevier and Chris A Kaiser. Formation and transfer of disulphide bonds in living cells. *Nature reviews Molecular cell biology*, 3(11):836–847, 2002.

[182] Philip J Hogg. Disulfide bonds as switches for protein function. *Trends in biochemical sciences*, 28(4):210–214, 2003.

[183] K. Degtyarenko. Bioinorganic motifs: towards functional classification of metalloproteins. *Bioinformatics*, 16(10):851–864, 2000.

[184] A. Rietsch and J. Beckwith. The genetics of disulfide bond metabolism. *Annu Rev Genet.*, 32(1):163–184, 1998.

[185] C. Savojardo, P. Fariselli, M. Alhamdoosh, P.L. Martelli, A. Pierleoni, and R. Casadio. Improving the prediction of disulfide bonds in eukaryotes with machine learning methods and protein subcellular localization. *Bioinformatics*, 27(16):2224–2230, 2011.

[186] A. Pierleoni, P.L. Martelli, P. Fariselli, and R. Casadio. Bacello: a balanced subcellular localization predictor. *Bioinformatics*, 22(14):e408–e416, 2006.

[187] Md M Islam, Xin Yao, and Kazuyuki Murase. A constructive algorithm for training cooperative neural network ensembles. *Neural Networks, IEEE Transactions on*, 14(4):820–834, 2003.

[188] Gianluca Pollastri, Darisz Przybylski, Burkhard Rost, and Pierre Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Structure, Function, and Bioinformatics*, 47(2):228–235, 2002.

[189] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, TN Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[190] S. Griep and U. Hobohm. Pdbselect 1992–2009 and pdbfilter-select. *Nucleic Acids Research*, 38(suppl 1):D318–D319, 2010.

[191] A. Passerini, M. Punta, A. Ceroni, B. Rost, and P. Frasconi. Identifying cysteines and histidines in transition-metal-binding sites using support vector machines and neural networks. *Proteins: Structure, Function, and Bioinformatics*, 65(2):305–316, 2006.

[192] T. Guo, S. Hua, X. Ji, and Z. Sun. Dbsubloc: database of protein subcellular localization. *Nucleic acids research*, 32(suppl 1):D122–D124, 2004.

[193] A. Bairoch, R. Apweiler, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, et al. The universal protein resource (uniprot). *Nucleic acids research*, 33(suppl 1):D154–D159, 2005.

[194] A.C.R. Martin. Mapping pdb chains to uniprotkb entries. *Bioinformatics*, 21(23):4297–4301, 2005.

[195] Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann, et al. Hidden markov support vector machines. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 3, 2003.

[196] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

[197] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9:768–786, 1998.

[198] A. Vullo and P. Frasconi. Disulfide connectivity prediction using recursive neural networks and evolutionary information. *Bioinformatics*, 20(5):653–659, 2004.

[199] J. Dyrlov Bendtsen, H. Nielsen, G. von Heijne, and S. Brunak. Improved prediction of signal peptides: Signalp 3.0. *Journal of molecular biology*, 340(4):783–795, 2004.

[200] Parag Singla and Pedro Domingos. Markov logic in infinite domains. *arXiv preprint arXiv:1206.5292*, 2012.

[201] Armin Biere. *Handbook of satisfiability*, volume 185. IOS Press, 2009.

[202] R Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1319–1325. Citeseer, 2005.

[203] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.

[204] Brian Milch, Luke S Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *Aaai*, volume 8, pages 1062–1068, 2008.

[205] Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *AAAI*, volume 5, pages 868–873, 2005.

[206] Tuyen N Huynh and Raymond J Mooney. Max-margin weight learning for markov logic networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 564–579. Springer, 2009.

[207] Stanley Fields and O Song. A novel genetic system to detect protein protein interactions. *Nature*, 340(6230):245–246, 1989.

[208] Benjamin A Shoemaker and Anna R Panchenko. Deciphering protein–protein interactions. part i. experimental techniques and databases. *PLoS computational biology*, 3(3):e42, 2007.

[209] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46, 2005.

[210] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.

[211] Howook Hwang, Brian Pierce, Julian Mintseris, Joël Janin, and Zhiping Weng. Protein–protein docking benchmark version 3.0. *Proteins: Structure, Function, and Bioinformatics*, 73(3):705–709, 2008.

[212] Bin Li and Daisuke Kihara. Protein docking prediction using predicted protein-protein interface. *BMC bioinformatics*, 13(1):7, 2012.

[213] Yanjun Qi and William Stafford Noble. Protein interaction networks: Protein domain interaction and protein function prediction. In *Handbook of Statistical Bioinformatics*, pages 427–459. 2011.

[214] Joel R Bock and David A Gough. Predicting protein–protein interactions from primary structure. *Bioinformatics*, 17(5):455–460, 2001.

[215] Yanjun Qi, Oznur Tastan, Jaime G Carbonell, Judith Klein-Seetharaman, and Jason Weston. Semi-supervised multi-task learning for predicting interactions between hiv-1 and human proteins. *Bioinformatics*, 26(18):i645–i652, 2010.

[216] Zhu-Hong You, Ying-Ke Lei, Jie Gui, De-Shuang Huang, and Xiaobo Zhou. Using manifold embedding for assessing and predicting protein interactions from high-throughput experimental data. *Bioinformatics*, 26(21):2744–2751, 2010.

[217] Zheng Xia, Ling-Yun Wu, Xiaobo Zhou, and Stephen TC Wong. Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. *BMC systems biology*, 4(Suppl 2):S6, 2010.

[218] Thanh-Phuong Nguyen and Tu-Bao Ho. Detecting disease genes based on semi-supervised learning and protein–protein interaction networks. *Artificial Intelligence in Medicine*, 54(1):63–71, 2012.

[219] Luc De Raedt. *Inductive logic programming*. 2010.

[220] Alvaro J González, Li Liao, and Cathy H Wu. Prediction of contact matrix for protein–protein interaction. *Bioinformatics*, 2013.

[221] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[222] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[223] Shawn M Gomez, William Stafford Noble, and Andrey Rzhetsky. Learning to predict protein–protein interactions from protein sequences. *Bioinformatics*, 19(15):1875–1881, 2003.

[224] M. Götte, X. Li, and M.A. Wainberg. Hiv-1 reverse transcription: a brief overview focused on structure-function relationships among molecules involved in initiation of the reaction. *Archives of biochemistry and biophysics*, 365(2):199–210, 1999.

[225] Zhi Wei Cao, Lian Yi Han, Chan Juan Zheng, Zhi Lang Ji, Xin Chen, Hong Huang Lin, and Yu Zong Chen. Computer prediction of drug resistance mutations in proteins REVIEWS. *Drug Discovery Today: BIOSILICO*, 10(7), 2005.

[226] Donn N Rubingh. Protein engineering from a bioindustrial point of view. *Current Opinion in Biotechnology*, 8(4):417–422, 1997.

[227] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *University Computing*, pages 629–682, 1994.

[228] Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi. kfoil: learning simple relational kernels. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, pages 389–394. AAAI Press, 2006.

[229] R.W. Shafer. Rationale and uses of a public hiv drug-resistance database. *Journal of Infectious Diseases*, 194(Supplement 1):S51–S58, 2006.

[230] Elisa Cilia, Stefano Teso, Sergio Ammendola, Tom Lenaerts, and Andrea Passerini. Predicting virus mutations through relational learning. In *Proceedings of the ECCB Workshop on Annotation, Interpretation and Management of Mutations (AIMM)*, 2012.

[231] E. De Clercq. Hiv inhibitors targeted at the reverse transcriptase. *AIDS research and human retroviruses*, 8(2):119–134, 1992.

[232] R.A. Spence, W.M. Kati, K.S. Anderson, and K.A. Johnson. Mechanism of inhibition of hiv-1 reverse transcriptase by nonnucleoside inhibitors. *Science*, 267(5200):988–993, 1995.

[233] Lothar Richter, Regina Augustin, and Stefan Kramer. Finding Relational Associations in HIV Resistance Mutation Data. In *Proceedings of Inductive Logic Programming (ILP)*, volume 9, 2009.

[234] M.J. Betts and R.B. Russell. Amino-acid properties and consequences of substitutions. *Bioinformatics for geneticists*, pages 311–342, 2003.

[235] W. R. Taylor. The classification of amino acid conservation. *Journal of Theoretical Biology*, 119(2):205–218, March 1986.

[236] Marco Punta, Penny C Coggill, Ruth Y Eberhardt, Jaina Mistry, John Tate, Chris Boursnell, Ningze Pang, Kristoffer Forslund, Goran Ceric,

Jody Clements, et al. The pfam protein families database. *Nucleic acids research*, 40(D1):D290–D301, 2012.

[237] Christian JA Sigrist, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Virginie Bulliard, Amos Bairoch, and Nicolas Hulo. Prosite, a protein domain database for functional characterization and annotation. *Nucleic acids research*, 38(suppl 1):D161–D166, 2010.

[238] GJ Bartlett, CT Porter, N Borkakoti, and JM Thornton. Analysis of catalytic residues in enzyme active sites. *Journal of Molecular Biology*, 324(1):105–121, 2002.

[239] Narayanan Eswar, Ben Webb, Marc A Marti-Renom, MS Madhusudhan, David Eramian, Min-yi Shen, Ursula Pieper, and Andrej Sali. Comparative protein structure modeling using modeller. *Current Protocols in Protein Science*, pages 2–9, 2007.

[240] Frances C Bernstein, Thomas F Koetzle, Graheme JB Williams, Edgar F Meyer, Michael D Brice, John R Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. The protein data bank. *European Journal of Biochemistry*, 80(2):319–324, 2008.

[241] Stephen Muggleton. Learning from positive data. In *Inductive Logic Programming Workshop*, pages 358–376, 1996.

[242] Niels Landwehr, Andrea Passerini, Luc Raedt, and Paolo Frasconi. Fast learning of relational kernels. *Mach. Learn.*, 78(3):305–342, March 2010.

[243] Diane E Bennett, Ricardo J Camacho, Dan Otelea, Daniel R Kuritzkes, Hervé Fleury, Mark Kiuchi, Walid Heneine, Rami Kantor, Michael R Jordan, Jonathan M Schapiro, Anne-Mieke Vandamme, Paul Sandstrom, Charles a B Boucher, David van de Vijver, Soo-Yon Rhee, Tommy F Liu, Deenan Pillay, and Robert W Shafer. Drug resistance mutations for surveillance of transmitted HIV-1 drug-resistance: 2009 update. *PloS one*, 4(3):e4724, 2009.

[244] Koen Deforche, Ricardo J Camacho, Zehave Grossman, Marcelo a Soares, Kristel Van Laethem, David a Katzenstein, P Richard Harrigan, Rami Kantor, Robert Shafer, and Anne-Mieke Vandamme. Bayesian network analyses of resistance pathways against efavirenz and nevirapine. *AIDS (London, England)*, 22(16):2107–15, October 2008.

[245] Elisa Cilia, Neils Landwehr, and Andrea Passerini. Relational feature mining with hierarchical multitask kfoil. *Fundamenta Informaticae*, 113(2):151–177, December 2011.

[246] Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*, 2012.