UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Multi-target Prediction Methods for Bioinformatics

Approaches for Protein Function Prediction and Candidate Discovery for Gene Regulatory Network Expansion

Luca Masera

Advisor
Prof. Prof. Enrico Blanzieri
Università degli Studi di Trento

June 23, 2019

Abstract

Biology is experiencing a paradigm shift since the advent of next generation sequencing technologies. The retrieved data largely exceeds the capability of biologists to investigate all possibilities in the laboratories, hence predictive tools able to guide the research are now a fundamental component of their workflow. Given the central role of proteins in living organisms, in this thesis we focus on their functional analysis and the intrinsic multi-target nature of this task. To this end, we propose different predictive methods, specifically developed to exploit side knowledge among target variables and examples.

As a first contribution we face the task of protein-function prediction and more in general of hierarchical-multilabel classification (HMC). We present Ocelot *a predictive pipeline for genome-wide protein characterization. It relies on a statistical-relational-learning tool, where the knowledge on the input examples is coded by the combination of multiple kernel matrices, while relations among target variables are expressed as logical constraints. Both, the mislabeling of examples and the infringement of logical rules are penalized by the loss function, but* Ocelot *do not forces hierarchical consistency. To overcome this limitation, we present* AWX*, a neural-networks output-layer that guarantees the formal consistency of HMC predictions.*

The second contribution is VSC*, a binary classifier designed to incorporate the concepts of subsampling and locality in the definition of features to be used as the input of a perceptron. A locality-based confidence measure is used to weight the contribution of maximum-margin hyper-planes built by subsampling pairs of examples of opposite class. The rationale is that local methods can be exploited when a multi-target task is expected, but not reflected in the annotation space.*

The third and last contribution are NES^2RA *and* OneGenE*, two approaches for finding candidates to expand known gene regulatory networks.* NES^2RA *adopts variable-subsetting strategies, enabled by volunteer distributed computing, and the PC algorithm to discover candidate causal relationships within each subset of variables. Then, ranking aggregators combine the partial results into a single ranked candidate genes list.* OneGenE *overcomes the main limitation of* NES^2RA*, i.e. latency, by precomputing candidate expansion lists for each transcript of an organism that are then aggregated on-demand.*

Keywords

Multi-target prediction; Hierarchical-multilabel classification; Protein functional analysis; Network expansion; Volunteer distributed computing

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Prof. Enrico Blanzieri, who guided and motivated me through all my Ph.D study, providing me precious ideas and advises not only work-wise. As a natural continuation, I would like to say a big thank the gene@home team, the collaborators at the Edmund Mach Foundation, as well as to Gabriela Viero and Andrea Passerini for having inspired me to purse the path of machine learning and bioinformatics. Our meetings probably were not the most time-efficient, but surely we have had good times with Valter's jokes and stories. Thank you.

A sincere acknowledgements to Davide Bacciu and Gianluca Pollastri for having read and reviewed this thesis. Your comments have been precious to highlight weak points and possible way of improvement. Moreover, I would like to thank the ICT Doctoral School secretariat staff for all the patience and help they gave me in the past years.

My deepest gratitude goes to my colleagues, that helped me survive these three years. The coffee-break at 4pm, the infinite walks in the corridors of DISI, the eternal sessions of pair programming and article writing, and the "intellectual" nights. I will miss all of this. A special acknowledgment is reserved for Daniele, who has been a fantastic friend to me. We shared unforgettable experiences in Canada and all around Trentino Alto-Adige. Thank you for having awakened my passion for mountains.

When you move to an apartment with three roommates, things can go terribly wrong. Fortunately that was not the case, and I found three splendid friends to share "la tana di Manci". My Ph.D. study would not have been the same without you guys, I am already looking forward to our next session of D&D.

After the new friends, I would like to acknowledge the old good PARVs. We changed and grew a lot in the last years, spreading ourselves even wider around the globe (thank you D. for being PARV-ambassador in North-America). Nonetheless we always find ways to share fantastic experiences and adventures, that keep us as close as at the time we shared the lanes on the athletic field. Our discussions and meetings are always a precious source of ideas and inspiration. Thank you all for being so non-representative of the average population.

I would like to say that most of the concepts and ideas expressed in this thesis come from the twelve or more hours of driving (Trento $\leftrightarrow$ Zurich) that I have done in uncountable weekends during these years, but that would simply be a lie. Our weekend migrations were not very relaxing, and probably affected our performance on Monday, but we held on. Thank you Sara for having come all along this journey at my side and having been "pazientissima". We finally made it.

In ultimo non posso che esprimere tutta la mia gratitudine alla mia famiglia. Mi avete accompagnato in questi i miei 20 e più anni di formazione, senza mai farmi mancare il vostro supporto, che le cose andassero bene o meno. "Lo studio è il tuo lavoro" mi dicevate quando non ero che un giovanotto, evidentemente ci vedevate più lontano di quanto potessi fare io.

Luca Masera

Contents

1 Introduction **1**

1.1 Contributions 3

1.2 Structure of the Thesis 5

1.3 Personal Contributions 6

2 Background **9**

2.1 Biological Background 9

2.1.1 From DNA to Protein 9

2.1.2 Regulation of Gene Expression 12

2.1.3 Transcriptomics technologies 15

2.1.4 Protein Structure 16

2.1.5 Gene Ontology 18

2.2 Kernel Methods 19

2.2.1 Semantic Based Regularization 20

2.3 Artificial Neural Networks 22

2.4 PC Algorithm 25

2.5 Ranking aggregators 26

2.6 BOINC 28

3 Multi Target Prediction **29**

3.1 Hierarchical Multilabel Classification 29

3.1.1 Formalization 31

3.1.2 True Path Rule 32

3.1.3 Evaluation metrics 32

3.2 Candidate Discovery for Network Expansion 34

3.2.1 Formalization 35

4 Combining Learning and Logical Constraints for Hierarchical Multilabel Classification of Protein Functions 37
4.1 Related Work . . . 39
4.2 Model Description . . . 40
4.2.1 Overview of the Prediction Pipeline . . . 40
4.2.2 Rules . . . 43
4.3 Results . . . 45
4.3.1 Data Processing . . . 45
4.3.2 Empirical analysis . . . 48
4.4 Conclusion . . . 53

5 Consistent Hierarchical-Multilabel Classification with Artificial Neural Networks 55
5.1 Model description . . . 56
5.2 Generalized TPR . . . 58
5.2.1 The gTPR holds for AWX . . . 59
5.2.2 Implementation . . . 59
5.3 Experimental setting . . . 60
5.4 Results . . . 62
5.5 Conclusion . . . 65

6 Binary Classification from Unknown Multilabel Annotation Space 67
6.1 Very Simple Classifier . . . 70
6.1.1 Hyperplane selection . . . 71
6.1.2 Hyperplane confidence . . . 71
6.1.3 Learning the hyperplane weights . . . 72
6.1.4 Characterization of the confidence in terms of Chebichev inequality . . . 73
6.2 Results . . . 75
6.2.1 Experimental setup . . . 75
6.2.2 Experiment 1 . . . 75
6.2.3 Experiment 2 . . . 77
6.2.4 Discussion . . . 79
6.3 Conclusion . . . 81

7 Gene Regulatory Network Expansion by Stratified variable Subsetting and Ranking Aggregation 83
7.1 Related Work . . . 84

7.2 NES²RA 84
7.3 NES²RA on the gene@home BOINC project 89
7.4 Evaluation 91
7.5 Conclusion 96

8 OneGenE: Regulatory Gene Network Expansion via Distributed Volunteer Computing on BOINC 99
8.1 OneGenE 100
8.2 Implementation 102
8.2.1 Performance 102
8.2.2 Benchmarks 104
8.2.3 Computational power and drawbacks 106
8.3 *Pseudomonas aeruginosa* 107
8.3.1 Experimental results 108
8.4 Conclusion 111

9 Conclusions 113

10 Publications 117

11 Released Software 119

Bibliography 121

List of Tables

5.1 Details of the benchmark datasets used in the experiments 60
5.2 ANN architecture used in the experiments 60
5.3 Performance of AWX with AUC($\overline{\text{PR}}$) 63
5.4 Performance of AWX with $\overline{\text{AUCPR}}$ 64
5.5 Performance of AWX with $\overline{\text{AUCPR}}_w$ 65

6.1 Details of the used datasets 76
6.2 Parameters' values used in the grid search 77
6.3 Average results of Experiment 1 78
6.4 Ranks the classifiers in Experiment 1 79

7.1 Performance comparison between `skeleton` and PC++ 91
7.2 Candidate genes list of the FOS LGN of *A. thaliana* produced by NES^{2}RA 93
7.3 NES^{2}RA precision performance using different aggregation methods 93
7.4 Ranked lists of the gadW LGN of *E. coli* 95
7.5 Cumulative BOINC statistics for the *E. coli* experiments. 96
7.6 Statistics of the workunits computational costs 96
7.7 Detailed BOINC statistics for NES^{2}RA Π_{L} on the *E. coli* data set. . 97

8.1 Optimization of the executable of the OneGenE application 104
8.2 Benchmark table for the OneGenE application 104
8.3 Top 10 candidate genes OneGenE 109

List of Figures

2.1 Central Dogma of Molecular Biology 10
2.2 Schematic representation of a gene regulatory network 14
2.3 Protein structure 17
2.4 Gene Ontology DAG 18

3.1 True path rule examples 30
3.2 Finding candidate for network expansion 35

4.1 Depiction of the Ocelot decision making process 41
4.2 Overall performance of all prediction methods on the Yeast dataset . 50
4.3 Breakdown of the performance of all methods at different GO term depth 51
4.4 Overall performance of DeepGO, Ocelot, GoFDR and the baseline on the Yeast dataset 52

5.1 From hierarchy to AWX 56
5.2 Comparison of `max` and ℓ-norms 57
5.3 Generalized true path rule example 58
5.4 Terms and leaves distribution by depth 61

6.1 Schematic representation of the hyperplane selection 71
6.2 Shape of the confidence measure 72
6.3 Results of Experiment 1 80
6.4 Results of Experiment 2 80

7.1 NES^{2}RA workflow. 85

8.1 Blocks scheme of the OneGenE architecture 101
8.2 Computing status page of the gene@home project 105
8.3 Estimated FLOPS per day for gene@home project 106
8.4 Output similarity by aggregator 110

Chapter 1

Introduction

The marriage between computer science and biology has a 60 years long history. Since the identification of the DNA as the molecule that contains the genetic information [Watson et al., 1953] and the works by Sanger to sequence proteins [Sanger, 1945] and DNA [Sanger et al., 1977], it was clear the necessity to systematically store and share the retrieved data. The *Atlas of Protein Sequence and Structure* by Margaret Dayhoff laid the foundations for the modern biological data banks by collecting all the available sequences, at that time 65, in a book that was published in 1965. This paved the way to subsequent works that aimed at finding similarities in biological sequences [Needleman and Wunsch, 1970; Smith and Waterman, 1981], and building evolution-based substitution matrices [Dayhoff and Schwartz, 1978]. On top of these milestones, the Human Genome Project (HGP) was formally launched in 1990 giving start to the era of *bionformatics*.

This ambitious project required more than ten years and $0.5 - 1$ billion dollars to be accomplished [Reuter et al., 2015], but in 2003 the complete human genome was finally unveiled. However, the Sanger sequencing technology used for the HGP was not scalable enough to be systematically applied on new organisms. To this end, the National Human Genome Research Institute created a 70 million dollars initiative to reduce the sequencing cost to $1000. This, and other large investments, led to the introduction of a plethora of high-throughput sequencing (HTS) or next generation sequencing (NGS) protocols, reducing the cost per sequenced megabase (one million DNA bases) from almost $10000 in 2001 to less then $0.1 nowadays[1]. NGS technologies represent one of the biggest breakthrough in biology since the work of Watson et al. [1953], opening the gates to previously unimaginable scenarios.

Proteins are among the most important molecules in living organisms. Their structure and capability to bond with other bio-molecules make protein extremely

[1]Data available at `https://www.genome.gov/sequencingcostsdata/`

versatile building blocks whose space of activity ranges from the very basic enzymatic reactions up to structural support of cells. Despite the fact that the genetic material required to assemble proteins is the same in each cell of an organism[2], the regulation of gene expression controls the relative abundance of proteins inducing cell differentiation and reaction to stimuli.

The amino-acid sequences of proteins (proteins are made of a combination of 20 different amino-acids) for the commonly studied organisms are publicly available on different data banks online, but also the *de-novo* sequencing of new organisms is feasible even for relatively small laboratories. Despite this, the functional annotation of these macro molecules is still an open issue. Indeed, the function accomplished by a protein is strongly related to its three-dimensional structure that is typically inquired through X-ray crystallography [Engh and Huber, 1991]. At the time of writing, 45970 distinct protein-sequences 3D structures are available, while more than 100 millions proteins have a known amino-acid sequence (500 thousand of which are manually curated)[3]. NGS protocols widened the gap between retrieved and experimentally-annotated data. Indeed, wet-lab procedures to precisely investigate molecules properties still require hours or days to be accomplished. For example, the protocol to verify experimentally the interaction between two proteins needs more then 2 days[4] and testing all pairs of proteins even in a small organism is therefore far from possible. Computational methods that can guide the work of biologist identifying the most likely targets are therefore of crucial importance, and are nowadays an essential component of the biological research.

The possibility to code protein sequences as *strings*, whose alphabet is the set of amino acids, was extremely tempting for computer scientists. Many computational approaches have been developed to face the supervised task of protein-function prediction (PFP), exploiting similarities of protein sequences to *transfer* annotation from known to unknown ones [Gong et al., 2016; Kulmanov et al., 2018]. The performance of this class of algorithm is good as long as proteins are evolutionary related, homologous, and performs better for predicting molecular functions and cellular component rather then biological processes[5] [Dessimoz et al., 2013; Rost et al., 2003]. More sophisticated approaches rely on the aggregation of different data sources [Yu et al., 2016; Li et al., 2016; Stuart et al., 2003] to cover a wider range of biological aspects.

Microarrays and RNA-seq technologies (see Section 2.1.3 for details) allowed bi-

[2]Exception made for haploid cells in Eukaryotes.
[3]Data from `https://www.rcsb.org/stats/` and `https://www.uniprot.org/`
[4]Time for co-immunoprecipitation protocol [Anderson, 1998]
[5]See Section 2.1.5 for definitions.

ologists to simultaneously measure gene expression levels of (all) genes transcripts from a population of cells. From the measurement of gene expression in different conditions are obtained observational data. On top of this data computational methods that capture causal relations among gene products are of the foremost importance. Simple correlation between pairs of transcripts is still widely used by biologists to explore and visualize high-dimensional data [Butte et al., 2000; Oldham et al., 2006], but it fails in distinguishing direct relations from indirect ones [Opgen-Rhein and Strimmer, 2007]. Taking into consideration the expression levels of the other transcript is therefore necessary to provide the user with more accurate predictions [Maathuis et al., 2010]. A wide range of pathologies is caused by the misregulation of specific genes or pathways, that may be over or under expressed [Lee and Young, 2013]. Being able to predict the genes that affect the regulation of those pathways is therefore crucial to identify new drugs targets [Imoto et al., 2007; Emmert-Streib et al., 2014; Aloraini and ElSawy, 2018].

1.1 Contributions

HTS protocols deeply shaped biological research in the last twenty years, generating huge amount of raw data. Providing accurate insight to biologists to guide their and accelerate the research process is therefore of the foremost importance. However, going beyond the standard binary classification framework is critical to improve prediction quality [Waegeman et al., 2018]. For this reason we face the task of multi-target prediction in bioinformatics. The contribution of this thesis is threefolds.

The first contribution is on the task of hierarchical-multilabel classification (HMC) that, as mentioned in the previous section, plays a crucial role in the protein function prediction (PFP) problem. To this aim we propose two solutions, the first one is task-driven, the second one is methodological. We propose Ocelot, a predictive pipeline specifically developed for genome-wide protein annotation. It is based on SBR, a state of the art statistical relational learning framework that incorporates fuzzy-logical constraints in the kernel machinery. The knowledge on the label structure and the protein-protein interactions are formulated as first-order logic rules and used as penalty in the model training. The genome-wide application of Ocelot on the model organism *Saccharomyces cerevisiae* showed that the proposed kernel, based on multiple biological features, and rules positively affect the predictions, outperforming the baseline and the state-of-the-art when trained in the same setting. It must be noticed that Ocelot penalizes the infringement of hierarchical con-

straints, but does not guarantee a formal consistency on the final result. In order to overcome this limitation, and the poor scalability, we propose AWX (Adjacency Wrapping Matrix), a novel neural network output layer. We generalized the true path rule to continuous prediction and proved that AWX predictions are consistent for each possible threshold. AWX performs the prediction on the leaf-nodes of the hierarchy and combine them with different aggregation strategies to obtain the prediction of the inner-nodes. The whole output is then jointly optimized by means of stochastic gradient descent. Thanks to the modularity of neural networks (NN) architectures, AWX is completely agnostic to the underlying topology and can therefore be employed on top of any deep NN. An extensive benchmark evaluation confirmed a significant improvement in performance of AWX with respect to state-of-the-art HMC-specific approaches.

Multi-target classification has multiple fields of application, but the annotation space is not always given. The second contribution of the thesis is VSC (Very Simple Classifier), a binary classifier that exploits the concept of locality to provide accurate classification also when a multi-target annotation space is expected but unknown. The idea behind locality is that decisions based on the portion of space "close" to the target example are preferable to the ones that take into consideration the whole input space [Bottou and Vapnik, 1992]. To incorporate this concept, VSC starts by sampling pairs of training examples belonging to opposite classes and for each of them builds the maximum margin hyper-plane. When a new example is evaluated, the pre-computed separation hyper-planes are weighted by means of a learned score and a locality-based confidence measure. We performed an extensive experimental evaluation on benchmark datasets and compared the results of VSC with other 10 general-purpose binary classifiers. VSC has competing performance on the tested datasets especially when the input has very distinct clusters within the same class. Moreover we show that the confidence measure plays an important role in the quality of the predictions.

The third and last contribution of the thesis regards the task of finding candidates to expand known gene regulatory networks. To this end we developed NES^{2}RA, a predictive pipeline capable of facing this task also for large scale genomes, as plants or human ones. NES^{2}RA takes as input a gene expression data matrix and a set of variables (transcripts) that are the target of a biological investigation. The transcriptome of the analyzed organism is subdivided in (almost) disjoint subsets (called tiles) where appearance probability of the input transcripts in each tile is controlled by a probability vector. The `skeleton` function of the PC algorithm [Kalisch and Bühlmann, 2007] is applied on the expression data of each tile to systematically

test conditional independence of variable pairs. This process is iterated thousand of times with different parameters' sets and the obtained partial results are aggregated by means of ranking aggregators. This task is computationally intensive and could require years on a single machine. Given the highly parallel nature of the algorithm we developed *gene@home*, a distributed volunteer research project based on the BOINC platform [Anderson, 2004b]. *gene@home* has an average throughput over 10 TFLOPs that allows us to drastically reduce the computation time of the ranked expansion lists. NES^{2}RA has been successfully applied on four GRN of *Vitis vinifera* related to climate change [Malacarne et al., 2018]. Despite the huge boost provided by the BOINC platform, NES^{2}RA computation is definitely not real-time and is therefore the best choice for an exploratory research. To overcome this issue we developed OneGenE, an evolution of NES^{2}RA, where partial expansion lists are pre-computed for each transcript and combined on the fly given a set of input variables. Up to now, we have pre-computed expansion lists for two bacterial organisms and two plants, respectively *Pseudomonas aeruginosa*, *Escherichia coli*, *Arabidopsis thaliana*, and *Vitis vinifera*. The expansion of the human transciptome in currently undergoing.

1.2 Structure of the Thesis

This chapter presents the motivations and the contributes achieved in this thesis. The rest of the manuscript is organized as follows.

Chapter 2 collects the biological and computational background useful for the thesis, while chapter 3 presents and formalize the task of multi-target prediction, with a focus on the tasks of hierarchical-multilabel classification and finding candidate for network expansion.

Chapter 4 and 5 present our contributes on hierarchical-multilabel classification applied to protein function prediction. Chapter 4 discusses Ocelot, a predictive pipeline for protein function prediction that has been published in:

> Teso, Stefano; Masera, Luca; Diligenti, Michelangelo, and Passerini, Andrea. Combining learning and constraints for genome-wide protein annotation. *BMC Bioinformatics*, 20(1):338, 2019

Chapter 5 discusses AWX, an artificial neural network component for consistent hierarchical-multilabel classification, presented in:

Masera, Luca and Blanzieri, Enrico. AWX: An Integrated Approach to Hierarchical-Multilabel Classification. In *Machine Learning and Knowledge Discovery in Databases 2018, Proceedings, Part I*, pages 322–336. Springer International Publishing, 2019a

Chapter 6 discusses VSC, a binary classifier that exploits locality for investigating problems where a multi-target structure is supposed but not known. The article has been presented in:

Masera, Luca and Blanzieri, Enrico. Very Simple Classifier: a Concept Binary Classifier to Investigate Features Based on Subsampling and Locality. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2019*, 2019b

Chapter 7 and 8 presents our contributes on finding candidate genes for network expansion. Chapter 7 discusses NES^{2}RA, our solution based on volunteer distributed computing for finding candidate genes that expand known gene regulatory networks. The work has already been published in:

Asnicar, Francesco; Masera, Luca; Coller, Emanuela; Gallo, Caterina; Sella, Nadir; Tolio, Thomas; Morettin, Paolo; Erculiani, Luca; Galante, Francesca; Semeniuta, Stanislau; Malacarne, Giulia; Engelen, Kristof; Argentini, Andrea; Cavecchia, Valter; Moser, Claudio, and Blanzieri, Enrico. NES2RA: Network Expansion by Stratified Variable Subsetting and Ranking Aggregation. *The International Journal of High Performance Computing Applications*, 32(3):380–392, aug 2016

Chapter 7 discusses OneGenE; based on NES^{2}RA, it aims at overcoming the main limitations of its predecessor. The article has been presented in:

Asnicar, Francesco; Masera, Luca; Pistore, Davide; Valtentini, Samuel; Cavecchia, Valter, and Blanzieri, Enrico. OneGene: Regulatory Gene Network Expansion via Distributed Volunteer Computing on BOINC. In *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2019

1.3 Personal Contributions

I am the second author of the submitted article discussed in Chapter 4. The work started in my master project [Masera, 2015]; I formulated the first order logic rules that describe the hierarchical relations, performed the experimental comparison with

related works, and contributed to the writing of the article.

I am first author of the published article discussed in Chapter 5. I conceived the model, implemented it, performed the experimental evaluation and contributed to the writing of the article.

I am first author of the submitted article discussed in Chapter 6. I contributed in the formulation of the model, implemented it, performed the experimental evaluation, and contributed to the writing of the article.

I am co-first author of the accepted articles discussed in Chapter 7 and 8. I contributed in the formulation of the pipelines, to their implementation, and contributed to the writing of the articles.

Chapter 2

Background

2.1 Biological Background

This section will introduce the biological background related to the work presented in the thesis, focusing on the most crucial processes and molecules that are the basis of life. Firstly we will follow the flux of genetic information, described by the central dogma of molecular biology, with a deeper look into the gene expression regulation mechanisms and the final product, i.e. proteins. Finally we will introduce the techniques employed to retrieve the data from biological samples and the bioinformatic resources that store and provide this information.

The concepts discussed in this section are taken from [Alberts et al., 2015].

2.1.1 From DNA to Protein

Proteins are the most important part of the cells and are involved in almost all vital processes of the cell and of the organism. They are essential for several crucial roles such as structural functions, movement, transport and storage of molecules, regulation of several processes and even more complex roles in big organisms such as humans with also other crucial functions as in body-protection (antibodies).

Proteins are large molecules with a chain form, composed of a series of smaller units called amino acids. There are 20 different types of amino acids (or actually 23, including Selenocysteine, Pyrrolysine and N-Formylmethionine, which are coded only in some species and are interpreted as stop codons). The different sequence of these 20 amino acids in every protein determines the way the protein will fold into her unique 3-dimensional structure which consequently determines its specific function. Being able to predict the folding of a protein knowing its sequence could therefore help in also predicting the function of the protein.

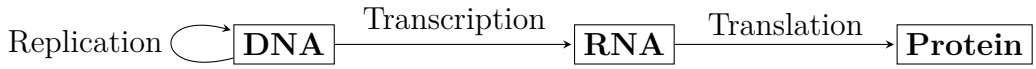


Figure 2.1: **Central Dogma of Molecular Biology**. The Central Dogma of Molecular Biology describes the three fundamental processes for life.

All cells and organisms (bacterial, virus and human) produce and use proteins. To build them every cell needs the information that is stored in the nucleic acids (RNA or DNA) and a process to translate this information from nucleic acids to amino acids.

The same type of process to read and translate the information from DNA to protein is used, with only secondary differences, in every cell, from bacterial to human, representing therefore a fundamental process, explained by the central dogma of the molecular biology (Figure 2.1). This process consists of two steps: transcription and translation, which together constitute the phenomenon of gene expression.

Transcription

The DNA is the holder of all information of the organism, including information required to build proteins. In 1950s Watson and Crick discovered the structure of DNA: a double helix composed from two complementary chains of nucleotides, bound together with hydrogen bonds. Four different nucleotides (cytosine, guanine, adenine and thymine) are bound together in a specific sequence to form a single strand DNA. This chain is then bound with hydrogen bonds according to the base pairing (adenine bonds thymine and cytosine bonds guanine) to form the double stranded DNA. In this double stranded DNA dwells all the information needed for the life of the cell and the organism. This information needs nevertheless to be translate in a structure which can suit the needs of the cell, i.e. from nucleotides to amino acids.

The first step of this process is called transcription, since the information hold in a DNA-gene is re-written in a similar code, called RNA, using similar nucleotides. The difference in the molecular structure of the two nucleic acids (DNA and RNA) is due to a different sugar: DNA contains the sugar deoxyribose, while RNA contains the sugar ribose. The languages of the two nucleic acids is also slightly different since RNA uses the base uracil instead of thymine, used in DNA.

The information stored in the DNA-gene needs to be accessible to start the process of transcription. Since this information could not be read if the two DNA

chains were bond, the DNA double helix must unwind in a region very closed to the gene that has to be transcribed: this region is called transcription bubble. The transcription is performed by enzymes called RNA polymerases, which link the DNA nucleotides in special recognizable regions called promoters and use the DNA sequence as template to form a RNA strand. The first site of transcription is called initiation site and represents the first nucleotides being transcribed from DNA into RNA.

There are slight differences between the transcription process in bacteria and in eukaryotes.

In bacteria, group of genes are transcribed together, starting from a common promoter. The bacterial polymerase recognizes and binds the promoter, opens the double stranded DNA and start the transcription. This promoter zone is mostly composed from several thymine and adenine bases, since they are easier to open due to only two hydrogen bonds instead of the three bonds of the bases guanine and cytosine.

In eukaryotes, like human, each gene has its own promoter and also starting zones called TATA-box (so called because of repeating thymine and adenine bases, similarly to bacteria). Differently from bacteria, in eukaryotes additional helper proteins (transcription factors) are needed to help the polymerase recognize and bind these promoters.

Once the polymerase is bond to the right zone of the DNA, the transcription can start: it takes place adding complementary RNA nucleotides to the template DNA sequence (using uracil instead of thymine). As well as initiating zones, there is at the end of the gene (or group of genes in bacteria) a termination zone which leads to the end of transcription in slightly different ways, which will not be discussed here. Once the transcription is finished, the RNA transcript, called now messenger-RNA (mRNA) is ready to undergo the next process: the translation.

In bacteria translation can take place even before the transcription is finished and the protein can be translated while the transcription of its gene is still undergoing. This is not possible in eukaryotes like human since transcription is taking place in the nucleus and translation in cytosol, needing a transport through the nucleus membrane before starting the transcription. Moreover, in eukaryotes several other processes called splicing happen to the mRNA before the translation takes place, changing its sequences and therefore possibly coding for different proteins, even if coming from the same gene transcription.

Translation

The second step of the central dogma of biology is the process in which the language will decode, from a sequence of nucleotides to a sequence of amino acids, building the different proteins.

The enzymes involved in this process are the ribosomes: these enzymes together, with other help proteins, are able to read the genetic code in which every triplet of nucleic acids (called codon) corresponds to a specific amino acid. The ribosomes are composed of two different subunits which are normally separated in the cytosol and join together only on specific initial region on the mRNA molecule to start the translation. This specific first region of every mRNA is known as untranslated region (UTR), since is a region only used to link mRNA to ribosome and it is not translated into protein.

A special triplet (AUG) which codes for methionine acts also in every mRNA as start codon, giving the signal to the ribosome to start the translation. The methionine added as very first amino acid in every new protein is then often removed after the end of the translation.

At the end of the mRNA three different codons (UAA, UAG, UGA) which do not code for any amino acid act as stop-codons.

The ribosome is not able alone to pair the different codons with the corresponding amino acid and needs helper proteins. Special RNA complexes, known as transfer-RNA (tRNA) are used as adaptor molecule. They present two ends: one end is composed of a specific triplet (anticodon) which pair to the complementary codon on the mRNA, the other end of tRNA attaches the corresponding amino acid.

After the assembly of the ribosome on mRNA with the discussed mechanisms, the elongation process of translation can take place. The ribosome allows the different tRNA to bind the codon on the mRNA and form peptide bonds between the corresponding amino acids to form a polypeptide chains and finally a protein. Several corrective processes are performed to prevent a mismatch of the codons. When the ribosome come to the stop codons, the translation is ended and the protein, already folded in its 3-dimensional structure, is ready to undergo further processes or to start its function.

2.1.2 Regulation of Gene Expression

Every cell of an organism contains exactly the same DNA. The gene expression is nevertheless very different in different type of cells, determining their different functions. There are very basic genes, involved in metabolic functions, which are

expressed at constant rates in every cell and other very specific sets of gene which are expressed just in some cells, after specific signals or changes of the environment.

The modulation of the gene expression is therefore vital for the cell and the organism and several different mechanisms are involved in it. The eukaryote cell presents all the following mechanisms.

DNA packaging The human DNA is composed of 3 billions base pairs that would be about 100cm in length if stretched. To be able to be included in every single cell, the DNA must be packaged. Chromatin is the name of packaged DNA, which gives it the known form of chromosome. This very sophisticated packaging is realized in several stages: at the very first level the double helix will wrap around proteins, called histones, forming the so called euchromatin. The euchromatin will then be coiled to form a more compressed DNA, called heterochromatin, which will be supercoiled to form the chromosomes.

In this very condensed form, the chromatin is not accessible to any process and must be open before the transcription process could start. Because of the very big size of DNA, only the needed regions of DNA will be open to allow the transcription, while the rest of DNA stays condensed. The transcription could even further be repressed due to the so-called methylation of the chromatin: it is a process of methylation of DNA that leads to heterochromatic silencing of specific, not needed, regions of DNA.

Transcription regulation As seen in the transcription paragraph, many regulatory systems are necessary to start the transcription of specific genes. As transcription enabling factors are needed to recognize the start-site and begin the process, many other contro-regulatory factors could repress the transcription of silenced genes. Control mechanisms could even act after the transcription process has ended, modifying the resulting RNA: the different processes are known as RNA-processing. The same transcript of RNA could undergo alternative splicing, which is a process of cutting and modifying the sequence of nucleic acids, resulting in different mRNAs that code for very different proteins, even though they initially came from the same DNA template. Other post-transcription processes to the RNA include capping and poly-A tail, to enable the RNA-transcript to exit from the cell nucleus, and other signaling modifications allowing the molecules to be transported in separated places for the translation, for example mitochondrial cytoplasm.

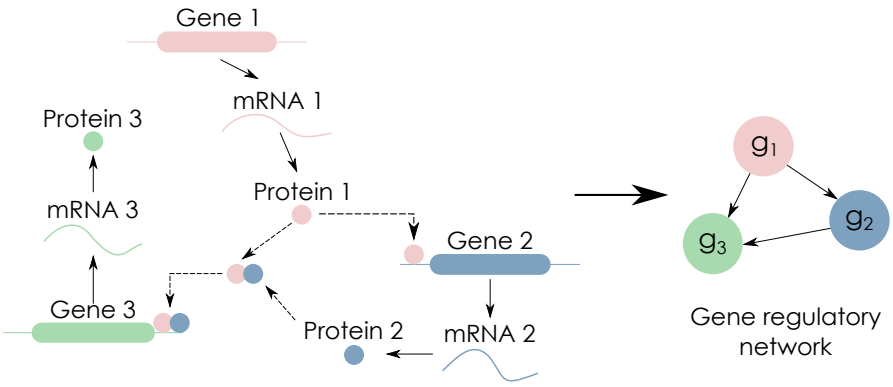


Figure 2.2: **Schematic representation of a gene regulatory network**. A complex biophysical model describes the interaction between three genes, involving both direct regulation (gene 2 by gene 1) and combinatorial regulation via complex formation (gene 3 by genes 1 and 2). The abstracted structure of the system is given in the (directed) network on the right. Figure and caption from [Huynh-Thu and Sanguinetti, 2019].

Stability of molecules Some regulating systems act modifying the stability i.e. the longevity of the mRNAs. If an mRNA would not be dismissed, it would be able to be continuously be translated in the corresponding protein. To allow for the building of just the exact number of needed protein, the mRNA transcript gets into modifications, targeting it to be degraded from specific enzymes (Ribonuclease).

Translation's efficacy and protein regulation The translation rate itself could also been regulated, leading to an inhibition or an increase of the efficacy of the protein-building process. Several regulating factors are involved in this process. The end-product of the whole process, the protein, could also undergo regulation-processes, as well as degradation process through specific enzymes called proteasome, which lead to an enhanced or inhibited function of the protein itself.

All of these mechanisms, and maybe even more, are present in the eukaryotic cells, which are more complex than the prokaryotic ones. Nevertheless, some of the described mechanisms are also present in prokaryotic cells. With this very sophisticated set of tools, organisms manage to express the proteins they can code when they are needed (in response to specific stimuli and situations, or in specific cell) to guarantee and maintain a well-working system.

Gene Regulatory Networks

Gene Regulatory Networks are a synthetic and convenient way of representing as graphs the functional interactions of the genes of an organism [Hasty et al., 2001b].

The representation abstracts away from the details of the actual underlying chain of events that produce an interaction between two genes, and draws it as an edge, possibly directed, between the corresponding nodes. Figure 2.2 provides an example, on how biological processes and interaction are translated into a gene regulatory network.

2.1.3 Transcriptomics technologies

In the previous section we have introduces the mechanisms that regulate gene expression allowing cell specialization and prompt response to stimuli. In order to study such mechanisms, researchers must be able to simultaneously measure the levels of expression of multiple gene products [Huynh-Thu and Sanguinetti, 2019]. However, the intrinsic limits in sensitivity and a relatively complex analysis pipeline make the analysis at proteomics level not a viable solution [Bantscheff et al., 2007]. Therefore, here we will discuss the main techniques to quantitatively assess expression at transcriptomics level, as a proxy to proteomics.

Methods for the quantification of RNA levels have largely improved in the last twenty years. Microarray technology first provided enormous impetus to the field in the late 90s [Brown and Botstein, 1999]. Microarrays exploit the technique of inverse hybridization to assess at the same time the expression levels of thousand of transcripts within a population of cells. For each mRNA that need to be measured, a set of DNA called *probes* is arranged on a substrate chip (array). Target mRNA, extracted from the biological sample, is converted to complementary DNA via reverse transcriptase and marked with fluorescent atoms. When hybridization is activated, cDNA molecules bounded to DNA *probes* can be identified simply by looking at their coordinates and their concentration is proportional to their luminescence.

Despite the huge breakthrough represented by the introduction of Microarrays, their design implicitly bounds their application allowing to measure only those mRNAs whose *probe* is present on the chip. This limits the discovery of unexpected behaviours and mechanisms, as in the case of long-non-coding RNAs, whose importance has been ignored for decades due to the lack of measurements.

Next generation sequencing (NGS) technologies filled this gap, drastically reducing the cost of sequencing per base by implementing highly parallel protocols. RNA-seq [Wang et al., 2009] is one of the most popular NGS technologies, that allows biologists to study the transcriptome by quantitatively measure the abundance of transcripts in biological samples. The sequencing protocol changes according to the technology, but we can depict the main steps. Firstly, the RNA from a population of cells is reverse transcribed in cDNA, and processed by high-throughput

sequencing technology, to obtain fragments of sequences called *reads*. *Reads* are then mapped on reference genome or transcriptome in order to identify and measure the abundance of the transcripts present in the sample. The so-obtained *counts* are typically normalized by length and gives a raw measurement of gene expression.

RNA-seq has several advantages over Microarrays. It is suitable for organism with no reference genome or transcriptome and, more in general, being not bounded by the *probes*, it does not limit the scope of research. Moreover, RNA-seq has a better resolution and dynamic range, with a better reproducibility of the results. The popularity of RNA-seq is indeed rapidly growing.

2.1.4 Protein Structure

Proteins are long polymers and amino acids are their monomers. Amino acids are organic molecules composed of a common backbone, consisting of amine (-NH_2) and carboxylic acid (-COOH) functional groups, along with a specific side-chain, that characterize each amino acid. Indeed, the physico-chemical properties of the side chain determine the structure and the shape of the whole protein.

After the translation, proteins fold themselves in complex three dimensional structures, which will widely determine their functionalities. During the folding process, not only the amino acids composition is important, but also the environment plays a critical role. Indeed, according to it, some amino acids may "prefer" to face the outside of the protein or the core. If the folding does not succeed the protein is useless or, in some cases, even dangerous. It is indeed known that many degenerative diseases like the Alzheimer, the Cystic fibrosis or the BSE (commonly known as "mad cow" disease) are caused by the misfolding of proteins. Therefore, there exist specialized proteins (called molecular chaperones) and biological process (translocation) that try to guarantee to proteins the right folding environment.

The structure of a protein is usually analyzed at four different abstraction levels (Figure 2.3):

- Primary structure: is the lowest of the four levels and is determined by the bare sequence of amino acids. Nowadays it is reasonably easy to obtain this information by translating the nucleotide triplets in coding sequences of genes obtained by genome sequencing, but is less informative than the higher levels.
- Secondary structure: represents how the peptidic backbone interacts with itself through hydrogen bonds. At this level specific folding shapes can be identified, i.e. the alpha helix and the beta sheet. This first level of folding occurs just after the translation (few milliseconds) and is mainly caused by hydrophobic

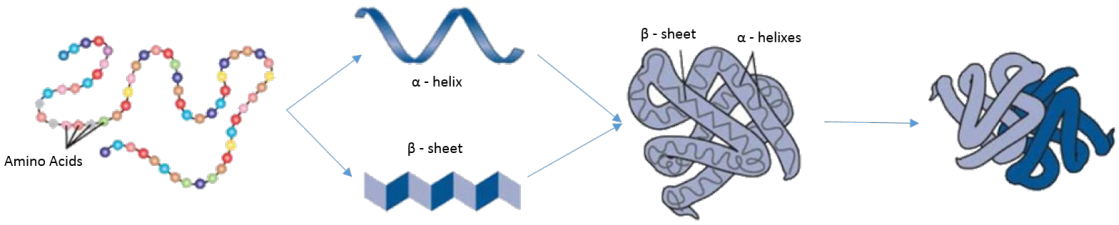


Figure 2.3: **Protein structure**. The figure shows the schematic representation of the four levels of the protein structure.

behaviors of the amino acids.

- Tertiary structure: describes how the secondary structures interact with themselves forming the three dimensional shape. The structure is stabilized through hydrogen bonds and disulfide bridges (covalent bonds created post-translation). Information about this levels are still hard and expensive to obtain. Bioinformatic methods are not powerful enough to face this problem, so techniques like NMR (Nuclear Magnetic Resonance) or X-ray crystallography have to be applied.
- Quaternary structure: some proteins are constituted by more than one peptidic chain or are part of protein complexes. These interactions are described by the quaternary structure. Studying this structure level is usually even harder than for the tertiary one. Due to the steric effects of protein complexes, they can not be easily crystallized and analyzed with X-ray crystallography. In these cases cryo-electron microscopy is used, allowing researchers to literally freeze the structure of a protein and investigate it through an electron microscope.

Even after having finished the folding process, the structure of a protein is not rigid and immutable. The three-dimensional structure can indeed change in time depending on the surrounding environment (temperature, pH, voltage, ion concentration, phosphorylation or ligand binding), which can actually trigger their functionalities. It is, for example, the case of myosin which interacts with ATP molecule. ATP molecules are very energetic and their hydrolysis induces a conformational change in the protein that, as a final result, allows the muscle contraction.

Evolution has deeply shaped living beings, giving them very different forms and features, from their appearance to their molecular level. Proteins are no exception. Despite this there are portions of them, which are strongly conserved between different proteins of the same organism but also between species. These pieces of proteins are called domains and are usually characterized by a compact structure and a specific folding. Protein domains are strongly related to their function and

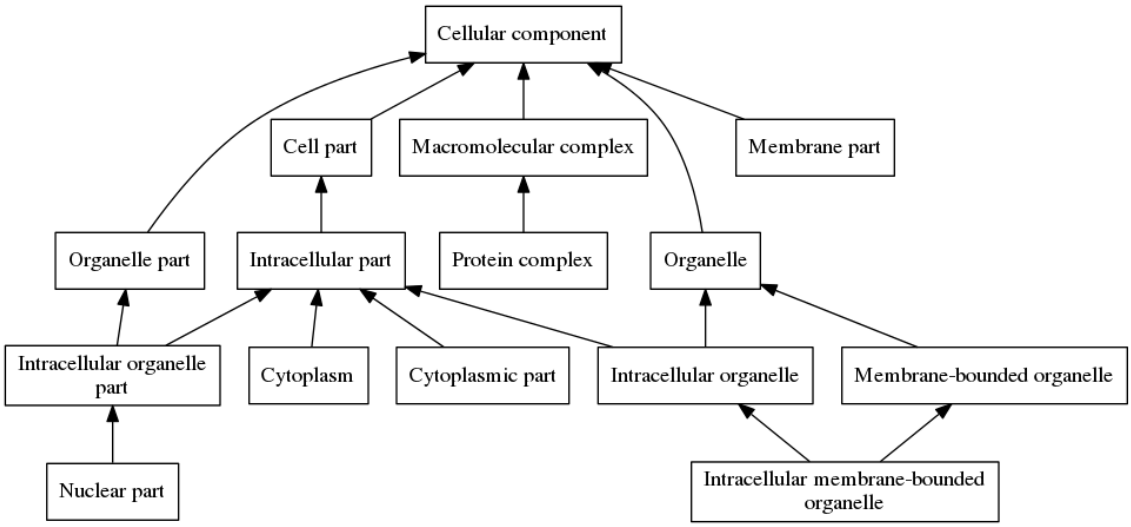


Figure 2.4: **Gene Ontology DAG**. The figure shows a portion of the cellular component hierarchy of GO. Note that the edges represent *is_a* relations and flow from the leaf terms to the root.

are combined in proteins like building blocks. Their conservation can be used as a key for identifying known domains in proteins according to their primary sequence.

2.1.5 Gene Ontology

The lack of a standard terminology in the biological field can lead to inefficient communication and ambiguous data sharing in the scientific community. Gene Ontology (GO) [Gene Ontology Consortium, 2001] is a bioinformatic project whose purpose is to fill this gap.

The Structure Gene Ontology is a manually-curated structured ontology with the aim to provide a unique and unequivocal description of all gene products. This information consist of an ID, a name, the GO domain and a natural language definition listing its main features. All the entries are organized in three direct acyclic graphs (DAGs) as shown in Figure 2.4, each one having as root a specific GO domain, which are:

- `cellular component`: the parts of a cell or its extra-cellular environment,
- `molecular function`: the elemental activities of a gene product at the molecular level, such as binding or catalysis,
- `biological process`: operations or sets of molecular events with a defined

beginning and end, pertinent to the functioning of integrated living units: cells, tissues, organs and organisms.

In addition to the annotation, relation plays a very important role in the GO structure definition, giving fundamental information about relationships between the various terms.

- *is_a*: is the main relation and forms the basic structure of GO. A GO term A is said to be in a *is_a* relation with a GO term B if A is a subtype of B. Therefore $A \rightarrow B$, but not vice-versa. This relation is transitive, indeed taking the *is_a* chain of mitochondrion, intra-cellular organelle and organelle as example, it is clear that a mitochondrion is an organelle and not every organelle is mitochondrion.

- *part_of*: is similar to the *is a* relation, but it express the concept of being part of something (cellular component) or take part in something (biological process and molecular function). As the *is_a* relation, *part_of* is transitive, indeed if A *part_of* B and B *part_of* C, then A *part_of* C. This relation can be trans-hierarchy, e.g. a Molecular Function can be *part_of* a Biological Process.

- *regulates*: this relation represents the ability of a GO term to directly affect the manifestation of another. It can be further specified if the regulation is positive (at the growth of the first the second grows) or negative (at the growth of the first the second decreases).

- *occurs_in*: this relation expresses a locational relationship between a BP and a CC term. It is probably the most informative trans-hierarchy relation, but unfortunately it occurs rarely in the GO DAGs.

2.2 Kernel Methods

Kernel methods are a class of machine learning algorithms for pattern analysis, whose best-known exponents are the Support Vector Machines (SVM) [Cortes and Vapnik, 1995]. Thanks to their theoretical firmness, their computational efficiency and flexibility, kernel methods are ubiquitously presents, not only in the machine learning literature, but also in fields like computational biology where they are often used as black boxes to solve the most diverse tasks.

The common ground among these methods is the use of *kernel functions*. These functions allow to operate in a high-dimensionality feature space, without the cost of computing the explicit inner product between feature vectors. This means that

kernel functions merge the expressiveness of non-linear machine learning algorithm with the efficiency of the linear ones. Intuitively *kernel functions* generalize the notion of dot product to arbitrary (even infinite-dimensional) input space and can be seen as a measure of similarity between objects. The flexibility and expressiveness of *kernel functions* arise from the fact that they can be reasonably easy applied, not only to points in the euclidean space, but also to data structures, such as graphs, trees or sequences.

Definition 1. Valid kernel function [Schölkopf et al., 2001]: Given a function $K : \mathbf{X} \times \mathbf{X} \to \mathbb{R}_{\geqslant 0}$ and a set of examples $\{x_1, ..., x_n\}$, the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ is defined as:

$$\boldsymbol{G}_{i,j} := K(x_i, x_j) \tag{2.1}$$

If the Gram matrix generated by K is positive semi-definite, i.e. if it satisfies the condition:

$$\sum_{i,j} c_i c_j K_{i,j} \geqslant 0, \quad \forall \mathbf{c} \in \mathbb{R}^n \tag{2.2}$$

then K is a valid kernel function.

Valid *kernel functions* are very important in machine learning because they always correspond to a dot product in some Reproducing Kernel Hilbert Space (RKHS). Moreover the Representer Theorem [Schölkopf et al., 2001] shows that problems in the form:

$$f^* = \arg\min_{f \in \mathcal{H}} \; c((x_1, y_1, f(x_1)), ..., (x_n, y_n, f(x_n)) \; + \; g(\|f\|) \tag{2.3}$$

where $\mathcal{H}$ is an appropriate RKHS, g a strictly monotonically increasing real-valued function and c an arbitrary cost function, admits representation in terms of kernel expansions of the form:

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \tag{2.4}$$

where $\alpha_i \in \mathbb{R}$ for all $i \in \{1, ..., n\}$. This theorem demonstrates that many learning methods admit solutions that can be expressed as expansion in the training data.

2.2.1 Semantic Based Regularization

Semantic Based Regularization (SBR) [Diligenti et al., 2012, 2017] is a state-of-the-art Statistical Relational Learning framework specifically designed to reason and learn with constraints and correlations among related prediction tasks. Entities, tasks and relations are encoded in SBR using First-Order Logic (FOL).

Knowledge Base and constraints. SBR is based on a variation of fuzzy generalizations of FOL, which have been first proposed by Novák [1987], and which can transform any FOL knowledge base into a set of real-valued constraints.

A *T-norm fuzzy logic* [Zadeh, 1965] generalizes Boolean logic to variables assuming values in $[0, 1]$. A T-norm fuzzy logic is defined by its T-norm $t(a_1, a_2)$ that models the logical AND. A T-norm expression behaves as classical logic when the variables assume the crisp values 0 (false) or 1 (true). Different T-norm fuzzy logics have been proposed in the literature. For example, given two Boolean values $\bar{a}_1, \bar{a}_2$ and their continuous generalizations a_1, a_2 in $[0, 1]$, the Łukasiewicz T-norm is defined as

$$(\bar{a}_1 \wedge \bar{a}_2) \quad \rightarrow \quad t(a_1, a_2) = \mathtt{max}(0, a_1 + a_2 - 1) \; . \tag{2.5}$$

The negation $\neg\bar{a}$ of a variable corresponds to $1-a$ in the Łukasiewicz T-norm. From the definition of the $\wedge$ and $\neg$ logic operators, it is possible to derive the generalized formulation for the $\vee$ operator via the DeMorgan law and the implication $\Rightarrow$ via the T-norm residuum. Other choices of the T-norm are possible, like the *minimum T-norm* defined as

$$(\bar{a}_1 \wedge \bar{a}_2) \; \rightarrow \; t(a_1, a_2) = \mathtt{min}(a_1, a_2). \tag{2.6}$$

We focus our attention on FOL formulas in the Prenex Normal Form, having all the quantifiers at the beginning of the expression. The quantifier-free part of the expression is an assertion in fuzzy propositional logic once all the quantified variables are grounded. Let us consider a FOL formula with variables $x_1, x_2, \ldots$, with values in the finite sets $\mathcal{X}_1, \mathcal{X}_2, \ldots$, and let $\mathcal{P}$ indicate the vector of predicates and $\mathcal{P}(\mathcal{X})$ be the set of all grounded predicates.

The degree of truth of a formula containing an expression E with a universally quantified variable x_i is the average of the T-norm generalization $t_E(\cdot)$, when grounding x_i over $\mathcal{X}_i$:

$$\forall x_i \;\; E\big(\mathcal{P}(\mathcal{X})\big) \quad \longrightarrow \quad \Phi_\forall(\mathcal{P}\big(\mathcal{X}\big)) := \frac{1}{|\mathcal{X}_i|} \sum_{x_i \in \mathcal{X}_i} t_E\big(\mathcal{P}(\mathcal{X})\big) \tag{2.7}$$

Building constraints from logic. Let us assume to be given a knowledge base KB, consisting of a set of FOL formulas. We assume that some of the predicates in the KB are unknown: the SBR learning process aims at finding a good approximation of each unknown predicate, so that the estimated predicates will satisfy the FOL formulas for the sample of the inputs. In particular, the function $f_j(\cdot)$ will be learned by a Kernel Machine as an approximation of the j-th unknown predicate p_j. Let $\boldsymbol{f} = \{f_1, \ldots, f_T\}$ indicate the vector of all approximated predicates and $\boldsymbol{f}\big(\mathcal{X}\big)$ indicate the output values for all possible groundings of the approximated predicates.

One constraint $1 - \Phi_i(\boldsymbol{f}(\mathcal{X})) = 0$ for each formula in the knowledge base is built by taking its fuzzy FOL generalization Φ_i, where the unknown predicates are replaced by the learned functions.

Cost function and training. Let us assume that a set of H functional constraints $1 - \Phi_h(\boldsymbol{f}) = 0, 0 \leqslant \Phi_h(\boldsymbol{f}) \leqslant 1,\ h = 1, \ldots, H$ describes how the functions should behave. Let $\boldsymbol{f}(\mathcal{X})$ be a vector collecting the values of the functions for each grounding. In order to enforce the functions to satisfy the constraints, the cost function C penalizes their violation on the sample of data:

$$C[\boldsymbol{f}(\mathcal{X})] = \sum_{k=1}^{T} ||f_k||^2 + \lambda_l \mathcal{L}(\boldsymbol{y}, \boldsymbol{f}(\mathcal{X})) + \sum_{h=1}^{H} \lambda_h \Big(1 - \Phi_h\big(\boldsymbol{f}(\mathcal{X})\big)\Big) \ , \tag{2.8}$$

where $\mathcal{L}(\boldsymbol{y}, \boldsymbol{f}(\mathcal{X}))$ is the loss with respect to the supervised examples $\boldsymbol{y}$, λ_l is the weight enforcing the fitting of the supervised patterns, λ_h is the weight for the h-th constraint and the first term is a regularization term penalizing non-smooth solutions such that $||f_k||^2 = \boldsymbol{w}_k^T \boldsymbol{G}_k \boldsymbol{w}_k$, where $\boldsymbol{G}_k, \boldsymbol{w}_k$ are the Gram matrix and the weight vector for the k function, respectively. The weights are optimized via gradient descent using a back-propagation schema, see [Diligenti et al., 2017] for more details.

Collective classification. The process of performing inference over a set of instances that are correlated is commonly referred to as *Collective classification* [Sen et al., 2008]. Collective classification takes advantage of the correlations by performing a collective assignment decision.

Let $\boldsymbol{f}(\mathcal{X}')$ be a vector collecting the groundings for all functions over the test data. Collective classification for SBR minimizes the following cost function to find the values $\bar{\boldsymbol{f}}(\mathcal{X}')$ respecting the FOL formulas on the test data:

$$C_{coll}[\bar{\boldsymbol{f}}(\mathcal{X}'), \boldsymbol{f}(\mathcal{X}')] = \mathcal{L}_{coll}\big(\bar{\boldsymbol{f}}(\mathcal{X}'), \boldsymbol{f}(\mathcal{X}')\big) + \sum_{h} \Big(1 - \Phi_h\big(\bar{\boldsymbol{f}}(\mathcal{X}')\big)\Big) \ . \tag{2.9}$$

where $\mathcal{L}_{coll}$ is a loss penalizing solutions that are not close to the prior values established by the trained kernel machines.

2.3 Artificial Neural Networks

Artificial neural networks (ANN), or more commonly just neural networks (NN), are a wide class of machine learning algorithms, that take inspiration from biological neural networks and are nowadays omnipresent in many fields, not only in computer

science. Their formulation stems from one of the very first learning models developed: the perceptron. It was introduced by Frank Rosenblatt in 1958 [Rosenblatt, 1958] with the idea to abstract the mechanism of brain neurons by collecting the input signals, summing them together and applying on top of this an activation function. However, the linear nature of the perceptron strongly limited the class of problem that could be tackled, and few decades have been required in order to reach the formulation of the multi layer perceptron (MLP) and its learning algorithm by back propagating errors [Werbos, 1974; Rumelhart et al., 1986] in order to bring ANN to the limelight. Their success was not very durable, indeed in the late '90 the introduction of kernel-based methods, ANN were almost ignored for two decades by the computer science community. Until 2012, when [Krizhevsky et al., 2012] largely outperformed the previous record on the Imagenet [Deng et al., 2009] competition, giving start to the era of deep leaning [LeCun et al., 2015].

When considering ANN the main distinguish to be done is between feed-forward and recurrent NN's. Under those classes, a plethora of task-specific implementations have been proposed to collect, transform, and transmit signals through the network. Feed-forward NNs are a class of ANN with no loops, where information moves only forward. They have no memory on previous inputs, so the output depends only on the current input. On the other hand, recurrent NNs (RNNs) have been developed exactly to catch relations among sequences of inputs, by looping on the input and keeping memory of previous states. Unfortunately RNNs are harder to train and suffer from the gradient vanishing problem [Hochreiter, 1998] that leads to flat gradient regions where learning is impossible. To overcome this problem, that afflicts deep feed-forward NN as well, several approaches have been proposed. LSTM (Long Short Term Memory) [Hochreiter and Schmidhuber, 1997] are RNNs with a forget gate that allows the networks to forget dependencies too distant in time. More general solutions involve activation functions, where the rectified linear unit (ReLU) [Nair and Hinton, 2010] and its variants [He et al., 2015; Maas et al., 2013] allowed the researchers to explore deeper network architectures.

Now consider a feed-forward ANN with L hidden layers, where $l \in \{1, \dots, L\}$ is the index of the l-th hidden layer.

Dense layers are the basic feed-forward ANN layers where $H^{(l)}$ hidden units (neurons) are arranged in parallel in layer h. The output of the i-th hidden unit is computed as

$$y_i^{(l)} = f(\mathbf{w}_i^{(l)} \mathbf{y}^{(l-1)} + b_i^{(l)}) \tag{2.10}$$

where $\mathbf{w}_i^{(l)}$ is the weight vector specific for the i-th unit, $b^{(l)}$ is the bias, $\mathbf{y}^{(l-1)}$ is the output of the previous layer of size $H^{(l-1)}$, and f an activation function. The complete output for the l-th layer can be formulated as matrix multiplication as follows:

$$\mathbf{y}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.11}$$

where $\mathbf{W}^{(l)}$ is a $H^{(l)} \times H^{(l-1)}$ weight matrix and $\mathbf{b}$ is the bias vector.

Convolutional layers are a class of feed-forward ANN layers developed to mimic receptive field of the human brain visual cortex. The very basic idea is to learn k n-dimensional[1] filters that scan the input looking for patterns. Let us consider the simplest case, with only one one-dimensional filter of size m, then the output of a convolutional layer is computed as follows.

$$y_i^{(l)} = f\left(\sum_{a=0}^{m-1} w_{i+a}^{(l)} \cdot y_{i+a}^{(l-1)} + b_i^{(l)}\right) \tag{2.12}$$

Note that if no padding is applied, the output will be $m-1$ units smaller. Convolutional layers incorporates a strong notion of locality, that is particularly useful when analyzing data where feature order is meaningful.

Abstract Feature Learning

Kernel methods have shown remarkable results in terms of adaptability and soundness, but require deep field-specific knowledge for feature engineering. The incredible amount of available data and the advent of powerful GPUs opened the gates to deep neural networks, where powerful features are not engineered, but directly learned by the model starting from raw data. Zeiler and Fergus [2014] showed a very interesting example of this behaviour, by visualizing the filters of the convolutional layers in a deep architecture for image categorization. The analysis of the filters in the shallow layers (closer to the input) showed activation pattern similar to Gabor filters [Jain and Farrokhnia, 1991], hence recognizing edges and very simple shapes. On the other hand, deeper filters were activated by abstract concepts, like faces or car wheels.

Unfortunately the analysis of deep NN provides typically no human-explainable interpretation, limiting therefore their applicability in fields where a reasonable explanation is required, as in medicine or finance. This opened a very fruitful field of

[1] Typically $n = 2$ because of the very popular application on images classification, but filters with $n = 1$ are often applied on sequences and $n > 2$ to multi-channel data.

Algorithm 1: `skeleton` procedure [Kalisch and Bühlmann, 2007].

Data: T set of variables, $E = n \times p$ data matrix
Input: Significance level α
Result: An undirected graph with causal relationship between transcripts
Graph $G \leftarrow$ complete undirected graph with nodes in T
$l \leftarrow -1$
while $l < |G|$ **do**
  $l \leftarrow l + 1$
  foreach $\exists u, v \in G$ *s.t.* $|Adj_G(u) \backslash \{v\}| \geqslant l$ **do**
    if $v \in Adj_G(u)$ **then**
      foreach $\mathbf{k} \subseteq Adj_G(u) \backslash \{v\}$ *s.t.* $|\mathbf{k}| = l$ **do**
        if u, v *are conditionally independent given* $\mathbf{k}$ *w.r.t.* E *with significance level* α **then**
          remove edge $\{u, v\}$ from G
return G

research and in the last years some interesting works have been published [Sturm et al., 2016; Zhang et al., 2018].

2.4 PC Algorithm

The work of Judea [Pearl, 2009] established causality as a fundamental concept of statistics and computing. The notion of causality is receiving a growing attention by the data mining community [Le et al., 2018], in applications where the focus is the discovery of cause-effect relationships from observational data. The PC algorithm [Spirtes and Glymour, 1991b] and its recent variants are among the so-called constraint-based algorithms whose main characteristic is to start *"with a fully connected undirected graph and using conditional independence (CI) tests to eliminate edges [...] the graph is oriented via a series of rules (identifying v-structures or colliders, avoiding cycles, etc.)"* [Raghu et al., 2018].

The PC algorithm takes as inputs an $n \times p$ data matrix and a significance level *alpha*, typically 0.05 or 0.01. It starts by computing the `skeleton` function (Algorithm 1). A fully connected graph G with n nodes is constructed and systematically CI is tested for adjacent nodes in the graph w.r.t. a separation set of increasing cardinality l. Note that with $l = 0$, the separation set is empty, hence the CI is tested pairwise. On the other hand, if a node in the graph has k neighbours $\binom{k-1}{l}$ tests are performed. This leads to a worst-case super-exponential complexity in the number of variables, that however becomes polynomial in the case the graph to be

reconstructed is sparse enough [Leclerc, 2008; Kalisch and Bühlmann, 2007].

An other important consideration is that the `skeleton` function is strongly dependent on the order of the input. Consider a clique of three nodes u, v, q, $l = 1$ and assume that each couple of nodes is conditionally independent given the third variable. As soon as the CI test between the first two variables is tested, the edge between them is immediately removed, limiting the possibility to test the CI of the remaining pairs. To overcome this issue different solution have been developed. A naïve order-independent solution, using correlation to test for CI, has been presented in Colombo and Maathuis [2014]. The concept is to postpone the deletion of edges the end of each cycle, with the clear drawback of being computationally more demanding. [Bacciu et al., 2013] explore various adaptation on the PC-algorithm, mainly based on mutual information, to evaluate the trade-off between quality of the reconstructed network and the computational feasibility of the approaches. The "test the weakest first" heuristic proposed in the article not only is capable of providing an order-independent solution, but also improves the quality of the reconstructed when combined with other strategies, while keeping the computational load unchanged.

Once the `skeleton` function terminates, a series of rules is applied in order to reconstruct a Markov equivalent DAG.

The discovery of causal relationships is particularly relevant for biological data and in fact a successful application of the PC algorithm to gene regulatory networks inference has been already reported [Maathuis et al., 2010]. In the work of Maathuis et al. [2010] the expression data of *Escherichia coli*, a bacterial model organism with a relatively small genome, was fed in a R implementation of the PC algorithm [Kalisch and Bühlmann, 2007]. However, the worst-case complexity of the algorithm and the sub-optimal implementation, make this approach not directly suitable for more complex organisms with tens of thousands of genes.

2.5 Ranking aggregators

The problem of ranking aggregation is strictly related to the voting theory. There are k candidates and n voters expressing a (partial) preference list over the candidates, a voting system aims at combining the preferences such that the result is a "consensus" of the voters [Dwork et al., 2001b]. Even though the problem stems from the ancient Greece, there are plenty of present-day applications, ranging from ensembles of learning machines [Valentini and Masulli, 2002], to the aggregation of search engine results [Dwork et al., 2001b].

The task of ranking aggregation can be formulated as follows. Given a set of ranked lists $\mathcal{L} = \{l_1, \ldots, l_n\}$, the ranking aggregation problem can be informally defined as combining the ranking in $\mathcal{L}$ in order to obtain a "better" final ordering $l^\star$.

Borda Count

The Borda count [Borda, 1781], consists in associating to each element u in a list l a Borda score $B_l(u)$. The final rank of the element u will be computed using an aggregation function $f(B_1(u), ..., B_n(u))$. Common aggregation function are the arithmetic mean or the median.

RankAggreg Package

RankAggreg [Pihur et al., 2009] algorithm, using local search, tries to find the list $l^\star$ defined as:

$$l^\star = \arg\min_l \left(\sum_{i=1}^{n} w_i d(l, l_i) \right)$$

where w_i is the weight associated with list l_i (usually $w_i = 1$) and d is the Kendall Tau or the Spearmann distance.

RobustRankAggreg Package

RobustRankAggreg [Kolde et al., 2012] is an R package which implements an aggregator based on order statistics. The assumption behind it is that the rank of each element U in lists in S is sampled from a distribution and, once the distribution is known, it can be compared with a null distribution obtained by sampling the ranking uniformly (non-informative ranking). The aggregator associates at each element a corrected p-value and the final ranking is obtained by sorting the elements by p-value.

Markov Chain method

Markov chain 4 (MC4) method have been proposed by [Dwork et al., 2001a]. MC4 method builds an ergodic markov chain, where each state represents a ranked element, using the ranking in $\mathcal{L}$. The chain is build starting from a ranked element U, then an element V different from U is sampled uniformly. If V is ranked lower for a majority of the lists in $\mathcal{L}$ where both of elements are present then, associate a probability to the transition from U to V else increase the probability of staying

in U. The final rank $l^\star$ is obtained by computing the stationary distribution of the chain and by sorting the elements by their stationary probability.

2.6 BOINC

In the era of big data, the computational resources required to deal with modern computational problems are constantly increasing. A viable alternative to expensive server clusters is the volunteer distributed computing (VDC), a distributed computational system that rely on the computational power provided by personal computing devices that would be otherwise idle.

The most famous framework for VDC is probably the Berkeley Open Infrastructure for Network Computing (BOINC) [Anderson, 2004b]. It was developed in the early 2000 by the Berkeley University in order to provide the computational power required by the SETI@home project [Anderson et al., 2002], whose aim is to search for extra-terrestrial intelligence by analyzing data coming from a radio telescope. In the following years, BOINC started hosting a plethora of scientific projects, ranging from mathematics and physics to biology and medicine [Das et al., 2007; Lombraña González et al., 2012; Monasterio et al., 2018]

BOINC projects have a typical master-slave architecture. The master node, server side, is composed by one or more nodes whose goal is to generate the work for slave-nodes, and then collect the processed results.

Chapter 3

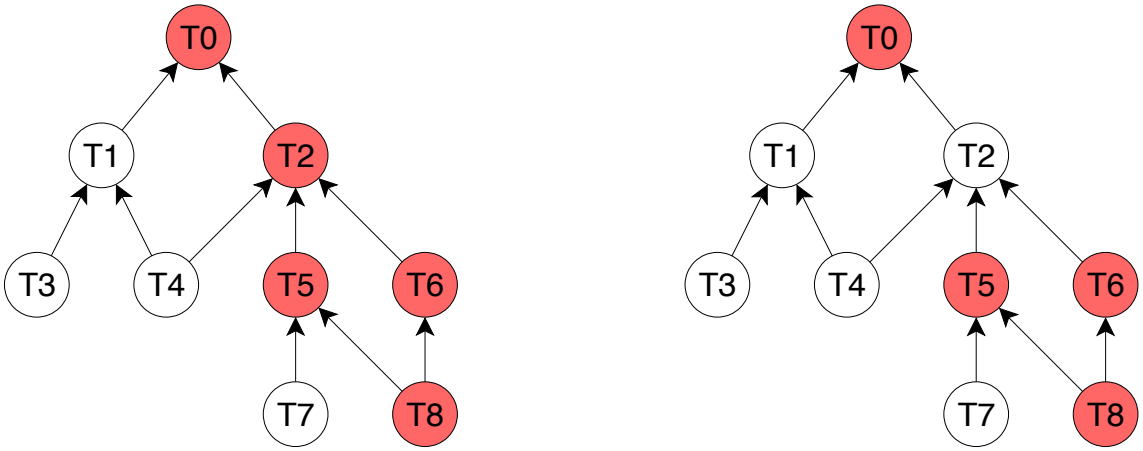
Multi Target Prediction

Many important decision problems require the simultaneous prediction of more then one target variable, possibly of different types [Waegeman et al., 2018]. Even though, most of the multi-target problems can be deconstructed in many single-target predictions and tackled with standard predictors, dedicated methods have been developed to face specific aspects of this interesting class of problems. The common ground among this class of dedicated methods is to exploit *side knowledge* that may arise from *a priori* information on the task, as in the case of hierarchical relations, or may be hidden in the data and the relations among target variables need to be unraveled by the predictor.

In this thesis, we will focus on two classes of multi-target prediction problems, that may be tagged under the names of hierarchical-multilabel classification and candidate discovery for network expansion. This choice stems from the central role they have in bioinformatics, the former in the well-known problem of protein function prediction, the latter for the expansion of known biological networks.

3.1 Hierarchical Multilabel Classification

The task of multilabel classification is an extension of binary classification, where more than one label may be assigned to each example [Sorower, 2010]. However, if the labels are independent, the task can be reduced without loss of generality to multiple binary tasks. Of greater interest is the case where there is an underlying structure that forces relations across the labels. These relations define a notion of consistency in the annotations, that can be exploited in the learning process to improve the prediction quality. This task goes under the name of hierarchical multilabel classification (HMC) and can be informally defined as the task of assigning a subset of consistent labels to each example in a dataset [Vens et al., 2008].



(a) TPR-compliant annotation.

(b) Not TPR-compliant annotation.

Figure 3.1: **True path rule examples**. Examples of hierarchical multilabel annotations, where the red-filled nodes are the predicted ones. Note that the edges represent "*is_a*" relations. In the right hierarchy T2 is not positively predicted, interrupting the path from T8 to T0, resulting therefore a not TPR-compliant annotation.

Knowledge is organized in hierarchies in a wide spectrum of applications, ranging from content-categorization [Soricut and Marcu, 2003; Sun and Lim, 2001] to medicine [Schriml et al., 2011] and biology [Ruepp et al., 2004; Gene Ontology Consortium, 2001; Murzin et al., 1995]. Hierarchies can be described by trees or direct acyclic graphs (DAG), where the nodes are the labels (we will refer to them as terms in the rest of the thesis) and the edges represent *is_a* relations that occurs between a child node and its parents[1]. These relations can be seen as a logical implication, because if a term is true then also its parents must be true. In other words, *"the pathway from a child term to its top-level parent(s) must always be true"* [Gene Ontology Consortium, 2001]. This concept was introduced by "The Gene Ontology Consortium" (GO) under the name of "true path rule" (TPR) to guarantee the consistency of the ontology with respect to the annotations, such that, whenever a gene product is found to break the rule, the hierarchy is remodelled consequently. Besides guaranteeing the consistency of the annotation space, the TPR can be forced also on the predictions, see example in Figure 3.1. Inconsistencies in the predictions have been shown to be confusing for the final user, who will likely not trust and reject them [Obozinski et al., 2008]. Even though there are circumstances where inconsistencies are accepted, we will focus on the strict case, where the TPR should hold for predictions as well.

HMC has a natural application in bioinformatics, where ontologies are widely

[1] A formal definition of child and parent nodes will be provided in the next section.

used as annotation space in predictive tasks. The critical assessment of functional annotation (CAFA) [Jiang et al., 2016b; Radivojac et al., 2013], for example, is the reference challenge for the protein function prediction community and uses the GO terms as annotations. The ontology comprises thousands of terms organized in three DAGs and the concepts expressed by some of those terms are so specific that just few proteins have been experimentally found belonging to them. Therefore, even though a perfect multilabel classification on the leaf nodes would solve the problem, the lack of examples forces the participants to exploit the hierarchical structure, by learning a single model [Sokolov and Ben-Hur, 2010] or by correcting the output of multiple models *a posteriori* [Gong et al., 2015].

As stated by Waegeman et al. [2018], dedicated methods for MTP have an edge over methods that do not exploit *side information*. This holds also for HMC, Vens et al. [2008] propose and compare three approaches based on predictive clustering trees. The *global* method called clus-HMC trains one decision-tree to cope with the entire classification problem. The proposed method is then compared with its naïve version clus-SC, which trains a decision-tree for each class of the hierarchy, ignoring the relationships between classes, and with clus-HSC, which explores the hierarchical relationships between the classes to induce a decision-tree for each class. The authors performed the experiments using biological datasets, and showed that the global method was superior both in the predictive performance and size of the induced decision tree. CLUS-HMC has been shown to have state-of-the-art performance, as reported in the study by Triguero and Vens [2015].

3.1.1 Formalization

This section formalizes the task of hierarchical multilabel classification and introduces the notation used in the rest of the thesis. Consider the hierarchy involved in the classification task described by a DAG $H = (\mathcal{T}, E)$, where $\mathcal{T} = \{t_1, \ldots, t_m\}$ is a set of m terms and $E = \{\mathcal{T} \times \mathcal{T}\}$ is a set of directed edges. In particular the edges in E represent "*is_a*" relations, i.e. given a sample x and $\langle t_u, t_v \rangle \in E$, t_u *is_a* t_v means that t_u implies t_v, $t_u(x) \implies t_v(x)$ for all x.

child t_u is a child of t_v iff $\langle t_u, t_v \rangle \in E$, $children(t_v)$ returns the children of t_v;
parent t_v is a parent of t_u iff $\langle t_u, t_v \rangle \in E$, $parents(t_u)$ returns the parents of t_u;
root a term t_v such that $parents(t_v) = \varnothing$;
leaf a term t_u such that $children(t_u) = \varnothing$, $\mathcal{F} = \{t_u | child(t_u) = \varnothing\}$ is the set of leaves;
ancestors the set of terms belonging to all the paths starting from a term to the

root, $ancestors(t_v)$ returns the set of ancestors of t_v;

descendants the set of terms belonging to the paths in the transposed graph H^T starting from a term to the leaves.

Let $\mathbf{X}$ be a set of i.i.d. samples in $\mathbb{R}^d$ drawn from an unknown distribution, and $\mathbf{Y}$ the set of the assignments $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ of an unknown labelling function $y : \mathbf{X} \to \mathcal{P}(\mathcal{T})$[2], namely $\mathbf{y}_i = y(\mathbf{x}_i)$. The function y is assumed to be consistent with the TPR (formalized in the next paragraph). Let $\mathcal{D}$ be the dataset $\mathcal{D} = \{\langle \mathbf{x}_1, \mathbf{y}_1 \rangle, \ldots, \langle \mathbf{x}_n, \mathbf{y}_n \rangle\}$ where $\mathbf{x}_i \in \mathbf{X}$, and $\mathbf{y}_i \in \mathbf{Y}$. For convenience the labels $\mathbf{y}_i$ assigned to the sample $\mathbf{x}_i$ are expressed as a vector in $\{0,1\}^m$ such that the j-th element of $\mathbf{y}$ is 1 iff $t_j \in y(\mathbf{x}_i)$. The hierarchical multilabel classification can be defined as the task of finding an estimator $\hat{y} : \mathbf{X} \to \{0,1\}^m$ of the unknown labelling function. The quality of the estimator can be assessed with a loss function $L : \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{T}) \to \mathbb{R}$, whose minimization is often the objective in the learning process.

3.1.2 True Path Rule

The TPR plays a crucial role in the hierarchical classification task, imposing a consistency over the predictions. The informal definition introduced in Section 3.1 can now be formalized within our framework. The *ancestors* function, that returns the terms belonging to all the paths starting from a node up to the root, can be computed by

$$ancestors(t_u) = \begin{cases} \left(\bigcup_{t_k \in par(t_u)} anc(t_k) \right) \cup par(t_u) & \text{if } par(t_u) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \tag{3.1}$$

where *anc* and *par* are shorthand abbreviations for the *parents* and *ancestors* functions.

Definition 2. The labelling function y observes the TPR iff

$$\forall t_u \in \mathcal{T}, t_u \in y(\mathbf{x}_i) \implies ancestors(t_u) \subset y(\mathbf{x}_i).$$

3.1.3 Evaluation metrics

HMC requires a dedicated class of metrics for performance evaluation. Zhang and Zhou [2014] reports an exhaustive set of those metrics highlighting properties and

[2] $\mathcal{P}(\cdot)$ is the power set of a given set.

use cases. We report here the definitions of the metrics used to evaluate the proposed approaches and compare them with the state-of-the-art.

Selecting optimal thresholds in the setting of HMC is not trivial, due to the natural unbalance of the classes. Indeed, by the TPR, classes that lay in the upper part of the hierarchy will have more annotated examples with respect to one on the leaves. Metrics that do not set thresholds, such as the area under the curve (AUC), are therefore very often used.

Metrics for HMC can be subdivided in two classes, i.e. term-centric metrics and example-centric metrics. The former, as the name suggests, focus on the goodness of the predictions for each term in the hierarchy, highlighting particularly-difficult terms or sub-hierarchies. The latter, instead, are computed example by example, giving insights on how the predictor performs on single predictions.

Term-centric Metrics The micro-averaged area under the precision recall curve ($\mathrm{AUC}(\overline{\mathrm{PR}})$) computes the area under a single curve, obtained computing the micro-averaged precision and recall of the m classes

$$\overline{Prec} = \frac{\sum_i^m TP_i}{\sum_i^m TP_i + \sum_i^m FP_i} \qquad \overline{Rec} \;\; = \frac{\sum_i^m TP_i}{\sum_i^m TP_i + \sum_i^m FN_i} \tag{3.2}$$

where TP_i, FP_i and FN_i are respectively the number of true positives, the false positives and the false negatives of the i-th term. It gives a global snapshot of the prediction quality but is not sensitive to the size of the classes.

Macro-averaged ($\overline{\mathrm{AUCPR}}$) and weighted ($\overline{\mathrm{AUCPR}}_w$) area under the precision recall curve, take more into account the classes with fewer examples. Both compute AUCPR_i for each class $i \in \{1, \ldots, m\}$, which are then averaged uniformly by the former and proportionally by the latter.

$$\begin{aligned} \overline{\mathrm{AUCPR}} &= \frac{1}{m} \sum_i^m \mathrm{AUCPR}_i \\ \overline{\mathrm{AUCPR}}_w &= \sum_i^m w_i \cdot \mathrm{AUCPR}_i \end{aligned} \tag{3.3}$$

where $w_i = v_i / \sum_j^m v_j$ with v_i the frequency of the i-th class in the dataset.

Example-centric Metrics As in the case of terms-centric metrics, the choice of the decision threshold τ is critical. The creators of the CAFA challenge propose this set of metrics to evaluate the competitors.

$$F_{\max} = \max_{\tau \in [0,1]} \frac{2\,\mathrm{pr}(\tau)\,\mathrm{rc}(\tau)}{\mathrm{pr}(\tau) + \mathrm{rc}(\tau)} \quad S_{\min} = \min_{\tau \in [0,1]} \sqrt{\mathrm{ru}(\tau)^2 - \mathrm{mi}(\tau)^2}$$

The $F_{\max}$ score is maximum value achieved by the F_1 score, i.e. the harmonic mean of the precision $\mathrm{pr}(\tau)$ and recall $\mathrm{rc}(\tau)$:

$$\mathrm{pr}(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{|P_i(\tau) \cap T_i|}{|P_i(\tau)|} \quad \mathrm{rc}(\tau) = \frac{1}{n} \sum_{i=1}^{n} \frac{|P_i(\tau) \cap T_i|}{|T_i|}$$

Here $P_i(\tau)$ is the set of terms for the i-th example, T_i is the set of true (observed) annotations, $m(\tau)$ is the number of examples with at least one predicted annotation at threshold τ, and n is the total number of examples. The $S_{\min}$ score is the minimum semantic distance, defined in terms of the remaining uncertainty (ru) and misinformation (mi):

$$\mathrm{ru}(\tau) = \frac{1}{n} \sum_{i=1}^{n} \sum_{f} \mathrm{ic}(f) [\![f \notin P_i(\tau) \wedge f \in T_i]\!]$$

$$\mathrm{mi}(\tau) = \frac{1}{n} \sum_{i=1}^{n} \sum_{f} \mathrm{ic}(f) [\![f \in P_i(\tau) \wedge f \notin T_i]\!]$$

where $\mathrm{ic}(f)$ is the information content of term f and $[\![\cdot]\!]$ is the 0-1 indicator function.

3.2 Candidate Discovery for Network Expansion

In the previous section we defined the task of HMC, where relations among (target) variables are exploited to refine the predictive task. In this section we will focus on the prediction of those relations starting from partial knowledge of a graph. Scientific research is often guided by incomplete prior knowledge, so starting from a set of variables that are known to be *related* and part of a common sub-network, the researchers may wonder if other variables are involved in the same network. The task of Candidate Discovery for Network Expansion (NE) aims at providing predictive tools that can help in this process.

NE approaches are of particular interest in computational biology. We are witnessing an exponential increase of sequencing data and gene expression data in the public databases. The collection and integration of these data sets has offered new opportunities and challenges to the field of computational biology. In particular, the analysis of the huge amount of available gene expression data can lead to discover causal relationships between the genes of an organism and link them to a specific biological process. However, to date these causal relationships are not yet well known, even when considering the most studied model organisms. It is very common in biological research, when studying a particular process, to start taking into account the prior available knowledge such as the genes participating in that process. In

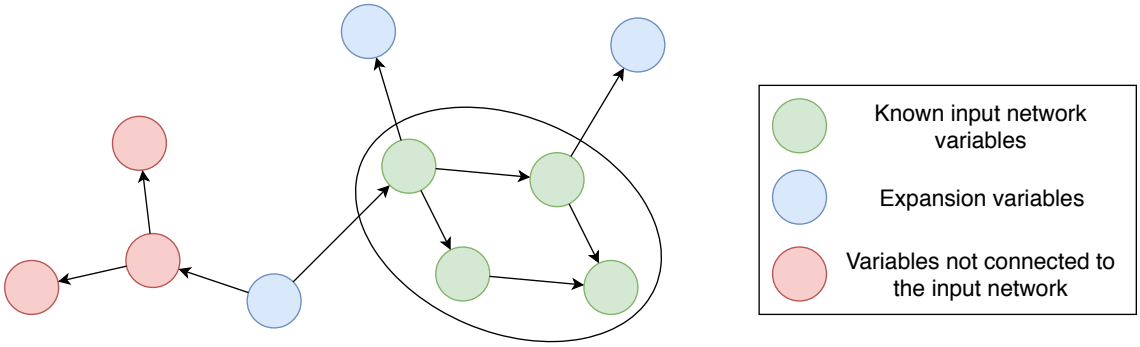


Figure 3.2: **Finding candidate for network expansion**. Given the input variables $\mathcal{S}$ (green nodes), the task of network expansion consist in ranking the remaining nodes $\mathcal{S} \setminus N$, such that the expansion variables (blue nodes) have an higher score then the one not connected (red nodes).

this scenario, methods that can suggest new candidate genes, which are potentially playing a role within a given gene network, are of essential importance for biologists. In particular, the Gene Network Expansion (GNE) task starts with a set of genes known to be interacting and searches for other candidate genes that regulate or are regulated by genes belonging to the input set of genes.

Network inference (NI) methods can be used to solve the task of NE. Indeed, a perfect solution for the NI task would perfectly solve also the NE task and consequently the task of finding candidate genes. However, the available NI methods are far from perfect and computationally very demanding due to the enormous size of the solution space. For instance, in the PC-algorithm the solution space is super-exponential in the number of nodes [Kalisch and Bühlmann, 2007].

3.2.1 Formalization

We generalize here the concepts defined by Asnicar et al. [2015a] for gene regulatory networks. Given a set $\mathcal{S}$ of variables and a golden-truth direct-graph $G = (\mathcal{S}, B)$ with $B \subset \mathcal{S} \times \mathcal{S}$ that represents the real causal relationships between the variables, it is possible to define the following tasks.

Task 1. Network inference. Given a subset of variables $N \subseteq \mathcal{S}$, find a (direct) graph $G = (N, B)$ where $B \subset N \times N$ is a relation between the elements of N, and G approximates the sub-graph in G obtained considering just the variables in N.

Task 2. Candidate discovery for network expansion (or simply network expansion). Given a graph $G = (N, B)$ where N is a subset of the variables of $\mathcal{S}$ and $B \subset N \times N$ is a relation between the elements of N, find a ranked list of elements of $\mathcal{S} \setminus N$ such that the elements of the list are connected or very near to the elements of N in G.

Chapter 4

Combining Learning and Logical Constraints for Hierarchical Multilabel Classification of Protein Functions

The advent of high-throughput experimental procedures comes both as an opportunity and as a challenge for computational approaches. On one hand, it allows to rely on unprecedented amounts of experimental data, such as sequential data at a genomic and meta-genomic scale as provided by next generation sequencing experiments. On the other hand, it calls for a change of scale for predictive approaches, from the focus on the analysis of individual biological sequences to the development of models characterizing the behaviour of all sequences in a given genome or metagenome [Friedberg, 2006].

This level of analysis requires to develop models capable of *jointly* performing predictions on multiple entities, accounting for the relationships between these entities in order to provide predictions which are consistent with the existing knowledge.

In this chapter we focus on two tightly-connected aspects of protein behaviour which are crucial in determining cell life, namely protein function and protein-protein interaction (PPI). By protein function we refer to the characterization of protein behaviour as formalized by the Gene Ontology Consortium (GO) [Ashburner et al., 2000]. Proteins mostly function through their interactions with other proteins, and predicting these interactions is thus at the heart of functional genomics [Keskin et al., 2008]. Furthermore, PPI play crucial roles both in the mechanisms of disease [Hopkins, 2008] and the design of new drugs [Csermely et al., 2013].

These predictive tasks are highly relational. GO hierarchies naturally enforce a set of taxonomic constraints between predictions. For instance, if a protein is annotated with a GO term it should also be annotated with the parents of this term as well as with its ancestors, all the way up to the root of the hierarchy. Protein-protein interaction predictions provide additional sources of constraints, as for instance two interacting proteins are more likely to be involved in the same process, while two proteins located in different cellular compartments are less likely to interact.

In this chapter we present OCELOT, a predictive model based on Semantic Based Regularization (SBR) [Diligenti et al., 2017], a statistical relational learning framework combining statistical learners with fuzzy-logic rules. For each GO term, a binary classifier is trained to predict whether a protein should be labelled with that term. A pairwise classifier is trained to predict whether pairs of proteins interact or not. All classifiers are implemented as kernel machines with kernels defined over multiple sources of information such as gene co-expression, sequence conservation profiles and protein domains (see Section 4.3.1 construction for the details). Consistency among predictions is enforced by a set of fuzzy-logic rules relating terms in the hierarchies and terms with PPI predictions (see Section 4.2.2 for details).

An extensive experimental evaluation over the Yeast genome shows the potential of the approach. Yeast was chosen as a reference genome because of the large amount of functional and interaction annotation available. Our results show that both hierarchical and term-interaction rules contribute in increasing prediction quality in all GO hierarchies, especially for the lower levels where less training examples are available. PPI predictions provide an additional boost in function prediction performance. The converse is not true, as function predictions do not contribute to improve PPI prediction quality. This is an expected result, as the latter task is comparatively simpler, and information tends to propagate from simpler tasks to more complex ones. When compared to alternative approaches, our model substantially improves over GoFDR [Gong et al., 2016], the only high-ranking system at the latest CAFA challenge [Jiang et al., 2016a] for which an implementation was readily available, when GoFDR is allowed to access Yeast proteins only (as our method does), and has comparable or better results (depending on the hierarchy and performance measure) when GoFDR is given full access to the UNIREF90 database of proteins. In addition, our system produces comparable results to DeepGO [Kulmanov et al., 2018], a deep learning-based method that relies on the true PPI network to produce its predictions.

4.1 Related Work

Protein function prediction methods can be roughly grouped in two classes. Sequence-based methods perform annotation transfer by leveraging sequence similarity only. They follow a two-step scheme: first candidate homologues are identified using tools like BLAST [Altschul et al., 1990] or PSI-BLAST [Altschul et al., 1997], then the annotations of the hits are transferred to the target based on various strategies. The underlying assumption is that homologues tend to share the same functions. Indeed, this is often the case for sequences with at least 60% similarity [Lee et al., 2007]. Targets that do not satisfy this condition are more challenging (they are referred to as "difficult targets" in CAFA parlance), and require finer-grained approaches. Recent approaches leverage deep learning architectures for analyzing the sequence data (e.g. [Kulmanov et al., 2018]). Some sequence-based methods additionally rely on sequence features such as (inferred) domains, motifs, or conserved residues, see e.g. [Gong et al., 2016].

Data-based methods instead gather functional hints from heterogeneous data sources, including physical interactions [Yu et al., 2016; Li et al., 2016], co-expression patterns [Stuart et al., 2003; Massjouni et al., 2006], and genetic context [Škunca et al., 2013; Sokolov et al., 2013], among others. Please see [Rentzsch and Orengo, 2009; Jiang et al., 2016a] for a list of frequently used sources. In this context, the key issue is how to appropriately integrate the sources while taking into account differences in format and reliability. The integration step is often carried out using statistical, probabilistic or machine learning tools.

Methods in both categories often do not enforce consistency among predictions. Those that do typically rely on a post-processing step to prune inconsistent annotations. More principled methods account for relations among GO terms directly in the training procedure, allowing annotation information to propagate across related terms. For instance, GOstruct [Sokolov and Ben-Hur, 2010; Sokolov et al., 2013] employs structured output support vector machines (SVM) [Joachims et al., 2009] to jointly predict all functional annotations of any target protein in a consistent manner. Ocelot follows the same principles, but relies on Semantic Based Regularization, a different, sound structured-output method. SBR has previously been applied to multi-level PPI prediction [Saccà et al., 2014]. Contrary to structured-output SVMs, SBR can be easily adapted to different prediction tasks by changing the consistency rules, as described in Methods. Further, SBR does not require to solve an optimization problem explicitly (as is the case for loss-augmented inference in structured-output SVMs [Joachims et al., 2009]) and can scale to larger tasks.

We note in passing that self-consistency alone is not enough to guarantee state-of-the-art results, as shown by the GOstruct results in the CAFA2 challenge [Jiang et al., 2016a]. More generally, despite the growing amount of "omics" data, which should favor data-based methods, sequence-based approaches proved to be hard to beat in practice [Hamp et al., 2013b], with some of them ranking among the top methods in the CAFA 2 competition [Jiang et al., 2016a]. For instance, GoFDR [Gong et al., 2016], an advanced sequence-based method, demonstrated excellent results in several categories, including eukaryotic genomes. Due to its excellent performance and immediate availability, we use GoFDR as the prime competitor in our experiments.

In addition, given the recent success of deep learning-based methods, we consider also the DeepGO approach of Kulmanov *et al.* [Kulmanov et al., 2018]. This approach applies a one-dimensional convolutional neural network (with max-pooling layers) to the sequence data in order to produce a hidden representation of the protein. Then, PPI information is also converted into a hidden representation via knowledge graph embeddings. These representations are fed into a neural network, whose structure mimics the target GO ontology. DeepGO has shown considerable performance, but, in contrast to our method, it requires interaction data to be available.

4.2 Model Description

4.2.1 Overview of the Prediction Pipeline

Genome-wide prediction of protein function and interaction involves inferring the annotations of all proteins in a genome. Ocelot approaches this problem by decomposing it into simpler prediction tasks, and exploits prior biological knowledge to reconcile the resulting predictions. Ocelot instantiates one task for every candidate GO term, i.e., deciding whether a given protein should be annotated with that term, plus a separate task for deciding whether a given protein pair interacts. The overall, genome-wide annotations are obtained by imposing consistency across the predictions of all tasks. See Figure 4.1 for a simplified depiction of our prediction pipeline.

In order to model the genome-wide prediction task, Ocelot employs Semantic Based Regularization (SBR) [Diligenti et al., 2012, 2017], a state-of-the-art Statistical Relational Learning framework specifically designed to reason and learn with constraints and correlations among related prediction tasks. Entities, tasks and relations are encoded in SBR using First-Order Logic (FOL). At the logical level,

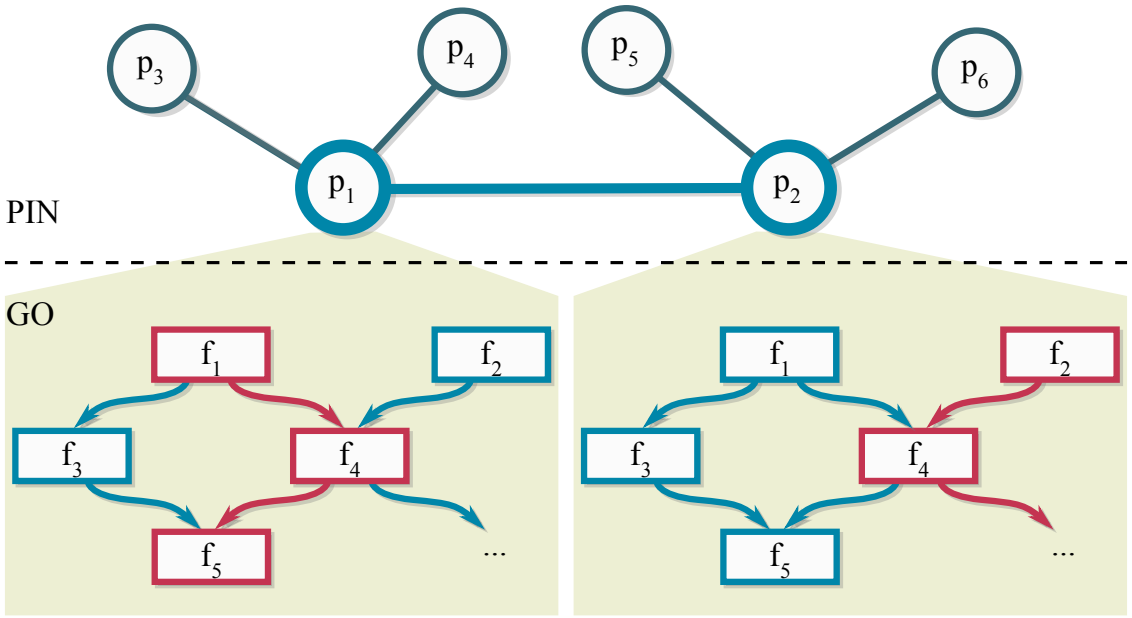


Figure 4.1: **Depiction of the Ocelot decision making process**. Above: predicted protein–protein interaction network, circles are proteins and lines represent physical interactions. Below: GO taxonomy, boxes are terms and arrows are *is_a* relations. Predicted annotations for proteins p_1 and p_2 (black): p_1 is annotated with terms f_1, f_4, f_5 and p_2 with f_2, f_4. The functional predictions are driven by the similarity between p_1 and p_2, and by consistency with respect to the GO taxonomy (e.g. f_1 entails either f_3 or f_4, f_2 entails f_4, etc.). The interaction predictions are driven by similarity between protein pairs (i.e. (p_1, p_2) against all other pairs) and are mutually constrained by the functional ones. For instance, since p_1 and p_2 do interact, OCELOT aims at predicting at least one shared term at each level of the GO, e.g. f_4 at the middle level. These constraints are not hard, and can be violated if doing so provides a better joint prediction. As an example, p_1 is annotated with f_1 and p_2 with f_2. Please see the text for the details.

proteins and terms are represented as constants p, p', f, f', etc, while annotations are modelled as predicates. OCELOT uses several predicates: a predicate $Fun_f(p)$ for each candidate term f, indicating whether protein p performs function f, and a separate predicate $Bound(p, p')$, encoding whether proteins p and p' are physically bound. The truth value of a predicate is either fixed, in case the corresponding annotation is already known, or automatically imputed by SBR. In the latter case, the predicate is said to be a "target" predicate, and the truth value is predicted by a kernel machine [Scholkopf and Smola, 2001; Borgwardt, 2011] associated to the predicate itself.

The kernel function, which lies at the core of kernel machines, measures the similarity between objects based on their representations. In our setting, a protein can be represented by the sequence of its residues, as well as by information about its

amino acid composition or phylogenetic profile: having similar sequences, composition or profiles increases the similarity between proteins. Given a kernel and an object x, a kernel machine is a function that predicts some target property of x based on its similarity to other objects for which that property is known. More formally, the function is:

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

This summation computes how strongly the property is believed to hold for x (if the sum is positive) or not (otherwise), and is often referred to as "confidence" or "margin". For instance, a kernel machine could predict whether a protein x resides in the nucleus or not. In this case, being similar to a protein x_i residing in the nucleus (positive w_i) drives the prediction toward a positive answer, while being similar to a protein x_i residing elsewhere (negative w_i) has the opposite effect. Note that designing an appropriate kernel is critical for predictive performance.

In SBR each target predicate is implemented as a kernel machine. The truth value of a predicate—applied to an uncharacterized protein— is predicted by the associated kernel machine. Given a set of kernel machines (or predicates), SBR employs FOL rules to mutually constrain their predictions. It does so by first translating the FOL rules into continuous constraints using T-norms, a procedure discussed more thoroughly in Section 2.2.1. Roughly, these constraints combine the confidences (margins) of the predicates appearing in the FOL rule into an overall confidence in the satisfaction of the rule.

In order to make the predictions of different tasks consistent with the rules, SBR computes a joint truth value assignment that maximizes the sum of 1) the confidences of the individual predicates, and 2) the confidence in the satisfaction of the rules. Informally, the optimal assignment y^* is obtained by solving the following optimization problem:

$$y^* = \text{argmax}_y \, \text{consist}(y, \text{kernel machines}) + \text{consist}(y, \text{rules})$$

The two terms represent the consistency of the inferred truth values and with respect to the predictions given by the kernel machines, and with respect to the rules derived from the FOL background knowledge, respectively. Notice that in this optimization problem, the rules act as *soft* constraints, encouraging assignments satisfying many rules with high confidence.

As for most other complex Statistical-Relational Learning models Getoor and Taskar [2007], this inference problem is not convex, which implies that we are restricted to finding local optima. SBR exploits a clever two-stage procedure to improve the quality of the obtained local optimum. In a first step, SBR disables the

constraints (by ignoring the second term of the equation above), thus obtaining individual predictions that fit the supervised data. This inference step is convex and can be solved efficiently to global optimality. In a second step, the obtained predictions are used as a starting point for the full inference procedure, where the constraints are turned back on. Empirically, this strategy was shown to achieve high-quality solutions, while being less computationally expensive than other non-convex optimization techniques Diligenti et al. [2017].

SBR can be used both in inductive and transductive mode. In the latter case, both training and test examples are provided during training, with labels for the training examples only. In this way, test examples can contribute via the rule consistency term even if their labels are not known. Semi-supervised approaches are known to boost predictive performance Zhu [2006], and fit the genome-wide prediction setting, where the full set of target proteins is available beforehand.

To summarize, functions and interactions of uncharacterized proteins are predicted based on similarity to other proteins and proteins pairs, respectively. The genome-wide predictions follow from applying consistency constraints, derived from biologically grounded FOL rules, to the low level predictions. In doing so, the constraints propagate information across GO terms and between the functional and interaction predictions.

4.2.2 Rules

Functional annotations are naturally subject to constraints. We consider both constraints entailed by the Gene Ontology and constraints imposed by the (partially predicted) protein–protein interaction network. SBR allows to express these through First-Order Logic rules, and to efficiently reason over them, even in the presence of inconsistencies. We proceed to describe the rules employed by our predictor.

Consistency with the GO hierarchies. The GO encompasses three domains, representing different aspects of protein function: biological process (BP), cellular component (CC), and molecular function (MF). Each domain specifies a term hierarchy, encoded as a directed acyclic graph: nodes are terms, while edges specify the specific-to-general *is_a* relation[1]. More general terms (parents) are logically implied by more specific ones (their descendants). For instance, all proteins annotated with "ribosome" as their Cellular Component must also be annotated with its ancestor term

[1] In this chapter we restrict ourselves to "*is_a*" relationships only, since the remaining GO relations, e.g. "*part_of*" and "`regulates`", occur too infrequently in the ontology.

"intracellular organelle". We encourage the OCELOT predictions to be consistent with the GO with the two following constraints.

First, terms imply their parents. If a protein p is annotated with a term f, then it must also be annotated with all of its parent terms. The converse also holds: if p is not annotated with f, then it can not be annotated with any of its children either. These constraints can be expressed as a single FOL statement:

$$Fun_f(p) \Longrightarrow \bigwedge_{f' \text{ parent of } f} Fun_{f'}(p) \qquad \forall\, p \forall\, f$$

Second, terms imply some of their children. If p is annotated with f, then it must be also annotated with at least one of the children of f:

$$Fun_f(p) \Longrightarrow \bigvee_{f' \text{ child of } f} Fun_{f'}(p) \qquad \forall\, p \forall\, f$$

Again, the converse also holds. These two rules are enforced for all GO aspects.

Note that if a protein is annotated (in the data) with a term f but with none of the children of f, the former may still result in the protein to be wrongly associated to a child term. We mitigate this applying the rules only to the upper levels of the hierarchy, where annotations are more abundant, as described below. Our empirical results show that, despite this issue, these rules provide non-negligible benefits in practice.

Consistency with the interaction predictions. Protein function and interactions are substantially intertwined: often a biological process is carried out through physical interaction, and interacting molecules must usually lie in the same (or close) cellular compartments. We tie functional annotations and interactions together by requiring that binding partners share at least one term at each depth of the corresponding domain. This rule can be encoded as:

$$Bound(p, p') \Longrightarrow \bigvee_{f \in \text{Domain}_l} (Fun_f(p) \wedge Fun_f(p')) \qquad \forall\, p, p', l$$

Here Domain_l is the set of GO terms appearing at depth l in the given domain. As above, the rule is soft. This rule is only applied to the BP and CC domains, as molecular function is less influenced by physical interactions. Further, we observed that this rule is mostly beneficial when applied to the top 5 levels of the CC taxonomy and 5 levels of the BP one. Its effect becomes irrelevant at the lower levels. Given that the rule is rather computationally expensive (as it involves all pairs of proteins p, p' in the genome and all terms at each depth l), we opted for applying it to the upper levels only.

4.3 Results

4.3.1 Data Processing

Annotations. We built a comprehensive genome-wide yeast dataset. All data was retrieved in August 2014. Protein sequences were taken from the Saccharomyces Genome Database (SGD) [Cherry et al., 2012]. Only validated ORFs at least 50 residues long were retained. The sequences were redundancy reduced with CD-HIT [Fu et al., 2012] using a 60% similarity threshold, leading to a set of 4745 proteins.

Functional annotations were also taken from SGD, while the GO taxonomy was taken from the Gene Ontology Consortium website[2]. Following common practice, automatically assigned (IEA) annotations were discarded. We also removed all obsolete terms and mismatching annotations, i.e. SGD annotations that had no corresponding term in the GO graph. The resulting annotations were propagated up to the root, i.e. if a sequence was annotated with a certain term, it was annotated with all its ancestor terms in the hierarchy. Since known annotations become more sparse with term specificity, we discarded the lowest levels of each GO hierarchy: we retained terms down to depth 9 for Biological Process and Molecular Function, and down to 6 for Cellular Component. We also dropped terms that had fewer than 20 annotations[3] . Dropped annotations were ignored in our performance evaluation. The resulting dataset includes 9730 positive annotations. All missing annotations were taken to be negative[4].

The protein–protein interaction network was taken from BioGRID [Chatr Aryamontri et al., 2015]. Only manually curated physical interactions were kept. After adding any missing symmetric interactions, we obtained 34611 interacting protein pairs. An equal number of non-interactions was sampled from the complement of the positive protein–protein interaction network uniformly at random. This procedure is justified by the overwhelming proportion of true non-interactions in the complement [Park and Marcotte, 2011]. All physical and functional interactions annotated in STRING 9.1 [Franceschini et al., 2013] were deleted from the complement prior to sampling, so to minimize the chance of sampling false negatives.

[2] `http://geneontology.org/page/download-ontology`

[3] Annotations of dropped child terms were aggregated into new "bin" nodes under the same parent. These terms provide useful supervision during training, and increase the satisfaction of Ocelot rules; see below for details.

[4] Some databases, e.g. NoGO [Youngs et al., 2014], do publish curated negative functional annotations. However, these resources do not yet provide enough annotations for training our predictor. Therefore, we resorted to sampling negative annotations from the non-positive ones, as is typically done. We adopted the same solution for negative interaction annotations [Blohm et al., 2013].

Kernels

In OCELOT, each learned predicate is associated to a kernel function, which determines the similarity between two proteins (or protein pairs). Following the idea that different sources provide complementary information [Rentzsch and Orengo, 2009; Yip et al., 2009; Sokolov et al., 2013], we computed a number of kernels, focusing on a selection of relevant, heterogeneous biological sources, intended to be useful for predicting both functions and interactions. Some very well known kernel for PFP have been excluded for the level of prior knowledge required for their construction. It is the case of graph kernels Ralaivola et al. [2005]; Borgwardt et al. [2005], that despite their biological and formal soundness rely on 3D protein structure, that is still not obtainable through high-throughput methodologies. The sources used in OCELOT include (i) gene *co-localization* and (ii) *co-expression*, (iii) protein *complexes*, (iv) protein *domains*, and (v) *conservation profiles*. Detailed explanations follow.

Co-localization Gene co-localization is known to influence the likelihood of proteins to physically interact [Yip et al., 2009], which is a strong indication of shared function [Yu et al., 2016; Li et al., 2016]. This information is captured by the gene co-localization kernel

$$K_{\text{coloc}}(p, p') = \exp\left(-\gamma|\text{pos} - \text{pos}'|\right).$$

Here $|\text{pos} - \text{pos}'|$ is the distance (measured in bases) separating the centroids of the genes encoding proteins p and p'. Closer centroids imply higher similarity. Genes located on different chromosomes have null similarity. Gene locations were obtained from SGD; γ was set to 1.

Protein complexes Similarly, protein complexes offer (noisy and incomplete) evidence about protein–protein interactions [Yip et al., 2009; Saccà et al., 2014]. We incorporated this information through a diffusion kernel

$$K_{\text{complex}}(p, p')$$

over the catalogue of yeast protein complexes [Pu et al., 2009]. Roughly speaking, similarity between proteins is proportional to the number of shared binding partners (and their shared partners, and so on) the two proteins have. The exact values are defined in terms of a diffusion process over the complex network. The contribution of more distant partners is modulated by a smoothness parameter β, set to 1 in our experiments. We refer the reader to [Kondor and Lafferty, 2002] for the mathematical details of diffusion kernels.

Co-expression kernel Co-expression also provides valuable information [Stuart et al., 2003]. The co-expression kernel is an inner product

$$K_{\text{coexp}}(p, p') = \langle \boldsymbol{e}, \boldsymbol{e}' \rangle$$

between vectors $\boldsymbol{e}$ and $\boldsymbol{e}'$ encoding the expression levels of p and p' across experimental conditions. The measurements were taken from two comprehensive sets of micro-array experiments [Spellman et al., 1998; Gasch et al., 2000] related to cell-cycle and environmental response in yeast.

Domains *Domains* often act as functional building blocks, so sharing the same domain is a strong indication of shared function [Fang and Gough, 2013]. We used InterPro [Mitchell et al., 2015] to infer the domains occurring in all proteins in the dataset. Presence of a domain in a protein p (resp. p') is encoded by an indicator vector $\boldsymbol{b}$ (resp. $\boldsymbol{b}'$): the k-th entry of $\boldsymbol{b}$ is 1 if the k-th domain was detected as present in p, and zero otherwise. Given this information, we defined a linear kernel over the indicator vectors, i.e.

$$K_{\text{dom}}(p, p') = \sum_k d_k d'_k.$$

Similarity is determined by the number of shared domains.

Conservation profiles Finally, we included phylogenetic information through a *profile* kernel [Kuang et al., 2005; Hamp et al., 2013a] over position-specific scoring matrices (PSSMs) obtained from the protein sequences. The PSSMs were computed with iterated PSI-BLAST (default parameters, two iterations) against the NCBI non-redundant sequence database (NR), as customary.

Each of the above kernels corresponds to a kernel 4865×4865 matrix. The matrices were normalized by the transformation

$$\hat{K}(p, p') = K(p, p') / \sqrt{K(p, p)\, K(p', p')}$$

and preconditioned by a small constant (10^{-6}) for numerical stability. Since OCELOT allows only a single kernel for each target term, we aggregated all the matrices into a single one through simple averaging:

$$K(p, p') = \frac{1}{5} \sum_{\text{all sources } s} \hat{K}_s(p, p').$$

This transformation equates to compounding information from all sources into a single kernel. More sophisticated strategies (e.g. assigning different weights to

different kernels) did not provide any benefits in our empirical analysis. Finally, the interaction predicate works on *pairs* of proteins, and thus requires a kernel between protein *pairs*. Following Saccà et al. [2014], we computed the pairwise kernel

$$K_{\text{pairwise}}((p, p'), (q, q'))$$

from the aggregate kernel $K(p, p')$ as follows:

$$K_{\text{pairwise}}((p, p'), (q, q')) = K(p, q) \cdot K(p', q') + K(p, q') \cdot K(p', q)$$

The pairwise kernel was also normalized and preconditioned.

4.3.2 Empirical analysis

We assessed the performance of OCELOT by comparing it against several competitors:

- $\text{GoFDR}_{\text{U90}}$: the state-of-the-art GoFDR prediction method [Gong et al., 2016] trained over all sequences in UNIREF90 [Suzek et al., 2015]. GoFDR is a state-of-the-art, sequence-based method that ranked very high in the CAFA 2 competition [Jiang et al., 2016a]. GoFDR[5] was shown to perform well on both "hard" and eukaryote targets. Note that UNIREF90 contains substantially more sequences than our own yeast genome dataset (including orthologues), giving $\text{GoFDR}_{\text{U90}}$ a significant advantage in terms of sequence information.
- $\text{GoFDR}_{\text{yeast}}$: GoFDR trained only on the same sequences used by OCELOT. Since only yeast sequences are considered, the parameters of PSI-BLAST (as used by GoFDR) were adjusted to capture even lower confidence alignments (namely by increasing the E-value threshold to 0.9 and the number of iterations from 3 to 4).
- BLAST: an annotation transfer approach based on BLAST, used as baseline in the CAFA2 competition[6].
- OCELOT with only GO consistency rules (i.e. no protein–protein interactions), and with no rules at all. We refer to these two baselines as $\text{OCELOT}_{\text{noppi}}$ and $\text{OCELOT}_{\text{indep}}$, respectively.

All methods were evaluated using a 10-fold cross-validation procedure: the proteins were split into 10 subsets, 9 of which were used for parameter estimation[7],

[5] Software taken from `http://gofdr.tianlab.cn`.

[6] Software taken from `https://github.com/yuxjiang/CAFA2`.

[7] OCELOT hyper-parameters: λ_l and λ_h penalize respectively the supervised label loss and the infringement of the structural constraints

and the remaining one for evaluation. The folds were constructed by distributing functional and interaction annotations among them in a balanced manner using a greedy procedure. Interactions were split similarly.

In addition, we also compared OCELOT against DeepGO [Kulmanov et al., 2018], a state-of-the-art deep learning approach that exploits sequence and PPI data. In contrast to the other methods, the results for DeepGO were obtained from its web interface[8]. Having no control over the ontology used by DeepGO, we had to limit the comparison to the overall perfomance computed on the terms in common between our and DeepGO's ontologies.

Performance measures. Following the CAFA2 procedure, predicted annotations were evaluated using both protein-centric and term-centric performance measures [Jiang et al., 2016a]. Protein-centric measures include the $F_{\max}$ and $S_{\min}$ scores, we also evaluated the predicted annotations using the F_1 score, i.e. the $F_{\max}$ score with τ fixed to 0.5. We used the Area under the Receiver Operating Characteristic Curve (AUC) for the term-centric evaluation.

Discussion. The overall performance of all predictors can be found in Figure 4.2. At a high level, all prediction methods tend to perform better than both the simple BLAST baseline, as expected, and $\text{GoFDR}_{\text{yeast}}$. This is hardly surprising: despite being configured to consider even distantly related homologues (by tweaking the PSI-BLAST parameters, as mentioned above), $\text{GoFDR}_{\text{yeast}}$ could not transfer any annotations to 1133 targets, as no alignment could be found in the yeast-only training set. Allowing GoFDR to access extra-genomic sequences solves this issue, as shown by the improved performance of $\text{GoFDR}_{\text{U90}}$ over $\text{GoFDR}_{\text{yeast}}$.

On the other hand, OCELOT, $\text{OCELOT}_{\text{noppi}}$, and $\text{OCELOT}_{\text{indep}}$, perform as well or better than $\text{GoFDR}_{\text{U90}}$ in terms of $F_{\max}$ and $S_{\min}$. The overall performance on BP and MF are rather close, while for CC the OCELOT-based methods offer a large improvement: the $F_{\max}$ and $S_{\min}$ of OCELOT are approximately 9% better (resp. higher and lower) than those of $\text{GoFDR}_{\text{U90}}$.

More marked improvements can be observed in the F_1 plots. The kernel-based methods perform as well or better than $\text{GoFDR}_{\text{U90}}$ in all GO domains. This holds despite the task being very class unbalanced (especially at the lower levels of the hierarchy), and the decision threshold being fixed at 0.5. In CC and MF, the biggest contribution comes from the hierarchy consistency rules. In contrast, consistency to the

[8]The DeepGO package does not provide a procedure for training the model on our yeast dataset. The predictions were retrived from `http://deepgo.bio2vec.net/deepgo/` on 14th June 2018.

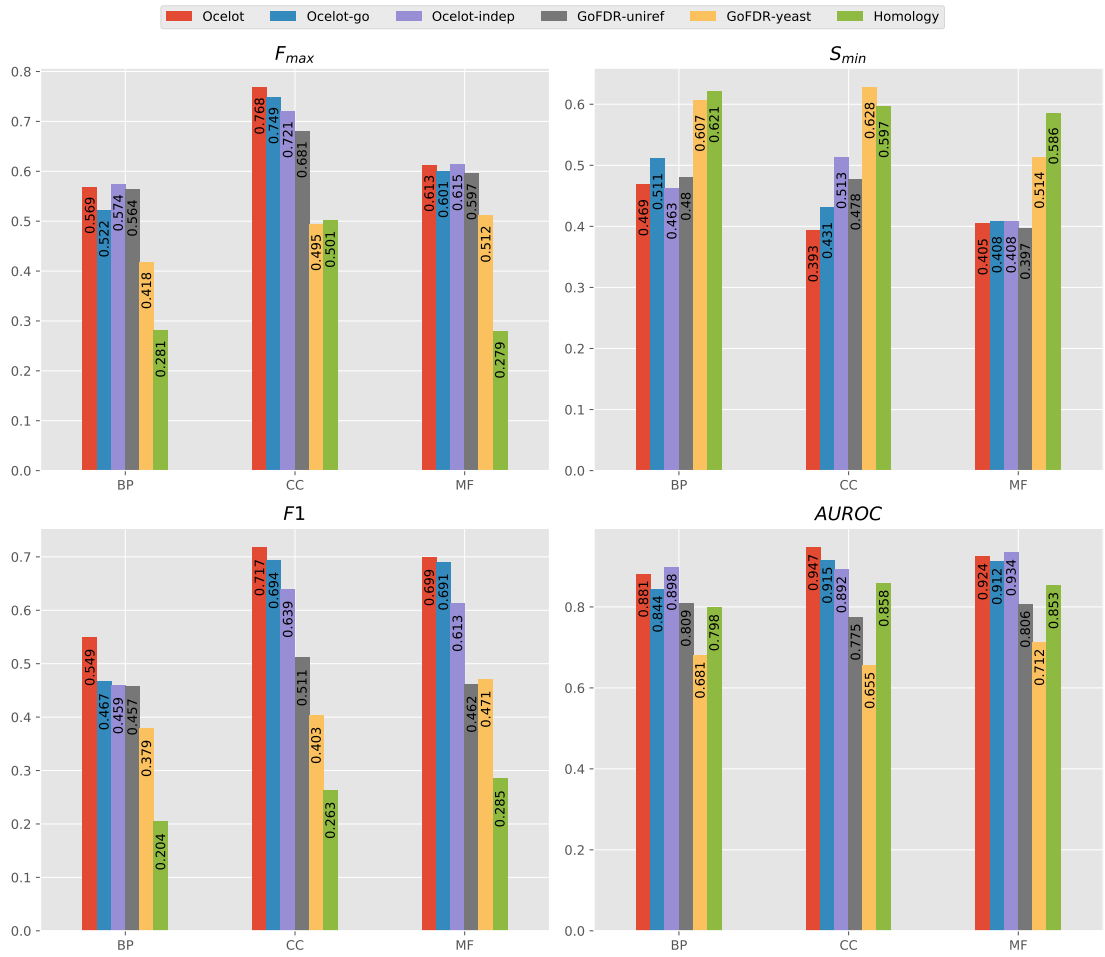


Figure 4.2: **Overall performance of all prediction methods on the Yeast dataset**. From top to bottom: $F_{\max}$, $S_{\min}$, F_1, and AUROC.

protein–protein interaction network seems to be the biggest factor for BP: OCELOT offers an 8% F_1 improvement over $\text{OCELOT}_{\text{indep}}$, $\text{OCELOT}_{\text{noppi}}$ and $\text{GoFDR}_{\text{U90}}$.

A breakdown of the performance at different term depths is provided in Figure 4.3. The general trend is the same as above: all methods outperform the baseline and $\text{GoFDR}_{\text{yeast}}$, and OCELOT with the full set of rules has the overall best performance. In all cases, the performance of the $\text{OCELOT}_{\text{indep}}$ is comparable to that of OCELOT at the top levels, however it quickly degrades with term depth. This implies that the consistency rules are successfully propagating the correct predictions down the hierarchy. This is especially evident for the cellular component domain. For the molecular function domain, the bottom levels are predicted as good as the top ones, and much better than the intermediate levels. This is actually an artifact of the sparsity in annotations at the lowest levels (recall that we dropped terms with

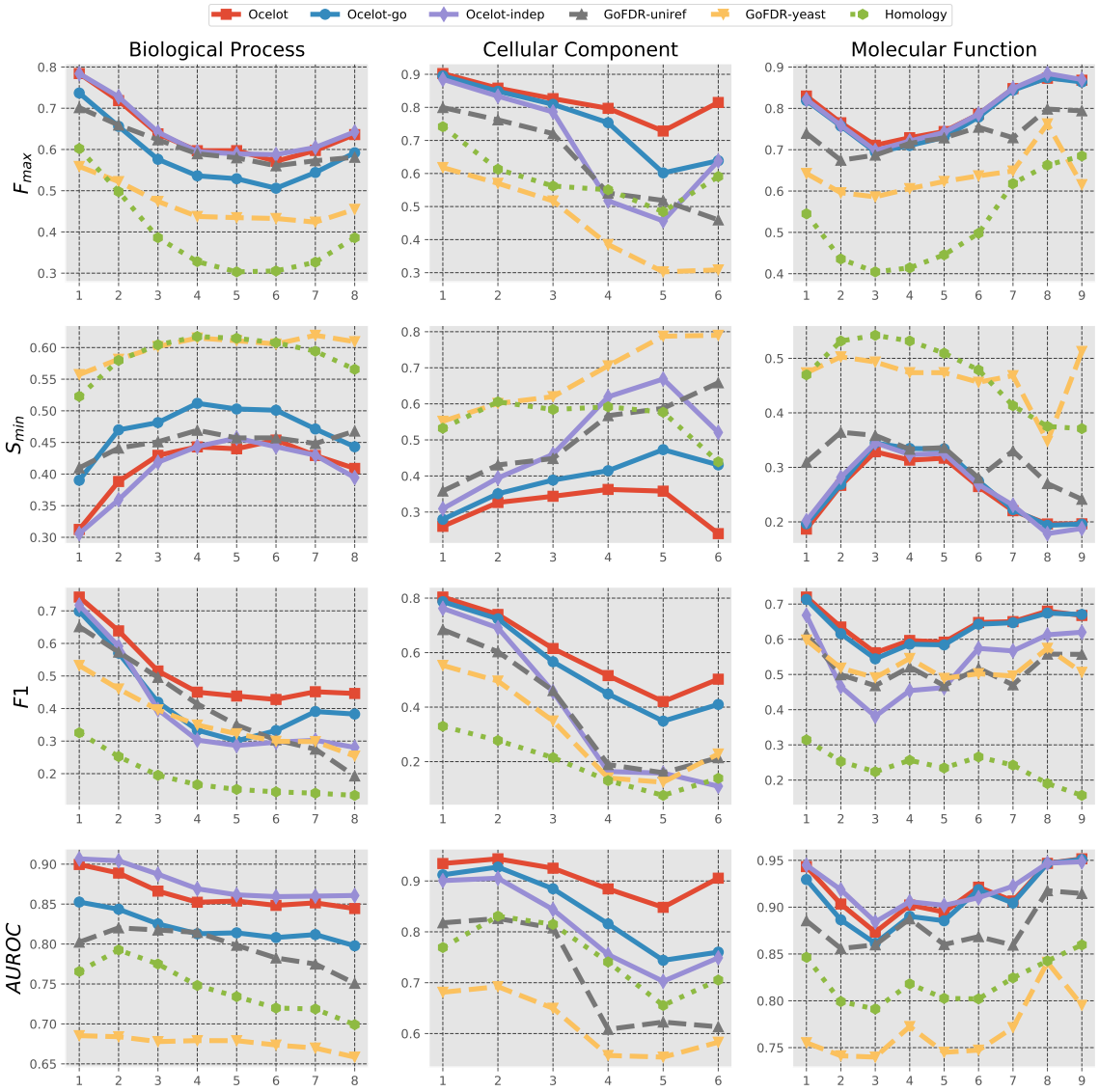


Figure 4.3: **Breakdown of the performance of all methods at different GO term depth**. From left to right: biological process, cellular component, and molecular function. From top to bottom: $F_{\max}$, $S_{\min}$, F_1 and AUROC.

less than 20 annotations, which drastically reduces the number of terms which are predicted in the lowest levels, especially for MF).

Few examples can help highlighting the role of the rules to enforce consistency in predictions. For example, the taxonomical consistency fixes some false-negative classifications for the `MAS2` protein, which is correctly re-assigned to the `mitochondrion` and `mitochondrial-part`. When also considering the consistency with respect of the PPI predictions, the `protein-complex` localization is also correctly predicted for the same protein. Note that the boost in performance given by the PPI rules is

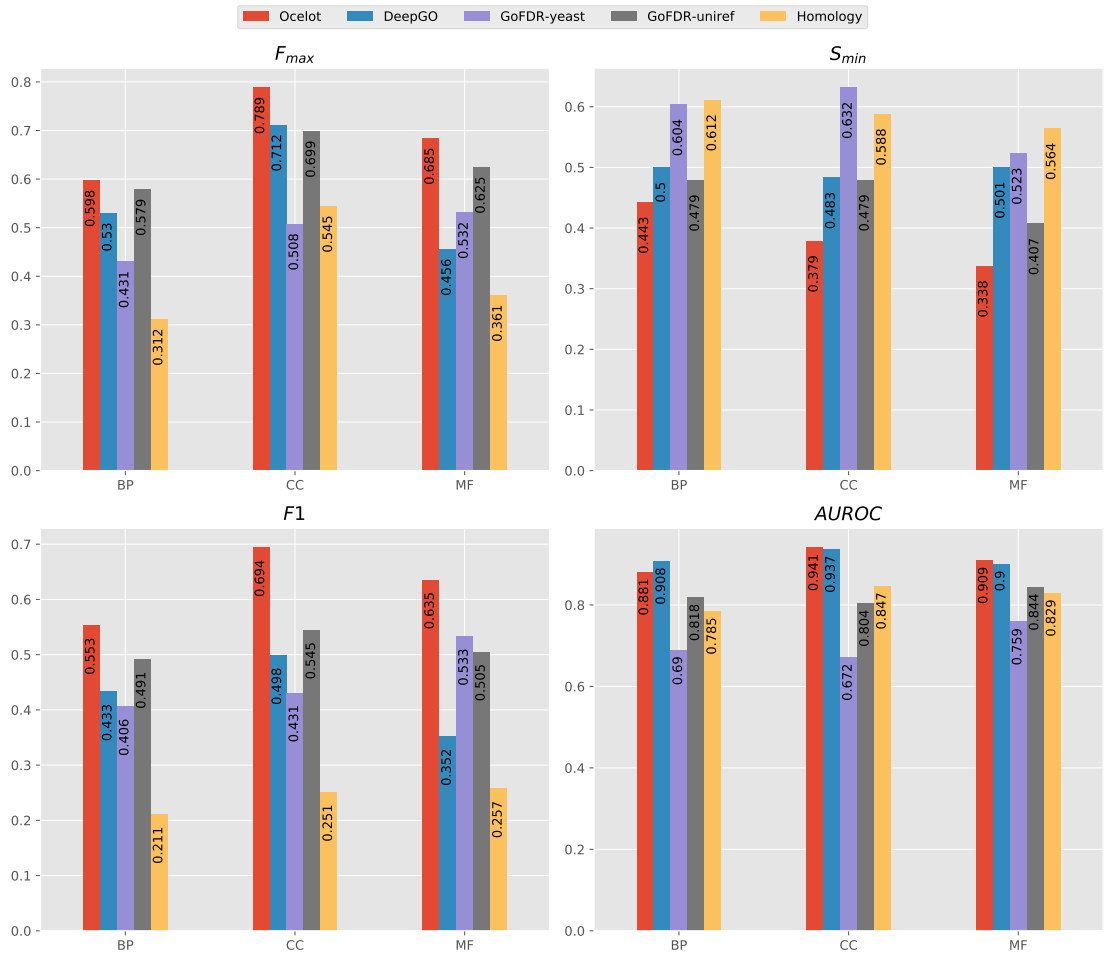


Figure 4.4: **Overall performance of DeepGO, Ocelot, GoFDR and the baseline on the Yeast dataset**. From top to bottom: $F_{\max}$, $S_{\min}$, F_1, and AUC.

achieved regardless of the fact that interactions are predicted and not observed. On the other hand PPI prediction performance (equal to 0.69 F_1) is not affected by the introduction of the GO rules. As already mentioned, this is an expected result as PPI prediction is a comparatively simpler task, and information tends to propagate from simpler tasks to more complex ones. A similar result has been observed in multi-level interaction prediction, where propagation goes from the protein to the domain and residue level but not viceversa [Saccà et al., 2014].

We also compare Ocelot to DeepGO, a state-of-the-art deep learning-based predictor [Kulmanov et al., 2018]. Since we could not train DeepGO on our ontology, we compare the methods only on the terms shared by our and DeepGO's ontology. The results are shown in Figure 4.4. The results confirm the ones obtained by Kulmanov et al. [2018], where DeepGO outperforms GoFDR in terms of AUC. On

the other hand, OCELOT and DeepGO perform comparably, in terms of AUC, with some slight variation between different aspects. Note that this holds regardless of the fact that DeepGO was trained on many more sequences than OCELOT, and that it uses true interaction data. In contrast, OCELOT has only access to yeast sequences, and only to predicted protein interactions. Most importantly, OCELOT outperforms DeepGO on all aspects for all other performance measures ($F_{\max}$, $S_{\min}$, and F_1). The performance of DeepGO is especially poor under the F_1 metric, showing that the predictor is not suitably calibrated against the natural decision threshold $\tau = 0.5$.

4.4 Conclusion

We introduced OCELOT, a predictive system capable of jointly predicting functional and protein-protein interaction annotations for all proteins of a given genome. The system combines kernel machine classifiers for binary and pairwise classification with a fuzzy logic layer enforcing consistency constraints along the GO hierarchy and between functional terms and interaction predictions. We evaluated the system on the Yeast genome, showing how the rule enforcement layer manages to substantially improve predictive performance in functional annotation, achieving results which are on par or better (depending on the GO domain and performance measure) than those of a state-of-the-art sequence-based approach fed with annotations from multiple genomes.

OCELOT can be extended in a number of directions. The system is currently conceived for intra-genome annotation. As a further research direction, OCELOT could be integrated with multiple kernel learning (MKL) approaches Sonnenburg et al. [2006]; Gönen and Alpaydın [2011]. Indeed, preliminary results conducted in Masera [2015] showed that single kernel performs differently with respect to the GO domain, highlighting how the average kernel may be a sub-optimal choice. A second major extension consists of adapting it to process multiple genomes simultaneously. This requires to incorporate both novel specialized predictors, like an orthology-based annotator [Pearson, 2013], and additional inter-genome rules, e.g. encouraging (predicted) orthologues to interact with the same partners. A third research direction consists in broadening the type of annotations provided by the system, by e.g. generalizing interaction prediction to the prediction of biochemical pathways [Gabaldón and Huynen, 2004]. Care must be taken in encoding appropriate rules in order to ensure consistent predictions without excessively biasing the annotation.

Chapter 5

Consistent Hierarchical-Multilabel Classification with Artificial Neural Networks

In the previous chapter we introduced Ocelot, a predictive pipeline that exploits Semantic Based Regularization, to perform genome-wide protein annotation. The system shows good results in the intra-organism setting, outperforming the state-of-the-art. However the kernel machinery has intrinsic scaling limitations. Indeed both explicit and implicit representation of the feature space come with large drawbacks when both the number of examples and features are very large. More-over, Ocelot penalizes the infringement of the hierarchical constraint, but does not guarantee the consistency of the prediction, and Obozinski et al. [2008] showed that users are more likely to discard inconsistent predictions.

The introduction of powerful GPU architectures brought artificial neural networks (ANNs) back to the limelight [Krizhevsky et al., 2012; Hinton et al., 2012; Srivastava et al., 2014]. The possibility to scale the learning process with highly-parallel computing frameworks allowed the community to tackle completely new problems or old problems with completely new and complex ANNs' topologies. However, ANN-based methods that account for hierarchical-multilabel classification (HMC) have not yet evolved consequently. Attempts to integrate ANNs and HMC have been conducted by Cerri et al. [2014, 2015]. They propose HMC-LMLP, a local model where for each term in the hierarchy an ANN is trained, that is fed with both the original input and with the output of models built for the parent terms. The performance are comparable with CLUS-HMC, however, because of the many models trained, the proposed approach is not scalable with deep learning ar-

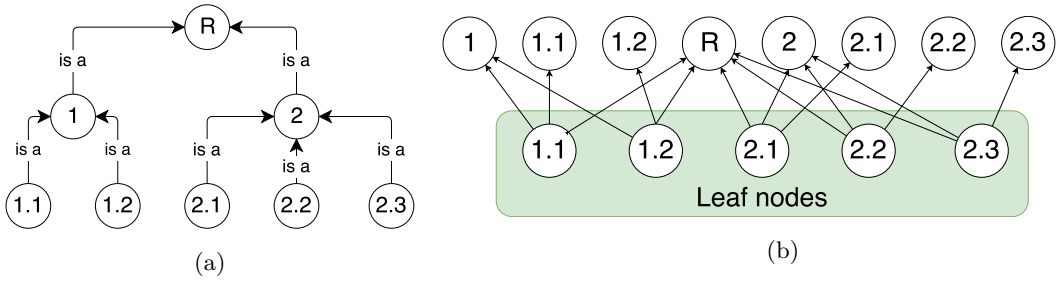


Figure 5.1: **From hierarchy to AWX**. **a**: Hierarchical tree structure. Sub-tree of the FunCat [Ruepp et al., 2004] annotation tree. **b**: Adjacency scheme described by E' starting from the adjacent tree.

chitectures that require a considerable amount of time for training. To the best of our knowledge there are no better models that exploit ANNs in the training process to specifically tackle the HMC problem.

In this chapter we present AWX (Adjacency Wrapping matriX), a novel ANN output component. We aim at filling the gap between HMC and ANNs left open in the last years, enabling HMC tasks to be tackled with the power of deep learning approaches. The proposed method incorporates the knowledge on the output-domain directly in the learning process, in form of a matrix that propagates the signals coming from the previous layers. The information flows from the leaves, up to the root, allowing a joint optimization of the predictions. We propose and discuss two approaches to combine the incoming signals, the first is based on the `max` function, while the second on ℓ-norms. AWX can be incorporated on top of any ANN, guaranteeing the consistency of the results with respect to the true path rule (TPR, see Section 3.1 for details). Moreover, we propose a generalization of the TPR to the continuous case and we prove that AWX is consistent with both definitions. Finally AWX is evaluated on ten benchmark datasets and compared against clus-HMC [Vens et al., 2008], that is the state-of-the-art, HMC-LMLP [Cerri et al., 2014, 2015] and the simple multi-layer perceptron MLP.

5.1 Model description

This section describes the AWX hierarchical output layer that we propose in this paper, using the notation introduced in Section 3.1. Consider an artificial neural network with L hidden layers and the DAG representing the hierarchy $H = (\mathcal{T}, E)$. Let

$$E' = \{\langle t_u, t_v \rangle | t_u \in \mathcal{F}, t_u = t_v \vee t_v \in \mathit{ancestors}(t_u)\}.$$

Note that for each $\langle t_u, t_v \rangle \in E'$ holds that $t_u \in \mathcal{F}$. Let $\mathbf{R}$ be a $|\mathcal{F}| \times m$ matrix that represents the information in E', where $r_{i,j} = 1$ iff $\langle t_i, t_j \rangle \in E'$ and 0 otherwise. Fig. 5.1b shows an example of the topology described by E'.

Now, let $\mathbf{y}^L$, $\mathbf{W}^L$ and $\mathbf{b}$ denote respectively the output, the weight matrix and the bias vector of the last hidden layer in the network and $\mathbf{r}_i$ the i-th column vector of $\mathbf{R}$. The AWX hierarchical layer is then described by the following equation

$$\begin{aligned} \mathbf{z} &= \mathbf{W}^L \cdot \mathbf{y}^L + \mathbf{b}^L \ , \\ \hat{y}_i &= \texttt{max}(\mathbf{r}_i \circ (f(z_1), \ldots, f(z_{|\mathcal{F}|}))^T) \end{aligned} \tag{5.1}$$

where $\circ$ is the symbol of the Hadamard product, `max` is the function returning the maximum component of a vector, and f is an activation function $f : \mathbb{R} \to [0, 1]$ (e.g. the sigmoid function). This constraint on the activation function is required to guarantee the consistency of the predicted hierarchy as we will show in Section 5.2.1.

Being $\mathbf{R}$ binary by definition, the Hadamard product in Equation 5.1, acts as a mask, selecting the entries of $\mathbf{z}$ corresponding to the non-zero elements of $\mathbf{r}_i$.

The `max` represents a straightforward way of propagating the predictions through the hierarchy, but it complicates the learning process. Indeed, the error can be back-propagated only through the maximum component of $\mathbf{r}_i$, leading to local minima. The `max` function can be approximated by the ℓ-norm of the incoming signals as follows

$$\hat{y}_i = \begin{cases} ||\mathbf{r}_i \circ f(\mathbf{z})||_\ell, & \text{if } \ < 1 \\ 1, & \text{otherwise.} \end{cases} \tag{5.2}$$

The higher ℓ, the more similar the results will be to the ones obtained by the `max`, the closer is ℓ to 1 the more each component of the vector contributes to the result. Figure 5.2 shows a two-dimensional example, and we can see that with $\ell = 5$ the norm is already a good approximation of the `max`. On the other hand, we can notice

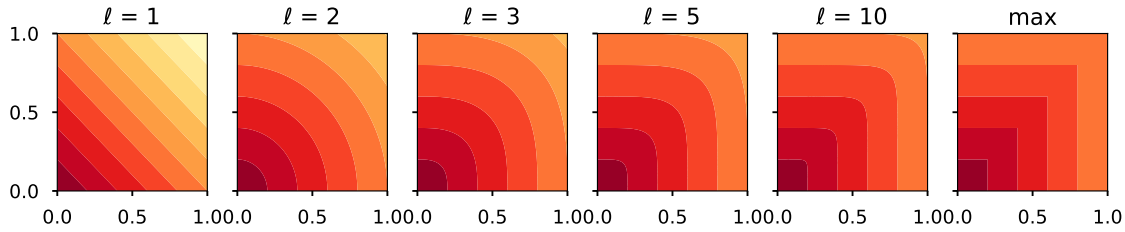
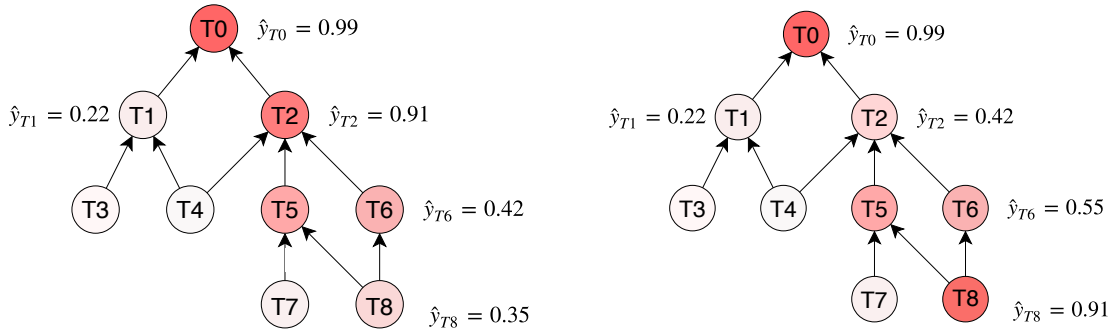


Figure 5.2: **Comparison of `max` and ℓ-norms**. Shape of the function $z = ||(x, y)^T||_\ell$ at different values of l and the comparison with the `max` function. Darker colors corresponds to lower values of z, while brighter ones to higher.



(a) gTPR-compliant annotation.

(b) Not gTPR-compliant annotation.

Figure 5.3: **Generalized true path rule example**. Examples of hierarchical multilabel annotations, where the intensity of filling color is proportional to the confidence of the prediction. In the right hierarchy $\hat{y}_{T8} > \hat{y}_{T6} > \hat{y}_{T2}$, so the prediction violates the gTPR. Note that with $\tau = 0.4$ the annotation would be consistent with the simple TPR.

that, even if the input is in $[0, 1]$, the output exceeds the range and must therefore be clipped to 1. Especially with ℓ close to 1, the ℓ-norm diverges from the `max`, giving output values that can be much higher than the single components of the input vector.

5.2 Generalized TPR

In section 3.1 we introduced and formalized the TPR. As a brief memorandum:

Definition 3. The labelling function y observes the TPR iff

$$\forall t_u \in \mathcal{T}, t_u \in y(\mathbf{x}_i) \implies ancestors(t_u) \subset y(\mathbf{x}_i).$$

The above definition holds for binary annotations, but, as mentioned in Section 3.1.3, hierarchical classifiers often do not set thresholds and predictions are evaluated based only on the output scores order. We introduce here a generalized notion of TPR, namely the generalized TPR (gTPR), that expands the TPR to this setting. Intuitively it can be defined by imposing a partial order over the DAG of predictions' scores. In this way the TPR is respected for each global threshold.

Definition 4. The gTPR is respected iff $\forall \langle t_u, t_v \rangle \in E$ is true that $\hat{y}_v \geqslant \hat{y}_u$.

This means that for each each couple of terms in a *is_a* relation, the prediction scores for the parent term must be grater or equal to the one of the child. In the extreme case of predictions that have only binary values, the gTPR clearly coincide with the TPR.

5.2.1 The gTPR holds for AWX

In this section we prove the consistency of the AWX output layer with respect to the gTPR, introduced in Section 5.2.

We want to show that $\forall < t_u, t_v > \in E, \hat{y}_v \geqslant \hat{y}_u$ holds for $\hat{y}_v, \hat{y}_u$ in Equation 5.1.

Proof. Note that Eq. 5.1 can be rewritten as $\hat{y}_v = max(\mathcal{C}_v)$, where

$$\mathcal{C}_v = \{f(z_u) | < t_u, t_v > \in E'\}$$

is the set of the contributions to the predictions coming from the leaf terms. In the special case of leaf terms, $t_u \in \mathcal{F}$, by construction, $\mathcal{C}_u = \{f(z_u)\}$ therefore $\hat{y}_u = f(z_u)$. We can express the statement of the thesis as $\forall < t_u, t_v > \in E$,

$$\hat{y}_v = max(\mathcal{C}_v) \geqslant max(\mathcal{C}_u) = \hat{y}_u \tag{5.3}$$

Being the max function monotonic, the above inequality holds if $C_u \subseteq C_v$.

Consider the base case $\langle t_u, t_v \rangle \in E$ such that $t_u \in \mathcal{F}$. It clearly holds that $\mathcal{C}_u = \{f(z_u)\} \subseteq \mathcal{C}_v$, because if $\langle t_u, t_v \rangle \in E$ then $\langle t_u, t_v \rangle \in E'$ and therefore $f(z_u) \in \mathcal{C}_v$.

Now consider two generic terms in a "*is_a*" relation $\langle t_u, t_v \rangle \in E$ and their contributions sets $\mathcal{C}_u$ and $\mathcal{C}_v$. By design

$$\forall\ t_k \in \mathcal{F}, \langle t_k, t_u \rangle \in E' \implies \langle t_k, t_v \rangle \in E'$$

and therefore

$$\{f(z_k) | \langle t_k, t_u \rangle \in E'\} \subseteq \{f(z_k) | \langle t_k, t_v \rangle \in E'\}$$
$$\mathcal{C}_u \subseteq \mathcal{C}_v,$$

and Equation 5.3 holds. □

The reasoning proceeds similarly for the estimator $\hat{y}_i$ in Equation 5.2, but in order to guarantee the consistency the input must be in $[0, +\infty)$ since the ℓ-norm is monotonic only in that interval.

5.2.2 Implementation

The model has been implemented within the Keras [Chollet et al., 2015] framework and a public version of the code is available at `https://github.com/lucamasera/AWX`. The choice of Keras was driven by the will of integrating AWX into deep-learning architectures, and at the time of writing Keras represents a widely-used framework in the area.

An important aspect to consider is that AWX is independent from the underlying network, and can therefore been applied to any ANN that requires a consistent hierarchical output.

Dataset	d	FunCat		GO	
		$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{F}\rvert$
Cellcycle	77	499	324	4122	2041
Church	27	499	324	4122	2041
Derisi	63	499	324	4116	2037
Eisen	79	461	296	3570	1707
Expr	551	499	324	4128	2043
Gasch1	173	499	324	4122	2041
Gasch2	52	499	324	4128	2043
Hom	47034	499	324	4128	2043
Seq	478	499	324	4130	2044
Spo	80	499	296	4116	2037

Table 5.1: **Details of the benchmark datasets used in the experiments**. d, $|\mathcal{T}|$ and $|\mathcal{F}|$ are respectively the dimensionality of the dataset, the number of terms and how many of them are leaves.

5.3 Experimental setting

In order to assess the performance of the proposed model, an extensive comparison was performed on the standard benchmark datasets[1]. The datasets cover different experiments [Vens et al., 2008] conducted on *S. cerevisiae* annotated with two ontologies, i.e. FunCat and GO. For each dataset are provided the train, the validation and the test splits of size respectively of *circa* 1600, 800, and 1200. The only exception is the Eisen dataset where, the examples per split are *circa* 1000, 500, and 800. FunCat is structured as a tree with almost 500 term, while GO is composed by three DAGs, comprising more then 4000 terms. The details of the datasets are reported in Table 5.1 while Figure 2 reports the distribution of terms and leaves per level. Despite having many more terms and being deeper, most of the GO terms lay above the sixth level, depicting an highly unbalance structure with just few branches reaching the deepest levels.

Table 5.2: **ANN architecture used in the experiments**.

`fully_connected`(`size=1000`, `activation` = `relu`, `l2` = 10^{-3})
`awx`(`activation` = `sigmoid`, `l2` = 10^{-3})

AWX has been tested both with the formulation in Eq. 5.1 (AWX_{MAX}) and with

[1] `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`

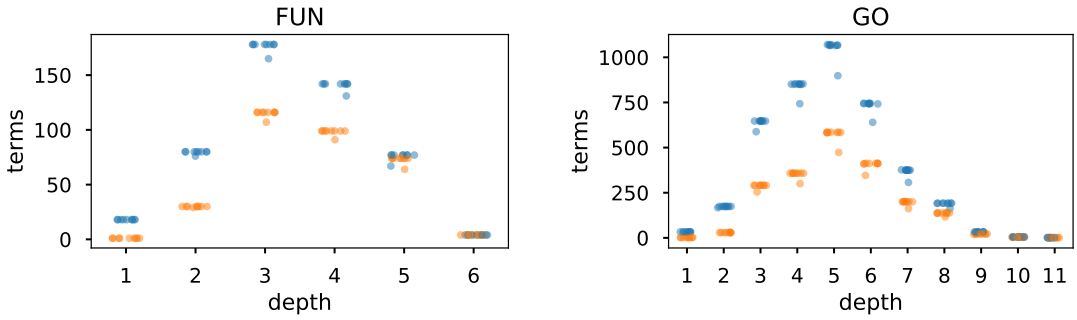


Figure 5.4: **Terms and leaves distribution by depth**. The figures show the distribution of terms and leaves by level. Each dataset has a blue marker for the number of terms and an orange one for the number of leaves.

the one of Eq. 5.2 ($\text{AWX}_{\ell=k}$ with $k \in \{1, 2, 3\}$). The overall scheme of the network employed in the experiments is reported in Table 5.2 and consists in just an hidden layer with 1000 units and AWX as output layer. The model has been trained with the ADAM optimizer algorithm [Kingma and Ba, 2014] ($lr = 10^{-5}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$) and loss function

$$L = \frac{1}{N} \sum_{i}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

for a maximum of 1000 epochs. An early stopping criterion with zero patience has been set on the validation set, such that, as soon as the performance on the validation test degrades, the learning is stopped.

The results of the four tested variants ($\text{AWX}_{\ell=1}$, $\text{AWX}_{\ell=2}$, $\text{AWX}_{\ell=3}$, AWX_{MAX}) have been evaluated with the metrics shown in Section 3.1.3 and compared against three other methods.

1. **HMC-clus** [Vens et al., 2008]: state-of-the-art model based on decision trees. The results reported are taken from the original work.

2. **HMC-LMLP** [Cerri et al., 2014, 2015]: The model is trained level by level, enriching the input space with the output of the previous level. As for HMC-clus, the reported results are taken from the original paper.

3. **MLP**$_{leaves}$: ANN trained only on the leaf terms with the same parameters of AWX. The prediction for the non-leaf terms are obtained by taking the `max` of the predictions for underlying terms in the hierarchy.

The comparison with MLP_{leaves} is crucial, because it highlights the impact of jointly learning the whole hierarchy with respect to inferring the prediction after the learning phase.

Both AWX and MLP_{leaves} are based on ANNs, so, in order to mitigate the effect of the random initialization of the weight matrix, the learning process has been repeated 10 times. We report the average results of the 10 iterations and the standard deviation ranges are reported in the caption of the tables. We performed a t-test with $\alpha = 0.05$ to assess the significativity of the difference with respect to the state-of-the-art and marked with * the results that passed the test.

No parameter tuning has been performed for the trained methods and the validation split has been used only for the early stopping criterion.

5.4 Results

In this section are reported and discussed the results obtained by the proposed method on ten benchmark datasets. Besides the comparison with the state-of-the-art, we will show the impact of AWX highlighting the differences with respect to MLP_{leaves}.

Table 5.3 reports the micro-averaged area under the precision recall curve ($\text{AUC}(\overline{\text{PR}})$). $\text{AWX}_{\ell=1}$ has a clear edge over the competitors, in both the ontologies. With the FunCat annotation, it is significantly better then CLUS-HMC six out of ten times and worse just in the Hom dataset, while with GO it wins nine out of ten times. $\text{AWX}_{\ell=1}$ clearly outperforms also the other AWX versions and MLP_{leaves} in all the datasets. We can notice that the performance tends to decrease for higher values of ℓ, and reaches the minimum with AWX_{MAX}. This can be explained by the distribution of the example-annotation: due to the TPR, the terms placed close to the root are more likely to be associated with more examples then the lower terms. With $\ell = 1$ each leaf, among the descendants, contributes equally to the prediction, boosting the prediction values of the upper terms.

Of great interest is the comparison between MLP_{leaves} and AWX_{MAX}. We can notice that AWX_{MAX}, despite being the worst performing version of AWX, always outperforms MLP_{leaves}. Remember that the main difference between the two architectures is that with AWX_{MAX} prediction are propagated at learning time and consequently optimized, while in MLP_{leaves} the predictions of the non-leaf terms are inferred offline.

Table 5.4 reports the macro-averaged area under the precision recall curves ($\overline{\text{AUCPR}}$). Unfortunately HMC-LMLP does not provide the score for this metric, but AWX clearly outperforms CLUS-HMC, both with the FunCat and with the GO annotations. We can notice that the differences are significant in all the datasets, with the exception of Hom, where the variance of the computed results is above 0.04

Table 5.3: **Performance of AWX with AUC($\overline{\mathbf{PR}}$)**. Bold values show the best preforming method on the dataset. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ and $[0.002, 0.008]$ respectively for FunCat and GO, with the only exception of Hom, where is an order of magnitude bigger.

		CLUS-HMC	HMC-LMLP	MLP_{leaves}	$\text{AWX}_{\ell=1}$	$\text{AWX}_{\ell=2}$	$\text{AWX}_{\ell=3}$	AWX_{MAX}
FunCat	Cellcycle	0.172	0.185	0.148*	**0.205***	0.189*	0.181*	0.174
	Church	0.170	0.164	0.102*	**0.173**	0.150*	0.136*	0.127*
	Derisi	**0.175**	0.170	0.112*	**0.175**	0.152*	0.142*	0.136*
	Eisen	0.204	0.208	0.196*	**0.252***	0.243*	0.234*	0.225*
	Expr	0.210	0.196	0.201	**0.262***	0.236*	0.229*	0.223*
	Gasch1	0.205	0.196	0.182	**0.238***	0.227*	0.217*	0.209
	Gasch2	0.195	0.184	0.150*	**0.211***	0.195	0.186	0.178*
	Hom	**0.254**	0.192	0.100*	0.107*	0.109*	0.106*	0.127*
	Seq	0.211	0.195	0.188*	**0.253***	0.234*	0.227*	0.218
	Spo	**0.186**	0.172	0.117*	0.179	0.159*	0.150*	0.143*
GO	Cellcycle	0.357	0.365	0.315*	**0.441***	0.406*	0.385*	0.362
	Church	0.348	0.347	0.272*	**0.440***	0.378*	0.355*	0.329
	Derisi	0.355	0.349	0.274*	**0.424***	0.376*	0.352	0.335*
	Eisen	0.380	0.403	0.347*	**0.481***	0.449*	0.426*	0.410*
	Expr	0.368	0.384	0.357*	**0.480***	0.437*	0.418*	0.407*
	Gasch1	0.371	0.384	0.346*	**0.468***	0.437*	0.416*	0.401*
	Gasch2	0.365	0.369	0.328*	**0.454***	0.417*	0.394*	0.379
	Hom	**0.401**		0.203*	0.264*	0.256*	0.242*	0.238*
	Seq	0.386	0.384	0.347*	**0.472***	0.429*	0.412*	0.397*
	Spo	0.352	0.345	0.278*	**0.420***	0.378*	0.355	0.336

with FunCat and 0.007 with GO. Within AWX is instead more difficult to identify a version that performs clearly better then the others, their performance are almost indistinguishable. Focusing on the differences between MLP_{leaves} and AWX_{MAX}, we can see that the former is outperformed fourteen out of twenty times by the latter. The advantage of the AWX layer in this case is not as visible as in terms of AUC($\overline{\text{PR}}$), because $\overline{\text{AUCPR}}$ gives equal weight to the curve for each class, ignoring the number of annotated examples. Within our setting, where most of the classes have few examples, this evaluation metric tends to flatten the results, and may not be a good indicator of the performance.

Table 5.5 reports the weighted-average of the area under the precision recall curves ($\overline{\text{AUCPR}}_w$). AWX has solid performance also considering this evaluation metric, outperforming significantly CLUS-HMC in most of the datasets. HMC-LMLP provides results only for the FunCat-annotated datasets, but appears to be not competitive with our method. Within the proposed variants of AWX, $\ell = 2$ has

Table 5.4: **Performance of AWX with $\overline{\text{AUCPR}}$**. Bold values show the best preforming method on the dataset. HMC-LMLP provides no results for this metric, so it has been removed from the table. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ for both FunCat and GO, with the only exception of Hom, where it is an order of magnitude bigger.

		CLUS-HMC	MLP_{leaves}	$\text{AWX}_{\ell=1}$	$\text{AWX}_{\ell=2}$	$\text{AWX}_{\ell=3}$	AWX_{MAX}
FunCat	Cellcycle	0.034	0.068*	0.075*	0.076*	0.077*	0.076*
	Church	0.029	0.040*	0.040*	**0.041***	0.040*	0.041*
	Derisi	0.033	0.047*	0.047*	0.048*	**0.049***	0.048*
	Eisen	0.052	0.095*	0.103*	0.104*	**0.106***	**0.106***
	Expr	0.052	0.114*	0.120*	**0.121***	**0.121***	0.120*
	Gasch1	0.049	0.101*	0.108*	0.110*	**0.111***	0.109*
	Gasch2	0.039	0.069*	0.080*	**0.078***	0.077*	0.078*
	Hom	0.089	0.116	0.095	0.086	0.112	**0.164**
	Seq	0.053	0.126*	0.121*	**0.126***	**0.126***	0.124*
	Spo	0.035	0.043*	0.045*	0.045*	0.045*	0.045*
GO	Cellcycle	0.021	0.057*	0.059*	0.057*	0.057*	0.057*
	Church	0.018	0.034*	0.030*	**0.032***	0.031*	0.031*
	Derisi	0.019	0.038*	0.041*	**0.040***	**0.040***	0.039*
	Eisen	0.036	0.082*	**0.091***	0.089*	0.088*	0.088*
	Expr	0.029	0.092*	**0.104***	0.102*	**0.104***	0.102*
	Gasch1	0.030	0.076*	0.086*	0.086*	0.086*	0.085*
	Gasch2	0.024	0.065*	0.067*	0.066*	0.067*	0.066*
	Hom	0.051	**0.071**	0.042	0.046	0.042	0.053
	Seq	0.036	**0.130***	**0.130***	**0.130***	0.128*	0.128*
	Spo	0.026	0.037*	0.038*	0.039*	**0.040***	0.038*

an edge over $\ell = 1$ and MAX, while is almost indistinguishable from $\ell = 3$. Moreover, comparing AWX_{MAX} and MLP_{leaves}, we can see that the former is systematically better then the latter.

The results reported for AWX (in all its variants) and MLP were not obtained tuning the parameters on the validation sets, but rather setting them *a priori*. The lack of parameter-tuning is clear on the Hom dataset. With all the considered evaluation metrics, this dataset significantly diverges with respect to the others. This behaviour is common also to CLUS-HMC, but unlikely the proposed methods, it has the best performance on this dataset. An explanation of this anomaly can be found in Table 5.1, where we can see the difference in the dimensionality. Hom dataset features are indeed two order of magnitude more then the second biggest dataset, i.e. Expr. The sub-optimal choice of the model parameters and the dimensionality could therefore explain the deviation of this dataset from the performance trend, and these aspects should be explored in the future.

Table 5.5: **Performance of AWX with $\overline{\text{AUCPR}}_w$.** Bold values show the best preforming method on the dataset. HMC-LMLP provides no data for the GO datasets. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ for both FunCat and GO, with the only exception of Hom, where it is an order of magnitude bigger.

		CLUS-HMC	HMC-LMLP	MLP_{leaves}	$\text{AWX}_{\ell=1}$	$\text{AWX}_{\ell=2}$	$\text{AWX}_{\ell=3}$	AWX_{MAX}
FunCat	Cellcycle	0.142	0.145	0.186*	0.200*	0.204*	**0.205***	0.203*
	Church	0.129	0.118	0.132	**0.139***	**0.139***	0.139*	0.138
	Derisi	0.137	0.127	0.144*	0.150*	**0.152***	**0.152***	0.151*
	Eisen	0.183	0.163	0.229*	0.246*	**0.254***	**0.254***	0.252*
	Expr	0.179	0.167	0.239*	**0.260***	**0.260***	0.258*	0.255*
	Gasch1	0.176	0.157	0.217*	0.234*	**0.241***	0.240*	0.237*
	Gasch2	0.156	0.142	0.183*	0.200*	**0.202***	**0.202***	0.200 *
	Hom	**0.240**	0.159	0.185	0.185	0.188	0.193	0.222
	Seq	0.183	0.112	0.232*	0.260*	**0.263***	0.262*	0.258*
	Spo	0.153	0.129	0.152	0.159	**0.161***	**0.161***	0.160*
GO	Cellcycle	0.335		0.372*	0.379*	0.384*	**0.385***	0.380*
	Church	0.316		0.325*	0.328*	**0.331***	0.329*	0.327*
	Derisi	0.321		0.331*	0.334*	**0.338***	0.337*	0.336*
	Eisen	0.362		0.402*	0.418*	**0.426***	0.423*	0.418*
	Expr	0.353		0.407*	0.424*	**0.430***	**0.430***	0.427*
	Gasch1	0.353		0.397*	0.410*	**0.421***	0.419*	0.416*
	Gasch2	0.347		0.379*	0.386*	**0.394***	0.393*	0.388*
	Hom	**0.389**		0.354*	0.342*	0.349	0.345*	0.356
	Seq	0.373		0.408*	0.431*	**0.436***	0.434*	0.430*
	Spo	0.324		0.333*	0.332	**0.339***	**0.339***	0.338*

AWX has solid overall performance. $\text{AUC}(\overline{\text{PR}})$ is the most challenging evaluation metric, where the improvement over the state-of-the-art is smaller, while with the last two metrics, AWX appears to have a clear advantage. The choice of the value for ℓ depends on the metric we want to optimize, indeed AWX performs the best with $\ell = 1$ considering $\text{AUC}(\overline{\text{PR}})$, while if we consider $\overline{\text{AUCPR}}$ or $\overline{\text{AUCPR}}_w$ values of $\ell = 2$ or $\ell = 3$ have an edge over the others. The direct comparison between MLP_{leaves} and AWX_{MAX} is clearly in favour of the latter, which wins almost on all the datasets. This highlights the clear impact of the proposed approach, that allows a jointly optimization of all the classes in the datasets.

5.5 Conclusion

In this chapter we have proposed a generalization to the continuous domain of the true path rule and presented a novel ANN layer, named AWX. The aim of this

component is to allow the user to compute consistent hierarchical predictions on top of any deep learning architecture. Despite the simplicity of the proposed approach, it appears clear that AWX has an edge over the state-of-the-art method CLUS-HMC. Significant improvements can be seen almost on all the datasets for the macro and weighted averaged area under the precision recall curves evaluation metric, while the advantage in terms of AUC($\overline{\text{PR}}$) is significant in six out of ten datasets.

Part of improvement could be attributed to the power and flexibility of ANN over decision trees, but we have shown that AWX systematically outperforms also HMC-LMLP, based on ANN, and MLP_{leaves}, that has exactly the same architecture as AWX, but with sigmoid output layer just for the leaf terms.

Further work should be focused to test the proposed methods with real-world challenging datasets, integrating AWX in deep learning architectures. As future research, it would be interesting to investigate the performance of AWX with semantic-based metrics, both globally or considering only the leaf terms.

Chapter 6

Binary Classification from Unknown Multilabel Annotation Space

The binary classification task occurs when "*the input is to be classified into one, and only one, of two non-overlapping classes*" [Sokolova and Lapalme, 2009]. In the case in which the classification rule is known only by a set of samples, the task represents one of the basic supervised tasks in machine learning, widely addressed in the literature [Quinlan, 1986; Cortes and Vapnik, 1995; Freund and Schapire, 1997; Breiman, 2001]. The effectiveness of each model, however, depends critically on the fact that the model is correct, namely its form represents the underlying generative phenomenon, provided the right choice of parameters. In this case, all the available data can be used to fit the model and an unbiased global model can be identified.

As noted by Hand and Vinciotti: "*However, the truth is that the model is hardly ever correct, and is sometimes ill-specified. There are almost always aspects of the relationships between the predictor variables and the response which are not reflected in the assumed model form*" [Hand and Vinciotti, 2003]. In other terms, in the majority of the cases the assumption of correctness of the model is not true and can therefore mislead global learning algorithms. An important example of this scenario is disease diagnosis, which consists in determining whether a patient is affected by a disease, given his medical record. Disease can occur in various facets, with different symptoms and disorders. Take for example the family of autoimmune diseases, and a classifier that predict whether or not a patient is affected by it. A multi-target predictor that takes into consideration all sub categories would be the preferable one, but the lack of training data or the intrinsic incompleteness of the annotation

space could make this approach not viable. Forced to binary classification, it is very difficult for global learning algorithms to distinguish sick patients from healthy ones due to the uneven distribution of examples. Hence, there is a clear need to treat these problem with local learning algorithms.

The notion of *locality* in learning has a long history. Local models appeared first in density-estimation [Parzen, 1962] and regression models [Nadaraya, 1964; Watson, 1964], where kernels were used to control the influence of the samples to the overall model. The classical k-Nearest Neighbors classifier [Cover and Hart, 1967] is inherently a local method. In Nearest Neighbor and derived methods the attention focuses on ways of defining the distances or metrics to be used to find the set of neighbors and on the transformations of the space [Wang and Sun, 2014; Dutta and Ghosh, 2016]. Moreover, theoretical results on k-Nearest Neighbors [Fukunaga and Hostetler, 1975] gave a glimpse of the power of local models and, more generally, Bottou and Vapnik [1992] established a fundamental result demonstrating that the local versions of base learners have better bounds on the generalization errors. The Vapnik and Bottou result leaves us with an effective strategy to improve classifiers by adding locality.

A straightforward way to achieve locality is to restrict the application of the learning method to local subsets of the samples. Following this approach local versions of the Support Vector Machine (SVM) has gained attention and empirically proved to be competitive [Segata and Blanzieri, 2010] and theoretically proved to be consistent [Hable, 2013]. Another way to achieve locality is to define functions that weights the effect of samples over the model whereas the learning step is performed on the whole dataset. This is the approach used in Radial Basis Function networks [Poggio and Girosi, 1990] and, most notably, in the popular SVM with RBF, i.e. gaussian kernels [Scholkopf et al., 1997].

Deep learning approaches [LeCun et al., 2015], in which general-use features are learnt and then used as an input of other layers within a multilayer perceptron architecture, have attracted a growing attention since the substantial improvement of their learning procedure [Hinton et al., 2006; Bengio et al., 2007; Poultney et al., 2006]. In these approaches, which proved to be very successful in several applications, different layers of simple processing units are stacked. The layers compute features of growing richness, the emphasis being on the actual deep representation discovered in the process and encoded in the parameters [Bengio et al., 2013]. The way the features are learnt in deep learning architectures can vary [Schmidhuber, 2015]. We mention here auto-encoders [Vincent et al., 2010] that map the input vectors onto themselves. Some of the approaches incorporates locality aspects, like

for example the classical convolutional neural networks [LeCun et al., 1998] where topological information about the features is exploited. A thorough discussion of locality in the fast-growing literature on deep learning is beyond the scope of this thesis. However, to the best of our knowledge, no explicit attempt to incorporate subsampling-based local models in features definition, in order to exploit the advantages guaranteed by the result of Vapnik and Bottou, has been presented yet.

In this chapter we present a novel approach to binary classification that is based on the idea of locality, and combine it with a classifier architecture typical of deep approaches. The main idea is to use a number of models to define linear separators, combine them with a confidence function that incorporates the information about the position of the samples and that uses the results as input of a single-layer perceptron. The rationale of the approach, which is motivated by the theoretical bound on local models given by Vapnik and Bottou, is to leverage the notion of locality to achieve good features that can be used in multi-layered classifiers.

In order to test the effectiveness of this idea, we defined a "concept" classifier called *Very Simple Classifier* (VSC) [Masera and Blanzieri, 2019b] that incorporates an extreme version of the approach. In the case of VSC the local models are built using just 2 samples, the confidence function is based on geometric considerations and we show that it modulates locality in a way that is based on the generalized Chebichev inequality. Finally the parameters of the final perceptron are found with a regularized pseudo inverse. VSC is tested on a battery of benchmark datasets against relevant competitors. Despite its simplicity, the results of VSC are surprisingly good, showing that VSC is competitive with MLP and it outperforms other classifiers in the binary classification task. An exploration in the parameter space completes the comparison with MLP.

The chapter is organized as follows. The remaining of the introduction is devoted to a brief notational introduction to the binary classification task. Section 6.1 presents the details of VSC whose empirical evaluation is presented in Section 6.2. Finally, we draw our conclusions.

Binary Classification Task

Let us assume to have an input normed space $\mathbb{R}^n$ and a set (of labels) $L = \{-1, 1\}$, and N samples $(\mathbf{x}_i, \mathbf{y}_i) \in S \times L$ for $i = 1, \ldots, N$ such that the $\mathbf{x}_i$ are i.i.d. variables of an unknown distribution $f(\mathbf{x})$. Let $\mathbf{y}_i = y(\mathbf{x}_i)$ with $y : S \to L$ be an unknown function that associates the sample $\mathbf{x}_i$ with its label $\mathbf{y}_i$. Hence, the binary classification is the task of finding an estimator function $\mathring{y} : S \to L$ such that the expectation of the loss function $E_f(\mathcal{L}(\mathring{y}(\mathbf{x}), y(\mathbf{x})))$ is minimized, where $\mathcal{L} : L \times L \to \mathbb{R}$. The

Algorithm 2: VSC learning algorithm

Data: training data $\mathbf{X}$, labels $\mathbf{y}$, number of hyperplanes k, regularization factor λ

$\mathcal{P} \leftarrow$ select k pairs of examples of opposite class
foreach $j \leqslant k$ **do**
  $\mathbf{h}_j \leftarrow$ compute max margins hyperplanes for $p_j \in \mathcal{P}$
foreach $i \leqslant |\mathbf{X}|$ **do**
  foreach $j \leqslant k$ **do**
    $\mathbf{X}'[i,j] \leftarrow \tanh(\langle \hat{\mathbf{x}}_i, \mathbf{h}_j \rangle)\ \mathcal{C}_{p_j}(\mathbf{x}_i)$
$\mathbf{w} \leftarrow (\mathbf{X}'^T\mathbf{X}' + \lambda\mathbf{I})^{-1}\mathbf{X}'^T\mathbf{y}$

typical choice of the loss function $\mathcal{L}$ is the 0/1 loss, i.e. $\mathcal{L}(u,v) = 1$ if $u = v$ and 0 otherwise.

6.1 Very Simple Classifier

From a structural point of view, VSC is similar to a three-layer MLP with $n+1$ nodes in the first layer, $k+1$ nodes in the second, and just one in the third. The extra nodes in the first and second layer are used as biases. Procedurally VSC introduces significant novel differences based on subsampling and locality. The main steps of VSC are (I) *the pair selection procedure*, (II) *the pre-computation of the separating hyperplanes*, (III) *the confidence measure for the hyperplanes*, and (IV) *the regularized weights learning*.

As shown by the pseudo-code in Algorithm 2, the learning procedure starts with the selection of k pairs of examples $p := (\mathbf{x}_p^+, \mathbf{x}_p^-)$ such that $y(\mathbf{x}_p^+) = 1$ and $y(\mathbf{x}_p^-) = -1$. Given the "concept" nature of the VSC, these pairs are selected randomly among the training set examples. The sampled pairs are then used to compute k separating hyperplanes (to be described in Section 6.1.1). Following the parallel with the MLP, the precomputed hyperplanes are used as fixed weights for the network between the first and the second layer. The activation function of the second layer is an hyperbolic tangent which is down-weighted by a confidence measure (to be defined in Section 6.1.2). Being the weights fixed, the output of the second layer can be computed without further learning procedures. With the matrix of the outputs $\mathbf{X}'$ and the labels for the training set, the weights between the second and the third layer can be easily learned with the product of pseudo inverting the matrix $\mathbf{X}'$ with the vector of labels $\mathbf{y}$.

The following sections have the purpose to give the reader further details and the rationale of the VSC steps.

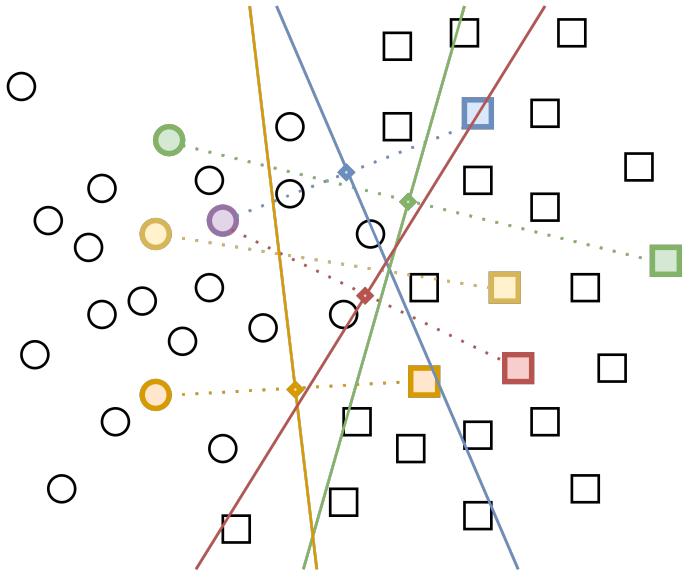


Figure 6.1: **Schematic representation of the hyperplane selection**.

6.1.1 Hyperplane selection

Given a pair of samples $p := (\mathbf{x}_p^+, \mathbf{x}_p^-)$ in the input space, a good separating hyperplane is the one that maximizes the margin. In this simple condition the maximum margin separating-hyperplane $\mathbf{h}_p$ is uniquely identified as the hyperplane perpendicular to $\mathbf{v}_p = \mathbf{x}_p^+ - \mathbf{x}_p^-$ and passing for their center $\mathbf{c}_p = (\mathbf{x}_p^+ + \mathbf{x}_p^-)/2$.

$$\mathbf{h}_p = (\mathbf{v}_p^1, \ldots, \mathbf{v}_p^n, \langle \mathbf{v}_p, \mathbf{c}_p \rangle)^T$$

where $\langle \cdot, \cdot \rangle$ is the inner product. There are, however, infinite formulations for this hyperplane. The canonical formulation for $\mathbf{h}_p$ by VSC is the hyperplane with unitary norm.

6.1.2 Hyperplane confidence

Each hyperplane selected at the previous stage depends only on 2 training samples, it is therefore important to add a confidence measure to limit its influence area. Let $\mathbf{x}_p^+$ and $\mathbf{x}_p^-$ be the samples used to build the hyperplane, and let $\mathbf{x}$ be the point to be classified with $\mathbf{h}_p$. Then the confidence measure $\mathcal{C}_p : \mathbb{R}^n \to (0, 1)$ is

$$\mathcal{C}_p(\mathbf{x}) = \sigma\left(\frac{d}{||\mathbf{x}_p^+ - \mathbf{x}||^2} + \frac{d}{||\mathbf{x}_p^- - \mathbf{x}||^2} - \frac{2d}{d^2}\right)$$

where $d = ||\mathbf{x}_p^+ - \mathbf{x}_p^-||/2$ and σ is the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. In the implementation a small $\epsilon = 0.01$ was added to each denominator in order to avoid

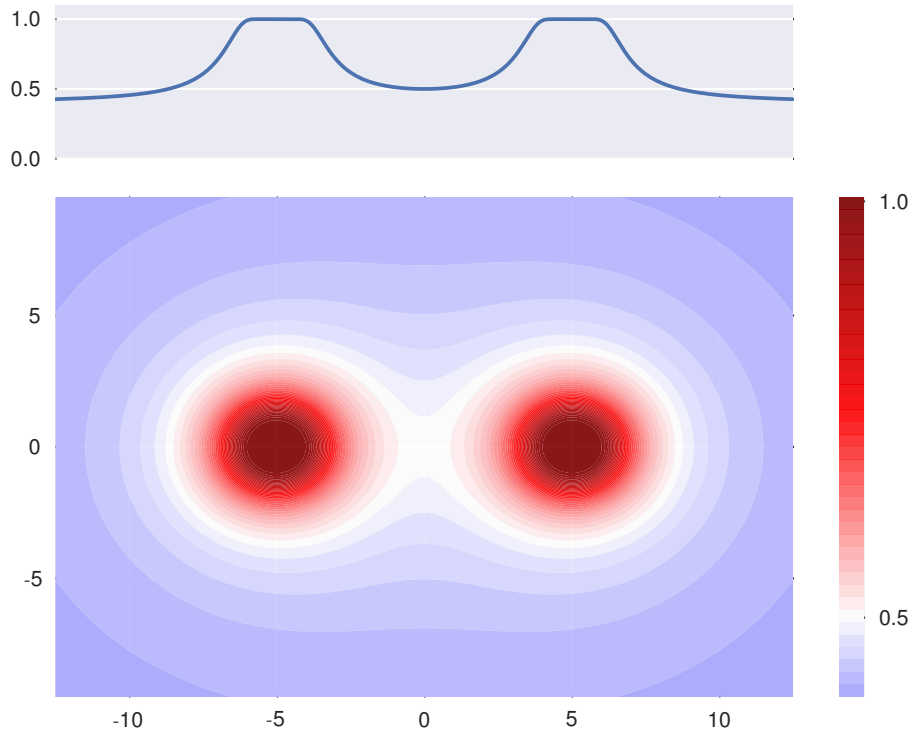


Figure 6.2: **Shape of the confidence measure**. The figure shows the heat map generated by the confidence measure with $\mathbf{x}_p^+ = (-5, 0)$ and $\mathbf{x}_p^- = (5, 0)$.

divisions by zero. In subsection 6.1.4 the formal characterization for this function will be made explicit, but the geometric intuition is that the confidence of $\mathbf{h}_p$ for the point $\mathbf{x}$ is high if $\mathbf{x}$ is close to $\mathbf{x}_p^+$ or to $\mathbf{x}_p^-$. The value of d plays the role of smoothing the confidence around $\mathbf{x}_p^+$ and $\mathbf{x}_p^-$, such that the higher the value of d, the wider and smoother the confidence region will be.

6.1.3 Learning the hyperplane weights

Once the hyperplanes $\mathbf{H}$ of the first layer have been selected, we can construct the matrix $\mathbf{X}'$:

$$\mathbf{X}' = (\mathbf{x}'_{i,j}) = \big(\tanh(\langle \hat{\mathbf{x}}_i, \mathbf{h}_j \rangle)\ \mathcal{C}_{p_j}(\mathbf{x}_i)\big)$$

where $\hat{\mathbf{x}} = (1, \mathbf{x}^1, \ldots, \mathbf{x}^n)^T$. $\mathbf{X}'$ is an $N \times k$ matrix where each entry $\mathbf{x}'_{i,j}$ is the result of the prediction for i-th training example $\mathbf{x}_i$ with only the j-th hyperplane $\mathbf{h}_j$ and the confidence measure. The weights for each hyperplane could be obtained by inverting the matrix $\mathbf{X}'$. In most cases, however, $k \neq N$, thus $\mathbf{X}'$ is not square and invertible. In order to compute the hyperplanes weights VSC takes advantage of the regularized pseudoinverse, also referred to as Tichonov regularization. This

choice is common in RBF networks and it has been used more recently in Extreme Learning machines (ELM) Huang et al. [2006] where the emphasis is on the speed of the computation. Thus

$$\mathbf{w} = (\mathbf{X'}^T\mathbf{X'} + \lambda\mathbf{I})^{-1}\mathbf{X'}^T\mathbf{y}$$

where $\mathbf{I}$ is the identity matrix of size $N \times N$. The effect of λ is to smooth the decision boundary, otherwise very prone to overfit: the higher the λ, the higher will be the regularization. In order to enhance the expressiveness of the VSC, a bias is added to this computation by adding 1 at the beginning of each line of the matrix $\mathbf{X'}$. The decision function for the VSC is thus:

$$y_{\text{VSC}}(\mathbf{x}) = \text{sign}\left(\sum_{p\in P} \mathbf{w}_p \tanh(\langle\hat{\mathbf{x}}, \mathbf{h}_p\rangle)\ \mathcal{C}_p(\mathbf{x})\right) + \mathbf{w}_0.$$

6.1.4 Characterization of the confidence in terms of Chebichev inequality

Given $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x} \in \mathbb{R}^n$ and x a multivariated random variable on $\mathbb{R}^n$, one of the generalization of the Chebichev inequality due to Grenader, as reported in Zhou and Hu [2012], can be written for the random variable $\mathbf{x}_1 - \mathbf{x}$ as:

$$Pr(||\mathbf{x}_1 - x|| \geqslant \epsilon) \leqslant \frac{E(||\mathbf{x}_1 - x||^2)}{\epsilon^2}.$$

If we choose to set ϵ as

$$\epsilon = \epsilon_1 = \frac{||\mathbf{x}_1 - \mathbf{x}||}{||\mathbf{x}_1 - \mathbf{x}_2||} \frac{\sqrt{E(||\mathbf{x}_1 - x||^2)}}{2} \tag{6.1}$$

than the inequality becomes:

$$Pr(||\mathbf{x}_1 - x|| \geqslant \epsilon_1) \leqslant 4\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{||\mathbf{x}_1 - \mathbf{x}||^2}$$

or equivalently

$$Pr(||\mathbf{x}_1 - x|| < \epsilon_1) \geqslant 1 - 4\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{||\mathbf{x}_1 - \mathbf{x}||^2}.$$

This inequality can be also written considering the point $\mathbf{x}_2$ and the corresponding ϵ_2. Summing up the two inequalities

$$2 \geqslant Pr(||\mathbf{x}_1 - x|| < \epsilon_1) + Pr(||\mathbf{x}_2 - x|| < \epsilon_2) \geqslant$$

$$2 - 4\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{||\mathbf{x}_1 - \mathbf{x}||^2} - 4\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{||\mathbf{x}_2 - \mathbf{x}||^2}$$

and dividing by $2||\mathbf{x}_1 - \mathbf{x}_2||$ we obtain

$$\frac{1}{||\mathbf{x}_1 - \mathbf{x}_2||} \geqslant \frac{1}{||\mathbf{x}_1 - \mathbf{x}_2||} \cdot \frac{Pr(||\mathbf{x}_1 - x|| < \epsilon_1) + Pr(||\mathbf{x}_2 - x|| < \epsilon_2)}{2} \geqslant$$

$$\geqslant \frac{1}{||\mathbf{x}_1 - \mathbf{x}_2||} - 2\frac{||\mathbf{x}_1 - \mathbf{x}_2||}{||\mathbf{x}_1 - \mathbf{x}||^2} - 2\frac{||\mathbf{x}_1 - \mathbf{x}_2||}{||\mathbf{x}_2 - \mathbf{x}||^2}.$$

If we set $\mathbf{x}_1 = \mathbf{x}_p^+$ and $\mathbf{x}_2 = \mathbf{x}_p^-$ in Equation 6.1 with corresponding ϵ_p^+

$$\epsilon_p^+ = \frac{||\mathbf{x}_p^+ - \mathbf{x}||}{||\mathbf{x}_p^+ - \mathbf{x}_p^-||} \frac{\sqrt{E(||\mathbf{x}_p^+ - x||^2)}}{2}$$

and analogous ϵ_p^-, and considering that the sigmoid is a monotonically increasing function we have

$$\mathcal{C}_p(\mathbf{x}) \geqslant \sigma\left(-\frac{Pr(||\mathbf{x}_p^+ - x|| < \epsilon_p^+) + Pr(||\mathbf{x}_p^- - x|| < \epsilon_p^-)}{d}\right).$$

The argument of the sigmoid function in the second term is at most zero, so that the bound does not guarantee that the confidence is bigger than 1/2. However, by considering the numerator approaching zero in the case of low probability of observing points relatively near to the pair, the bound can provide a lower bound to the confidence. In fact, an hyperplane spans a whole subspace and so its contribution to the prediction of a point $\mathbf{x}$ should be higher for a small probability to observe points relatively near to the pair that generated the hyperplane. In other terms, the data are less "local" and the confidence of the hyperplane contribution as a global predictor should be higher. Moreover, by considering the denominator increasing, the higher the distance between the points of the pair the higher is the confidence, this means that the two points are far apart and the simple model built with them, namely the max-margin classifier, should be applied in a wide range.

The above bound guarantees that the model is applied also non locally when the condition apply. This helps to clarify that VSC incorporates locality in a negative sense, by increasing the confidence of models that have chances of being less local. This prevents the direct application of the bound for local versions. Models that are local, however, are still applied locally for geometric considerations. In fact, the confidence is very high near the points of the pairs and it has a saddle point equal to 1/2 in the center of the pair.

6.2 Results

6.2.1 Experimental setup

VSC has been implemented in Python 2.7 following the scikit-learn standards. This choice allowed us to conveniently compare the VSC with other 9 well-known classifiers implemented in the scikit-learn suite Pedregosa et al. [2011], i.e. MLP, SVM with linear, polynomial and RBF kernels, AdaBoost, naïve Bayes, decision tree, random forests, and k-nearest neighbours classifiers. Moreover, we compared the performances of VSC with the ones of the Python implementation[1] of ELM Huang et al. [2006].

The experiments have been conducted on 22 datasets retrieved from the Keel archive Alcalá et al. [2011] with the only criteria of being binary classification problems with no categorical features. At the time of download (January, 2016), all the available datasets that satisfy the requirements have been taken into account. Given the relatively small size of the above datasets we added a more challenging real-world dataset Uzilov et al. [2006], whose results are discussed separately. The details of the datasets are reported in Table 6.1. The data have been normalized with a standard normalization by removing the mean and scaling to unit variance for each feature. The performances have been assessed with a 10-fold stratified cross validation. An internal 5-fold cross validation has been performed to assess the models' parameters. For each classifier has been used a grid search on the 5 values reported in Table 6.2. The $F1$ score has been preferred to the accuracy metric for presenting the results for its better robustness to unbalanced classes.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The statistical significance is assessed with a paired two-tailed t-test with significance level $\alpha = 0.05$.

The evaluation of the VSC has been organized in two main experiments, which will be explained in the following subsections.

6.2.2 Experiment 1

Experiment 1 has the goal of comparing the performance of VSC with those obtained by the competitors on all the datasets. Table 6.3 reports the complete results, and a graphical representation of the statistically-significant results is shown in Figure 6.3. Moreover, Table 6.4 reports the rankings obtained by the classifiers on the datasets.

[1] `https://github.com/dclambert/Python-ELM`

	# Examples	# Features	Pos (%)	Ref
appendicitis	106	7 (7/ 0)	80	Alcalá et al. [2011]
banana*	5300	2 (2/ 0)	55	Alcalá et al. [2011]
bands	365	19 (13/ 6)	63	Alcalá et al. [2011]
bupa	345	6 (1/ 5)	58	Alcalá et al. [2011]
coil2000	9822	85 (0/85)	94	Alcalá et al. [2011]
haberman	306	3 (0/ 3)	74	Alcalá et al. [2011]
heart	270	13 (1/12)	56	Alcalá et al. [2011]
hepatitis	80	19 (2/17)	84	Alcalá et al. [2011]
ionosphere	351	33 (32/ 1)	64	Alcalá et al. [2011]
magic	19020	10 (10/ 0)	65	Alcalá et al. [2011]
mammographic	830	5 (0/ 5)	51	Alcalá et al. [2011]
monk-2*	432	6 (0/ 6)	53	Alcalá et al. [2011]
phoneme	5404	5 (5/ 0)	71	Alcalá et al. [2011]
pima*	768	8 (8/ 0)	65	Alcalá et al. [2011]
ring*	7400	20 (20/ 0)	51	Alcalá et al. [2011]
sonar	208	60 (60/ 0)	53	Alcalá et al. [2011]
spambase	4597	57 (57/ 0)	61	Alcalá et al. [2011]
spectfheart	267	44 (0/44)	79	Alcalá et al. [2011]
titanic	2201	3 (3/ 0)	68	Alcalá et al. [2011]
twonorm*	7400	20 (20/ 0)	50	Alcalá et al. [2011]
wdbc	569	30 (30/ 0)	63	Alcalá et al. [2011]
wisconsin	683	9 (0/ 9)	65	Alcalá et al. [2011]
cod-rna	488565	8 (8/ 0)	66	Uzilov et al. [2006]

Table 6.1: **Details of the used datasets**. Between brackets the number of continuous and discrete features respectively. (*) Synthetic datasets.

If the average $F1$ measures on the 10 folds diverge less than 0.001, then the same rank is assigned to the classifiers.

In order to test the effect of the confidence measure in VSC, we performed additional runs on the same 22 datasets of a modified version of VSC with confidence identically forced to 1, namely $\mathcal{C}_p(\mathbf{x}) \equiv 1$. In order to investigate just the impact of the confidence measure, all the other parameters have been fixed to a default parameter of $k = 100$ hyperplanes and regularization factor $\lambda = 1$. The result is showed in Figure 6.3b. With the exception of 7 datasets VSC outperforms the modified VSC. In particular, the only dataset in which the VSC is outperformed by the modified version with statistical significance is monk-2, which is a synthetic dataset with discrete features that, as shown in Table 6.3, is particularly hard for VSC.

From the analysis of Table 6.4, VSC emerges as the classifier with the second-best overall ranking, preceded just by the SVM with RBF kernel. VSC has solid performances among the 22 datasets taken into consideration, outperforming (with statistical significance) the competitors 64 times, and being worse in just 21 cases. VSC obtains the best performance in just 2 datasets but in 13 datasets VSC is never

Classifier	Parameter	Values
VSC	λ	$0.001, 0.01, 0.1, 1, 10$
	k	$25, 50, 100, 250, 500$
SVM RBF	C	$0.001, 0.01, 0.1, 1, 10$
	γ	$1/n$
MLP	activation	$tanh, logistic, relu$
	# hidden nodes	$25, 50, 100, 250, 500$
AdaBoost	estimators	$25, 50, 100, 250, 500$
	learning rate	$0.001, 0.01, 0.1, 1, 10$
K-NN	K	$1, 2, 5, 10, 25, 50$
Random Forest	# estimators	$5, 10, 25, 50, 100$
SVM linear	C	$0.001, 0.01, 0.1, 1, 10$
SVM poly	pol. degree	$2, 3$
	C	$0.001, 0.01, 0.1, 1, 10$
ELM	# hidden nodes	$25, 50, 100, 250, 500$

Table 6.2: **Parameters' values used in the grid search**. Naïve Bayes and decision tree have no parameter to be cross-validated.

significantly worse of the competitors. Moreover, with the exception of 4 datasets (ionosphere, mammographic, monk-2, and sonar), VSC is always significantly better more times than the other way around.

Having identified the SVM with RBF kernel as the main competitor of VSC, we tested both methods on the more challenging dataset of cod-rna [Uzilov et al., 2006]. Given the size of the dataset, no parameter selection has been conducted and the default parameter of the models have been used. In particular the parameters for VSC are $k = 100$ and $\lambda = 10$, while SVM has been trained with $C = 1$ and $\gamma = 1/n$. Also in this experiment the results are very close, with the SVM with RBF kernel and VSC obtaining an $F1$ score of 0.970 and VSC 0.965 respectively.

6.2.3 Experiment 2

Experiment 2 has the goal of studying how the performance of the VSC changes with the variation of the parameters k and λ, with a special focus on their relationship. The values chosen for this investigation are $K = \{25, 50, 100, 250, 500\}$ and $\Lambda = \{0.1, 1, 10\}$. The results on each of the 22 datasets are normalized with respect to the corresponding performance of VSC with $\lambda = 1$ and $k = 100$ and are shown in Figure 6.4a. In order to assess the impact of sub-sampling pairs of samples with different classes we run a modified version of VSC. Instead of sampling pairs from the data, the modified VSC randomly selects points uniformly in the ranges of the features and then builds the hyper-planes. In this case the data are used only for computing the ranges, in particular, without using the information on their class.

	VSC	SVM RBF	MLP	AdaBoost	K-NN	Random Forest	SVM linear	SVM poly	ELM	Naïve Bayes	Decision Tree
appendicitis	0.926	0.915	0.899	0.926	0.922	0.912	0.907	**0.929**	0.808▾	0.901	0.867▾
banana	0.916	**0.918**	0.911	0.767▾	0.911▾	0.904▾	0.711▾	0.668▾	0.913	0.710▾	0.885▾
bands	0.705	0.741	0.686	0.720	0.730	0.697	**0.748**	0.722	0.700	0.080▾	0.547▾
bupa	0.735	0.770	0.764	0.761	0.696	**0.774**	0.752	0.723	0.716	0.511▾	0.705
coil2000	**0.969**	**0.969**	**0.969**	**0.969**	**0.969**	0.963▾	**0.969**	**0.969**	**0.969**	0.101▾	0.937▾
haberman	0.852	0.831	0.844	0.842	**0.854**	0.806▾	0.847	0.843	0.817▾	0.841▾	0.727▾
heart	0.858	0.870	**0.871**	0.861	0.867	0.848	0.864	0.859	0.827	0.858	0.768▾
hepatitis	0.894	0.876	0.884	0.918	0.889	0.913	0.860	**0.921**	0.840	0.637▾	0.842▾
ionosphere	0.927	**0.958**▵	0.947▵	0.948	0.924	0.936	0.907	0.926	0.898	0.910	0.903
magic	0.900	0.908▵	0.905▵	0.888▾	0.885▾	**0.911**▵	0.848▾	0.877▾	0.886▾	0.813▾	0.860▾
mammographic	0.817	0.847▵	0.818	**0.850**▵	0.812	0.806	0.803	0.803	0.814	0.813	0.778▾
monk-2	0.931	0.972▵	0.990▵	**1.000**▵	0.909	0.986▵	0.809▾	0.769▾	0.891▾	0.868	**1.000**▵
phoneme	0.916	0.923▵	0.896▾	0.876▾	0.933▵	**0.937**▵	0.843▾	0.859▾	0.908	0.821▾	0.911
pima	**0.834**	0.829	0.831	0.827	0.833	0.815	0.832	0.820▾	0.812▾	0.816	0.764▾
ring	0.974	0.979▵	0.973	0.970	0.842▾	0.949▾	0.786▾	0.968▾	0.915▾	**0.980**▵	0.882▾
sonar	0.544	0.649	0.671	**0.744**▵	0.718	0.722▵	0.665	0.715▵	0.540	0.564	0.634
spambase	0.945	0.946	0.949	0.952	0.917▾	**0.954**▵	0.940▾	0.931▾	0.806▾	0.829▾	0.914▾
spectfheart	0.873	0.880	0.869	0.856	**0.889**	0.880	0.874	0.877	0.856	0.744▾	0.848
titanic	0.862	0.860	0.853▾	0.848▾	0.858	0.863	0.847▾	0.852▾	**0.865**	0.842▾	0.862
twonorm	0.978	0.978	0.978	0.972▾	0.977	0.971▾	0.978	0.977	0.952▾	**0.979**	0.843▾
wdbc	0.971	**0.984**	0.982	0.981	0.974	0.970	0.982▵	0.968	0.908▾	0.944▾	0.949▾
wisconsin	0.976	0.975	0.975	0.968▾	0.973	0.973	**0.977**	0.967	0.968	0.971	0.958▾
Average	0.878	0.890	0.885	0.884	0.876	0.886	0.852	0.861	0.846	0.752	0.836
Median	0.908	0.912	0.897	0.882	0.889	0.911	0.848	0.868	0.860	0.825	0.861

Table 6.3: **Average results of Experiment 1**. Experimental F1 measure obtained on the analyzed datasets. The best results for each dataset are marked in bold. Results that are statistically better and worse with respect to our method are marked with ▵ and with ▾ respectively.

The runs were with the same set of parameters as above, and the results, which are normalized as before, are shown in Figure 6.4b.

In Figure 6.4a, the boxplot corresponding to VSC with $\lambda = 1$ and $k = 100$ is a single line due to the normalization. Variations of k from 100 corresponds for $\lambda = 1$ to very limited variations of the performance. There is some sensitivity of the results when λ is small: in particular with $k = 500$ and $\lambda = 0.1$ VSC shows the worst variation. At higher values of λ the effect of k appears to be mitigated, and when $\lambda = 10$ high values of k produces results that are even better of the ones obtained with the initial choice of the parameters.

In Figure 6.4b are reported the results obtained by changing the pairs selection step such that the pairs that drive the hyper-planes construction are not sub-sampled from the dataset, they are instead uniformly sampled in the range of the data. We

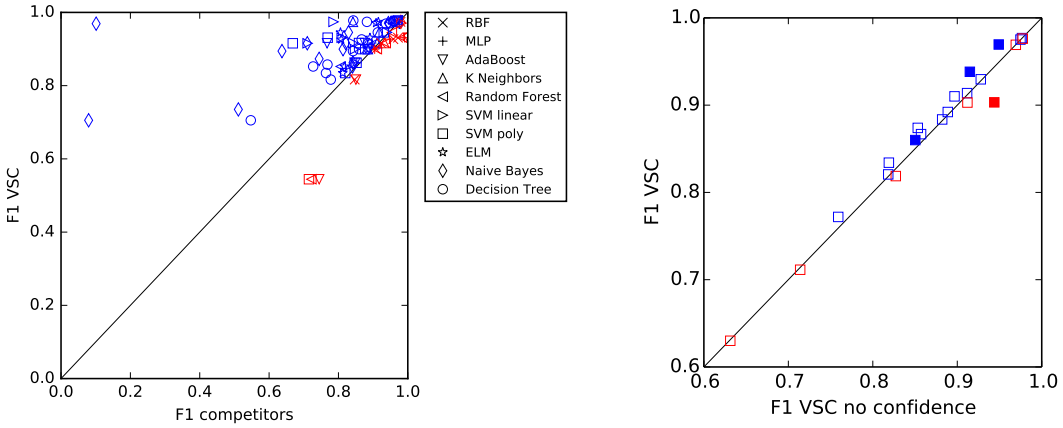
	Avg	appendicitis	banana	bands	bupa	coil2000	haberman	heart	hepatitis	ionosphere	magic	mammographic	monk-2	phoneme	pima	ring	sonar	spambase	spectfheart	titanic	twonorm	wdbc	wisconsin
VSC	4.14	2	2	6	6	1	2	7	4	5	4	4	6	4	1	3	10	5	6	3	2	6	2
SVM RBF	3.27	5	1	2	2	1	8	2	7	1	2	2	5	3	5	2	7	4	2	5	2	1	3
MLP	4.23	9	4	9	3	1	4	1	6	3	3	3	3	7	4	4	5	3	7	7	2	2	3
AdaBoost	4.59	2	8	5	4	1	6	5	2	2	5	1	1	8	6	5	1	2	8	9	8	4	8
K-NN	4.86	4	4	3	10	1	1	3	5	7	7	7	7	2	2	10	3	8	1	6	6	5	5
Random For.	5.18	6	6	8	1	9	10	9	3	4	1	8	4	1	9	7	2	1	2	2	9	7	5
SVM linear	6.00	7	9	1	5	1	3	4	8	9	10	9	10	10	3	11	6	6	5	10	2	2	1
SVM poly	6.32	1	11	4	7	1	5	6	1	6	8	9	11	9	7	6	4	7	4	8	6	8	10
ELM	7.86	11	3	7	8	1	9	10	10	11	6	5	8	6	10	8	11	11	8	1	10	11	8
Naïve Bayes	8.59	8	10	11	11	11	7	7	11	8	11	6	9	11	8	1	9	10	11	11	1	10	7
Dec. Tree	8.82	10	7	10	9	10	11	11	9	10	9	11	1	5	11	9	8	9	10	3	11	9	11

Table 6.4: **Ranks the classifiers in Experiment 1**. Rankings of the classifiers on the datasets ordered by average $F1$ score on the 10 folds. If the scores diverge less than 0.001, then the same rank is assigned to the classifiers.

can see that renouncing to the pairs' sub-sampling in the training data produces a slight, but systematic, decrease in the performances and an increase in the variability. Notice also the increased number of outliers in the low part of the plot. This modified version of VSC resemble the ELM approach, where the first layer of the multi-layer architecture is set to random noise. By comparing the results of VSC and ELM in Table 6.3 with those in Figure 6.4b, we can see that the confidence measure added to the separating hyper-planes plays a key role in the performance. Indeed the results obtained with this modifies version of VSC are still comparable with the original, while ELM's results are not even close. With high values of λ and k the performances increase suggesting that "extreme" version of VSC could be worth exploring; we may pay, however, the price of choosing the parameters within a context of more variable performance.

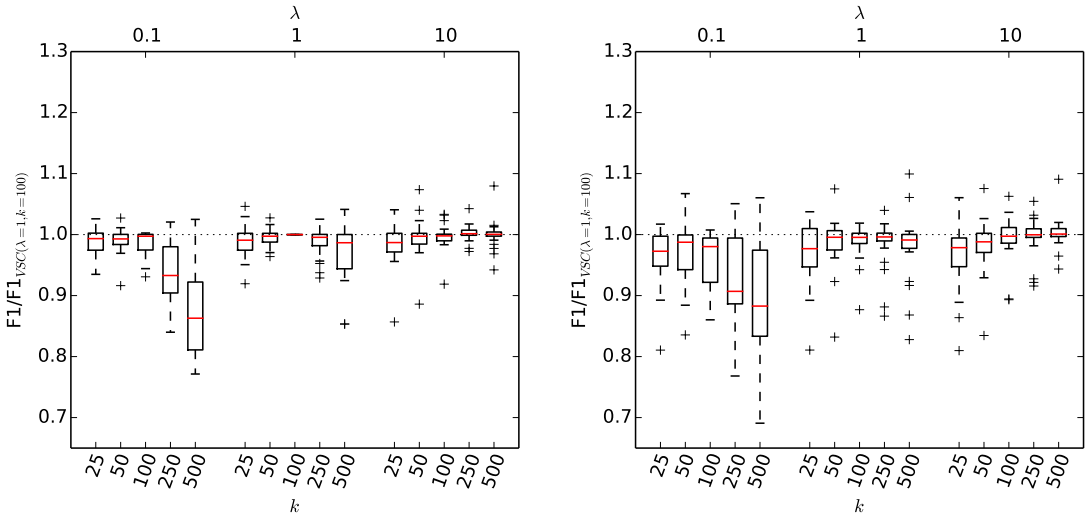
6.2.4 Discussion

Despite the simplicity of the method, VSC obtains very competitive results on the pool of analyzed datasets, which vary for size, number of features, and origin of the data. Under some aspects VSC may be similar to the extreme learning machines; however the in-sample pair selection, although naive, shows relevant advantages over the random weights initialization. Moreover, with Experiment 1, the effectiveness of the confidence measure can be appreciated. This highlights the importance of



(a) (b)

Figure 6.3: **Results of Experiment 1**. The left plot shows the statistically significant data of Table 6.3. The datasets in which VSC outperforms the competitor are marked in blue. The red marks, on the contrary, the datasets for which the competitor achieves better results. The symbol identifies the competitor. The right plot shows the impact of the confidence measure. The datasets in which VSC has $F1$ higher than the VSC with modified confidence are marked in blue. The red marks, on the contrary, are the datasets for which the modified confidence is better. The filled marks represent results statistically significant.



(a) VSC (b) VSC with uniformly sampled pairs

Figure 6.4: **Results of Experiment 2**. Box plots of the performance of VSC on the 22 datasets by varying the regularization parameter λ and the number of hyper-planes k. For each dataset, the $F1$ values are normalized with the $F1$ score of the VSC($\lambda = 1, k = 100$) on the same dataset.

limiting locally the influence of features that are non-global by construction.

As expected, it is difficult to find a clear winner in the challenge of general purpose classifier, because there is no general purpose classification task. Each problem has its own peculiarities and therefore certain classifiers fit better the data then others. The SVM with RBF kernel, which is also a local method, seems to have an edge on the VSC and the other analyzed competitors, corroborating the hypothesis that locality plays an important role in the performance in the general case. On the other hand SVM models have no fixed size and can reach high size, especially with RBF kernel, if the ideal decision boundary is complex, leading to slow prediction times. VSC on this aspects offers a fixed size model, whose dimension depends on the parameters imposed by the user.

The current version of VSC involves the inversion of the $N \times N$ matrix, whose size depends on the size of the problem. With the increasing size of the current datasets this can quickly become a problem, but this learning step can be replaced by a $L2$ regularized regression. This change would allow us to tackle bigger problems and extend the model to multi-label classification by learning simultaneously the weights for each label.

SVM with RBF kernel obtains better average ranking with respect to VSC, but both methods have very comparable overall performances. Indeed, despite having a relevant difference in the average $F1$ value, the median on the same metric is very close. This difference is driven by the bad averaged results obtained by VSC on the sonar dataset, which are however not significantly worse than the one obtained by SVM RBF.

From the experiments, we can therefore appreciate the very good adaptability of the VSC, which obtains good and competitive results on most of the datasets.

6.3 Conclusion

We have presented VSC, a "concept" classifier designed to test the idea that features which are based on the notion of locality can be effectively incorporated in a multi-layer perceptron architecture. Max-margin hyper-planes are defined on a subset of the pairs of the samples with different classes and a confidence measure characterized in terms of Chebichev inequality is defined. Results of runs with different values of the regularization parameter and number of pairs show the effectiveness of the approach in terms of quality of the results. The competitors are overperformed with the exception of SVM with RBF kernel, confirming the theoretical assumptions. The effectiveness of the confidence measure is also empirically verified.

The motivation of the work was to investigate the possibility that locality can produce features of high quality to be included in more complex architectures. Further studies will be important to evaluate the scalability in terms of size and dimensionality of the datasets. The results, however, are very encouraging and future work will be devoted to the identification of pair selection strategies that can maximize the effectiveness of the approach and the integration with deep learning. In particular we believe that VSC can be effectively be incorporated in such frameworks as an initial layer where no explicit order, spatial or temporally, is present in the input features.

Chapter 7

Gene Regulatory Network Expansion by Stratified variable Subsetting and Ranking Aggregation

Biological processes are often regulated at the transcriptional level, via gene regulatory networks (GRN) [Hasty et al., 2001a] comprising regulatory genes, known as transcription factors, and regulated genes. So far, in most cases, only a small fraction of the genes involved in a GRN are known, and usually collected in Local Gene Networks (LGNs) that are subsets of genes known to be causally connected. These genes are discovered through ad hoc experiments testing *in vivo* the hypothesis that a given gene participates in a specific biological process. Thus, there is a urgent need to fill this knowledge gap in order to have a better picture of most biological processes and translate biology into medical, biotechnological, and agricultural applications. A major contribution in this field has come from the new sequencing technologies, which dramatically increased the sequence output capacity and equally decreased the cost per sequenced base[1].

In this chapter, we present NES^{2}RA [Asnicar, Masera et al., 2016] (Network Expansion by Stratified Subsetting and Ranking Aggregation) a method for finding candidate genes for expanding GRN. NES^{2}RA generalizes our previous proposal NESRA [Asnicar et al., 2015a] with the main difference being that it is now possible to model with a probability the presence of the genes of the network to be expanded in the subsets, namely the sampling is stratified. Both NESRA and

[1]Wetterstrand KA. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP) Available at: `www.genome.gov/sequencingcosts`. Accessed January 2019.

NES^{2}RA are based on the PC-algorithm that we run on our gene@home project [Asnicar et al., 2015b], developed on the BOINC platform [Anderson, 2004a]. We evaluate NES^{2}RA on real data on model organisms (*Arabidopsis thaliana* and *Escherichia coli*), and compare it against NESRA and ARACNE.

7.1 Related Work

The PC-algorithm [Spirtes and Glymour, 1991a] discovers causal relationships among variables by systematically testing the conditional independence of two nodes given subsets of their adjacent nodes. The computational cost of the PC-algorithm is exponential in the number of nodes, but it behaves reasonably in the case of sparse, scale-free networks [Maathuis et al., 2010], like biological networks [Barabási, 2003]. The PC-algorithm has been comprehensively presented and evaluated by Kalisch and Bühlmann [2007] and applied to gene network reconstruction [Maathuis et al., 2010]. The PC-algorithm has also been successfully employed in other network inference approaches [Tan et al., 2008, 2011; Wang et al., 2010; Zhang et al., 2012]. The results of the PC-algorithm depend on the order of the nodes in the input file; the order-independent version is called PC-stable [Colombo and Maathuis, 2012].

At the time of writing, other popular methods for network inference (NI) are the BAyesian Network inference with Java Objects (BANJO [Hartemink, 2005]), Network Inference by Reverse-engineering (NIR [Gardner et al., 2003]), and the Algorithm for the Reconstruction of Accurate Cellular NEtworks (ARACNE [Margolin et al., 2006a,b]). The last one has been empirically proved to be the state-of-the-art NI method [Allen et al., 2012]. The available reconstruction methods applied to genome wide data are computationally demanding due to the huge size of the solution space [Kalisch and Bühlmann, 2007]. Moreover, as we will see here, these methods are not accurate enough in order to use the results to perform a network expansion [Marbach et al., 2012].

7.2 NES^{2}RA

NES^{2}RA is an improved and generalized version of NESRA [Asnicar et al., 2015a], which considers as input data the LGN and the probability of each gene of the LGN to be included in the subsets, the set of parameters to be used, and the gene expression levels for the considered organisms. The inclusion of the LGN in the subsetting step improves the quality of the results (as we will see in Section 7.4), because the composition of each subset is influenced by the LGN nodes added.

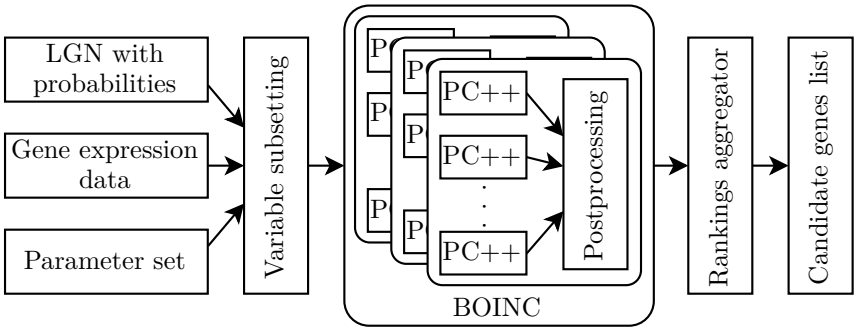


Figure 7.1: NES²RA workflow.

The vector of probabilities is a representation of the knowledge of the user (e.g., a biologist) about the presence of specific genes in the network. Probability 1 means that the gene is definitely in the network, whereas probability 0 means that there is no knowledge about the presence of the gene in the network. Depending on the probabilities the genes will be included in the data for the run of the PC-algorithm. If all the probabilities are zero NES²RA coincides with its previous version NESRA [Asnicar et al., 2015a]. The high-level structure of NES²RA is described in Figure 7.1 and Algorithm 3.

Algorithm 3: Pseudo-code of NES²RA.

Data: $\mathcal{S}$ set of candidate transcripts, $\mathcal{S}_{\text{LGN}}$ set of LGN transcripts, E expression data, a vector $\Pi = (\pi_1, \ldots, \pi_l, \ldots, \pi_{|\mathcal{S}_{\text{LGN}}|})$ of the probabilities of each $g_l \in \mathcal{S}_{\text{LGN}}$ to be in the LGN.

Input: I set of number of iterations, D set of the subset dimensions, A set of the significance levels α, k maximum length of the candidate gene lists

Result: ordered list of candidate transcripts

```
L ← ∅                                                    // L set of ordered lists
foreach α ∈ A do
    foreach d ∈ D do
        foreach i ∈ I do
            L ← L∪ RP(S, S_LGN, E, Π, i, d, α)           // call Algorithm 4
L ← top(L, k)                          // cut each list in L to the first k elements
return Ranking_aggregation(L)
```

The ranking procedure (RP) presented in Algorithm 4 is composed of three main steps, which respectively: (1) create the subsets, (2) execute several calls of the `skeleton` procedure of the PC-algorithm (Algorithm 1), and finally, (3) compute the transcripts frequency that defines the order of each ranking.

The RP takes as parameters the number of iterations i, the dimension of the subset d, the significance level α for the `skeleton`, and the probability vector Π for the

genes of the LGN. The output of the `skeleton` depends on the order of the inputs. Hence iterating i times its application mitigates this effect, reaching a more stable solution. The RP returns a ranked list of k elements that is partially computed on the gene@home BOINC project, while the frequencies calculation and the rankings aggregation are executed off-line. The novelty of NES^{2}RA is in Step 1 of the ranking procedure (Algorithm 4), where we take into consideration the knowledge of the LGN with its associated probabilities.

NES^{2}RA systematically and iteratively applies subsetting on the whole data set in order to randomly select genes that will be processed with the `skeleton` procedure. The subsetting is controlled by the iterations i and subset size d parameters. In NES^{2}RA the subsetting is stratified, and the genes of the LGN can have an increased probability of being in the subsets. In fact, for a given pair of subset size d and iteration i, a first selection, controlled by the probability vector Π, specifies which genes of the LGN are present in the subsets. The genes of the LGN that are not selected in the first selection are considered, together with the other candidate genes, for a second selection with uniform probability. Finally, a third selection restricted to the genes not already present in the current subset, permits to complete the last subset whenever not yet of the desired dimension d.

The overall effect of the probability vector Π in Algorithm 4 is such that the probability of a gene g to be present in the h-th subset of genes $T_{h,i}$ at the i-th iteration is given by:

$$P(g \in T_{h,i}) = \begin{cases} \pi_l + (1-\pi_l)\frac{d-\sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|}\pi_m}{|\mathcal{S}|-\sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|}\pi_m}, & \text{if } g = g_l \in \mathcal{S}_{\text{LGN}} \\ \frac{d-\sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|}\pi_m}{|\mathcal{S}|-\sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|}\pi_m}, & \text{if } g \in \mathcal{S}\backslash\mathcal{S}_{\text{LGN}} \end{cases} \tag{7.1}$$

where $\mathcal{S}$ is the set of candidate genes, $\mathcal{S}_{\text{LGN}}$ is the set of genes of the LGN, d with $|\mathcal{S}_{\text{LGN}}| < d \leqslant |\mathcal{S}|$ is the subset dimension, π_l is the l-th component of Π corresponding to the probability of the g_l gene of the LGN to be selected in the first selection, and $\sum_{m=1}^{|\text{LGN}|}\pi_m$ is the expected number of LGN genes selected after the first selection. The last subset of each iteration is a special case: the third selection can intervene for its completion and the formula above does not hold anymore in a rigorous way. The exact correction of the fractional term of Equation 7.1 requires a more detailed analysis that is beyond the current aim to illustrate the effect of Π.

The probability of a gene $g_l \in \mathcal{S}_{\text{LGN}}$ of being in a subset is the convex combination of the probability of being in the subset of a gene that is not in the LGN, controlled by the parameter π_l. For a gene g_l of the LGN, if $\pi_l = 1$ then $P(g_l \in T_{h,i})|_{\pi_l=1} = 1$

Algorithm 4: NES^{2}RA ranking procedure (RP).

Data: $\mathcal{S}$ set of candidate transcripts, $\mathcal{S}_{\text{LGN}}$ set of LGN transcripts, E expression data, $\Pi = (\pi_1, \ldots, \pi_l, \ldots, \pi_{|\mathcal{S}_{\text{LGN}}|})$ probability vector for each $g_l \in \mathcal{S}_{\text{LGN}}$

Input: $i \geqslant 1$ number of iterations, d subset dimension, α significance level

Result: l, ordered list of candidate transcripts

foreach $g \in \mathcal{S}$ **do**
  $p_g \leftarrow 0$, $f_g \leftarrow 0$

`/* Step 1: Subsets creation */`

foreach j, $1 \leqslant j \leqslant i$ **do**
  $h \leftarrow 1$, $\mathcal{S}_{temp} \leftarrow \mathcal{S} \backslash \mathcal{S}_{\text{LGN}}$
  while $\mathcal{S}_{temp} \neq \varnothing$ **do**
    foreach $g_l \in \mathcal{S}_{LGN}$ **do**
      with probability π_l: $T_{h,j} \leftarrow T_{h,j} \cup \{g_l\}$
    $\mathcal{S}_{temp} \leftarrow \mathcal{S}_{temp} \cup (\mathcal{S}_{\text{LGN}} \backslash T_{h,j})$
    while $|T_{h,j}| < d$ **do**
      uniformly random select $g \in \mathcal{S}_{temp}$
      $T_{h,j} \leftarrow T_{h,j} \cup \{g\}$
      $\mathcal{S}_{temp} \leftarrow \mathcal{S}_{temp} \backslash \{g\}$
      $p_g \leftarrow p_g + 1$
    if $\mathcal{S}_{temp} = \varnothing$ *and* $|T_{h,j}| < d$ **then**
      while $|T_{h,j}| < t$ **do**
        uniformly random select $g \in \mathcal{S} \backslash T_{h,j}$
        $T_{h,j} \leftarrow T_{h,j} \cup \{g\}$
        $p_g \leftarrow p_g + 1$
    $h \leftarrow h + 1$
  $N_j \leftarrow h$

`/* Step 2: skeleton */`

foreach j, $1 \leqslant j \leqslant i$ **do**
  foreach h, $1 \leqslant h \leqslant N_j$ **do**
    $R_{h,j} \leftarrow skeleton(T_{h,j}, E, \alpha)$ `// Algorithm 1`

`/* Step 3: Frequency computations */`

foreach $g \in \mathcal{S}$ **do**
  foreach $q \in \mathcal{S}_{LGN}$ **do**
    foreach j, $1 \leqslant j \leqslant i$ **do**
      foreach h, $1 \leqslant h \leqslant N_j$ **do**
        if $g \in Adj_{R_{h,j}}(q)$ **then**
          $l \leftarrow l \cup \{g\}$ `// adjacent nodes of` q `in` $R_{h,j}$
          $f_g \leftarrow f_g + 1$
  $f'_g \leftarrow f_g / (p_g * |\mathcal{S}_{\text{LGN}}|)$ `// Normalized frequency`

return l ordered w.r.t. f'_g

and the l-th gene is present in all the subsets. Alternatively, if $\pi_l = 0$ then

$$P(g_l \in T_{h,i})|_{\pi_l=0} = \frac{d - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m}{|\mathcal{S}| - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m},$$

namely the same probability of a gene that is not in the LGN. The probability of Equation 7.1 can be written as:

$$\begin{aligned} P(g_l \in T_{h,i}) &= \pi_l + (1 - \pi_l) P(g_l \in T_{h,i})|_{\pi_l=0}) = \\ &= \pi_l (1 - P(g_l \in T_{h,i})|_{\pi_l=0}) + P(g_l \in T_{h,i})|_{\pi_l=0}. \end{aligned}$$

Setting π_l permits to modulate the probability of the presence of each gene g_l in the subsets. In the case $\pi_l = 0$ for all the genes of the LGN the probability becomes $P(g \in T_{h,i}) = d / |\mathcal{S}|$ for each gene and NES^{2}RA corresponds to NESRA where the probability of presence of the genes of the LGN in the subsets is the same of the other genes.

The number of executions of the `skeleton` procedures (Algorithm 1) that are generated in NES^{2}RA by the parameter d are:

$$\begin{aligned} O_{\text{PC}} = &\mathbb{E}\left(\#\text{runs at iteration } j\right) = \\ &\left\lceil \frac{|\mathcal{S}| - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m}{d - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m} \right\rceil \times P\left(\bigcap_h T_{h,j} \backslash S_{\text{LGN}} \neq \varnothing \right) + \\ + &\frac{|\mathcal{S}| - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m}{d - \sum_{m=1}^{|\mathcal{S}_{\text{LGN}}|} \pi_m} \times P\left(\bigcap_h T_{h,j} \backslash S_{\text{LGN}} = \varnothing \right) \end{aligned} \tag{7.2}$$

`skeleton` executions. Note that this formula is independent from the specific iteration j.

For each pair d, i, NES^{2}RA executes a number of `skeleton` procedures, given by Equation 7.2. The results of these executions are combined in a single list of genes, ranked by their number of appearances. The `skeleton` procedure produces a graph, providing the causal relationships between nodes, but not their directions. The PC-algorithm estimates the orientation of the edges after the `skeleton` procedure, and the orientation steps do not remove or add any edge. The execution of the `skeleton` procedure indeed produces the most important information that we want to exploit in NES^{2}RA: the existence of an edge between two nodes. Such edge, in fact, represents the existence of a causal relationship between the two nodes, even though we do not know its direction.

Finally, NES^{2}RA produces the list of candidate genes by applying different ranking aggregation methods on the ranked lists. These methods comprise a base technique, i.e. the *number of appearances*, and more sophisticated methods, namely

Borda Count [Borda, 1781] and MC4 heuristic [Lin, 2010]. The method we considered as baseline is the *number of appearances*, that counts how many rankings have a certain gene, i.e. the more a gene is present, the higher its position in the aggregated rank is. The Borda Count method consists in constructing a matrix $A(m, n)$ with m rows and n columns, corresponding to the genes and the rankings, respectively. The element a_{ij} is the rank of gene i on ranking j, and a statistic for each gene is computed on the rows of the matrix. The two statistics that we used are the mean (BC-mean) and the minimum (BC-min) of the elements. MC4 heuristic is an aggregator based on Markov chains. It consists in computing the transition matrix of the pairwise comparison of all the rankings for each gene. A step in the Markov chain assigns a higher probability to a gene q if $rank(q) < rank(p)$ for a majority of the lists that ranked both p and q [Dwork et al., 2001b]. So that, the steady state of the chain assigns higher probability to the genes with higher ranks. To avoid a non unique stationary distribution, MC4 has as parameter the significance level α_{MC4} for which we considered two values: 0.05 and 0.01.

Both NESRA and NES²RA exploit the gene@home project for computing the first two steps of ranking procedure.

7.3 NES²RA on the gene@home BOINC project

Nowadays, the literature reports several successful research projects that exploit the power of volunteer grid computing in order to achieve their goals [Anderson et al., 2002; Das et al., 2007]. BOINC, for instance, is an open-source framework particularly convenient for projects that require a large amount of computation, but do not have access to suitable resources. NES²RA requires O_{PC} executions (see Equation 7.2 in Appendix) that can be easily parallelized. Therefore, we decided to exploit the gene@home [Asnicar et al., 2015b] BOINC project, hosted by the TN-Grid platform[2], to distribute the computation of the `skeleton` procedure (Step 2 of Algorithm 4).

Every BOINC project is composed by several components, such as the work generator and the validator. The aim of the former is to create the workunits that will be then distributed to the volunteers, while the latter validates the results of the finished workunits. The validator performs a bitwise comparison of two workunits that have been computed by two different machines. This step is required to ensure the consistency of the results. We designed and developed our custom work generator using the Python language and two C++ wrappers to interface the work generator

[2] `http://gene.disi.unitn.it/test/`

Algorithm 5: Partial correlations function.

Input: i, j analyzed variables, $\mathbf{k}$ separation set

Result: $\rho_{i,j|\mathbf{k}}$

$l \leftarrow |\mathbf{k}|$; $M \leftarrow [i, j, k_1, \ldots, k_l]$; $d \leftarrow |M|$

Initialize the $d \times d$ matrix $\boldsymbol{\rho}$ s.t. $\boldsymbol{\rho}[u][v] \leftarrow correlation(M[u], M[v])$

for $n = 1$ **to** l **do**
  for $u = 0$ **to** $l - n$ **do**
    for $v = u + 1$ **to** $d - n$ **do**
      $$\boldsymbol{\rho}[u][v] \leftarrow \frac{\boldsymbol{\rho}[u][v] - \boldsymbol{\rho}[u][d-n] \cdot \boldsymbol{\rho}[v][d-n]}{\sqrt{(1 - \boldsymbol{\rho}^2[u][d-n])(1 - \boldsymbol{\rho}^2[v][d-n])}}$$

return $\boldsymbol{\rho}[0][1]$

and the Python scripts with the BOINC framework. The subsets creation (Step 1 of Algorithm 4) is implemented in the work generator. Each workunit corresponds to many runs of the `skeleton` procedure (Step 2 of Algorithm 4), and the duration is estimated in order to inform the volunteers about execution times, a fundamental aspect in any BOINC project.

The core of gene@home is the client application. To date, it is available for both 32 and 64 bit architectures, for three operating systems: Linux, Windows, and Mac OS X. The original implementation of the `skeleton` procedure, present in the `pcalg` R package [Kalisch et al., 2012; Hauser and Bühlmann, 2012], is not suitable for high-performance volunteer-computing projects due to both its software requirements, i.e. the R interpreter and numerous R packages, and its low speed and high memory consumption. To the best of our knowledge, no alternative open-source implementation of the `skeleton` procedure is available, thus we implemented our C++ version, namely PC++[3]. The PC++ implementation makes use of efficient data structures and avoids the storing of the separation sets, which are not needed in NES^{2}RA, to reduce the memory usage. Moreover, the original recursive computation of the partial correlation (Proposition 2 [Kalisch and Bühlmann, 2007]) has been replaced with an iterative version based on dynamic programming technique [Cormen et al., 2001], shown in Algorithm 5. This solution reduces the complexity of the computation from $O(3^l)$ to $O(l^3)$, where l corresponds to the size of the separation sets. These optimizations and the possibility to natively integrate PC++ into the BOINC client application drastically decreased the computational time and memory usage.

In Table 7.1 we report the detailed comparison between the PC++ and the `skeleton` procedure of the `pcalg` package, conducted on the *Escherichia coli* data

[3]Publicly available at `https://bitbucket.org/francesco-asnicar/pc-boinc`

Table 7.1: **Performance comparison between `skeleton` and PC++.** Comparison between `skeleton` and PC++ in terms of running time and memory usage on the *E. coli* data set using different subset sizes.

		$d = 50$	$d = 100$	$d = 200$	$d = 500$	$d = 4065$
`skeleton`	time (s)	86.13	924.96	15470.62	169869.69	timed out
	RAM (MB)	95.23	104.83	145.88	200.11	
PC++	time (s)	0.36	3.82	69.59	716.88	94525.63
	RAM (MB)	5.85	7.85	13.73	35.51	506.75

set. From the results in Table 7.1 we can appreciate that the PC++ implementation gained a speed-up of more than 200 and decreased in the memory usage of an order of magnitude[4]. For a fair comparison, we modified the original `skeleton` procedure to avoid storing the separation sets. No data is available for the `skeleton` on subset size $d = 4065$, because it reached the two-weeks time limit we imposed. We estimated the execution time of `skeleton` for $d = 4065$ via a regression analysis on the other subset sizes, in more than 200 days.

The post-processing phase in NES^{2}RA consists in a two-step pipeline. The workunits results are firstly combined on the volunteers' local machines by the client applications, in order to reduce the size of the data to be uploaded. Lastly, the partially aggregated results are aggregated on the server.

7.4 Evaluation

In this section we report the results of three different experiments that assess the performance of NES^{2}RA. The aim of the first experiment is to biologically evaluate the results of NES^{2}RA in comparison with NESRA, ARACNE, and the PC-algorithm. The second experiment, instead, has the goal to analyze the impact of the probability vector Π on the final expansion list. Finally, we analyze the computational aspects of NES^{2}RA executed by the gene@home project.

The data set considered in the first experiment is composed by gene expression hybridizations for the *Arabidopsis thaliana* plant model organism, namely microarray expression values publicly available in the Plex database [Dash et al., 2012]. The data set comprises 393 hybridization experiments of the GeneChip Arabidopsis ATH1 Genome Array that encompass 22810 probe sets. The LGN that we used for *A. thaliana* is the Flower Organ Specification Gene Regulatory Network (FOS). The

[4]The experiments were executed on an Intel® Core™ i5-4590 processor at 3.30 GHz, with 8 GB of RAM running a 64 bit Linux with the 3.19.0-32 kernel.

FOS gene network has been characterized and validated *in vivo* by the use of specific mutants [Espinosa-Soto et al., 2004], and it is composed by 15 genes connected by 54 causal relationships [Sanchez-Corrales et al., 2010]. In this case the presence of the genes in the network is certain and so the vector Π of NES^{2}RA has all its components set to 1.

The second experiment has been conducted on the bacterial model organism *Escherichia coli*. The data set contains 4065 genes for 2470 hybridizations and it is publicly available in the COLOMBOS [Meysman et al., 2014] database. The LGN considered is a transcription factor network called gadW collected from the EcoCyc [Keseler et al., 2013] database. The gadW LGN is composed of 13 nodes connected by 12 edges, and it is involved in the acid resistance system of *E. coli*. In this experiment we compared the results of NES^{2}RA using two probability vectors: Π_{H} and Π_{L}. The former has just the probability of the hub node, the gadW gene, set to 1 while the probabilities of all the other genes are set to 0. The latter instead, has all the entries set to 1. The same experimental setup has been used to assess the computational aspects of NES^{2}RA.

We assessed the biological validity of the results by performing a bibliographic research, classifying genes in four different classes, as follows:

Class 1 collects genes reported to be biologically or functionally related to the genes in the LGN;

Class 2 contains genes not reported to be directly related with the input network, but reported to be related to genes of Class 1;

Class 3 comprises all the genes described in the literature that were reported not to be related with the input network or with the genes of Class 1;

Class 4 are genes for which no description was found in the available literature.

When we found a gene belonging to Class 1 or Class 2 we considered it to be a true positive, while a gene falling in Class 3 or Class 4 was considered a false positive. The precision of the genes in the candidate output list is the ratio between the number of true positives and the sum of true positives and false positives. Other measures, like F1 and Recall, can not be computed on real organisms data sets because no complete ground truth is available. We can only exploit the manually curated classification that we have performed for the resulting genes provided by the methods considered.

Table 7.2 reports an example of the candidate genes list for the FOS LGN of *A. thaliana*, produced by NES^{2}RA. The list has a precision value of 80% and it

Table 7.2: **Candidate genes list of the FOS LGN of *A. thaliana* produced by NES^{2}RA**.

Rank	AffyID	Gene	Annotation	Class
1	259089_at	AT3G04960	Similar to unknown protein	Class 1 [Lee et al., 2005]
2	248496_at	AT5G50790	ATSWEET10	Class 3 [Chen et al., 2012]
3	265441_at	AT2G20870	Cell wall protein precursor	Class 1 [Cai et al., 2007]
4	255644_at	AT4G00870	Basic helix-loop-helix (bHLH) family protein	Class 2 [Hu et al., 2003]
5	261375_at	AT1G53160	SPL4 (SQUAMOSA PROMOTER BINDING PROTEIN-LIKE 4)	Class 1 [Lal et al., 2011]
6	249939_at	AT5G22430	Similar to unknown protein	Class 1 [Zik and Irish, 2003]
7	255448_at	AT4G02810	FAF1 (FANTASTIC FOUR 1)	Class 1 [Wahl et al., 2010]
8	245842_at	AT1G58430	RXF26	Class 1 [Shi et al., 2011]
9	256259_at	AT3G12460	DEDDy 3'-5' exonuclease domain-containing protein	Class 4
10	260355_at	AT1G69180	CRC (CRABS CLAW)	Class 1 [Lee et al., 2005]

Table 7.3: **NES^{2}RA precision performance using different aggregation methods**. Results for NESRA BC-mean and ARACNE have been computed in [Asnicar et al., 2015a]

Method	k=5	k=10	k=20	k=55
NES^{2}RA N of appearances	0.57	0.57	0.57	0.51
NES^{2}RA BC-mean	0.80	0.90	0.75	0.51
NES^{2}RA BC-min	0.80	0.88	0.80	0.51
NES^{2}RA MC4 ($\alpha_{\text{MC4}} = 0.05$)	0.80	0.90	0.75	0.51
NES^{2}RA MC4 ($\alpha_{\text{MC4}} = 0.01$)	0.80	0.90	0.75	0.51
NESRA BC-mean	0.90 ± 0.098	0.65 ± 0.049	0.63 ± 0.038	0.43 ± 0.016
ARACNE	0.20	0.30	0.35	0.45

has been obtained aggregating 60 different ranked lists using the MC4 method with the parameter $\alpha_{\text{MC4}} = 0.01$. The values considered for this run are: $I = \{100, 250, 500, 1000, 1500, 2000\}$, $D = \{50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000\}$, and $A = \{0.05\}$. The gene AT3G12460, ranked in position 9, is considered as a false positive. However, only a biological wet-lab validation could rule out if it is actually involved in the FOS LGN.

Table 7.3 shows the precision values of NES^{2}RA using the same set of experiments presented in Table IV by Asnicar et al. [2015a]. Using the very same set of parameters for NES^{2}RA we aggregated only the six rankings that have the same values as in NESRA [Asnicar et al., 2015a]: iterations $I = \{100, 500, 2000\}$, subset dimensions $D = \{1000, 2000\}$ and $A = \{0.05\}$. It is possible to see that the best

performances are obtained with list lengths of $k = 5$, 10, and 20. In particular, if we compare these results of NES^{2}RA with the results of NESRA (Table 7.3) we can see that NES^{2}RA has better precision when considering longer lists ($k = 55$). Moreover, NES^{2}RA prove to have better precision when compared with ARACNE. NES^{2}RA can be also compared with the PC-algorithm and the PC-stable using their precision values reported in [Asnicar et al., 2015a], which are 0.39 ± 0.03 and 0.43, respectively. It can be seen that NES^{2}RA outperforms both the PC-algorithm and the PC-stable in the task of finding candidates for GNE.

Although, the PC-algorithm, PC-stable, and ARACNE are used for a NI task, here we compared them to a GNE approach. In order to do such a comparison we obtained a pseudo-expansion list by running each method on the whole data set of *A. thaliana*. Their results were then filtered w.r.t. the FOS LGN, selecting only the edges connected with at least a node in the LGN. Then, when possible, the results were sorted according to the p-value provided by the methods.

Table 7.4 reports the results obtained with different probability vectors to expand the gadW LGN of *E. coli* with parameters $A = \{0.01, 0.05\}$, $D = \{100, 200\}$, and $I = \{80, 100, 500, 1000, 1500, 2000\}$. Here NESRA can also be interpreted as a special case of NES^{2}RA where the probability vector Π is set to 0 for each gene in the LGN. Additionally, we analysed the results of ARACNE on the same data set for the same LGN, using the same analysis applied in Asnicar et al. [2015a]. Interestingly, both NESRA and NES^{2}RA produce a higher quality expansion list, in particular when considering only the first five genes in the output lists. We manually curated the classification of the genes found, and report in bold the genes are classified as either Class 1 or Class 2, and in italics the ones belonging to either Class 3 or Class 4. It can be noticed that the injection of prior knowledge in form of presence probability in the subsets, positively impacts the final quality of the expansion list. Indeed, as we can see from Table 7.4, the more prior-knowledge is used, the more precise the expansion lists are, reaching up to 90% precision.

Table 7.5, Table 7.6, and Table 7.7 report the BOINC statistics regarding the experiments conducted on the *E. coli* data set. The FLOPs for the workunits have been computed on the basis of the computation time and the FLOPS of the hosts machines, determined by BOINC running a Whetstone benchmark [Curnow and Wichmann, 1976]. In the gene@home project, each workunit is computed twice in order to be able to validate the results, as explained in Section 7.3. Thus, the actual FLOPs required for a NES^{2}RA experiment would be half of the ones reported in Table 7.5. By comparing the average throughput of gene@home and the FLOPs required for executing NES^{2}RA we see that, by exploiting the volunteer computa-

Table 7.4: **Ranked lists of the gadW LGN of *E. coli***. True positives are shown in bold, false positives in italics. Below the gene name are reported the references to the bibliography.

	1	2	3	4	5
NESRA	**hdeD** [Masuda and Church, 2003]	**ybaS** [Lu et al., 2013]	**yhiM** [Tucker et al., 2002]	**ybaT** [Tucker et al., 2002]	**cfa** [Tucker et al., 2002]
$\text{NES}^2\text{RA}\ \Pi_{\text{H}}$	**hdeD** [Masuda and Church, 2003]	**yhiM** [Tucker et al., 2002]	**ybaS** [Lu et al., 2013]	**ybaT** [Tucker et al., 2002]	**cfa** [Tucker et al., 2002]
$\text{NES}^2\text{RA}\ \Pi_{\text{L}}$	**hdeD** [Masuda and Church, 2003]	**yhiM** [Tucker et al., 2002]	**cbpA** [Tucker et al., 2002]	**ybaT** [Tucker et al., 2002]	**appC** [Hayes et al., 2006]
ARACNE	**hdeD** [Masuda and Church, 2003]	**ybaS** [Lu et al., 2013]	*dps* [Dukan and Touati, 1996; Gundlach and Winter, 2014]	*aidB* [Volkert and Nguyen, 1984]	*sra* [Akira, 1986; Izutsu et al., 2001]
	6	7	8	9	10
NESRA	**cbpM** [Tucker et al., 2002]	*aidB* [Volkert and Nguyen, 1984]	*kch* [Milkman, 1994]	**cbpA** [Tucker et al., 2002]	*yjfbQ* [Kim et al., 2010]
$\text{NES}^2\text{RA}\ \Pi_{\text{H}}$	**cbpA** [Tucker et al., 2002]	*kch* [Milkman, 1994]	**ycaC** [Tucker et al., 2002]	**cbpM**, *cueR* [Tucker et al., 2002; Stoyanov et al., 2001]	-
$\text{NES}^2\text{RA}\ \Pi_{\text{L}}$	**ybaS** [Lu et al., 2013]	*aidB* [Volkert and Nguyen, 1984]	**hyaF** [Hayes et al., 2006]	**hyaA** [Hayes et al., 2006]	**hyaC** [Hayes et al., 2006]
ARACNE	**cbpA** [Tucker et al., 2002]	**ybaT** [Tucker et al., 2002]	*elaB* [Yoshida et al., 2012]	*yegP* [Kumar et al., 2016]	*talA* [Weber et al., 2006]

Table 7.5: Cumulative BOINC statistics for the *E. coli* experiments.

	#RP runs	Workunits	Hosts	Hosts GigaFLOPS	Tot. PetaFLOP
NESRA	24	6424	141	2.76 ± 0.65	177.59
NES^{2}RA Π_{H}	24	6528	138	2.98 ± 0.88	173.57
NES^{2}RA Π_{L}	24	7150	151	2.85 ± 0.68	156.80

Table 7.6: **Statistics of the workunits computational costs**. Statistics computed on the *E. coli* data set.

	GigaFLOP per workunit			
	Min	Max	Avg $\pm$ SD	RSD
NESRA	0.04	186.65	13.67 $\pm$ 19.03	1.39
NES^{2}RA Π_{H}	1.00	134.57	13.21 $\pm$ 16.90	1.28
NES^{2}RA Π_{L}	0.04	228.68	10.86 $\pm$ 15.17	1.40

tional power, we could execute a NES^{2}RA experiment in about 2.5 days. The real execution time, however, may vary depending on several factors, such as the number of different experiments running at the same time on gene@home. Moreover, the double validation required by the gene@home project can increase the completion time of an experiment. Table 7.6 reports a summary of the workunit computational costs for the experiments conducted on *E. coli*. Table 7 presents the details of the computational effort requested by a run of NES^{2}RA, for each combination of the parameters A, D, and I. The number of the workunits is $\left\lceil O_{\mathrm{PC}} * \frac{i}{100} \right\rceil$ were O_{PC} is computed with Equation 7.2 where the second term is zero, in this case. The computational effort requested for each run of the RP function, varies within the same run as it is apparent consider the minimum and the maximum GigaFLOPS values. It is also worth to note that the pair (α, d) determines the computational cost required by a single PC++ execution.

7.5 Conclusion

We presented NES^{2}RA, our novel approach for generating ranked candidate genes lists, which expands known LGNs starting from gene expression data. It exploits iterated variable subsetting and ranking aggregation, as our previous proposal NESRA [Asnicar et al., 2015a], allowing the user to integrate the available prior knowledge on the network that has to be expanded. This makes possible to model the biologist

Table 7.7: Detailed BOINC statistics for NES^{2}RA Π_L on the *E. coli* data set.

α	d	i	Workunits	GigaFLOP per workunit					GigaFLOP
				Min	Max	Avg ± SD	RSD	Sum	per PC++
0.01	100	80	38	0.66	3.39	2.52 ± 0.54	0.21	191.75	0.05
		100	47	0.68	5.28	2.65 ± 0.63	0.24	248.78	0.05
		500	235	0.68	6.11	2.52 ± 0.60	0.24	1186.04	0.05
		1000	470	0.66	8.41	2.64 ± 1.00	0.38	2493.82	0.05
		1500	705	0.63	8.67	2.45 ± 0.95	0.39	3461.11	0.05
		2000	940	0.04	28.07	2.32 ± 1.18	0.51	4492.96	0.05
	200	80	18	3.17	14.93	11.51 ± 2.54	0.22	414.29	0.24
		100	22	6.13	18.32	12.05 ± 3.00	0.25	530.25	0.24
		500	110	3.17	18.13	9.92 ± 2.82	0.28	2182.22	0.20
		1000	220	3.11	24.72	10.38 ± 3.10	0.30	4567.99	0.21
		1500	330	2.91	40.92	10.64 ± 6.14	0.58	7024.33	0.21
		2000	440	1.18	40.47	9.96 ± 4.44	0.45	8771.42	0.20
0.05	100	80	38	1.50	11.06	5.65 ± 1.39	0.25	429.41	0.11
		100	47	1.48	11.16	5.82 ± 1.46	0.25	547.30	0.12
		500	235	1.49	10.94	4.64 ± 1.30	0.28	2180.19	0.09
		1000	470	2.01	19.22	5.16 ± 1.96	0.38	4863.63	0.10
		1500	705	1.40	13.00	4.77 ± 1.88	0.39	6719.52	0.10
		2000	940	1.44	228.68	4.99 ± 5.37	1.08	9403.84	0.10
	200	80	18	14.49	115.36	46.83 ± 17.49	0.37	1686.03	0.96
		100	22	23.18	56.14	40.58 ± 9.21	0.23	1785.45	0.81
		500	110	12.21	88.43	38.44 ± 10.90	0.28	8571.81	0.78
		1000	220	12.19	98.01	38.79 ± 13.43	0.35	17184.80	0.78
		1500	330	12.02	93.27	41.20 ± 15.54	0.38	27358.36	0.83
		2000	440	5.45	134.84	43.28 ± 14.84	0.34	40513.54	0.92

knowledge about the presence of certain genes in the LGN that is translated into a higher probability of presence of these genes in the variable subsets generated. The injection of such prior knowledge shows encouraging results. NES^{2}RA relies on the computational power provided by the gene@home BOINC project, hosted by the TN-Grid platform [Asnicar et al., 2015b]. We exploit the gene@home project for extensive executions of the PC++ algorithm, while all the post-processing, ranking, and aggregation analyses are performed off-line. The parallel nature of our approach

together with the efficient implementation of the PC-algorithm (namely, PC++), allow us to easily distribute the computational work using the gene@home project. We evaluated the performances of NES^{2}RA on the FOS LGN of the model plant *Arabidopsis thaliana*. NES^{2}RA outperforms both ARACNE, which is been proven to be a state-of-the-art NI method [Allen et al., 2012], and our previous proposal NESRA [Asnicar et al., 2015a]. The runs on the gadW network of *Escherichia coli* confirmed the good results and permitted the assessment of the computational load of our application. Considering the performances of NES^{2}RA, its ability to scale with respect to the size of the input data, and the quality of the results, we plan to perform an extensive evaluation using different types of data that encompass several organisms, which include the bacterial model organism *Escherichia coli* and the eukaryote organism *Vitis vinifera*.

Chapter 8

OneGenE: Regulatory Gene Network Expansion via Distributed Volunteer Computing on BOINC

In order to overcome some of the limitations of the original PC algorithm, Asnicar et al. [2015a] suggested that by subsetting the input variables and then combining the results of the subsets, it could allow to address both bigger genomes (i.e., larger number of genes) and also increase the precision of the results. In the previous chapter we showed that by employing a stratified subsetting strategy, in which the appearance of the input genes in the subsets is ruled by a confidence value provided as input, can further increase the quality of the results. In this last case, the task can informally be defined as of Gene Network Expansion (GNE), namely finding the genes that are connected to a given Local Gene Network (LGN). The resulting algorithm, Network Expansion Stratified Subsetting and Ranking Aggregation (NES^{2}RA), was applied to four networks of *Vitis vinifera* data [Malacarne et al., 2018].

Since 2014 we have been running a distributed-computation project, called gene@home, for the specific task of GNE [Cavecchia et al., 2017]. The gene@home project has hosted the runs of both NESRA and NES^{2}RA and is now hosting the OneGenE computation. The gene@home project runs on TN-Grid[1] an infrastructure based on BOINC [Anderson, 2004b], an open-source and popular framework for distributed volunteer computing, in which we distribute to the clients host a C++ implemen-

[1] https://gene.disi.unitn.it/test/

tation of the `skeleton` function of the PC algorithm (PC++). The overall computational power provided by the users to the project (with a peak of more than 20 TFLOPS as of 20/09/2018) allows us to highly scale-up on both the number and the dimension of the GNE tasks. In particular, it has become now feasible to systematically expand each single gene of a genome with up to 30K genes, producing a resource for both biologists and bioinformaticians that can be further used for real-time LGN expansion, gene regulatory network inference, and interactive browsing.

The contribution of this chapter is two-fold. On the one side we present the algorithm One Gene Expansion (OneGenE) [Asnicar, Masera et al., 2019], its rationale, and the way we plan to use the resulting expansions. In particular, we will see to what extent OneGenE can be used to mimic in real-time the results computed by NES^{2}RA. On the other side we give details about the gene@home project and the current implementation of OneGenE, its performance and the management that we did in order to run it.

Finally, in order to validate our approach, we present the application of OneGenE to the expression data of *Pseudomonas aeruginosa*, a bacterium responsible for fatal infections and capable of developing antibiotic-resistance.

8.1 OneGenE

OneGenE is a novel method to compute ranked candidate gene lists that expand known local gene regulatory networks given expression data. As its predecessor NES^{2}RA, OneGenE is based on the systematic and iterative application of the `skeleton` function of the PC algorithm on subsets of the input data, but it aims to overcome the main criticality of NES^{2}RA, i.e. large latency, by pre-computing partial results on the BOINC platform.

Figure 8.1 shows the blocks scheme of the OneGenE architecture, highlighting the different platforms and the two computational stages:

- **Pre-computation step**: candidate expansion lists are pre-computed for each gene of the target organism (Algorithm 6) exploiting the BOINC platform;
- **Real-time interaction**: the user provides an input LGN and OneGenE computes on-the-fly the final candidate expansion list, by aggregating the pre-computed intermediate results by means of state-of-the-art ranking aggregators (Algorithm 7).

The input of the pipeline consists in a $n \times m$ gene expression data matrix E,

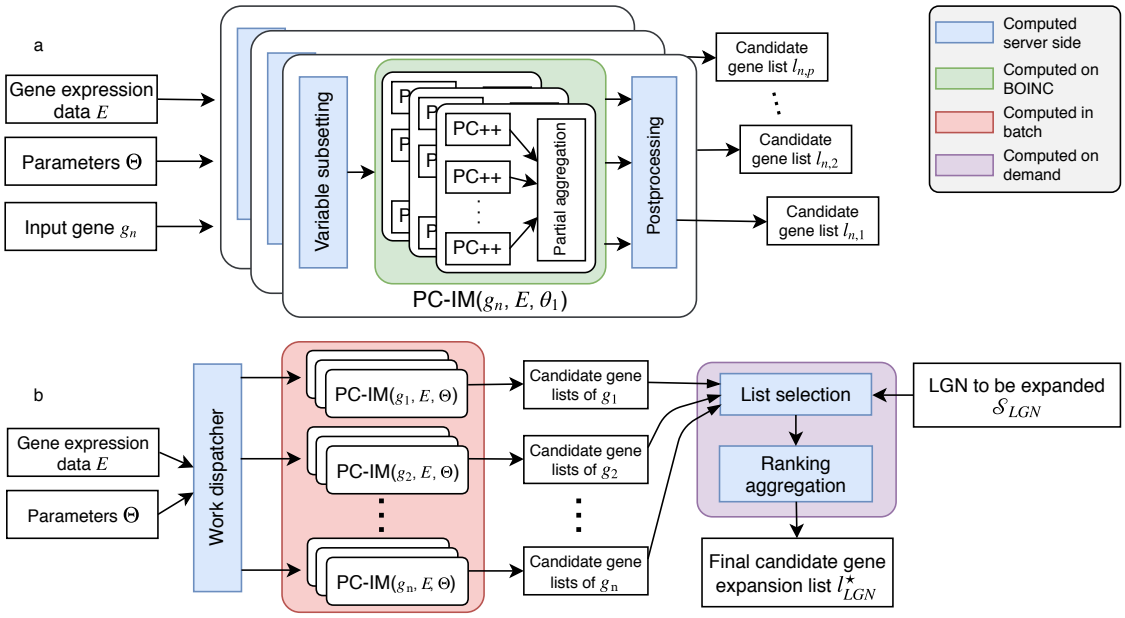


Figure 8.1: **Blocks scheme of the OneGenE architecture**. Subfigure *a* shows the detail of the computation for a single gene g, while Subfigure *b* shows the overview of OneGenE, highlighting the two computational stages.

where $n = |\mathcal{S}|$ is the number of transcripts $\mathcal{S}$ and m the number of conditions, and a set of parameter tuples

$$\Theta = \{(\alpha, d, i) \mid \alpha \in A,\ d \in D,\ i \in I\},$$

where A, D and I are respectively the sets of alpha values, tile sizes (namely subset sizes), and number of iterations. For each transcript g in $\mathcal{S}$, p instances of PC-IM (Figure 8.1a) are executed on the BOINC platform, where $p = |\Theta| = |A| \cdot |D| \cdot |I|$. One PC-IM is a special case of NES^{2}RA, that receives as input-LGN a single gene g with probability vector $\Pi = \mathbf{1}$, i.e. the gene g is present in each subset. Unlike NES^{2}RA, no ranking aggregator is applied at this stage, so each PC-IM returns a candidate expansion list $l_{g,\theta}$ for each tuple of parameters $\theta = (\alpha, d, i)$ for $\theta \in \Theta$. The candidate gene expansion lists are stored on a local server, and, once all the intermediate results have been computed, OneGenE is ready to be queried by the user with input LGNs.

Let $\mathcal{S}_{LGN}$ be the set of transcripts in an input LGN and $l_{g,\theta}$ is the candidate expansion list of the gene g with the parameter tuple θ. The final candidate gene expansion list is obtained by combining the set of partial results

$$\mathcal{L} = \{l_{g,\theta} \mid g \in \mathcal{S}_{LGN},\ \theta \in \Theta\}$$

by means of a ranking aggregator (see Section 2.5).

Algorithm 6: OneGenE: Precomputation Step.

Data: S set of transcripts, E expression data.

Input: I set of values of number of iterations, D set of values of the subset dimension, A set of values of the significance level α

Result: a set of ordered lists of candidate transcripts

$L \leftarrow \varnothing$ `// L set of ordered lists`

foreach $g \in S$ **do**

  foreach $\theta = (\alpha, d, i) \in A \times D \times I$ **do**

    `//` $l_{g,\theta} \leftarrow$ `RP(`S`, {`g`}, `E`,1, `i`, `d`, `α`) call NES`^{2}`RA Ranking Procedure (Algorithm 4) namely:`

    foreach $j \leqslant i$ **do**

      Randomly generate a minimal collection of subsets of dimension d of S such that g is in every subset and each transcript is in at least one subset

    foreach *subset* **do**

      Run the PC-`skeleton` function (Alg. 1) on the expression data E restricted to the transcripts of the subset and generate a network.

    foreach γ *adjacent to* g *in the networks* **do**

      $f_\gamma \leftarrow$ #networks s.t. γ, g are adjacent

      $f'_\gamma \leftarrow f_\gamma/$(#subsets that contain γ)

    $l_{g,\theta} \leftarrow$ genes ordered with respect to f'_γ

    `// The gene` g `is present in the subsets with probability 1 so RP takes the name of PC-IM and the call above could be denoted as` $l_{g,\theta} \leftarrow$ `PC-IM (`S`,{`g`},`E`,`i`,`d`,`α`)`

    $L \leftarrow L \cup l_{g,\theta}$

return L

8.2 Implementation

Since year 2014 CNR-IMEM, in collaboration with the University of Trento, has run TN-Grid, a computing infrastructure based on the BOINC platform. TN-Grid is hosting gene@home, a project developed with the Edmund Mach Foundation [Cavecchia et al., 2017] with the goal to expand genetic regulatory networks with putative causal relationships by analyzing gene expression data, using the NESRA and NES^{2}RA algorithms. The gene@home project is now also running OneGenE.

8.2.1 Performance

As shown in Section 8.1, the OneGenE application running on the gene@home BOINC server, has the following main sets of parameters: I (number of iterations), D (the subset dimensions, i.e. the tile sizes), and A (the set of α to be used in the statistical test of the PC algorithm). These parameters need to be carefully chosen by balancing the execution speed of the application, the accuracy of the results and

Algorithm 7: OneGenE: Real time interaction

Data: $L = \{l_{g,\theta}\}$ a set of ordered lists of candidate transcripts
Input: k maximum length of the lists
Result: ordered list of candidate transcripts

while *true* **do**
  $\mathcal{L}_{temp} \leftarrow \varnothing$ `// $\mathcal{L}_{temp}$ set of ordered lists`
  User enters/selects/edits S_{LGN} set of LGN transcripts `// Lists selection`
  foreach $l_{g,\theta} \in L$ *such that* $g \in S_{LGN}$ **do**
    $\mathcal{L}_{temp} \leftarrow \mathcal{L}_{temp} \cup \{l_{g,\theta}\}$
  $\mathcal{L}_{temp} \leftarrow \text{top}(\mathcal{L}_{temp}, k)$ `// cut each list in` $\mathcal{L}_{temp}$ `to the first` k `elements`
  visualize l^*_{LGN}=Ranking_aggregation($\mathcal{L}_{temp}$)

the statistical errors, and their values depends on the expression dataset in input. Additionally, the parameter *n_pc*, the number of PC algorithm executions collected in a single workunit (the BOINC unit of work that is distributed to the volunteers), and the *cut_off*, a way of shortening the size of the output file by removing the lesser present interactions found in the ranked output list of each single workunit, have to be selected in order to optimize server and client performance and the overall network bandwidth. Due to the large running times expected for completing a OneGenE run, it is crucial to identify the optimal parameter values at the very early stages of the computational experiment.

The OneGenE experiments are intrinsically slow. The core application, initially written in the R language, was first rewritten in C++ (obtaining a substantial execution speed increase), adapted to BOINC using its API, compiled for different computing platforms (Windows x32/x64, Linux x32/x64, Mac OS and Linux ARM), and made publicly available to the volunteers[2]. One of the BOINC volunteers (Daniel Frużyński, Motorola Solutions Systems Polska Sp.z.o.o), in his spare time, made significant performance improvements to the code, profiling it using Callgrind (Valgrind/Linux) [Developers, 2010], by

- removing two top bottlenecks in code (range check, better I/O performance);
- rewriting of the correlation function, focusing on unaligned load/store instructions and unnecessary memory writes;
- introducing templated versions of most performance-critical functions;
- adding SIMD (Single instruction, multiple data), SSE-AVX-FMA hardware-related optimization;

[2] `https://bitbucket.org/francesco-asnicar/pc-boinc`

Table 8.1: **Optimization of the executable of the OneGenE application**. The table shows the results computed on the *E. coli* dataset, reporting in the last column the relative gain computed w.r.t. the previous version in the table.

version	SIMD	Organism	# of Transcripts	# of Conditions	α	Tile size	Time (s)	Rel. gain
0.09	—	*E. coli*	4065	2470	0.05	200	51.80	—
0.10	sse2	*E. coli*	4065	2470	0.05	200	26.29	1.97
0.11	sse2	*E. coli*	4065	2470	0.05	200	16.62	1.58
1.00	sse2	*E. coli*	4065	2470	0.05	200	10.60	1.57

Table 8.2: **Benchmark table for the OneGenE application**. The benchmark have been computed with application version 1.00, considering the *E. coli* and *P. aeruginosa* organisms, showing how the execution time is influenced by the tile size and α parameters.

Organism	# of Transcripts	# of Conditions	Tile size	α	Time (s)	ETA (years)
E. coli	4065	2470	100	0.05	2.4	1.6
E. coli	4065	2470	200	0.05	10.6	3.6
E. coli	4065	2470	100	0.01	0.9	0.6
E. coli	4065	2470	200	0.01	4.0	1.4
E. coli	4065	2470	500	0.05	180.1	26.1
E. coli	4065	2470	1000	0.05	609.4	49.0
P. aeruginosa	5524	238	100	0.05	0.4	0.5
P. aeruginosa	5524	238	500	0.05	0.5	0.1
P. aeruginosa	5524	238	1000	0.05	1.0	0.1
P. aeruginosa	5524	238	1500	0.05	2.0	0.2
P. aeruginosa	5524	238	2000	0.05	3.6	0.2

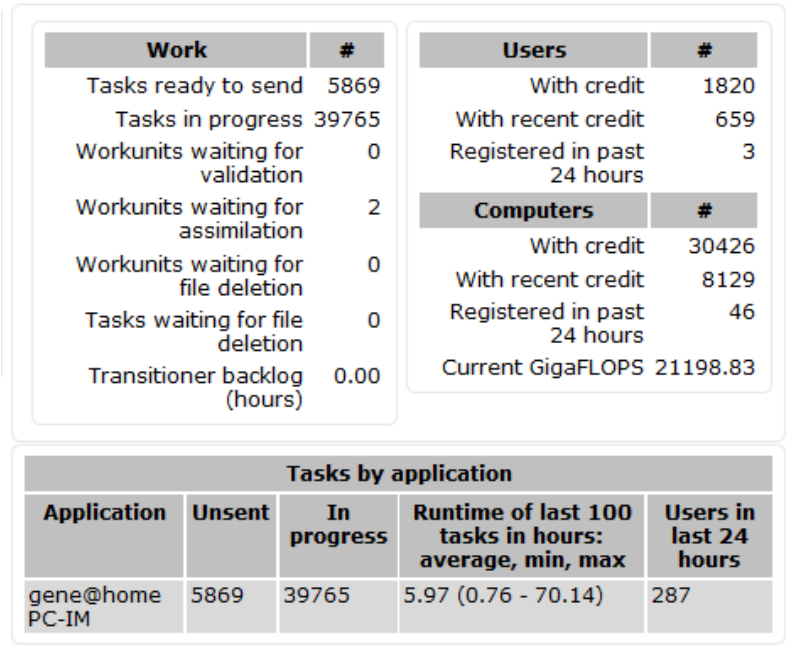
- using Gray code (reflected binary code, RBC, [Frank, 1953]) for generating combinations (version 1.0).

Table 8.1 shows the achieved relative speed-ups, for versions 0.09-1.00 of the application executable, where version 1.00 is the one currently running on gene@home.

8.2.2 Benchmarks

The benchmarks have been executed on the bacterial model organism *E. coli* using the COLOMBOS dataset [Moretto et al., 2016] and on *P. aeruginosa*, the latter will be the considered for further analysis and discussed in Section 8.3.1.

Among the parameters of a BOINC workunit, an important one is the estimate of the computing time, which allows the server to perform efficient scheduling and assign the so-called *credits* (virtual reward for the volunteers) in a fair way. If a



Work	#
Tasks ready to send	5869
Tasks in progress	39765
Workunits waiting for validation	0
Workunits waiting for assimilation	2
Workunits waiting for file deletion	0
Tasks waiting for file deletion	0
Transitioner backlog (hours)	0.00

Users	#
With credit	1820
With recent credit	659
Registered in past 24 hours	3
Computers	**#**
With credit	30426
With recent credit	8129
Registered in past 24 hours	46
Current GigaFLOPS	21198.83

Tasks by application

Application	Unsent	In progress	Runtime of last 100 tasks in hours: average, min, max	Users in last 24 hours
gene@home PC-IM	5869	39765	5.97 (0.76 - 70.14)	287

Task data as of 20 Sep 2018, 9:07:50 UTC

Figure 8.2: **Computing status page of the gene@home project**.

volunteer participates in many BOINC projects, appropriate compute time estimate allows the BOINC client to better choose it's scheduling parameters (cache, priority, resource shares), thus minimizing deadline misses. The calculation is usually done by running a small number of randomly generated PC algorithm executions on a benchmark machine, getting the execution time and calculating the needed FLOPS with the formula $running\_time * host\_flops * scale\_factor$. We therefore built up a benchmark table for a small set of Prokaryotes (Table 8.2).

The reference machine used for the benchmark is an Intel I7-4770k workstation running Linux, with 8GB RAM, Hyper-threading enabled, and a theoretical computational power of 4374.07 MFLOPS and 16809.68 MIPS (average values given by the BOINC client using standard Whetstone/Dhrystone synthetic benchmark system). Benchmarks were run using only one thread, keeping the others free. The *Time* column shows the averaged time needed for completing five single PC++. The *ETA* column (calculated by assuming 1000 as number of iterations) gives an estimate of the time needed for completing a OneGenE pre-computation step for all $g \in \mathcal{S}$ using all the eight threads of the reference machine. For *E. coli* dataset, there is a significant change of the execution time by increasing just the *tile size* parameter from 100 to 1000.

The speed of our flow-controlled *work generator* (one of the Boinc server key

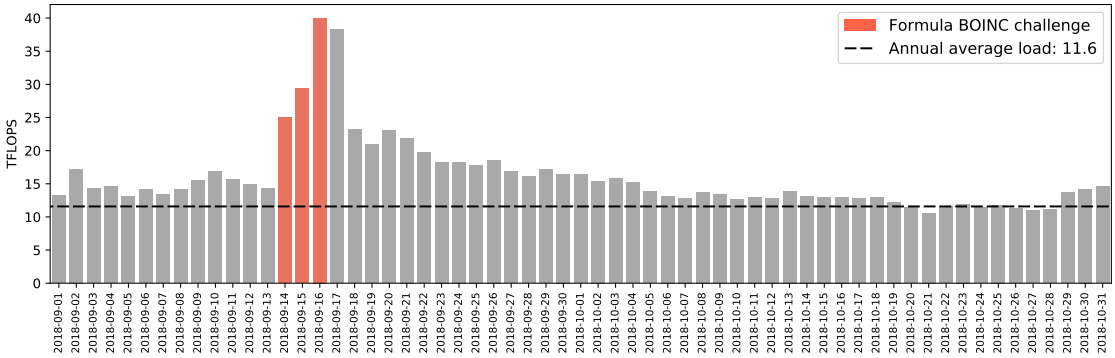


Figure 8.3: **Estimated FLOPS per day for gene@home project**. Data are courtesy of Willy de Zutter (`https://boincstats.com/`)

components, the one that actually builds the workunits, written in the Python language) depends only on the number of transcripts of the dataset and the *tile size*. In the specific case of *P. aeruginosa*, being the workunit's computational time that short (as seen in Table 8.2), the *work generator* was not able to make enough workunits to satisfy the volunteers' requests for jobs. This led to a decrease of the BOINC system computational power.

8.2.3 Computational power and drawbacks

The use of a volunteer-based distributed computing system, like the BOINC framework, has some drawbacks. Any workunit is sent to at least two different volunteers, with a deadline (4 days in our set up). The returned results, i.e. the application output files containing the expansion list, are considered correct only if returned before the deadline and if at least two of them are bit-wise identical (*homogeneous redundancy*), if not they are sent to other volunteers. This procedure, while obviously minimizing client-side computational errors, practically halves our theoretical computing power. Moreover, for expanding a single transcript, the system distributes several workunits, and the slowest one to be completed determines the overall computation time. Another issue is that BOINC is a *volatile* resource, only relying on volunteers, therefore the overall available computing power is difficult to predict and to maintain in time. However, being active in solving volunteer's problems and taking care of communications issues we were able to involve a large number of volunteers, many of them with a large number of powerful computers (a snapshot of the server's status page is shown in Figure 8.2). In the last year we achieved an average power of 11.6 TFLOPS (using about 350 of our reference computer), and this value could be further increased by upgrading our server resources. Figure 8.3 shows the

computing power of TN-Grid in a 60 days range. The high peak around 2018-09-16 is due to a *competition* among volunteers that attracted a significant number of them to the project, just for the duration of the competition (3 days), this peak value is now very close to our theoretical power limit, without using non-volatile resources, such as HPC.

8.3 *Pseudomonas aeruginosa*

The bacterial organism *Pseudomonas aeruginosa* is an ubiquitous species that can be an opportunistic pathogen causing acute and chronic infections in patients with a compromised immune system. In particular, patients affected by Cystic Fibrosis (CF) are susceptible to airways infections mainly caused by *P. aeruginosa*. About 30% of babies affected by CF encounter this pathogen within their first year of life and can experience acute infections. This percentage increase up to 50% if we consider instead a span of time within their three years of life. One of the main reasons that make *P. aeruginosa* a dangerous bacteria lies in its genome plasticity and ability to acquire resistances, in particular to antibiotics. About 70% of adult CF patients that are colonized by *P. aeruginosa* still decease [Moradali et al., 2017].

In this work we retrieve from the COLOMBOS database [Moretto et al., 2016] an expression dataset of *P. aeruginosa* and in particular of the strain PAO1. The expression dataset retrieved is a collection of gene expression values collected from public resources, like Gene Expression Omnibus e ArrayExpress, and incorporates several different types of data, like RNA-seq and microarray. These data undergo: (1) a quality-control step whose aim is to ensure that gene expression measures are consistent; and (2) a normalization step that allows comparison between gene and different experiments.

The retrieved expression dataset is composed of 5,654 genes and for each of them there are 561 expression measures. Expression values here are defined as the log-ratio with respect to a house-keeping gene (i.e., a gene known to be not affected by the experiment and for which its expression is considered basal). The log-ratio normalization can be defined as: $log_2(G/B)$, where G represents the gene expression value of the gene G, and B represent the gene expression value of the basal gene G. This normalization allows the comparison of gene expression measures obtained with different techniques and technologies as in this way it is reported the magnitude of the expression of the a gene with respect to a reference, and not its actual value that can depend on the experiment conditions.

The retrieved dataset has been further cleaned by removing the experiments

that do not have expression values for more than 25% of the genes of *P. aeruginosa* PAO1 strain, reducing the original collection of 561 experiments to 238 experiments. After the removal of experiments with missing values, we checked genes for missing expression values removing such genes that still had missing expression values for more than 25% of the experiments, removing 130 genes from the initial set yelding a dataset of 5,524 genes. Experiments and genes retained after these two cleaning steps that still had missing values, for statistical stability reasons of the PC algorithm, we replaced the missing values with the median computed on the available values.

For the experiments, performed on the cleaned expression dataset of *P. aeruginosa* PAO1 strain, we used the following parameters: $I = \{5000, 10000\}$, $D = \{750, 1000, 1250, 1500\}$, and $A = \{0.01, 0.05\}$. These set of parameters for the OneGenE algorithm result in a total of 16 expansion lists for each gene, which were subsequently aggregated by means of five different aggregators: MC4 [Dwork et al., 2001a], BC-mean, BC-median, RankAggreg [Pihur et al., 2009], and RobustRankAggreg [Kolde et al., 2012].

8.3.1 Experimental results

For the experiments we decided to study a well-known biological pathway in Bacteria called quorum-sensing. Quorum-sensing is a form of "communication" used by Bacteria that allows them to regulate gene expression of cell density-dependent genes. *P. aeruginosa* has two known quorum-sensing systems regulated by *las* and *rhl* gene families, respectively. These two quorum-sensing gene families regulates the expression of many factors, and among them they can control virulence, pathogenicity, and biofilm development [Li et al., 2007]. Our experiments focused on four *P. aeruginosa* genes belonging to the two quorum-sensing networks: *lasI*, *lasR*, *rhlI*, and *rhlR*.

Table 8.3 shows the top 10 genes of final candidate expansion lists for NES^{2}RA and OneGenE, executed with the same set of parameters and aggregated using different ranking aggregators, and Figure 8.4 highlights the similarity of these lists by means of the averaged and plain Jaccard dissimilarities [Greene et al., 2014].

One of the main goal of OneGenE is to overcome the high latency of NES^{2}RA, thus we are interested in being able to recreate the NES^{2}RA candidate expansion lists as precisely as possible. According to Figure 8.4, Borda-Count mean is the best ranking aggregator in this scope. This results was highly expected, because in the post-processing of NES^{2}RA genes found connected to the LGN are normalized by its size. In other words, if we consider each gene in $\mathcal{S}_{LGN}$ as independent, the resulting expansion list of NES^{2}RA is the average of the contributions of the connected genes.

We performed a bibliographic research of the obtained candidate expansion list in

Table 8.3: **Top 10 candidate genes OneGenE**. Top 10 candidate genes for *lasI*, *lasR*, *rhlI*, and *rhlR* networks by ranking aggregator. Bold genes correspond to *class* 1 results, *i.e.* genes related to Quorum Sensing (true positives); while the other are *class* 2 and *class* 3. The last line report the percentage of true positives (TP) for each list.

	NES^{2}RA	OneGenE				
	MC4	BC-Mean	BC-Median	RankAggreg	RobustRank
1	***rsaL*** [Rampioni et al., 2007]	***aprI*** [Wagner et al., 2004]	***rsaL*** [Rampioni et al., 2007]	***rsaL*** [Rampioni et al., 2007]	*clpP2*	***rsaL*** [Rampioni et al., 2007]
2	*clpP2*	***rhlA*** [Pesci et al., 1997]	*clpP2*	*clpP2*	***rsaL*** [Rampioni et al., 2007]	***aprI*** [Wagner et al., 2004]
3	***aprI*** [Wagner et al., 2004]	**PA0122** [Gilbert et al., 2009]	***xcpP*** [Chapon Hervé et al., 1997]	***xcpP*** [Chapon Hervé et al., 1997]	PA2592	***xcpP*** [Chapon Hervé et al., 1997]
4	***xcpP*** [Chapon Hervé et al., 1997]	***lasA*** [Toder et al., 1994]	***aprI*** [Wagner et al., 2004]	***aprI*** [Wagner et al., 2004]	PA3496	**PA0122** [Gilbert et al., 2009]
5	PA5061	***rhlB*** [Pearson et al., 1997]	PA5061	PA5061	***xcpP*** [Chapon Hervé et al., 1997]	***rhlA*** [Pesci et al., 1997]
6	***lasA*** [Toder et al., 1994]	PA3347	*tal*	*tal*	PA3010	*clpP2*
7	PA2592	*clpP2*	***lasA*** [Toder et al., 1994]	***lasA*** [Toder et al., 1994]	PA5178	*mscL*
8	PA4312	PA4141	**PA0122** [Gilbert et al., 2009]	**PA0122** [Gilbert et al., 2009]	PA1245	*prfA*
9	PA3529	PA3010	PA4312	PA4312	**PA0122** [Gilbert et al., 2009]	PA3347
10	PA3496	***xcpP*** [Chapon Hervé et al., 1997]	PA3529	PA3529	***aprI*** [Wagner et al., 2004]	***rhlB*** [Pearson et al., 1997]
TP	40%	60%	50%	50%	40%	60%

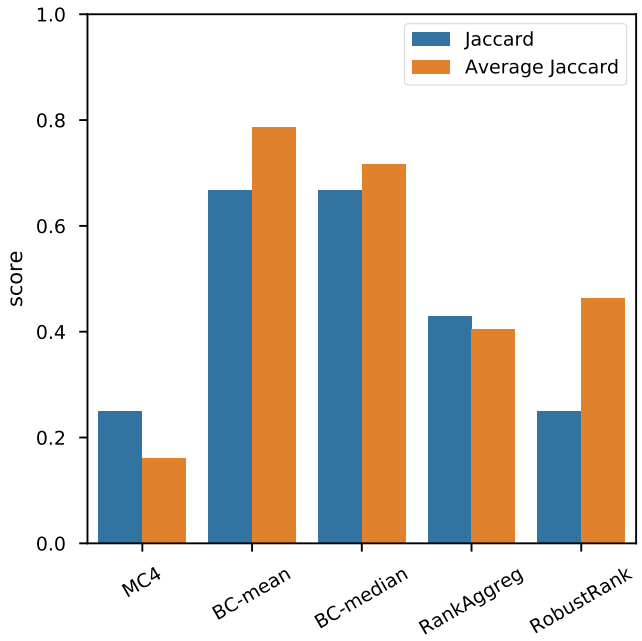


Figure 8.4: **Output similarity by aggregator**. Jaccard and Averaged Jaccard coefficients between NES^{2}RA candidate expansion list and OneGenE with different ranking aggregators.

order to assess the biological relevance of the results. Table 8.3 reports the collected information, such that

- **class 1**: are genes with known relations with the input LGN (marked in bold);
- **class 2**: are known genes, but there are no evidence of relations with the input LGN (reported as gene names in italic);
- **class 3**: are unknown genes (reported with their gene ID).

In the evaluation we consider the worst case, where all the unknown relations are wrong, so both class 2 and class 3 are considered not correct and only class 1 results are true positives. Interestingly, all the OneGenE candidate expansion lists match or outperform the results of NES^{2}RA. In particular, we can notice that both MC4 and RobustRankAggreg, that have the lowest values of list similarity w.r.t. NES^{2}RA, are instead retrieving the best candidate expansion lists.

The gene network expansion approach has the aim of suggesting to biologists potentially relevant genes found to be connected to the provided set of genes of interest. In biology, it is hardly the case for which we have a full knowledge about the function of each gene in a given organism. So, if we look at the expansion lists obtained we can notice that there is one gene (*clpP2*) always appearing in all lists, but that it has never been studied before whether it plays a role with respect to

the quorum-sensing network. In the literature, the *clpP2* gene has been previously reported to be involved in protein degradation of malformed proteins. This can suggests that: (1) it is possible that by studying this particular gene we can find a new interaction within the quorum-sensing network's genes, or (2) it is possible that *P. aeruginosa* was under a condition of stress (at least for the quorum-sensing network genes) when the experiments were conducted.

8.4 Conclusion

The expansion of gene regulatory networks is a task that can benefit from the application of causal discovery methods such as the PC algorithm. Here we built on our previous proposals of subsetting (NESRA) and stratified subsetting (NES^{2}RA) of variables for permitting the application of the PC algorithm to problems with an higher number of variables. The main idea is to systematically expand each gene of a genome and then let the user query with a local gene network and presenting the aggregation of the lists. We called this approach OneGenE.

The implementation of the pre-computation step of OneGenE runs on a BOINC-based distributed architecture that has proven to be reliable and scalable to permit a systematic expansion of each gene of an entire genome. We gave details of the management of the system and we argue that it represents a viable way for addressing the application of OneGenE to other genomes.

We tested the aggregation part of OneGenE and we have shown that its application to *P. aeruginosa* produces in the case of the quorum-sensing network satisfactory results. In particular, the aggregation of the expansion lists of the involved gene not only is able mimic the behavior of NES^{2}RA but, with the most sophisticated ranking aggregation technique, is able to improve on it. However, the test is rather limited and further extensive testing would be needed in order to draw definitive conclusions on the comparison of the methods.

Further work is required in order to systematically applying OneGenE to other organisms and to provide a user-friendly web interface to browse and aggregate the results.

Chapter 9

Conclusions

Proteins are the *workhorse* molecules of life, playing key roles in all biological processes. An accurate functional annotation of these macro molecules and their relations is therefore crucial to unveil the deepest mechanisms of life. To this end next generation sequencing technologies came both as an opportunity and a challenge, leading to a paradigm shift in biology research. Biologist have now access to previously unimaginable amount of data, but are forced to exploit computational tools to unveil the information hidden in the data. In this thesis we explored the intrinsic multi-target nature of this data, providing new predictive tools to deal with them. The common ground among the proposed tools is the possibility to exploit the flux of information among target-variables or examples in order to increase the prediction quality.

Chapter 4 presents Ocelot, a predictive pipeline for protein function prediction (PFP). Ocelot is based on Semantic Based Regularization [Diligenti et al., 2012, 2017], a kernel method developed to incorporate relational knowledge in form first order logic (FOL) rules in the learning process. We built two set of rules, the first one reflecting the taxonomic structure imposed by the gene ontology annotation space, the second one binding interacting proteins with target variables. Ocelot has been applied genome-wide on the eukaryotic model organism *Saccharomyces cerevisiae* after removing homologies to asses its performance. We compared it against GoFDR [Gong et al., 2016] and DeepGO [Kulmanov et al., 2018]. As baseline we used the BLAST-based classifier provided by [Jiang et al., 2016a] and two versions of Ocelot, one without protein-interaction rules and one with no rules at all. Ocelot outperform the competitors, both the one that trained only the yeast data, and the one that have access to UNIREF90 [Suzek et al., 2015]. Moreover, the comparison with the baselines shows that the incorporation of prior knowledge in form of FOL rules effectively constrains the learning process, especially when

the natural threshold $\tau = 0.5$ is chosen. OCELOT is a solid predictive pipeline for intra-genome annotation, but we are planning to expand the set of rules in order to adapt it to the simultaneous annotation of multiple genomes, or the prediction of biochemical pathways.

Artificial neural networks (ANNs) have shown excellent predictive performance, but previous works specifically facing hierarchical-multilabel classification (HMC) [Cerri et al., 2014, 2015] are far from optimal. In chapter 5 we present AWX [Masera and Blanzieri, 2019a], an ANN output layer specifically developed to provide hierarchically consistent multilabel predictions. AWX jointly predicts and optimize the whole target hierarchy: leaf-terms have learned weights and are *directly* predicted, while inner nodes are predicted by combining the leaf-term scores by means of the `max` function or l-norms. By doing this, the prediction for the terms close to the root (typically easier to predict), is forced to flow through the leaf-terms, which are by far more difficult to predict. We introduced a generalization of the true path rule (TPR) to continuous prediction to guarantee hierarchical consistency independently to the chose threshold value τ and we proved that AWX is consistent both under the TPR and its generalized version. To assess AWX performance we evaluated it on different biological benchmark datasets, annotated both with GO (DAG structure) and FunCat (tree structure). AWX outperformed the state-of-the-art Clus-HMC [Vens et al., 2008] and the previous ANN-based approach, namely HMC-MLP. But more importantly, we showed that the AWX architecture effectively exchanges information among target variables, as demonstrated by the comparison with the baseline method MLP_{levaes} that is constantly outperformed by AWX. We released the code of AWX as a component of Keras, a popular framework for ANN programming, so that it could be easily integrated in any ANN architecture that requires a consistent hierarchical multilabel prediction.

In Chapter 6 we face the problem of having a multi-target prediction task in the data, that is however not reflected by a binary annotation space. We present VSC [Masera and Blanzieri, 2019b] as a possible solution, a "concept" classifier designed to test the idea that features based on the notion of locality can be effectively incorporated in an ANN architecture. VSC defines max-margin hyperplanes for a subset of opposite-class pairs and a confidence measure characterized in terms of Chebichev inequality. The confidence measure weights the contribution of the computer hyper planes given the target example, giving more importance to those planes that are "closer" to the target example. We experimentally evaluated VSC on 22 benchmark datasets. The impact of the locality-based confidence measure was assessed by introducing a baseline version of VSC where the confidence was

fixed to 1, while the general performance were compared against 10 general-purpose binary classifiers. The evaluation showed that VSC effectively incorporates the concept of locality by means of the confidence measure obtaining significantly better results. VSC proved to be a valid binary classifier against the tested competitors, outperforming them with the exception of SVM with RBF kernel. This is however not contradictory with our thesis, since RBF kernel is also a local method.

Chapter 7 presents NES^{2}RA [Asnicar, Masera et al., 2016], an evolution of its predecessor NESRA [Asnicar et al., 2015a], a predictive pipeline designed to search for candidates to expand known gene regulatory networks (GRN). Both approaches subdivide the whole set of genes of the input organism in (almost) disjoint subsets, reconstruct a causal network for each subset and then combine the results with ranking aggregators. NES^{2}RA enriches its predecessor by allowing the researcher to modulate the appearance probability of the genes to be expanded thanks to a probability vector. The rationale is to increase the focus on the target genes by increasing their chances to be part of the separation set, i.e. set of genes that is used to test the conditional independence. GRNs, also for model organisms are yet only partially known, therefore no gold truth is available to perform a systematic evaluation of the proposed approaches. In order to assess the quality of the predictions, a literature validation was performed for two GRNs, one belonging to *Arabidopsis thaliana*, the other for *Escherichia coli*. NES^{2}RA outperformed its predecessor NESRA and ARACNE [Margolin et al., 2006a], predicting correctly 9 out 10 possible candidates to expand both the considered networks. In collaboration with Edmund Mach Foundation, we successfully applied NES^{2}RA to four GRN of *Vitis vinifera* related to climat change [Malacarne et al., 2018]. In the same article we compare NES^{2}RA with simple correlation, highlighting the advantages of considering simultaneously subsets of genes (separation sets) when deciding over the independence of a pair of genes.

The BOINC-based project *gene@home* provides NES^{2}RA with more then 10 TFLOPS of average computational power, but the computation of a single candidate expansion list may require from few hours to days, making NES^{2}RA not suitable for preliminary exploratory researches. In order to overcome this high latency, we developed OneGenE [Asnicar, Masera et al., 2019], where candidate expansion lists are precomputed for each transcript t in an organism using $\pi_t = 1$ and then combined on demand to expand target networks. We tested OneGenE on the bacterial organism *Pseudomonas aeuruginosa* and performed a literature research to validate the expansion of a subset of the Quorum sensing network. Both the MC4 [Dwork et al., 2001a] and RobustRank [Kolde et al., 2012] ranking aggre-

gators provide solid results, with 60% true positive in the first 10 retrieved results, outperforming even NES^{2}RA (40% of true positives). OneGenE has been already precomputed genome-wide for *Pseudomonas aeuruginosa*, *Escherichia coli*, *Vitis vinifera*, and is currently under execution the expansion of the human expression data of the FANTOM 5 project [Forrest et al., 2014]. Given the positive results obtained in by the preliminary evaluation on *Pseudomonas aeuruginosa*, we hope in the next future to release the precomputed data and develop a web-resource where researchers can easily expand gene networks of interest.

Chapter 10

Publications

International Journals

Teso, Stefano; Masera, Luca; Diligenti, Michelangelo, and Passerini, Andrea. Combining learning and constraints for genome-wide protein annotation. *BMC Bioinformatics*, 20(1):338, 2019

Malacarne, Giulia; Pilati, Stefania; Valentini, Samuel; Asnicar, Francesco; Moretto, Marco; Sonego, Paolo; Masera, Luca; Cavecchia, Valter; Blanzieri, Enrico, and Moser, Claudio. Discovering causal relationships in grapevine expression data to expand gene networks. a case study: Four networks related to climate change. *Frontiers in Plant Science*, 9:1385, 2018

Asnicar, Francesco; Masera, Luca; Coller, Emanuela; Gallo, Caterina; Sella, Nadir; Tolio, Thomas; Morettin, Paolo; Erculiani, Luca; Galante, Francesca; Semeniuta, Stanislau; Malacarne, Giulia; Engelen, Kristof; Argentini, Andrea; Cavecchia, Valter; Moser, Claudio, and Blanzieri, Enrico. NES2RA: Network Expansion by Stratified Variable Subsetting and Ranking Aggregation. *The International Journal of High Performance Computing Applications*, 32(3):380–392, aug 2016

International Conferences

Asnicar, Francesco; Masera, Luca; Pistore, Davide; Valtentini, Samuel; Cavecchia, Valter, and Blanzieri, Enrico. OneGene: Regulatory Gene Network Expansion via Distributed Volunteer Computing on BOINC. In *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2019

Masera, Luca and Blanzieri, Enrico. Very Simple Classifier: a Concept Binary

Classifier to Investigate Features Based on Subsampling and Locality. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2019*, 2019b

Masera, Luca and Blanzieri, Enrico. AWX: An Integrated Approach to Hierarchical-Multilabel Classification. In *Machine Learning and Knowledge Discovery in Databases 2018, Proceedings, Part I*, pages 322–336. Springer International Publishing, 2019a

Asnicar, Francesco; Erculiani, Luca; Galante, Francesca; Gallo, Caterina; Masera, Luca; Morettin, Paolo; Sella, Nadir; Semeniuta, Stanislau; Tolio, Thomas; Malacarne, Giulia; Engelen, Kristoff; Argentini, Andrea; Cavecchia, Valter; Moser, Claudio, and Blanzieri, Enrico. Discovering candidates for gene network expansion by distributed volunteer computing. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 3, pages 248–253, Aug 2015a

Asnicar, Francesco; Sella, Nadir; Masera, Luca; Morettin, Paolo; Tolio, Thomas; Semeniuta, Stanislau; Moser, Claudio; Blanzieri, Enrico, and Cavecchia, Valter. TN-Grid and gene@home project: Volunteer Computing for Bioinformatics. In *BOINC: FAST 2015, Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development*, pages 1–15, 2015b

Cavecchia, Valter; Asnicar, Francesco; Masera, Luca; Blanzieri, Enrico, and Moser, Claudio. GENE@HOME: Improving and optimizing the scientific pipeline. In *Proceedings of the XIII International Scientific Conference on Optoelectronic Equipment and Devices in Systems of Pattern Recognition, Image and Symbol Information Processing*, Kursk, Russia, may 16-19 2017

Chapter 11

Released Software

During my PhD activity I implemented and released the following pieces of software.

AWX

AWX is implemented within the Keras framework, because of its popularity and modularity. AWX can indeed be stacked on top of any NN architecture, and provide the user with a consistent hierarchical-multilabel prediction.

The code is publicly available at `https://github.com/lucamasera/AWX`.

NES^{2}RA

We developed a stand-alone version of the NES^{2}RA pipeline. It relies on PC++, the efficient C++ implementation of the `skeleton` function.

The code is publicly available at `https://github.com/lucamasera/NESSRA`.

Bibliography

Akira, Wada. Analysis of *Escherichia coli* ribosomal proteins by an improved two dimensional gel electrophoresis. I. Detection of four new proteins. *Journal of Biochemistry*, 100(6):1583–1594, 1986.

Alberts, Bruce; Bray, Dennis; Hopkin, Karen; Johnson, Alexander D; Lewis, Julian; Raff, Martin; Roberts, Keith, and Walter, Peter. *Essential cell biology*. Garland Science, 2015.

Alcalá, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L., and Herrera, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

Allen, J. D. and others, . Comparing statistical methods for constructing large scale gene networks. *PLoS ONE*, 7 (1):e29348, 2012.

Aloraini, Adel and ElSawy, Karim M. Potential breast anticancer drug targets revealed by differential gene regulatory network analysis and molecular docking: Neoadjuvant docetaxel drug as a case study. *Cancer informatics*, 17: 1176935118755354, 2018.

Altschul, Stephen F; Gish, Warren; Miller, Webb; Myers, Eugene W, and Lipman, David J. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

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

Anderson, David P. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA, 2004a. IEEE Computer Society. ISBN 0-7695-2256-4.

Anderson, David P. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA, 2004b. IEEE Computer Society. ISBN 0-7695-2256-4.

Anderson, David P and others, . SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

Anderson, Neil G. Co-immunoprecipitation. In *Protein Targeting Protocols*, pages 35–45. Springer, 1998.

Ashburner, M; Blake, J A; Botstein, D; Butler, H; Cherry, J M; Davis, A P; Dolinski, K; Dwight, S S; Eppig, J T; Harris, M A, and others, . Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–9, May 2000.

Asnicar, Francesco; Erculiani, Luca; Galante, Francesca; Gallo, Caterina; Masera, Luca; Morettin, Paolo; Sella, Nadir; Semeniuta, Stanislau; Tolio, Thomas; Malacarne, Giulia; Engelen, Kristoff; Argentini, Andrea; Cavecchia, Valter; Moser, Claudio, and Blanzieri, Enrico. Discovering candidates for gene network expansion by distributed volunteer computing. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 3, pages 248–253, Aug 2015a.

Asnicar, Francesco; Sella, Nadir; Masera, Luca; Morettin, Paolo; Tolio, Thomas; Semeniuta, Stanislau; Moser, Claudio; Blanzieri, Enrico, and Cavecchia, Valter. TN-Grid and gene@home project: Volunteer Computing for Bioinformatics. In *BOINC: FAST 2015, Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development*, pages 1–15, 2015b.

Asnicar, Francesco; Masera, Luca; Coller, Emanuela; Gallo, Caterina; Sella, Nadir; Tolio, Thomas; Morettin, Paolo; Erculiani, Luca; Galante, Francesca; Semeniuta, Stanislau; Malacarne, Giulia; Engelen, Kristof; Argentini, Andrea; Cavecchia, Valter; Moser, Claudio, and Blanzieri, Enrico. NES2RA: Network Expansion by Stratified Variable Subsetting and Ranking Aggregation. *The International Journal of High Performance Computing Applications*, 32(3):380–392, aug 2016.

Asnicar, Francesco; Masera, Luca; Pistore, Davide; Valtentini, Samuel; Cavecchia, Valter, and Blanzieri, Enrico. OneGene: Regulatory Gene Network Expansion via Distributed Volunteer Computing on BOINC. In *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2019.

Bacciu, Davide; Etchells, Terence A.; Lisboa, Paulo J. G., and Whittaker, Joe. Efficient identification of independence networks using mutual information. *Computational Statistics*, 28(2):621–646, Apr 2013. ISSN 1613-9658. doi: 10.1007/s00180-012-0320-6. URL `https://doi.org/10.1007/s00180-012-0320-6`.

Bantscheff, Marcus; Schirle, Markus; Sweetman, Gavain; Rick, Jens, and Kuster, Bernhard. Quantitative mass spectrometry in proteomics: a critical review. *Analytical and bioanalytical chemistry*, 389(4):1017–1031, 2007.

Barabási, Albert-László. Linked: how everything is connected to everything and what it means for business, science and everyday life. *New York: Penguin*, 2003.

Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H., and others, . Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

Bengio, Y.; Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

Blohm, Philipp; Frishman, Goar; Smialowski, Pawel; Goebels, Florian; Wachinger, Benedikt; Ruepp, Andreas, and Frishman, Dmitrij. Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis. *Nucleic acids research*, page gkt1079, 2013.

Borda, J.C. *Mémoire sur les élections au Scrutin*. Histoire de l' Académie Royale des Sciences, 1781.

Borgwardt, Karsten M. *Kernel Methods in Bioinformatics*, pages 317–334. Springer, Berlin, Heidelberg, 2011.

Borgwardt, Karsten M; Ong, Cheng Soon; Schönauer, Stefan; Vishwanathan, SVN; Smola, Alex J, and Kriegel, Hans-Peter. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

Bottou, L.n and Vapnik, V. Local learning algorithms. *Neural computation*, 4(6):888–900, 1992.

Breiman, L. Random forests. *Machine learning*, 45(1):5–32, 2001.

Brown, Patrick O and Botstein, David. Exploring the new world of the genome with dna microarrays. *Nature genetics*, 21(1s):33, 1999.

Butte, Atul J; Tamayo, Pablo; Slonim, Donna; Golub, Todd R, and Kohane, Isaac S. Discovering functional relationships between rna expression and chemotherapeutic susceptibility using relevance networks. *Proceedings of the National Academy of Sciences*, 97(22):12182–12186, 2000.

Cai, X. and others, . A Putative CCAAT-Binding Transcription Factor Is A Regulator of Flowering Timing in Arabidopsis. *Plant Physiol.*, 145(1):98–105, Sep 2007.

Cavecchia, Valter; Asnicar, Francesco; Masera, Luca; Blanzieri, Enrico, and Moser, Claudio. GENE@HOME: Improving and optimizing the scientific pipeline. In *Proceedings of the XIII International Scientific Conference on Optoelectronic Equipment and Devices in Systems of Pattern Recognition, Image and Symbol Information Processing*, Kursk, Russia, may 16-19 2017.

Cerri, Ricardo; Barros, Rodrigo C., and de Carvalho, André C.P.L.F. Hierarchical multi-label classification using local neural networks. *J. Comput. Syst. Sci.*, 80(1):39–56, 2014. ISSN 00220000.

Cerri, Ricardo; Barros, Rodrigo C., and de Carvalho, Andre C. P. L. F. Hierarchical classification of Gene Ontology-based protein functions with neural networks. *2015 Int. Jt. Conf. Neural Networks*, pages 1–8, 2015.

Chapon Hervé, Virginie; Akrim, Mohammed; Latifi, Amel; Williams, Paul; Lazdunski, Andrée, and Bally, Marc. Regulation of the xcp secretion pathway by multiple quorum-sensing modulons in pseudomonas aeruginosa. *Molecular Microbiology*, 24(6):1169–1178, 6 1997. ISSN 1365-2958.

Chatr Aryamontri, Andrew; Breitkreutz, Bobby-Joe; Oughtred, Rose; Boucher, Lorrie; Heinicke, Sven; Chen, Daici; Stark, Chris; Breitkreutz, Ashton; Kolas, Nadine; O'Donnell, Lara, and others, . The biogrid interaction database: 2015 update. *Nucleic acids research*, 43(D1):D470–D478, 2015.

Chen, L. Q. and others, . Sucrose efflux mediated by SWEET proteins as a key step for phloem transport. *Science*, 335(6065):207–211, Jan 2012.

Cherry, J Michael; Hong, Eurie L; Amundsen, Craig; Balakrishnan, Rama; Binkley, Gail; Chan, Esther T; Christie, Karen R; Costanzo, Maria C; Dwight, Selina S; Engel, Stacia R, and others, . Saccharomyces genome database: the genomics resource of budding yeast. *Nucleic acids research*, 40(D1):D700–D705, 2012.

Chollet, François and others, . Keras, 2015.

Colombo, Diego and Maathuis, Marloes H. Order-independent constraint-based causal structure learning. *arXiv preprint arXiv:1211.3295*, 2012.

Colombo, Diego and Maathuis, Marloes H. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

Cormen, Thomas H. and others, . *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1): 21–27, January 1967.

Csermely, Peter; Korcsmáros, Tamás; Kiss, Huba JM; London, Gábor, and Nussinov, Ruth. Structure and dynamics of molecular networks: A novel paradigm of drug discovery. *Pharmacology & Therapeutics*, 138(3):333–408, 2013.

Curnow, Harold J and Wichmann, Brian A. A synthetic benchmark. *The Computer Journal*, 19(1):43–49, 1976.

Das, Rhiju and others, . Structure prediction for CASP7 targets using extensive all-atom refinement with Rosetta@home. *Proteins: Structure, Function, and Bioinformatics*, 69(S8):118–128, 2007.

Dash, S. and others, . PLEXdb: gene expression resources for plants and plant pathogens. *Nucleic Acids Res.*, 40 (Database issue):D1194–1201, Jan 2012.

Dayhoff, Margaret O and Schwartz, Robert M. A model of evolutionary change in proteins. In *In Atlas of protein sequence and structure*. Citeseer, 1978.

Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

Dessimoz, Christophe; Skunca, Nives, and Thomas, Paul D. Cafa and the open world of protein function predictions. *Trends Genet*, 29(11):609–10, 2013.

Developers, Valgrind. Callgrind: a call-graph generating cache and branch prediction profiler, 2010.

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

Diligenti, Michelangelo; Gori, Marco, and Saccà, Claudio. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

Dukan, SAM and Touati, Daniele. Hypochlorous acid stress in *Escherichia coli*: resistance, DNA damage, and comparison with hydrogen peroxide stress. *Journal of Bacteriology*, 178(21):6145–6150, 1996.

Dutta, S.t and Ghosh, A. On some transformations of high dimension, low sample size data for nearest neighbor classification. *Machine Learning*, 102(1):57–83, 2016.

Dwork, Cynthia; Kumar, Ravi; Naor, Moni, and Sivakumar, D. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001a. ACM. ISBN 1-58113-348-0.

Dwork, Cynthia; Kumar, Ravi; Naor, Moni, and Sivakumar, Dandapani. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001b.

Emmert-Streib, Frank; Dehmer, Matthias, and Haibe-Kains, Benjamin. Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. *Frontiers in cell and developmental biology*, 2:38, 2014.

Engh, Richard A and Huber, Robert. Accurate bond and angle parameters for x-ray protein structure refinement. *Acta Crystallographica Section A*, 47(4):392–400, 1991.

Espinosa-Soto, C. and others, . A Gene Regulatory Network Model for Cell-Fate Determination during *Arabidopsis thaliana* Flower Development That Is Robust and Recovers Experimental Gene Expression Profiles. *Plant Cell*, 16(11):2923–2939, Nov 2004.

Fang, Hai and Gough, Julian. A domain-centric solution to functional genomics via dcgo predictor. *BMC bioinformatics*, 14(3):S9, 2013.

Forrest, Alistair RR; Kawaji, Hideya; Rehli, Michael; Baillie, J Kenneth; De Hoon, Michiel JL; Haberle, Vanja; Lassmann, Timo; Kulakovskiy, Ivan V; Lizio, Marina; Itoh, Masayoshi, and others, . A promoter-level mammalian expression atlas. *Nature*, 507(7493):462, 2014.

Franceschini, Andrea; Szklarczyk, Damian; Frankild, Sune; Kuhn, Michael; Simonovic, Milan; Roth, Alexander; Lin, Jianyi; Minguez, Pablo; Bork, Peer; von Mering, Christian, and others, . String v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(D1):D808–D815, 2013.

Frank, Gray. Pulse code communication, March 17 1953. US Patent 2,632,058.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Friedberg, Iddo. Automated protein function prediction–the genomic challenge. *Briefings in Bioinformatics*, 7(3): 225–242, 2006.

Fu, Limin; Niu, Beifang; Zhu, Zhengwei; Wu, Sitao, and Li, Weizhong. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

Fukunaga, K. and Hostetler, L. k-nearest-neighbor bayes-risk estimation. *Information Theory, IEEE Transactions on*, 21(3):285–293, May 1975.

Gabaldón, T and Huynen, M A. Prediction of protein function and pathways in the genome era. *Cell Mol Life Sci*, 61(7-8):930–44, Apr 2004.

Gardner, T. S. and others, . Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, Jul 2003.

Gasch, Audrey P; Spellman, Paul T; Kao, Camilla M; Carmel-Harel, Orna; Eisen, Michael B; Storz, Gisela; Botstein, David, and Brown, Patrick O. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

Gene Ontology Consortium, . Creating the gene ontology resource: design and implementation. *Genome Res.*, 11 (8):1425–33, 2001. ISSN 1088-9051.

Getoor, Lise and Taskar, Ben, editors. *Introduction to statistical relational learning*. MIT Press, 2007.

Gilbert, Kerrigan B.; Kim, Tae Hoon; Gupta, Rashmi; Greenberg, E. Peter, and Schuster, Martin. Global position analysis of the pseudomonas aeruginosa quorum-sensing transcription factor lasr. *Molecular Microbiology*, 73(6): 1072–1085, 9 2009. ISSN 1365-2958.

Gönen, Mehmet and Alpaydın, Ethem. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

Gong, Qingtian; Ning, Wei, and Tian, Weidong. GoFDR: A Sequence Alignment Based Method for Predicting Protein Functions. *Methods*, 93:3–14, 2015. ISSN 1095-9130.

Gong, Qingtian; Ning, Wei, and Tian, Weidong. Gofdr: A sequence alignment based method for predicting protein functions. *Methods*, 93:3–14, 2016.

Greene, Derek; O'Callaghan, Derek, and Cunningham, Pádraig. How many topics? stability analysis for topic models. *CoRR*, abs/1404.4606, 2014.

Gundlach, Jasmin and Winter, Jeannette. Evolution of *Escherichia coli* for maximum HOCl resistance through constitutive expression of the OxyR regulon. *Microbiology*, 160(8):1690–1704, 2014.

Hable, R. Universal consistency of localized versions of regularized kernel methods. *The Journal of Machine Learning Research*, 14(1):153–186, 2013.

Hamp, Tobias; Goldberg, Tatyana, and Rost, Burkhard. Accelerating the original profile kernel. *PloS one*, 8(6): e68459, 2013a.

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

Hand, D. and Vinciotti, V. Local versus global models for classification problems: fitting models where it matters. *The American Statistician*, 57(2):124–131, 2003.

Hartemink, A. J. Reverse engineering gene regulatory networks. *Nat. Biotechnol.*, 23(5):554–555, May 2005.

Hasty, J. and others, . Computational studies of gene regulatory networks: in numero molecular biology. *Nat. Rev. Genet.*, 2(4):268–279, Apr 2001a.

Hasty, Jeff; McMillen, David; Isaacs, Farren, and Collins, James J. Computational studies of gene regulatory networks: in numero molecular biology. *Nature Reviews Genetics*, 2(4):268, 2001b.

Hauser, Alain and Bühlmann, Peter. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13:2409–2464, 2012.

Hayes, Everett T and others, . Oxygen limitation modulates pH regulation of catabolism and hydrogenases, multidrug transporters, and envelope composition in *Escherichia coli* K-12. *BMC Microbiology*, 6(1):1, 2006.

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Hinton, G. E; Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7): 1527–1554, 2006.

Hinton, Geoffrey; Deng, Li; Yu, Dong; Dahl, George E; Mohamed, Abdel-rahman; Jaitly, Navdeep; Senior, Andrew; Vanhoucke, Vincent; Nguyen, Patrick; Sainath, Tara N, and others, . Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Hochreiter, Sepp. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

Hu, W. and others, . Isolation, sequence analysis, and expression studies of florally expressed cDNAs in *Arabidopsis*. *Plant Mol. Biol.*, 53(4):545–563, Nov 2003.

Huang, G.; Zhu, Q., and Siew, C. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1): 489–501, 2006.

Huynh-Thu, Vân Anh and Sanguinetti, Guido. Gene regulatory network inference: an introductory survey. In *Gene Regulatory Networks*, pages 1–23. Springer, 2019.

Imoto, Seiya; Tamada, Yoshinori; Savoie, Christopher J, and Miyanoaa, Satoru. Analysis of gene networks for drug target discovery and validation. In *Target Discovery and Validation Reviews and Protocols*, pages 33–56. Springer, 2007.

Izutsu, Kaori and others, . *Escherichia coli* ribosome-associated protein SRA, whose copy number increases during stationary phase. *Journal of Bacteriology*, 183(9):2765–2773, 2001.

Jain, Anil K and Farrokhnia, Farshid. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.

Jiang, Yuxiang; Oron, Tal Ronnen; Clark, Wyatt T; Bankapur, Asma R; D'Andrea, Daniel; Lepore, Rosalba; Funk, Christopher S; Kahanda, Indika; Verspoor, Karin M; Ben-Hur, Asa, and others, . An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome biology*, 17(1):184, 2016a.

Jiang, Yuxiang and others, . An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome biology*, pages 1–17, 2016b. ISSN 1474-760X.

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

Kalisch, Markus and Bühlmann, Peter. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.*, 8:613–636, May 2007. ISSN 1532-4435.

Kalisch, Markus and others, . Causal Inference Using Graphical Models with the R Package `pcalg`. *Journal of Statistical Software*, 47(11):1–26, 2012.

Keseler, I. M. and others, . EcoCyc: fusing model organism databases with systems biology. *Nucleic Acids Res.*, 41(Database issue):D605–612, Jan 2013.

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

Kim, Juhan and others, . Three serendipitous pathways in *E. coli* can bypass a block in pyridoxal-5'-phosphate synthesis. *Molecular Systems Biology*, 6(1), 2010.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolde, Raivo; Laur, Sven; Adler, Priit, and Vilo, Jaak. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012.

Kondor, Risi Imre and Lafferty, John D. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

Krizhevsky, Alex; Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Kuang, Rui; Ie, Eugene; Wang, Ke; Wang, Kai; Siddiqi, Mahira; Freund, Yoav, and Leslie, Christina. Profile-based string kernels for remote homology detection and motif extraction. *J Bioinform Comput Biol*, 3(03):527–550, 2005.

Kulmanov, Maxat; Khan, Mohammed Asif, and Hoehndorf, Robert. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

Kumar, Ashwani and others, . Conditional Epistatic Interaction Maps Reveal Global Functional Rewiring of Genome Integrity Pathways in *Escherichia coli*. *Cell Reports*, 2016.

Lal, S.; Pacis, L. B., and Smith, H. M. Regulation of the *SQUAMOSA PROMOTER-BINDING PROTEIN-LIKE genes/microRNA156* Module by the Homeodomain proteins PENNYWISE and POUND-FOOLISH in *Arabidopsis*. *Mol Plant*, 4(6):1123–1132, Nov 2011.

Le, Thuc Duy; Zhang, Kun; Kıcıman, Emre; Hyvärinen, Aapo, and Liu, Lin. Preface: The 2018 acm sigkdd workshop on causal discovery. In *Proceedings of 2018 ACM SIGKDD Workshop on Causal Discovery*, pages 1–3, 2018.

Leclerc, Robert D. Survival of the sparsest: robust gene networks are parsimonious. *Molecular Systems Biology*, 4 (1), 2008. ISSN 1744-4292.

LeCun, Y.; Bottou, L.; Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y.; Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

Lee, David; Redfern, Oliver, and Orengo, Christine. Predicting protein function from sequence and structure. *Nature Reviews Molecular Cell Biology*, 8(12):995–1005, 2007.

Lee, J.Y. and others, . Activation of *CRABS CLAW* in the Nectaries and Carpels of Arabidopsis. *The Plant Cell*, 17(1):25–36, 2005.

Lee, Tong Ihn and Young, Richard A. Transcriptional regulation and its misregulation in disease. *Cell*, 152(6): 1237–1251, 2013.

Li, Luen-Luen; Malone, Jane E., and Iglewski, Barbara H. Regulation of the pseudomonas aeruginosa quorum-sensing regulator vqsr. *Journal of Bacteriology*, 189(12):4367–4374, 2007. ISSN 0021-9193.

Li, Zhanchao; Liu, Zhiqing; Zhong, Wenqian; Huang, Menghua; Wu, Na; Xie, Yun; Dai, Zong, and Zou, Xiaoyong. Large-scale identification of human protein function using topological features of interaction network. *Scientific reports*, 6, 2016.

Lin, S. Rank Aggregation Methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):555–570, 2010.

Lombraña González, D; Harutyunyan, A; Segal, B; Zacharov, I; McIntosh, E; Jones, PL; Giovannozzi, M; Rivkin, L; Marquina, MA; Skands, P, and others, . Lhc@home: A volunteer computing system for massive numerical simulations of beam dynamics and high energy physics events. In *Conf. Proc.*, volume 1205201, pages 505–507, 2012.

Lu, Peilong and others, . L-glutamine provides acid resistance for *Escherichia coli* through enzymatic release of ammonia. *Cell Research*, 23(5):635–644, 2013.

Maas, Andrew L; Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

Maathuis, Marloes H; Colombo, Diego; Kalisch, Markus, and Bühlmann, Peter. Predicting causal effects in large-scale systems from observational data. *Nature Methods*, 7(4):247, 2010.

Malacarne, Giulia; Pilati, Stefania; Valentini, Samuel; Asnicar, Francesco; Moretto, Marco; Sonego, Paolo; Masera, Luca; Cavecchia, Valter; Blanzieri, Enrico, and Moser, Claudio. Discovering causal relationships in grapevine expression data to expand gene networks. a case study: Four networks related to climate change. *Frontiers in Plant Science*, 9:1385, 2018.

Marbach, D. and others, . Wisdom of crowds for robust gene network inference. *Nat. Methods*, 9(8):796–804, Aug 2012.

Margolin, A. A. and others, . Reverse engineering cellular networks. *Nat Protoc*, 1(2):662–671, 2006a.

Margolin, A. A. and others, . ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006b.

Masera, Luca. Multiple protein feature prediction with statistical relational learning. *arXiv preprint arXiv:1609.08391*, 2015.

Masera, Luca and Blanzieri, Enrico. AWX: An Integrated Approach to Hierarchical-Multilabel Classification. In *Machine Learning and Knowledge Discovery in Databases 2018, Proceedings, Part I*, pages 322–336. Springer International Publishing, 2019a.

Masera, Luca and Blanzieri, Enrico. Very Simple Classifier: a Concept Binary Classifier to Investigate Features Based on Subsampling and Locality. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2019*, 2019b.

Massjouni, Naveed; Rivera, Corban G, and Murali, TM. Virgo: computational prediction of gene functions. *Nucleic acids research*, 34(suppl_2):W340–W344, 2006.

Masuda, Nobuhisa and Church, George M. Regulatory network of acid resistance genes in *Escherichia coli*. *Molecular Microbiology*, 48(3):699–712, 2003.

Meysman, P. and others, . COLOMBOS v2.0: an ever expanding collection of bacterial expression compendia. *Nucleic Acids Res.*, 42(Database issue):D649–653, Jan 2014.

Milkman, Roger. An *Escherichia coli* homologue of eukaryotic potassium channel proteins. *Proceedings of the National Academy of Sciences*, 91(9):3510–3514, 1994.

Mitchell, Alex; Chang, Hsin-Yu; Daugherty, Louise; Fraser, Matthew; Hunter, Sarah; Lopez, Rodrigo; McAnulla, Craig; McMenamin, Conor; Nuka, Gift; Pesseat, Sebastien, and others, . The interpro protein families database: the classification resource after 15 years. *Nucleic acids research*, 43(D1):D213–D221, 2015.

Monasterio, V.; Castro, J., and Carro, J. Denis: Solving cardiac electrophysiological simulations with volunteer computing. *PLoS One*, 13(10):e0205568, 2018. ISSN 1932-6203.

Moradali, M. Fata; Ghods, Shirin, and Rehm, Bernd H. A. Pseudomonas aeruginosa lifestyle: A paradigm for adaptation, survival, and persistence. *Frontiers in Cellular and Infection Microbiology*, 7:39, 2017. ISSN 2235-2988.

Moretto, Marco; Sonego, Paolo; Dierckxsens, Nicolas; Brilli, Matteo; Bianco, Luca; Ledezma-Tejeida, Daniela; Gama-Castro, Socorro; Galardini, Marco; Romualdi, Chiara; Laukens, Kris; Collado-Vides, Julio; Meysman, Pieter, and Engelen, Kristof. Colombos v3.0: leveraging gene expression compendia for cross-species analyses. *Nucleic Acids Research*, 44(D1):D620–D623, 2016.

Murzin, Alexey G; Brenner, Steven E; Hubbard, Tim, and Chothia, Cyrus. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

Nadaraya, E. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Needleman, Saul B and Wunsch, Christian D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Novák, Vilém. First-order fuzzy logic. *Studia Logica*, 46(1):87–109, 1987.

Obozinski, Guillaume; Lanckriet, Gert; Grant, Charles; Jordan, Michael I, and Noble, William Stafford. Consisten probabilistic outputs for protein function prediction. *Genome biology*, 9(65):1–19, 2008. ISSN 14604140.

Oldham, Michael C; Horvath, Steve, and Geschwind, Daniel H. Conservation and evolution of gene coexpression networks in human and chimpanzee brains. *Proceedings of the National Academy of Sciences*, 103(47):17973–17978, 2006.

Opgen-Rhein, Rainer and Strimmer, Korbinian. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC systems biology*, 1 (1):37, 2007.

Park, Yungki and Marcotte, Edward M. Revisiting the negative example sampling problem for predicting protein–protein interactions. *Bioinformatics*, 27(21):3024–3028, 2011.

Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076, 1962.

Pearl, Judea. *Causality*. Cambridge university press, 2009.

Pearson, J P; Pesci, E C, and Iglewski, B H. Roles of pseudomonas aeruginosa las and rhl quorum sensing systems in control of elastase and rhamnolipid biosynthesis genes. *Journal of Bacteriology*, 179(18):5756–5767, 1997. ISSN 0021-9193.

Pearson, William R. An introduction to sequence similarity ("homology") searching. *Curr Protoc Bioinformatics*, Jun 2013.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pesci, E C; Pearson, J P; Seed, P C, and Iglewski, B H. Regulation of las and rhl quorum sensing in pseudomonas aeruginosa. *Journal of Bacteriology*, 179(10):3127–3132, 1997. ISSN 0021-9193.

Pihur, Vasyl; Datta, Susmita, and Datta, Somnath. Rankaggreg, an r package for weighted rank aggregation. *BMC Bioinformatics*, 10(1):62, Feb 2009. ISSN 1471-2105.

Poggio, T. and Girosi, F. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, Sep 1990.

Poultney, C.; Chopra, S.; LeCun, Y., and others, . Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

Pu, Shuye; Wong, Jessica; Turner, Brian; Cho, Emerson, and Wodak, Shoshana J. Up-to-date catalogues of yeast protein complexes. *Nucleic acids research*, 37(3):825–831, 2009.

Quinlan, J. R. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Radivojac, Predrag; Clark, Wyatt T; Oron, Tal Ronnen; Schnoes, Alexandra M; Wittkop, Tobias; Sokolov, Artem; Graim, Kiley; Funk, Christopher; Verspoor, Karin; Ben-Hur, Asa; Pandey, Gaurav, and Yunes, Jeffrey M. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227, 2013. ISSN 1548-7091.

Raghu, Vineet K.; Poon, Allen, and Benos, Panayiotis V. Evaluation of causal structure learning methods on mixed data types. In *Proceedings of 2018 ACM SIGKDD Workshop on Causal Disocvery*, volume 92 of *Proceedings of Machine Learning Research*, pages 48–65, London, UK, 20 Aug 2018. PMLR.

Ralaivola, Liva; Swamidass, Sanjay J; Saigo, Hiroto, and Baldi, Pierre. Graph kernels for chemical informatics. *Neural networks*, 18(8):1093–1110, 2005.

Rampioni, Giordano; Schuster, Martin; Greenberg, Everett Peter; Bertani, Iris; Grasso, Marco; Venturi, Vittorio; Zennaro, Elisabetta, and Leoni, Livia. Rsal provides quorum sensing homeostasis and functions as a global regulator of gene expression in pseudomonas aeruginosa. *Molecular Microbiology*, 66(6):1557–1565, 12 2007. ISSN 1365-2958.

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

Reuter, Jason A; Spacek, Damek V, and Snyder, Michael P. High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597, 2015.

Rosenblatt, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Rost, Burkhard; Liu, Jinfeng; Nair, Rajesh; Wrzeszczynski, Kazimierz O, and Ofran, Yanay. Automatic prediction of protein function. *Cellular and Molecular Life Sciences CMLS*, 60(12):2637–2650, 2003.

Ruepp, Andreas; Zollner, Alfred; Maier, Dieter; Albermann, Kaj; Hani, Jean; Mokrejs, Martin; Tetko, Igor; Güldener, Ulrich; Mannhaupt, Gertrud; Münsterkötter, Martin, and Mewes, H. Werner. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545, 2004. ISSN 03051048.

Rumelhart, David E; Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Saccà, Claudio; Teso, Stefano; Diligenti, Michelangelo, and Passerini, Andrea. Improved multi-level protein–protein interaction prediction with semantic-based regularization. *BMC bioinformatics*, 15(1):103, 2014.

Sanchez-Corrales, Y. E. and others, . The *Arabidopsis thaliana* flower organ specification gene regulatory network determines a robust differentiation process. *J. Theor. Biol.*, 264(3):971–983, Jun 2010.

Sanger, Frederick. The free amino groups of insulin. *Biochemical Journal*, 39(5):507, 1945.

Sanger, Frederick; Nicklen, Steven, and Coulson, Alan R. Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467, 1977.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Scholkopf, B.; Sung, Kah-Kay; Burges, C.J.C.; Girosi, F.; Niyogi, P.; Poggio, T., and Vapnik, V. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *Signal Processing, IEEE Transactions on*, 45(11):2758–2765, Nov 1997. ISSN 1053-587X.

Scholkopf, Bernhard and Smola, Alexander J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

Schölkopf, Bernhard; Herbrich, Ralf, and Smola, Alex J. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.

Schriml, Lynn Marie; Arze, Cesar; Nadendla, Suvarna; Chang, Yu-Wei Wayne; Mazaitis, Mark; Felix, Victor; Feng, Gang, and Kibbe, Warren Alden. Disease ontology: a backbone for disease semantic integration. *Nucleic acids research*, 40(D1):D940–D946, 2011.

Segata, N. and Blanzieri, E. Fast and scalable local kernel machines. *Journal of Machine Learning Research*, 11: 1883–1926, 2010.

Sen, Prithviraj; Namata, Galileo; Bilgic, Mustafa; Getoor, Lise; Galligher, Brian, and Eliassi-Rad, Tina. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

Shi, J. X. and others, . SHINE Transcription Factors Act Redundantly to Pattern the Archetypal Surface of Arabidopsis Flower Organs. *PLoS Genet.*, 7(5):e1001388, May 2011.

Škunca, Nives; Bošnjak, Matko; Kriško, Anita; Panov, Panče; Džeroski, Sašo; Šmuc, Tomislav, and Supek, Fran. Phyletic profiling with cliques of orthologs is enhanced by signatures of paralogy relationships. *PLoS computational biology*, 9(1):e1002852, 2013.

Smith, Temple F and Waterman, Michael S. Comparison of biosequences. *Advances in applied mathematics*, 2(4): 482–489, 1981.

Sokolov, Artem and Ben-Hur, Asa. Hierarchical classification of gene ontology terms using the gostruct method. *Journal of bioinformatics and computational biology*, 8(02):357–376, 2010.

Sokolov, Artem; Funk, Christopher; Graim, Kiley; Verspoor, Karin, and Ben-Hur, Asa. Combining heterogeneous data sources for accurate functional annotation of proteins. *BMC bioinformatics*, 14(3):S10, 2013.

Sokolova, M. and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

Sonnenburg, Sören; Rätsch, Gunnar; Schäfer, Christin, and Schölkopf, Bernhard. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7(Jul):1531–1565, 2006.

Soricut, Radu and Marcu, Daniel. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003.

Sorower, Mohammad S. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18, 2010.

Spellman, Paul T; Sherlock, Gavin; Zhang, Michael Q; Iyer, Vishwanath R; Anders, Kirk; Eisen, Michael B; Brown, Patrick O; Botstein, David, and Futcher, Bruce. Comprehensive identification of cell cycle–regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular biology of the cell*, 9(12):3273–3297, 1998.

Spirtes, P. and Glymour, C. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991a.

Spirtes, Peter and Glymour, Clark. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991b.

Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Stoyanov, Jivko V; Hobman, Jon L, and Brown, Nigel L. CueR (YbbI) of *Escherichia coli* is a MerR family regulator controlling expression of the copper exporter CopA. *Molecular Microbiology*, 39(2):502–512, 2001.

Stuart, Joshua M; Segal, Eran; Koller, Daphne, and Kim, Stuart K. A gene-coexpression network for global discovery of conserved genetic modules. *science*, 302(5643):249–255, 2003.

Sturm, Irene; Lapuschkin, Sebastian; Samek, Wojciech, and Müller, Klaus-Robert. Interpretable deep neural networks for single-trial eeg classification. *Journal of neuroscience methods*, 274:141–145, 2016.

Sun, Aixin and Lim, Ee-Peng. Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 521–528. IEEE, 2001.

Suzek, Baris E.; Wang, Yuqi; Huang, Hongzhan; McGarvey, Peter B., and Wu, Cathy H. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Tan, M. and others, . Combining multiple types of biological data in constraint-based learning of gene regulatory networks. In *Computational Intelligence in Bioinformatics and Computational Biology, 2008. CIBCB '08. IEEE Symposium on*, pages 90–97, Sept 2008.

Tan, M. and others, . Influence of prior knowledge in constraint-based learning of gene regulatory networks. *IEEE/ACM Trans Comput Biol Bioinform*, 8(1):130–142, 2011.

Teso, Stefano; Masera, Luca; Diligenti, Michelangelo, and Passerini, Andrea. Combining learning and constraints for genome-wide protein annotation. *BMC Bioinformatics*, 20(1):338, 2019.

Toder, D S; Ferrell, S J; Nezezon, J L; Rust, L, and Iglewski, B H. lasa and lasb genes of pseudomonas aeruginosa: analysis of transcription and gene product activity. *Infection and Immunity*, 62(4):1320–1327, 1994. ISSN 0019-9567.

Triguero, Isaac and Vens, Celine. Labelling strategies for hierarchical multi-label classification techniques. *Pattern Recognit.*, pages 1–14, 2015. ISSN 00313203.

Tucker, Don L; Tucker, Nancy, and Conway, Tyrrell. Gene Expression Profiling of the pH Response in *Escherichia coli*. *Journal of Bacteriology*, 184(23):6551–6558, 2002.

Uzilov, Andrew V; Keegan, Joshua M, and Mathews, David H. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

Valentini, Giorgio and Masulli, Francesco. Ensembles of learning machines. In *Italian Workshop on Neural Nets*, pages 3–20. Springer, 2002.

Vens, Celine; Struyf, Jan; Schietgat, Leander; Džeroski, Sašo, and Blockeel, Hendrik. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, 73(2):185–214, 2008. ISSN 08856125.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y., and Manzagol, P. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

Volkert, Michael R and Nguyen, Dinh C. Induction of specific *Escherichia coli* genes by sublethal treatments with alkylating agents. *Proceedings of the National Academy of Sciences*, 81(13):4110–4114, 1984.

Waegeman, Willem; Dembczyński, Krzysztof, and Hüllermeier, Eyke. Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery*, pages 1–32, 2018.

Wagner, Victoria E.; Gillis, Richard J., and Iglewski, Barbara H. Transcriptome analysis of quorum-sensing regulation and virulence factor expression in pseudomonas aeruginosa. *Vaccine*, 22:S15 – S20, 2004. ISSN 0264-410X. Immunological Approaches against Nosocomial Infections.

Wahl, V. and others, . The FANTASTIC FOUR proteins influence shoot meristem size in *Arabidopsis thaliana*. *BMC Plant Biology*, 10(1):285, 2010.

Wang, F. and Sun, Jimeng. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2):534–564, 2014.

Wang, M. and others, . Inferring large-scale gene regulatory networks using a low-order constraint-based algorithm. *Mol Biosyst*, 6(6):988–998, Jun 2010.

Wang, Zhong; Gerstein, Mark, and Snyder, Michael. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57, 2009.

Watson, G. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

Watson, James D; Crick, Francis HC, and others, . Molecular structure of nucleic acids. *Nature*, 171(4356):737–738, 1953.

Weber, Arnim; Kögl, Stephanie A, and Jung, Kirsten. Time-Dependent Proteome Alterations under Osmotic Stress during Aerobic and Anaerobic Growth in *Escherichia coli*. *Journal of Bacteriology*, 188(20):7165–7175, 2006.

Werbos, Paul. Beyond regression: new tools for prediction and analysis in the behavioral sciences. 01 1974.

Yip, Kevin Y; Kim, Philip M; McDermott, Drew, and Gerstein, Mark. Multi-level learning: improving the prediction of protein, domain and residue interactions by allowing information flow between levels. *BMC bioinformatics*, 10 (1):241, 2009.

Yoshida, Hideji and others, . YqjD is an Inner Membrane Protein Associated with Stationary-Phase Ribosomes in *Escherichia coli*. *Journal of Bacteriology*, 194(16):4178–4183, 2012.

Youngs, Noah; Penfold-Brown, Duncan; Bonneau, Richard, and Shasha, Dennis. Negative example selection for protein function prediction: the nogo database. *PLoS Comput Biol*, 10(6):e1003644, 2014.

Yu, Guoxian; Fu, Guangyuan; Wang, Jun, and Zhu, Hailong. Predicting protein function via semantic integration of multiple networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(2):220–232, 2016.

Zadeh, Lofti A. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Zeiler, Md and Fergus, Rob. Visualizing and understanding convolutional networks. *Comput. Vision–ECCV 2014*, 8689:818–833, 2014. ISSN 978-3-319-10589-5.

Zhang, Min Ling and Zhou, Zhi Hua. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, 2014. ISSN 10414347.

Zhang, Quanshi; Wu, Ying Nian, and Zhu, Song-Chun. Interpretable convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8827–8836, 2018.

Zhang, X. and others, . Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28(1):98–104, Jan 2012.

Zhou, L. and Hu, Z. Chebyshev's inequality for banach-space-valued random elements. *Statistics & Probability Letters*, 82(5):925–931, 2012.

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

Zik, M. and Irish, V. F. Global identification of target genes regulated by APETALA3 and PISTILLATA floral homeotic gene action. *Plant Cell*, 15(1):207–222, Jan 2003.