

University of Liège - Faculty of Applied Sciences
Department of Electrical Engineering and Computer Science



Supervised inference of biological networks with trees

Application to genetic interactions in yeast

Marie Schrynemackers

Supervisor
Prof. Pierre Geurts

PhD dissertation

November 2014

Summary

Networks or graphs provide a natural representation of molecular biology knowledge, in particular to model relationships between biological entities such as genes, proteins, drugs, or diseases. Because of the effort, the cost, or the lack of the experiments necessary to the elucidation of these networks, computational approaches for network inference have been frequently investigated in the literature. In this thesis, we focus on supervised network inference methods. These methods exploit supervised machine learning algorithms to train a model for identifying new interacting pairs of nodes from a training sample of known interacting and possibly non-interacting pairs and additional measurement data about the network nodes.

Our contributions in this area are divided into three parts. First, the thesis examines the problem of the assessment of supervised network inference methods. Indeed, their reliable validation (in silico) poses a number of new challenges with respect to standard classification problems, related to the fact that pairs of objects are to be classified and to the specificities of biological networks. We perform a critical review and assessment of protocols and measures proposed in the literature. Through theoretical considerations and in silico experiments, we analyze in depth how important factors influence the outcome of performance estimation. These factors include the amount of information available for the interacting entities, the sparsity and topology of biological networks, and the lack of experimentally verified non-interacting pairs. From this analysis, we derived specific guidelines so as to how best exploit and evaluate machine learning techniques for network inference.

Second, we systematically investigate, theoretically and empirically, the exploitation of tree-based methods for network inference. We consider these methods in the context of the two main generic classification-based approaches for network inference: the local approach, which trains a separate model for each network node, and the global approach, which trains a single model over pairs of nodes. We present and formalize these two approaches, extending the former for the prediction of interactions between two unseen network nodes, and discuss their specializations to tree-based methods, highlighting their interpretability and drawing links with clustering techniques. Extensive experiments are carried out with these methods on various biological networks that clearly highlight that these methods are competitive with existing methods. The interpretability of the resulting method family is illustrated on a drug-protein interaction network.

In the last part of the thesis, we built on the experience gained in the two previous parts to try to predict at best the genetic interaction network in yeast *S.cerevisiae*. For that purpose, we collected a large dataset, assembling 4 millions gene pairs that were experimentally tested in the context of 11 different studies and 23 sets of measurements to use as gene input features for the inference. Through several cross-validation experiments on the resulting dataset, we showed that predicting genetic interactions is indeed possible to some useful extent and that actually in some settings, the accuracy of computational methods is not very far from that of experimental techniques.

Résumé

Les réseaux, ou graphes, fournissent une représentation naturelle de nos connaissances de la biologie moléculaire. Ils permettent en particulier de modéliser des relations entre des entités comme des gènes, des protéines, des médicaments, ou des maladies. Du à la difficulté de mise œuvre, au coût, ou simplement à la non-existence des expériences nécessaires à l'élucidation de ces réseaux, des approches informatiques pour l'inférence de réseaux ont souvent été examinées dans la littérature. Dans cette thèse, nous nous focalisons sur les méthodes d'inférence supervisée de réseaux. Ces méthodes exploitent des algorithmes d'apprentissage automatique pour apprendre un modèle permettant d'identifier de nouvelles paires de noeuds en interaction, à partir d'un ensemble d'apprentissage d'interactions connues, et de données supplémentaires mesurées sur les noeuds du réseau.

Nos contributions dans ce domaine sont divisées en trois parties. Tout d'abord, la thèse examine le problème de l'évaluation des méthodes d'apprentissage supervisé de réseaux biologiques. En effet, une validation (in silico) fiable pose un certain nombre de difficultés par rapport aux problèmes de classification standard, provenant du fait que des paires d'objets doivent être classées, et des spécificités des réseaux biologiques. Nous faisons un compte rendu critique et une évaluation des protocoles et mesures proposés dans la littérature. A travers des considérations théoriques et des expériences in silico, nous analysons en profondeur comment d'importants facteurs influencent le résultat de l'estimation de la performance. Ces facteurs incluent la quantité d'information disponible pour les entités qui interagissent, la dispersion et la topologie des réseaux biologiques, et le manque de paires ayant été expérimentalement vérifiées comme n'interagissant pas. A partir de cette analyse, nous avons extrait des directives spécifiques afin d'exploiter au mieux et d'évaluer les techniques d'apprentissage automatique pour l'inférence de réseaux biologiques.

Ensuite, nous examinons systématiquement, théoriquement et empiriquement, l'exploitation des méthodes d'arbres pour l'inférence de réseaux. Nous considérons ces méthodes dans le contexte des deux principales approches génériques de classification pour l'inférence de réseaux : l'approche locale, qui apprend un modèle séparé pour chaque noeud du réseau, et l'approche globale, qui apprend un seul modèle sur toutes les paires de noeuds. Nous présentons et formalisons ces deux approches, en étendant la première aux prédictions des interactions entre deux noeuds non vus, et discutons de leurs spécialisations aux méthodes d'arbres, en mettant en évidence leur interprétabilité et en faisant des liens avec les techniques de clustering. De nombreuses expériences sont réalisées avec ces méthodes sur divers réseaux biologiques, montrant clairement que ces méthodes sont compétitives avec celles existantes. L'interprétabilité des familles de méthodes résultantes est illustrée sur un réseau d'interactions médicaments-protéines.

Dans la dernière partie de la thèse, nous nous appuyons sur l'expérience acquise dans les deux premières parties pour essayer de prédire au mieux le réseau d'interactions génétiques de la levure *S.cerevisiae*. Dans ce but, nous avons collecté un large ensemble de données, comprenant 4 millions de paires de gènes ayant été testées expérimentalement dans le contexte de 11 études différentes,

et 23 ensembles de mesures à utiliser comme caractéristiques d'entrée pour l'inférence. A travers plusieurs expériences de validation croisée sur l'ensemble de données résultant, nous montrons que la prédiction des interactions génétiques est en effet possible dans une certaine mesure et que dans certains contextes, la précision des méthodes informatiques n'est pas très loin de celle des techniques expérimentales.

Acknowledgments

At the end of this work, I wish to sincerely thank all the people who contributed to its realization and allow, through their support and their advice, to complete it successfully.

First of all, I thank particularly my advisor, Pierre Geurts, for all the time he has spent to guide and supervise this research. I would like to say how much I have appreciated his great help and his constant presence and availability. I have also been very sensitive to his generosity and kindness. He has motivated me and encouraged me all along this five years, and this thesis would definitely not have seen the day without him.

I also address many thanks to Louis Wehenkel, who first supervised my master's thesis, which was the prelude of this whole work. He is the one who let me discover the world of research and made me want to do this PhD. Moreover he has always been present to proofread and give precious advice, during all these years.

I am thankful to Madan Babu, who has given me a warm welcome in the Laboratory of Molecular Biology in Cambridge during my master. He has provided me the material necessary to start this work, and very enthusiastic advice. I am also grateful to Robert Küffner, with whom I have had the opportunity to share and discuss. I thank him for his precious remarks, and the time he gave to me.

I would like to express my appreciation to all the members of the jury, for devoting time and interest to the reading and evaluation of this dissertation.

Many thanks of course to all my colleagues from the GIGA bioinformatics, that made me work in a joyous and relaxed atmosphere. Jean-Michel Begon, Vincent Botta, Pierre-Yves Gilson, Fabien Heuze, Renaud Hoyoux, Vân Anh Huynh-Thu, Arnaud Joly, Florence Lemahieu, Gilles Louppe, Raphaël Marée, Loïc Rollus, Yannick Schutz, Olivier Stern, Benjamin Stévens and Stéphane Wenric drove these years very pleasant and unforgettable.

Finally, I thank warmly all my family and friends, for their support and love, during the realization of this thesis. I address all my affection to my parents and grandparents, and to my brother. I thank especially Jean-François who has always been present to lift my spirits and make me happy.

Contents

1	Introduction	13
1.1	Biological network inference	13
1.2	Contributions	16
1.2.1	Performance evaluation	16
1.2.2	Supervised network inference with tree-based methods	17
1.2.3	Predicting genetic interactions in Yeast	18
1.3	Outline of the manuscript	19
1.4	Publications and dissemination of the results	20
2	Background	23
2.1	Biological networks	24
2.2	Supervised learning	26
2.2.1	Decision trees	28
2.2.2	Decision tree ensembles	30
2.2.3	Support vector machines	32
2.2.4	Output kernel trees	35
2.3	Supervised network inference	37
2.3.1	Problem definition	37
2.3.2	Network inference methods	38
2.4	Unsupervised and semi-supervised biological network inference	42
2.5	Discussion	44
3	Evaluating supervised network inference methods	45
3.1	Introduction	46
3.2	Evaluation measures	47
3.2.1	Binary predictions	47
3.2.2	ROC curves	49
3.2.3	Precision-recall curves	50
3.2.4	Comparison of ROC and PR curves	51
3.2.5	Other measures and curves	55
3.2.6	Discussion	56
3.3	Evaluation protocols	57
3.3.1	Cross-validation on pairs	59
3.3.2	Cross-validation on nodes	59
3.3.3	Discussion	60

3.3.4	Illustration	61
3.4	Lack of negative examples	64
3.4.1	Training a model	66
3.4.2	Evaluating a model	69
3.4.3	Illustration	72
3.5	Impact of heavy-tailed node degree distribution	74
3.5.1	Experiment on a simple network	74
3.5.2	Experiments on the DREAM5 network	76
3.6	Merging pair rankings	78
3.6.1	Objective	79
3.6.2	Constructing the optimal merged ranking	81
3.6.3	Illustration	85
3.6.4	Discussion	88
3.7	Conclusion	89
4	Tree-based methods for biological network inference	91
4.1	Introduction	92
4.2	Two different approaches	92
4.2.1	Global approach	92
4.2.2	Local approach	93
4.3	Tree-based ensemble methods	94
4.3.1	Global approach	95
4.3.2	Local approach	95
4.3.3	Interpretability	96
4.3.4	Implementation and computational issues	97
4.4	Experiments	97
4.4.1	Datasets	97
4.4.2	Protocol	99
4.4.3	Results	99
4.4.4	Comparison with related works	106
4.5	Illustration of interpretability of trees	110
4.5.1	Interpretability of single decision trees	110
4.5.2	Clustering with ensembles of trees	116
4.5.3	Pair-based feature ranking	122
4.6	Discussion	124
5	Predicting genetic interactions in yeast	127
5.1	Introduction	128
5.2	Background	129
5.2.1	Yeast <i>Saccharomyces cerevisiae</i>	129
5.2.2	Genetic interactions	129
5.2.3	Experimental techniques	132
5.3	Datasets	136
5.3.1	Training networks	136
5.3.2	Input datasets	141

5.4	Cross-validation experiments	144
5.4.1	Cross-validation across individual networks	145
5.4.2	Computational versus experimental predictions	154
5.4.3	Cross-validation across the ensemble of all networks	157
5.5	Predictions of unlabeled pairs	163
5.5.1	Global ranking of the unlabeled pairs	163
5.5.2	Validation with Gene Ontology	164
5.6	Conclusion	170
6	Conclusion	173
6.1	Main findings and conclusions	173
6.2	Limitations and future research directions	175
6.2.1	Methods	175
6.2.2	Biological applications	176

Chapter 1

Introduction

In this introduction, we first give the context and the motivations of our research. We then present our main contributions and describe the organization of the manuscript.

1.1 Biological network inference

Since the beginning of the genomic revolution, detailed biological information about cellular components has been becoming available. In 2004, the first genome sequence of human was completely mapped (International Human Genome Mapping Consortium, 2004), and allowed to identify human genes. Since then, the time to sequence a human genome has been decreasing exponentially until reaching 24 hours in 2013, for a cost of hundreds of euros (Debré and Gall, 2014). Like for human, genomes of many other species have now been sequenced, and it is thus possible to analyze the ensemble of a genome, instead of studying single genes. Genes encode the information required to assemble proteins. The ensemble of all proteins expressed by a genome is called the proteome. One gene can specify more than one protein, due to alternative splicing or post-translational modifications. Human genome has been estimated to contain more than 20,000 genes, while 30,000 human proteins have been already identified (Kim *et al.*, 2014).

All the information gained about genes and proteins through sequencing is not sufficient however to study cellular activities. An important step for understanding the properties of cellular systems is indeed to find and understand the relationships that exist between their components: which proteins regulate the expression of which gene, which proteins interact with each other to form complexes, which drug interacts with which protein, etc. These relationships are typically modeled through networks. (**Figure 1.1**)

Mathematically, a network (or graph) is an ensemble of objects, called nodes, and an ensemble of edges connecting pairs of nodes. An edge may be directed if it links one node towards another, or undirected if it does not assign an order to the nodes it connects. An edge can also be weighted by a number indicating its strength. A network can be homogeneous if it connects only nodes of the same kind, or bipartite when edges are linking nodes of two kinds. Given this flexibility, networks are very natural as a mathematical language to model relationships between biological entities such as genes, proteins, drugs, or diseases, and are indeed ubiquitous to model them. The semantics of nodes and edges in biological networks can be very diverse. The most studied networks include among others:

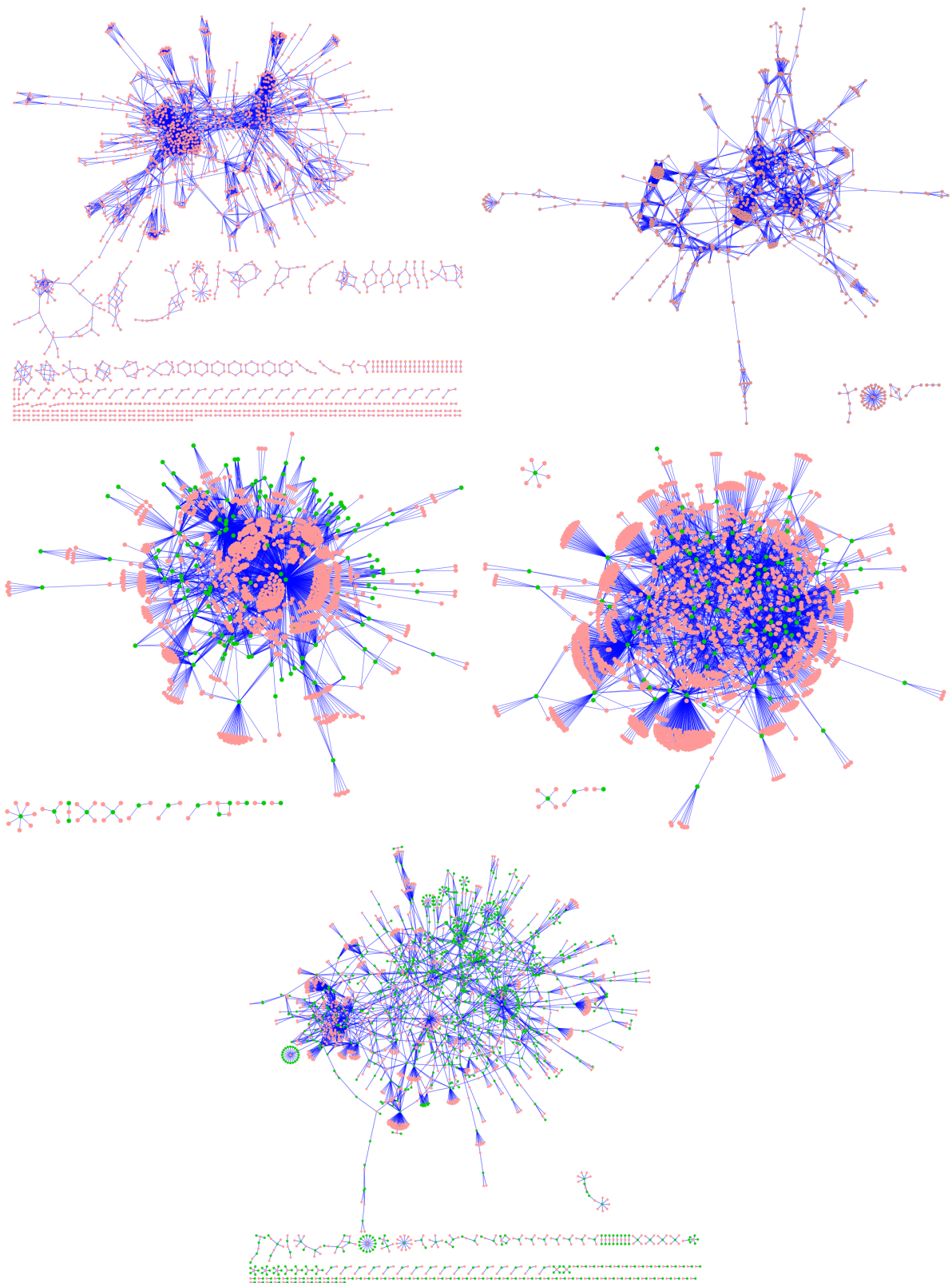


Figure 1.1: Examples of networks that will be discussed in Chapter 4: From top to bottom and left to right, a protein-protein interaction network of yeast *S.cerevisiae*, a metabolic network of yeast, a regulatory network of bacteria *E.coli*, a regulatory network of yeast and a drug-protein interaction network of human.

- Protein-protein interaction networks, in which the nodes are proteins and two proteins are connected by an undirected edge if they physically interact. Variants of this concept introduce an edge when two proteins belong to a same protein complex, or if they belong to a same biological pathway.
- Metabolic networks, where nodes are metabolites and each directed edge represents a biochemical reaction that transforms one metabolite into another and is labeled by the enzymatic protein that catalyzes this reaction.
- Gene regulatory networks, composed of two sets of nodes, genes and regulatory entities such as, e.g., transcription factors (i.e., proteins) or miRNAs. Here, a directed edge from a regulator to a gene indicates that this regulator regulates the expression of a given gene.
- Networks of drug-protein interactions, where interactions for instance denote the affinity of a drug to a given protein.

The mapping of the biological networks relative to an organism of interest is very crucial if one wants to understand the functioning of this organism. The elucidation of these networks is therefore one of the main goals and challenges of systems biology.

In theory, most of these networks, but not all, can be identified from lab experiments. In practice however, these experiments can be extremely difficult to set up and be very costly. These difficulties, together with the very large number of potential pairs to be tested, prevent the application of experimental techniques in a high-throughput manner to elucidate complete networks. In addition, the results of these experiments are most of time very noisy and plagued by missing values. As a consequence, the knowledge we have about most of these networks is very partial and can also be very noisy. On the other hand, more and more experimental data become nevertheless available about the biological entities involved in these networks (genes, proteins, drugs, etc.). Motivated by the availability of these data, several computational approaches have emerged in the literature that try to infer biological networks by integrating relevant data sources. These methods complement nicely experimental techniques.

Network inference methods can be classified into two main families: unsupervised and supervised methods (Maetschke *et al.*, 2013). In order to predict interactions, unsupervised inference methods generally derive a score expressing the confidence for a pair of nodes to interact, based on the analysis of some experimental data related to the nodes (e.g., expression measurements for genes, PFAM domains or phylogenetic profiles for proteins, chemical structure descriptors for drugs). More sophisticated approaches derive a mathematical model of the interactions at play in the network and learn the parameters of this model, that include the underlying network structure, from available experimental data. While effective, unsupervised methods nevertheless require some strong prior understanding of what defines an interaction, in order to design accurate scores or models. They are also intrinsically network specific, as the same score or model can obviously not be used to predict different kinds of interactions.

In contrast with unsupervised methods, supervised approaches provide a generic solution that can handle in principle any kinds of networks. In addition to node-related measurements, these approaches requires some partial knowledge of the network to predict in the form of a training sample of known interacting and non-interacting pairs of nodes. They then rely on some supervised learning algorithm to automatically construct a model that can subsequently be exploited to identify potentially interacting pairs among the remaining unknown pairs in the network. This model computes

its predictions from a set of features defined either on each network node or directly on pairs of nodes. In addition to being generic, by taking advantage of known interactions, supervised methods typically perform better than unsupervised ones. They can in principle be applied to infer any kind of networks or graphs: homogeneous, bipartite, directed or undirected, weighted or unweighted. Supervised methods have been applied to predict several biological networks: protein-protein interaction networks (Park and Marcotte, 2011; Tastan *et al.*, 2009; Yip and Gerstein, 2008), metabolic networks (Yamanishi and Vert, 2005; Bleakley *et al.*, 2007; Geurts *et al.*, 2007), gene regulatory networks (Cerulo *et al.*, 2010; Mordelet and Vert, 2008), epistatic gene networks (Ryan *et al.*, 2010; Ulitsky *et al.*, 2009), drug-protein interaction networks (Takarabe *et al.*, 2012; Cheng *et al.*, 2012; Yu *et al.*, 2012; Bleakley and Yamanishi, 2009; Yamanishi *et al.*, 2008).

1.2 Contributions

In this thesis, we focus on the problem of the supervised inference of biological networks. Our contributions in this domain are structured around three main questions that have not been answered so far in the literature, or only partially:

- How to **evaluate** supervised methods for the inference of biological networks in a fair and unbiased way and, as a corollary, how to best apply them to truly infer a real network?
- How to best exploit **tree-based ensemble methods** for supervised network inference and how do these methods compare with existing methods from the literature?
- How well can we predict **genetic interactions in yeast** using supervised network inference methods?

We motivate these three questions below in turn and detail our main contributions in the process of answering them.

1.2.1 Performance evaluation

The problem of the validation of standard supervised learning methods has been discussed extensively in the literature. Because both training and validation need to be performed on the basis of the same learning sample of labeled data, special care is needed to avoid any source of bias in the validation step. This implies for example the use of intricate and potentially nested cross-validation rounds for tuning method parameters, perform model selection, and assess the performance of the final model.

The problem of supervised network inference can be casted as a supervised classification problem on pairs of objects and thus the validation of inference methods requires the same special care as standard classification methods. However, since pairs, and not single objects, need to be classified, the traditional cross-validation techniques are not sufficient to get a fair and comprehensive assessment of the performance one can expect from these methods in practical applications. In particular, as shown by some authors, the quality of the prediction for a given pair strongly depends on the amount of labeled pairs in the training data that involves its two nodes. It is indeed typically much more difficult to predict pairs that involve nodes not represented in the training network. In consequence, one needs to adapt cross-validation techniques to account for this effect. In addition, several distinctive features of biological networks also impact the way supervised network inference methods should be evaluated or applied. These include for example their high sparsity (that makes

the corresponding classification problem extremely imbalanced), the lack of true negative examples (as it is merely impossible to measure the absence of an interaction), as well as particular topological properties of biological networks (e.g., their scale-free nature).

Although these different biases and difficulties have already been sporadically highlighted and discussed in the literature on biological network inference, our goal in this thesis is to provide a comprehensive and in-depth review and discussion of these problems and from this discussion to derive specific guidelines so as to how best exploit machine learning techniques for network inference.

Our main contributions in this part of the thesis are as follows:

- The problem of supervised network inference is formalized as a problem of classification on pairs. This general formalization encompasses most types of networks that are met in biology (homogeneous or bipartite, directed or undirected, weighted or unweighted). (*Section 2.3.1*)
- We provide a critical review of the different metrics that have been used in the literature to quantify the performance of network inference methods. From this review, we highlight the most appropriate metrics according to different application scenarios. (*Section 3.2*)
- We clearly highlight the differences in model performance according to the amount of information available in the training data about the nodes in the tested pair. We then explain how cross-validation techniques should be adapted to assess this effect. (*Section 3.3*)
- We discuss the impact of the lack of negative interactions in the training data, both at the training and at the validation stage. In particular, we show how to correct performance metrics in order to get a more realistic assessment of the model performance in such settings. (*Section 3.4*)
- We show that the heavy-tailed degree distribution often met in biological networks introduces a positive bias in the evaluation of methods. We show how to adapt baselines for supervised network inference in order not to overestimate the capability of inference methods to extract meaningful information from the input features. (*Section 3.5*)
- We propose a simple algorithm to merge several independent rankings of disjoint sets of pairs from estimated precision-recall curves for each of these ranking. We prove that this algorithm optimizes the precision-recall curve of the merged ranking when precision-recall curves of the individual rankings are perfect estimates of the true curves. (*Section 3.6*)

These different analyses are supported by *in silico* experiments on artificial (gene regulatory and co-expression) networks.

1.2.2 Supervised network inference with tree-based methods

As already mentioned, network inference consists in learning a classifier on pairs of nodes. Mainly two approaches have been investigated in the literature to adapt existing classification methods for this problem (Vert, 2010). The first one, called the global approach, considers this problem as a standard classification problem on an input feature vector obtained by combining the feature vectors of each node from the pair. The second approach, called local, trains a different classifier for each node separately, aiming at predicting its direct neighbors in the graph. These two approaches have been mainly exploited with support vector machine (SVM) classifiers, in part because several efficient kernels have been proposed to compare pairs of objects.

In this thesis, we would like to systematically investigate, theoretically and empirically, the exploitation of tree-based methods (Breiman *et al.*, 1984; Breiman, 2001; Geurts *et al.*, 2006a) in the context of the local and global approaches for supervised biological network inference. The choice of this particular family of methods is motivated by several interesting properties of these methods. They are non-parametric methods that can deal with a large number of features and samples and do not make any assumption (e.g., linearity) about the input-output relationship. Their predictive performance, when used within ensembles, is competitive with the best classification methods (Fernández-Delgado *et al.*, 2014). They can be directly adapted to handle a vectorial or a kernelized output space, which allows to propose alternative solutions for network inference, as shown later in this thesis. Last but not least, they can also provide interpretable information about the input-output relationship, either directly through the tree structure (in case of single trees) or in the form of a ranking of the input features from the most to the least relevant (in case of ensembles). This latter property is very interesting in the context of biological network inference, as it means that these methods can provide potentially useful information about the mechanisms that underly the interactions.

Although tree-based methods have already been used for the inference of specific biological networks, how to best exploit them for network inference and how well they perform in comparison with other methods on various biological networks are still open questions. Our main contributions in this part of the thesis are as follows:

- We formalize the local and global approaches and in the process, we extend the local approach along two lines. First, we propose an original two-step procedure that allows with this method to make predictions for pairs of nodes that have no known connections in the training network. Second, we propose to use in this context multi-output methods, which drastically reduces the requirement in terms of model size of the local approach (with no significant loss in terms of predictive performance with tree-based methods). (*Section 4.2*)
- We study in details the specialization of the local and global approaches to tree-based methods. In particular, we discuss implementation and computational complexity issues. We also draw some interesting connections between these methods and (bi-)clustering techniques that shed some light on the assumptions behind these methods when applied for network inference. (*Section 4.3*)
- We provide a systematic and large-scale evaluation of these methods on several homogeneous and bipartite networks taken from the literature. The goal of this evaluation is to compare the different methodological variants proposed in the thesis, as well as to compare tree-based methods with other methods from the literature. To the best of our knowledge, no previous study has considered simultaneously as many of these networks. (*Section 4.4*)
- Finally, we propose several semi-automatic procedures to exploit tree-based methods for extracting interpretable explanations about their predictions. These procedures are illustrated on a drug-protein interaction network. (*Section 4.5*)

1.2.3 Predicting genetic interactions in Yeast

An important type of biological network is genetic interaction networks. There is an interaction (epistasis) between two genes if the effect of the mutation of one of the genes is modified, negatively

or positively, by a mutation of the other gene. The knowledge of these interactions is crucial to understand the functions of genes and their products and more globally to elucidate the functional and organizational principles of biological systems. The information that can be extracted from these interactions however strongly depends on the global knowledge of the interaction network, and not only on the knowledge of some specific individual interactions. Experimental techniques exist that allow to measure these interactions (or absence thereof) and, in yeast *S.cerevisiae* about 4 millions pairs of genes have already been experimentally tested for an interaction in the context of 11 different studies. Although this number is already very impressive, these 4 millions pairs nevertheless only cover about 10% of all the pairs that can be defined from the ~ 6000 yeast genes. The use of computational inference techniques is thus very interesting to complete these experimentally confirmed interactions.

In this last part of the thesis, our goal is to first see how well it is possible to complete the genetic interaction network of the yeast *S.cerevisiae* using supervised network inference methods, and second to find the best way to apply supervised inference methods to actually infer this network. Our steps to achieve this goal are as follows:

- To constitute our training set for the inference, we collected and pre-processed a very large set of about 4 millions experimentally measured gene pairs (collected from 11 different studies carried out between 2005 and 2013) and for each Yeast gene a vector of more than 11,000 features (from 22 different sets) to describe them. (*Section 5.3*)
- On this dataset, we carried out several cross-validation experiments with three main goals: (1) evaluate the performance of inference methods, considering separately positive and negative interactions, (2) assess the relevance of the different feature sets individually and in combination, and (3) compare the performance of computational methods with those of experimental techniques. For this latter step, we exploited the fact that several experimental results are available for a significant number of gene pairs, given the overlap between the different studies. (*Section 5.4*)
- Finally, based on the experience gained in the rest of the thesis, we applied at best network inference techniques to get a global prediction of the complete genetic interaction network of the Yeast. We assessed the relevance of the predicted interactions by measuring the enrichment of the interacting pairs in genes that share similar function. (*Section 5.5*)

1.3 Outline of the manuscript

The main body of the manuscript is divided into four chapters.

In Chapter 2, we provide the reader with the required background in terms of biological networks (Section 2.1), machine learning (Section 2.2), and network inference methods, supervised (Section 2.3) and unsupervised (Section 2.4).

Chapter 3 is devoted to our study of the evaluation of supervised network inference methods. We first present in Section 3.2 common evaluation measures used in the context of biological network inference and discuss their main advantages and drawbacks. We then talk in Section 3.3 about evaluation protocols that allow to differentiate the predicted pairs according to the number of its nodes that are covered in the training network. We analyze the main characteristics of biological networks that may impact the performance evaluation of the methods, namely the lack of negative

examples (Section 3.4) and the heavy-tailed node degree distribution (Section 3.5). Finally, we describe in Section 3.6 the new approach that we developed to merge pairs predicted with different models and associated to different performance levels.

In Chapter 4, we explore the use of tree-based methods in the context of the local and global approaches for supervised biological network inference. Local and global approaches are detailed in Section 4.2 and then particularized for tree-based methods in Section 4.3. Experiments are carried out with these methods on 10 different networks in Section 4.4, where tree-based methods are also compared with other methods from the literature. Finally, the interpretability of these methods is illustrated in Section 4.5 by several experiments on a drug-protein interaction network.

In Chapter 5, we provide an in-depth application of supervised inference methods for the prediction of genetic interactions in yeast *S.cerevisiae*. After a presentation of the necessary background about genetic interactions in Section 5.2, we present the training data, both the known set of interactions and the gene input features, in Section 5.3. Various cross-validation experiments are reported in Section 5.4. Section 5.5 concludes this chapter with an application of the best identified methods and feature sets to infer the complete genetic network of yeast.

The thesis ends in Chapter 6 with a discussion of our main findings and several suggestions of future research directions.

1.4 Publications and dissemination of the results

Chapter 3 is an extended version of the following published paper:

- (Schrynemackers *et al.*, 2013) *On protocols and measures for the validation of supervised methods for the inference of biological networks*, Marie Schrynemackers, Robert Küffner, and Pierre Geurts. *Frontiers in Genetics*, 4(262).

With respect to this latter paper, Section 3.6 about the merging of several rankings is new and unpublished.

Chapter 4 is based on the following manuscript available on arXiv.org and submitted for publication:

- (Schrynemackers *et al.*, 2014) *Classifying pairs with trees for supervised biological network inference*, Marie Schrynemackers, Louis Wehenkel, M. Madan Babu, and Pierre Geurts. CoRR, abs/1404.6074.

Section 4.5 about the interpretability is however not part of this manuscript. Results of this chapter have furthermore been presented at the following international conferences:

- UGR meeting, Université de la Grande Région, Lultzhausen (Luxembourg), June 19-20 2012 (*Oral presentation*)
- MLCB 2012, workshop on Machine Learning in Computational Biology, at the twenty-sixth annual conference on Neural Information Processing Systems (NIPS 2012), Lake Tahoe (USA), December 7 2012 (*Poster presentation*)

Results in Chapter 5 are new and unpublished. The idea of inferring genetic interaction in yeast and preliminary results on much limited data have been presented at the following conferences:

- BBC 2009, the fifth Benelux Bioinformatics Conference, Liège, December 14-15 2009
(1 page abstract in the conference proceedings and oral presentation)
- MLSB 2010, the fourth international workshop on Machine Learning in Systems Biology, at the eleventh International Conference on Systems Biology (ICSB 2010), Edinburgh (UK), October 15-16 2010 *(Poster presentation)*

Chapter 2

Background

This thesis focuses on the problem of the supervised inference of biological networks. In this chapter, we provide the required background in terms of biological networks, supervised learning and network inference. In Section 2.1, we first informally define biological networks, which are any networks that relate to biological systems, and expose their main representatives in the literature on network inference. We then introduce supervised learning in Section 2.2. The general goal of supervised learning methods is to infer input-output relationship from a training sample of input-output pairs. Among these methods, we describe in details decision trees and random forests methods, that are at the heart of the approaches developed in the thesis. We also present in some details support vector machines, that have been extensively used for network inference and with which we will compare our results. In Section 2.3, we define formally the problem of supervised network inference and introduce the notations that will be used throughout the thesis. We then review the most representative works in this domain. Finally, while the thesis focuses on supervised techniques, Section 2.4 discusses unsupervised and semi-supervised approaches to this problem.

Contents

2.1	Biological networks	24
2.2	Supervised learning	26
2.3	Supervised network inference	37
2.4	Unsupervised and semi-supervised biological network inference	42
2.5	Discussion	44

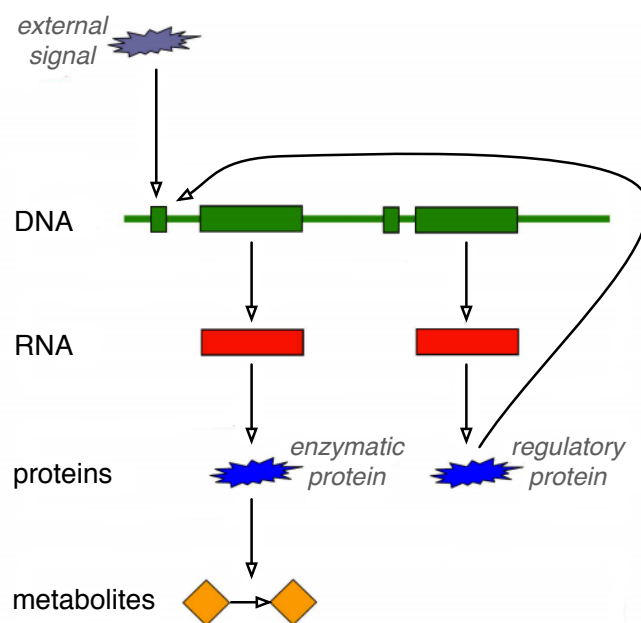


Figure 2.1: Central dogma of molecular biology: genes (DNA sequence) are transcribed into RNA sequence, which is translated into proteins. Figure modified from Bachmaier *et al.* (2013)

2.1 Biological networks

As discussed in the introduction, networks are ubiquitous in biology to represent relationships between biological entities such as genes, proteins, or enzymes. Mathematically, a network or graph¹ is defined as set V of objects, called nodes, and a set E of edges, i.e., pairs of nodes that are connected in the graph. An edge may be directed if it links one node towards another, or undirected if it does not differentiate the nodes it connects. It can also be weighted by a number indicating its strength. A graph is said to be bipartite if its nodes can be divided into two sets V_1 and V_2 such that each edge connect one node in V_1 to one node in V_2 .

A biological network is any network that applies to biological systems. The semantics associated to the nodes and edges in biological networks can be very diverse. The main players of most of these networks are however biochemical compounds such as proteins or genes, whose behavior in living systems is mostly governed by the famous central dogma of molecular biology. The most prominent biological networks are defined in relation with this central dogma and it is therefore useful to briefly remind it here. In a very simplified view, this dogma expresses that genes, encoded in DNA sequences, are transcribed into RNA sequences which are themselves translated into proteins (see **Figure 2.1** for a schematic representation). Proteins are macromolecules which are very important for the life of the cell, and which serve various functions. For example, some proteins are involved in structural support and maintain cell shape, others are involved in interactions with the outside world.

¹Although some authors make a difference between a network and a graph, we will use these two terms indistinctly in this thesis.

To perform their functions, proteins are typically not acting alone but rather in strong interactions with other proteins. Among the most studied biological networks, we have the so-called *protein-protein interaction network*. Nodes in this network are proteins and two proteins can be connected, with an undirected edge, for several reasons (Qi *et al.*, 2006): first, if they directly and physically interact or bind with each other, second if they belong to a same protein complex (they do not directly interact but are connected through other proteins), and third, in a more abstract way, if they belong to a same biological pathway (ie., if they participate to a common specific biological function).

Several important networks represent the interactions of specific proteins with other biochemical compounds. A first type of proteins of interest are enzymatic proteins. These proteins catalyze biochemical reactions that occur in cells (**Figure 2.1**). These biochemical reactions are collected into the so-called *metabolic network*. This network can either be seen as a directed network that connects two chemical compounds, e.g. metabolites, if the first is transformed into the second by a biochemical reaction. Each edge is then labeled by the enzymatic protein that catalyzes the corresponding reaction. In the context of network inference, the metabolic network is often simplified into an undirected network between enzymes, where two enzymes are connected if they catalyze two successive reactions in a metabolic pathway (Yamanishi and Vert, 2005). A second type of proteins of interest are regulatory proteins. These proteins bind to specific regulatory sequence of DNA, act to switch genes on and off and thereby regulate the transcription of genes (**Figure 2.1**). These regulations are compiled into *gene regulatory networks*. These networks are bipartite, with one set of nodes corresponding to genes and the other set of nodes to regulatory proteins. There is an edge between a protein and a gene if the former regulates the expression of the latter. The protein is then called a transcription factor for the gene. This network can be equivalently seen as directed network between genes, where there is a link from one gene to another if the first gene encodes a protein that regulates the expression of the second gene. A simplified view of the same network is the *gene co-regulation network* that connects two genes with an undirected edge, if they share some transcription factors. Proteins can also interact with drugs, with drugs modulating the function of proteins. These interactions play a key role in the context of drug discovery and they are compiled into *drug-protein interaction networks*. These networks are bipartite, where nodes from the first set are proteins and nodes from the second set are drugs.

Another kind of networks that is relative to genes are *epistatic networks*. Nodes in this network are genes and there is an undirected interaction between two genes when the presence or absence of one gene modifies the effect of another gene, positively or negatively, in terms of some phenotype (e.g., cell fitness). More precisely, a cell is mutated by deleting a gene g_a and another one is mutated by deleting a gene g_b . The two cell fitnesses are measured and from these measures, an estimation of the fitness of a third mutated cell, where both genes g_a et g_b would be deleted, is computed. If the observed fitness in this doubly mutated cell is (significantly) lower than the estimated fitness, then we consider the interaction between genes g_a and g_b to be negative, and in the contrary, we consider it to be positive. More details about this type of networks and the way these interactions are actually measured will be provided in chapter 5.

As a last example, biological networks can also be more abstract and less directly related to physical interactions between biochemical compounds. An example of network in this category can be found from the *OMIM* (Online Mendelian Inheritance in Man) compendium (McKusick, 1998). This network is a bipartite graph where the first set of nodes represents human genes and the second

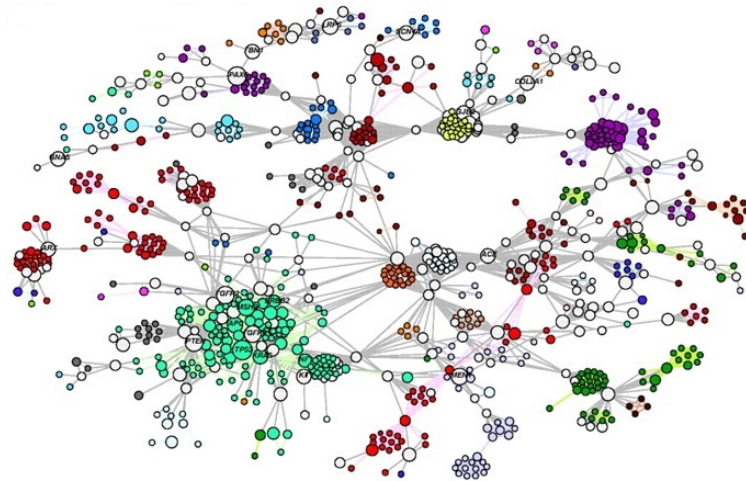


Figure 2.2: Example of biological network: OMIM links a human gene and a disease if a relationship has been found between them. Figure taken from Goh *et al.* (2007).

set of nodes represents phenotypes (e.g. diseases). A gene and a phenotype are linked by an edge in the network if a relationship between them have been highlighted by biologists (**Figure 2.2**).

2.2 Supervised learning

Machine learning is a branch of artificial intelligence which focuses on learning models from data. Supervised learning is the dominant methodology in machine learning (Cunningham *et al.*, 2008), in which a model is trained from labeled data (e.g., by human) that can then be used to predict the label of new data (**Figure 2.3**). More formally, the goal of supervised learning can be defined as follows:

Given a learning sample LS composed of input-output pairs:

$$LS = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y})^N$$

such that $\mathbf{x}_i \in \mathcal{X}$ is the input feature vector of the i -th example and $y_i \in \mathcal{Y}$ is its label, find a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that best approximates the unknown labels of new input vectors.

Typically, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$ or \mathcal{Y} is discrete. If the output is quantitative, the problem is called a regression problem and if the output is qualitative it is called a classification problem. In a binary classification problem, typically $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$.

To assess the quality of the predictions of a model, we need a notion of error that measures the difference between a predicted label $f(\mathbf{x})$ and the true label y for a given input \mathbf{x} . This error is defined through a loss function $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ that quantifies the discrepancy between two labels and can be used to assess prediction errors. The choice of the loss function depends on the learning problem being solved. A typical loss function for regression problems is the squared error loss function: $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ and a typical loss function for classification problems is the 0-1 loss function: $L(y, f(\mathbf{x})) = \mathbf{1}(y \neq f(\mathbf{x}))$, where $\mathbf{1}$ is the indicator notation.

Inputs				Output
X_1	X_2	X_3	X_4	Y
1.0	0.1	0.8	0.0	TRUE
0.5	0.4	1.0	0.8	FALSE
0.8	0.9	0.7	0.9	TRUE
\vdots				\vdots
0.7	0.8	0.7	0.4	?

Supervised learning
 $\rightarrow Y = f(X_1, X_2, X_3, X_4)$

Figure 2.3: Supervised learning is the machine learning task of inferring a function from labeled training data. In other words, it consists in using a set of input-output pairs to estimate a function that can predict the output associated to new inputs

The loss function L leads to the definition of the *risk* for a function f , also called the *generalization error*, which is the expected value of the loss function over the joint distribution of input-output pairs: $R(f) = \mathbb{E}[L(y, f(\mathbf{x}))]$. The risk can be estimated empirically from the learning set (or any other set) as:

$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(\mathbf{x}_i)).$$

The learning problem is then expressed as finding a classifier f in a given set \mathcal{F} , called the hypothesis space, that minimizes the empirical risk \hat{R} .

Most learning algorithms depend on a *complexity parameter* that determines the size of the hypothesis space. This parameter needs to be tuned to reach optimal performance. Indeed, if the model is too simple, it will not capture the regularities of the data and will be inaccurate (high bias). The model is then said to *underfit* the data. On the contrary, if the model is too complex, it will capture all regularities in the training data and will lead to a small empirical risk. But it will also fit the noise or the randomness of the learning set and will not generalize well to new data (high variance). The model is then said to *overfit* the data. The problem of simultaneously minimizing the bias and the variance (by tuning the complexity parameter of the model) is called the *bias-variance tradeoff*.

Given this tradeoff, a typical application of supervised learning techniques on a given dataset involves the following steps, based on cross-validation. First, the data is randomly partitioned into three parts (assuming that the size of the dataset is large enough):



Several models are learned on the training set by changing the value of the complexity parameter. Each of these models is assessed on the validation set and the complexity value that corresponds to the best model is used to retrain a final model on the union of the training and validation sets. The error of this final model is then estimated on the test set. When data is parsimonious, more reliable error estimates can be obtained at all steps by averaging results over several random data splits.

Many algorithms were developed to solve supervised learning problems. Among the most popular ones, we find naive bayes classifiers, k -nearest neighbors, support vector machines, neural networks, and decision trees. For more information about them, see e.g. the following references (Hastie *et al.*, 2001; Bishop, 2006; Murphy, 2012). In this work, we mainly exploit decision tree algorithms

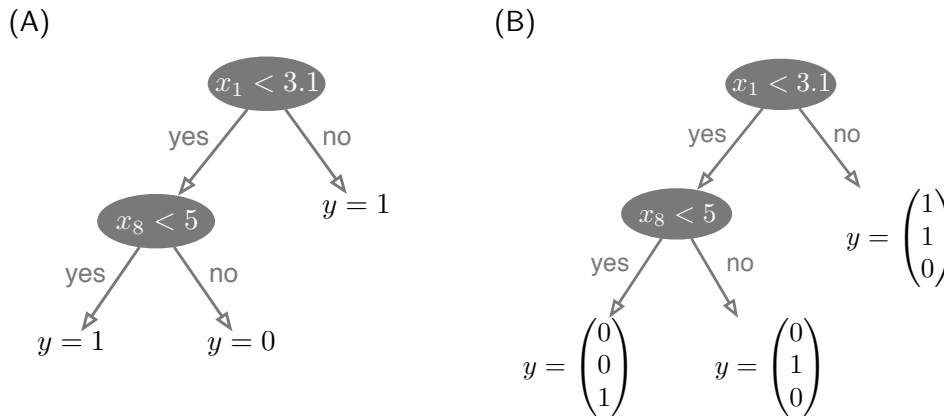


Figure 2.4: The single-output classification tree (A) and the multi-output classification tree (B) represent input-output models

and tree-based ensemble methods. They are thus presented in details respectively in Sections 2.2.1 and 2.2.2. Section 2.2.3 also presents kernel-based support vector machines. These methods have been extensively used for supervised network inference, and they will be compared with tree-based ensemble methods in Chapter 4. Finally, we describe the output kernel tree algorithm in Section 2.2.4, which uses ensemble of trees and which will be also compared with our methods.

2.2.1 Decision trees

The decision tree method (Breiman *et al.*, 1984; Quinlan, 1993) is a very popular supervised learning algorithm that can handle both classification and regression problems. Its main advantages with respect to other methods are the interpretability of the models that it produces and its non-parametric nature. We describe successively the hypothesis space explored by this method, its learning algorithm, and a generalization to handle multi-outputs. At the end of the section, we discuss interpretability and tree-derived feature importance scores.

Hypothesis space. In the decision tree method, input-output models are represented by trees. Each interior node of a tree is labeled with a test defined from the input features and each successor branch of this node corresponds to a potential issue of the test. Typically, tree tests are binary and based on one feature at a time. In the case of a numerical feature, they compare the value of a feature to a discretization threshold, also called cut-point. Each terminal node or leaf is labeled with a value of the output, i.e., a class (or a class probability distribution) in the case of classification, or a number, in the case of regression (see **Figure 2.4A** for an example). To predict the output given some input value \mathbf{x} , the tree is traversed from the root node to a tree leaf by choosing the branches according to the tree tests given the input x . The label associated to the leaf is then retrieved and provided as the final output prediction for this x .

Learning stage. The tree is grown in a top-down recursive manner. Initially, the root of the tree is associated with the N examples of the learning set: $S = \{(\mathbf{x}_i, y_i) \text{ with } i = 1, 2, \dots, N\}$. A new test $X_j \leq s$, i.e. a feature $j \in \{1, \dots, d\}$ and a cut-point s , is selected on the basis of the learning set S to split this node into a left and right child. The left child is associated with the subset S_l of

examples from S such that $X_j \leq s$ and the right child is associated with the subset S_r of examples from S such that $X_j > s$. The same process is then repeated recursively to develop further each of the two new successors nodes and their associated training sets. We stop splitting a node when either the output or all features are constant for the examples within this node or when some stop splitting criterion is satisfied. Several such criteria have been defined in the literature (Breiman *et al.*, 1984). In this thesis, we will only stop splitting a node when the number of examples in this node is less than a given pre-defined parameter n_{min} . Finally, terminal nodes, or *leaves*, are labeled according to the examples they contain and to the loss function. In the case of regression, a leaf is often labeled with the average output value of the examples that reach this leaf, as this is the output that minimizes square loss. In the case of classification, a leaf is often labeled with the majority class among the examples that reach this leaf, as this is the output that minimizes 0-1 loss.

At each node split, the pair of feature j and cut-point s must be selected so that the outputs in each of the two children are as similar as possible. More formally, an *impurity* function $Q(\cdot)$ is defined that measures how similar the outputs are in a given subset S of examples and the optimal test is defined as the pair (j^*, s^*) that maximizes the average impurity reduction brought by the split:

$$\begin{aligned} (j^*, s^*) &= \max_{(j,s)} \left[Q(S) - \frac{|S_l|}{|S|} \cdot Q(S_l) + \frac{|S_r|}{|S|} \cdot Q(S_r) \right] \\ &= \min_{(j,s)} [|S_l| \cdot Q(S_l) + |S_r| \cdot Q(S_r)], \end{aligned} \quad (2.1)$$

where $|\cdot|$ denotes the cardinality of a subset. For a regression problem, the impurity is typically measured by the variance of y :

$$Q(S) = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} (y_i - \bar{y})^2 \quad \text{with} \quad \bar{y} = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} y_i \quad (2.2)$$

For a classification problem, different measures of impurity exist, including the Gini index:

$$Q(S) = \sum_{c \in C} p_c(S)(1 - p_c(S)) \quad (2.3)$$

and the log entropy:

$$Q(S) = - \sum_{c \in C} p_c(S) \log p_c(S) \quad (2.4)$$

where C is the set of all possible classes and $p_c(S)$ is the proportion of objects in S that are labeled with the class c .

Computationally, (2.1) is solved by scanning all features and all possible cut-points (at most $N - 1$ when S contains N examples). When implemented properly, the computational complexity of the tree construction algorithm as described in this section can be shown to be $O(dN \log N)$ in average, where N is the size of the learning sample and d is the number of features (Louppe, 2014). When we have chosen the impurity measure, the only parameters of the method is the parameter n_{min} that adjusts the complexity of the tree. Fixing it to a too small value would lead to a large tree that would overfit the data, while a too large value would lead, on the contrary, to underfitting. Post-pruning algorithms also exist to automatically determine the tree complexity by using cross-validation techniques (Breiman *et al.*, 1984).

Multi-output decision trees. Beyond standard classification and regression settings, decision trees can be extended in a straightforward way to predict a vectorial output $y \in \mathcal{Y} = \mathbb{R}^l$ (Blockeel *et al.*, 1998). With respect to the algorithm described above, only the leaf labeling and the impurity function need to be adapted. Labels at tree leaves become vectors from \mathbb{R}^l (see **Figure 2.4B**) and they are computed as the average (or center of mass) of the output vectors in the subset S of training examples that fall into the leaf:

$$\frac{1}{|S|} \sum_{(x_i, y_i) \in S} \mathbf{y}_i. \quad (2.5)$$

The impurity function Q used to determine the optimal test at each tree node can be defined as the average squared euclidean distance between each output vector and the center of mass:

$$Q(S) = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} \|\mathbf{y}_i - \bar{\mathbf{y}}\|^2 \quad \text{with } \bar{\mathbf{y}} = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} \mathbf{y}_i \quad (2.6)$$

This latter expression is also equivalent to the average of the impurity in (2.2) over all outputs.

Interpretability and feature importances. In addition to their simplicity, one important advantage of decision trees with respect to other methods is their interpretability (Hastie *et al.*, 2001). A tree recursively partitions the input space into rectangular regions that are each described by a conjunction of tests based on the input features that can be directly read from the tree.

Moreover, a tree allows to quantitatively assess the relative importances or contributions of the different variables in the prediction of the output. Breiman *et al.* (1984) proposed to measure the importance of a variable by the weighted sum of the impurity reductions due to the splits at all nodes of the tree where this variable is tested. More precisely, denoting by $\Delta Q(\mathcal{N})$ the impurity reduction at node \mathcal{N} :

$$\Delta Q(\mathcal{N}) = Q(S_{\mathcal{N}}) - \frac{|S_{\mathcal{N},l}|}{|S_{\mathcal{N}}|} \cdot Q(S_{\mathcal{N},l}) + \frac{|S_{\mathcal{N},r}|}{|S_{\mathcal{N}}|} \cdot Q(S_{\mathcal{N},r}), \quad (2.7)$$

where $S_{\mathcal{N}}$, $S_{\mathcal{N},l}$ and $S_{\mathcal{N},r}$ are the set of examples associated respectively to \mathcal{N} and its left and right children. The importance of a variable X_i in a tree \mathcal{T} is then computed as:

$$I(X_i, \mathcal{T}) = \sum_{\mathcal{N} \in \mathcal{T} | v(\mathcal{N}) = X_i} \frac{|S_{\mathcal{N}}|}{|S_{\mathcal{T}}|} \Delta Q(\mathcal{N}) \quad (2.8)$$

where \mathcal{T} also denotes the set of interior nodes in the tree, S is the training set from which the tree was grown, and $v(\mathcal{N})$ is the variable tested at node \mathcal{N} .

2.2.2 Decision tree ensembles

A major drawback of decision trees is their very high variance (Geurts, 2002). A small change in the training data can greatly modify the tree, in part because a modification of a test at some node in the tree will have an impact on all its descendants. As a consequence of this high variance, the accuracy of decision trees is typically below that of other methods.

An efficient way to improve decision trees by reducing their variance is to use them in the context of ensemble methods. The main idea of ensemble methods is to aggregate the predictions of several individual models, either by averaging (in the case of regression) or by a majority vote (in the case of

classification). There exist in the literature several techniques to generate an ensemble of models to be aggregated (see Hastie *et al.* (2001) for a review). In this thesis, we will focus on methods that aggregate models generated independently of each other by introducing appropriate randomization into the learning procedure. These methods improve generalization essentially by reducing the variance with respect to the original non-randomized method. When applied to decision trees, these methods are commonly referred to as Random Forests techniques (Breiman, 2001).

The theory (Hastie *et al.*, 2001; Louppe, 2014) show that efficient variance reductions are obtained with ensembles by combining very diverse models, i.e. models whose errors are as much as possible uncorrelated. Several random forests techniques have been proposed in the literature to generate randomized ensembles of decision trees. We expose here below three popular approaches, that randomize the training set, the features that are searched at each tree node, and/or the cut-points for numerical features. These three methods can be applied whatever the nature of the output (classification, regression, or multi-outputs).

Bagging

In bagging (the acronym of *bootstrap aggregating*) (Breiman, 1996), diversity comes from the sampling of the objects of the learning set. Given a learning set LS of size N , bagging generates M replicate data sets, each consisting of N objects, drawn at random *with replacement* from LS . This kind of sample is known as a bootstrap sample.

Random Forests

Random Forests denote both generic tree-based ensemble methods based on randomization and a particular instance of this framework proposed by Breiman (2001). In this latter instance, diversity comes from bootstrap sampling, like in bagging, and from a random selection of the features at each tree node. More precisely, each tree is trained from a different subset of training objects obtained by bootstrapping, and for each node in each tree, the best split is searched from a randomly chosen (without replacement) set of K features (instead of the full set of d features). For a dataset with d features, $K = \sqrt{d}$ has been shown to be a good general default value, at least in classification (Breiman, 2001; Geurts *et al.*, 2006a). In bagging, correlation between the trees can happen when few features are strong predictors and are selected in many trees. In such case, the extra level of randomization introduced in Random Forest decorrelates the trees.

Extra-trees

In the Extra-Trees algorithm (acronym for *extremely randomized trees*) (Geurts *et al.*, 2006a), there is no bootstrap sampling, and the N objects of the learning set are used to build each tree. Diversity comes from the randomization of the features when splitting each test node, as in Random Forests, to which the authors have added a randomization of the cut-point. Since Extra-Trees is the supervised learning algorithm that will be mainly used throughout this thesis, let us detail the main steps of its node splitting procedure. Let S denote the set of objects corresponding to the node to split:

- K features $\{X_1, \dots, X_K\}$ are randomly selected, among all non-constant (in S) candidates features;

- K splits s_1, \dots, s_K are drawn randomly, where $s_i = [X_i < x_{i,c}]$. Each cut point $x_{i,c}$, $i = 1, \dots, K$, is uniformly drawn in $[x_{i,\min}^S, x_{i,\max}^S]$, where $x_{i,\min}^S$ and $x_{i,\max}^S$ are respectively the minimum and the maximum values of X_i in S .
- A split s_ϕ is selected among them so as to maximize impurity reduction as expressed in (2.1).

A tree node becomes a leaf that will not be split further if one of these three criteria is satisfied:

- $|S| < n_{\min}$,
- all attributes are constant in S ,
- the output is constant in S .

Because of the randomization of the cut-point, the node splitting procedure is much simpler than in Bagging and Random Forests for example, which makes the algorithm more efficient. It compares also very well in terms of accuracy with these two methods (Geurts *et al.*, 2006a; Louppe, 2014). As described, the algorithm depends on three parameters: K , n_{\min} , and M , the number of trees in the ensemble. The default values for these parameters are $K = \sqrt{d}$ (where d is the total number of features), $n_{\min} = 2$ and $M = 100$. Unless otherwise stated, these are the default values that we will use in our experiments throughout the thesis.

Interpretability

With ensembles, the straightforward interpretability that we had with single trees is mostly lost. It is however still possible to compute the relative importance of the features from an ensemble, by averaging the importance scores computed from each tree within the ensemble:

$$I_{ens}(X_i) = \frac{1}{T} \sum_{t=1}^T I(X_i, \mathcal{T}_t), \quad (2.9)$$

where $I(X_i, \mathcal{T}_t)$ for each tree \mathcal{T}_t is computed as in (2.8). The resulting ensemble importance is typically much more stable than the importance of each individual tree. Although originally a heuristic, this importance measure has been shown in (Louppe *et al.*, 2013) to present interesting theoretical properties.

2.2.3 Support vector machines

In this section we describe support vector machines (SVM), introduced by Cortes and Vapnik (1995) to solve binary classification problems. We first consider the linear case and then show how this method can produce a nonlinear boundary by working in a kernelized input space (Hastie *et al.*, 2001)

Large margin separation

Each of the N objects of the learning set is represented by a pair (\mathbf{x}, y) where $y \in \{-1, +1\}$ is a binary label and $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ is a point from \mathbb{R}^d . In \mathbb{R}^d , we can define a linear classifier through a linear function of the following form:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

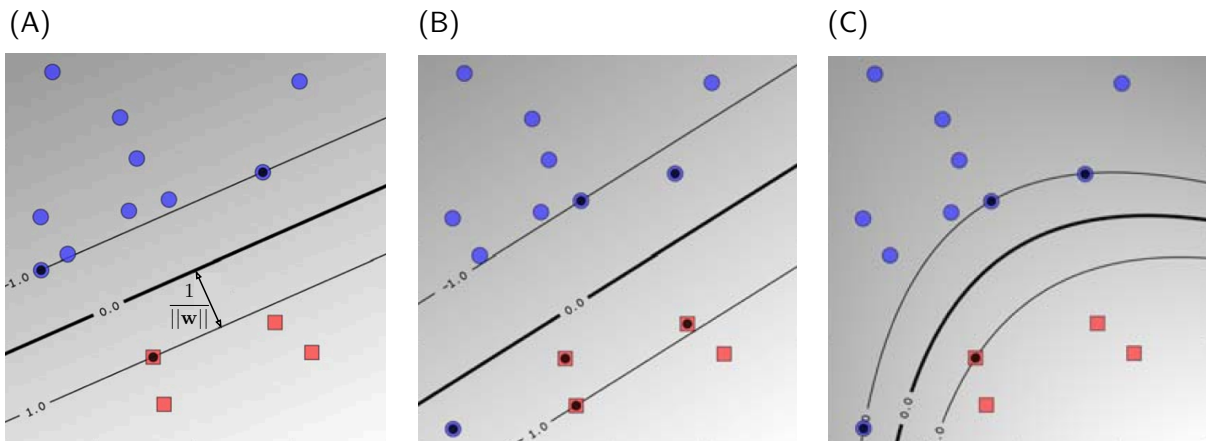


Figure 2.5: (A) The positive and negative objects are linearly separable, and the distance between the boundary and the closest points is equal to $\|\mathbf{w}\|$. This margin will be maximized. (B) By adding some slack variable, some points are allowed to be within the margin or on the wrong side of the boundary. (C) By choosing a polynomial kernel of degree 2, we made the boundary nonlinear. Figure adapted from (Ben-Hur *et al.*, 2008)

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, b is the bias term, and $\langle \cdot, \cdot \rangle$ denotes the scalar product between two vectors. Function f assigns a score to each input vector \mathbf{x} and a classifier can be obtained by taking the sign of this score: predicted class is equal to $+1$ if $f(\mathbf{x}) > 0$ and equal to -1 if $f(\mathbf{x}) < 0$. Such classifier divides the input space into two half spaces, where the separation is carried out by the hyperplane defined by the equation $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. When $d = 2$, this decision boundary corresponds to a line.

When the points of the two classes are linearly separable, there are actually an infinite number of hyperplanes, i.e., an infinite number of combinations (\mathbf{w}, b) , that realize a perfect separation. The idea of the SVM method in this case is to look for an hyperplane, among those that perfectly separate the data, that maximizes the margin, where the margin is defined as the distance of the closest point to the decision boundary. This feels a very safe choice intuitively and it is also supported by theory (Cortes and Vapnik, 1995). To simplify the mathematical formulation of the corresponding optimization problem, one should note first that b can always be chosen such that the hyperplane is half way between the closest positive and negative objects that are called the *support vectors* (the hyperplane would not maximize the margin if it was not the case). In addition, given that the model remains unaffected if both \mathbf{w} and b are rescaled in a similar way, \mathbf{w} can always be scaled such that $f(\mathbf{x}) = \pm 1$ for these support vectors. In this case, the distance between the decision boundary and the closest points, i.e. the margin, is equal to $\frac{1}{\|\mathbf{w}\|}$ (see **Figure 2.5A**).

To find \mathbf{w} and b that maximize the margin, we must then solve the following optimization problem:

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to} && y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, \dots, N. \end{aligned} \quad (2.10)$$

The inequality constraints ensure that $\text{sign}[y_i] = \text{sign}[\langle \mathbf{w}, \mathbf{x}_i \rangle + b]$ (all the objects of the learning set are correctly classified) and that $\|\langle \mathbf{w}, \mathbf{x}_i \rangle + b\| > 1$ (no point lies inside the margin).

When the points of the two classes are not linearly separable, the above optimization problem has no solution (as the constraints can not be all satisfied). In this case, one needs to relax the constraints so that some points are allowed to be on the wrong side of the boundary. This is achieved by introducing *slack variables* ξ_j :

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ & \text{subject to} && y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\ & && \xi_i \geq 0, \forall i. \end{aligned} \quad (2.11)$$

The parameter $C > 0$ is the *cost* that we have to pay for misclassifications. The separable case corresponds to $C = \infty$. This parameter controls some bias-variance tradeoff and therefore needs to be adjusted to the problem at hand (e.g., by cross-validation). Optimization problem (2.11) is called the *soft margin SVM*, while problem (2.10) is called the *hard margin SVM*. Note that the soft margin SVM is interesting also in the case of linearly separable data. It indeed allows to increase further the margin at the expense of misclassifying a few points, which can lead to a better bias-variance tradeoff for some problems.

Problem (2.11) is a convex optimization problem, because the objective function is quadratic with linear inequality constraints. It can therefore be solved using the method of Lagrange multipliers. The Lagrangian for this problem is written as:

$$L(\mathbf{w}, b, \alpha, \mu) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_i \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - (1 - \xi_i)] - \sum_i \mu_i \xi_i \quad (2.12)$$

where α_i and μ_i are the Lagrange multipliers. The dual problem consists in maximizing L subject to $\nabla_{\mathbf{w}, b, \alpha, \mu} L = 0$, $\alpha_i > 0$ and $\xi_i > 0$. Setting the different derivatives to zero, we get

$$\begin{aligned} \mathbf{w} &= \sum_i \alpha_i y_i \mathbf{x}_i \\ \sum_i \alpha_i y_i &= 0 \\ \mu_i &= C - \alpha_i \end{aligned} \quad (2.13)$$

which can be substituted into (2.12) to give the following dual optimization problem:

$$\begin{aligned} & \underset{\alpha_i}{\text{maximize}} && L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ & \text{subject to} && \sum_i y_i \alpha_i = 0, \\ & && 0 \leq \alpha_i \leq C, \forall i. \end{aligned} \quad (2.14)$$

The optimal values of α_i can then be plugged into (2.13) to find the weight vector \mathbf{w} solution to (2.11). In this formulation, each parameter α_i is associated to a learning sample object and the support vectors are the objects such that $\alpha_i > 0$.

Nonlinear classifiers with kernels

While the soft margin SVM is introduced to address non linear separability due to noise, a classification problem can also be non linearly separable because the true decision frontier is non linear.

A straightforward way to learn a non linear decision boundary with the SVM approach is to first map the original data into some new vectorial space by using a non linear function defined on the input space, $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$, and then to fit a linear model in this new space. The score function of the resulting classifier will become $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$, which remains linear with respect to the parameters \mathbf{w} and b but becomes non linear with respect to the inputs \mathbf{x} . For example, a quadratic classifier can be obtained by using the following map:

$$\phi(\mathbf{x}) = [x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_d, \dots, \sqrt{2}x_{d-1}x_d, \sqrt{2}x_1, \dots, \sqrt{2}x_d, 1], \quad (2.15)$$

which incorporates the products of all pairs of features.

Given the mapping ϕ , the Lagrangian from (2.14) becomes

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \quad (2.16)$$

From (2.13), the classification function becomes

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} + b = \sum_i \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b. \quad (2.17)$$

Thus, both the Lagrangian and the classification function involve only dot products between transformed feature space vectors through the mapping ϕ . As a consequence, the explicit computation of the mapping ϕ is not required to train an SVM model or to make predictions from it, as soon as it is possible to compute (explicitly) dot products in this space. A kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a function that for a pair of input feature vectors, outputs a real number that can be interpreted as a dot product corresponding to some (implicit) mapping ϕ :

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle.$$

This is the case as soon as K is a symmetric positive semi-definite function (Shawe-Taylor and Cristianini, 2004). By replacing every dot products in (2.16) and (2.17) by an appropriate kernel function, one can thus train an SVM in the corresponding feature space without having to explicitly compute the representation of the examples in this space. This is called the kernel trick in the SVM literature. Using kernels is especially convenient when the dimension D of the feature space is very high (or even infinite), while dot products (i.e., kernels) in this space can be computed efficiently.

Kernels have been defined for many data types (Shawe-Taylor and Cristianini, 2004). Popular generic choices when $\mathcal{X} = \mathbb{R}^d$ are the polynomial and Gaussian kernels:

$$\begin{aligned} K^{\text{polynomial}}(\mathbf{x}, \mathbf{x}') &= (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^p \\ K^{\text{Gaussian}}(\mathbf{x}, \mathbf{x}') &= \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2). \end{aligned}$$

The mapping ϕ corresponding to the polynomial kernel with $p = 2$ is given in (2.15), while the dimension of the feature space corresponding to the Gaussian kernel can be shown to be infinite. In what follows, we will review specific kernels that have been proposed in the context of supervised network inference.

2.2.4 Output kernel trees

The Output kernel trees (OK3) algorithm is a generalization of multi-output trees that has been developed by Geurts *et al.* (2006b). We describe it here because it has been used in (Geurts *et al.*, 2007) for supervised network inference (see Section 2.3.2).

Output kernel trees are an adaption of decision trees to make (implicit) predictions in the feature space corresponding to a kernel. It is based on the fact that the impurity function Q in (2.6) can be written using only dot products between output vectors. Indeed, using the following expansion of the square euclidean distance:

$$\|\mathbf{y} - \mathbf{y}'\|^2 = \langle \mathbf{y}, \mathbf{y} \rangle + \langle \mathbf{y}', \mathbf{y}' \rangle - 2\langle \mathbf{y}, \mathbf{y}' \rangle,$$

we have:

$$Q(S) = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} \|\mathbf{y}_i - \frac{1}{|S|} \sum_{(x_i, y_i) \in S} \mathbf{y}_i\|^2 \quad (2.18)$$

$$= \frac{1}{|S|} \sum_{i \in S} \langle \mathbf{y}_i, \mathbf{y}_i \rangle - \frac{1}{|S|^2} \sum_{i, j \in S} \langle \mathbf{y}_i, \mathbf{y}_j \rangle, \quad (2.19)$$

$$(2.20)$$

which makes use only of dot products. Let us assume that we have a learning sample of input vectors $LS = \{x_1, \dots, x_N\}$ and a (positive semi-definite) kernel matrix K that corresponds to dot products between output vectors in some feature space \mathcal{H} , where $(K)_{i,j} = k_{i,j} = \langle \phi_i, \phi_j \rangle$ with $\phi_i \in \mathcal{H}, i = 1, \dots, N$ the (unknown) output feature vectors corresponding to the learning sample objects. Only from this information, it is thus possible to grow a tree using as an impurity function:

$$\begin{aligned} Q(S) &= \frac{1}{|S|} \sum_{i \in S} \langle \phi_i, \phi_i \rangle - \frac{1}{|S|^2} \sum_{i, j \in S} \langle \phi_i, \phi_j \rangle, \\ &= \frac{1}{|S|} \sum_{i \in S} k_{i,i} - \frac{1}{|S|^2} \sum_{i, j \in S} k_{i,j}. \end{aligned}$$

Each leaf L of the resulting tree is implicitly labeled by the center of mass in \mathcal{H} of the outputs of the objects that fall into that leaf, i.e.:

$$\phi_L = \frac{1}{|S|} \sum_{i \in S} \phi_i,$$

where S is the subset of objects falling into L . The latter can not be explicitly computed only from kernel values. However, it is possible to compute dot products between two such vectors from kernels only and therefore make kernel predictions for two new objects. Indeed, let us consider two (new) input feature vectors \mathbf{x} and \mathbf{x}' falling respectively into leaves L and L' containing objects S and S' from the learning sample. Then, a prediction of the kernel between these two objects can be obtained as follows:

$$\hat{K}(\mathbf{x}, \mathbf{x}') = \langle \phi_L, \phi_{L'} \rangle = \frac{1}{|S||S'|} \sum_{i \in S} \sum_{j \in S'} \langle \phi_i, \phi_j \rangle = \frac{1}{|S||S'|} \sum_{i \in S} \sum_{j \in S'} k_{i,j}. \quad (2.21)$$

Randomization-based methods, such as Random Forests or Extra-Trees, can be easily extended to this output kernel framework and provide here the same kind of improvement, by variance reduction, as with standard classification or regression trees (Geurts *et al.*, 2006b).

Applications of this method are numerous. When an explicit output y is available, the method can be used to grow a tree with an alternative loss function that can be written as the squared distance into the feature space of some kernel. It allows also to handle prediction problems where

the output can not easily be represented by a vector of numbers but for which it is possible to define a kernel. To make explicit output predictions in this case, it is necessary however to solve a pre-image problem (see Geurts *et al.* (2006b)). Finally, the method can also be used to generalize a kernel over some input spaces, using predictions in the form (2.21). An application of this idea for supervised network inference is discussed in Section 2.3.2.

2.3 Supervised network inference

In this section, we first define the problem of supervised network inference more formally and lay out the notations for the rest of the thesis. We then review existing approaches to solve this problem.

2.3.1 Problem definition

For the sake of generality, let us assume that we have two finite sets of nodes, $U_r = \{n_r^1, \dots, n_r^{N_{U_r}}\}$ and $U_c = \{n_c^1, \dots, n_c^{N_{U_c}}\}$ of respective sizes N_{U_r} and N_{U_c} . A network connecting these two sets of nodes can then be defined by an adjacency matrix Y of size $N_{U_r} \times N_{U_c}$, such that $y_{ij} = 1$ if the nodes n_r^i and n_c^j are connected and $y_{ij} = 0$ if not. Actually, the subscripts r and c stand respectively for *row* and *column*, referring to the rows and columns of the targeted adjacency matrix Y . Y thus defines a *bipartite graph* over the two sets U_r and U_c . Standard graphs defined on only one family of nodes, that we call *homogeneous graphs*, can nevertheless be obtained as special cases of this general framework by considering only one set of nodes (i.e. $U = U_r = U_c$). Undirected or directed graphs can then both be represented using a symmetric or an asymmetric adjacency matrix Y .

For example, in the case of protein-protein interaction networks, $U_c = U_r$ is the set of all proteins of a given organism and the adjacency matrix is symmetric. A drug-protein interaction network can be modeled as a bipartite graph where U_r and U_c are respectively the sets of proteins and drugs of interest, and element y_{ij} of Y is equal to 1 if protein n_r^i interacts with drug n_c^j , 0 otherwise. A regulatory network can be modeled either as a bipartite graph where U_c is the set of all genes of the organism of interest and U_r is the set of all candidate transcription factors among them or equivalently by an homogeneous graph and an asymmetric adjacency matrix, where $U_c = U_r$ is the set of all genes and $y_{ij} = 1$ if gene n_i regulates gene n_j , 0 otherwise.

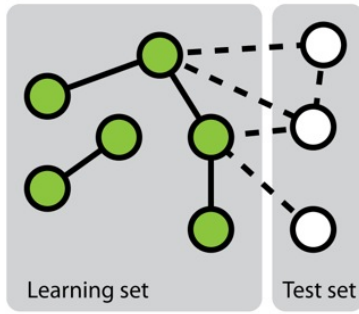
In addition, we assume that each node n (in both sets) is described by a feature vector, denoted $x(n)$, typically lying in \mathbb{R}^p . For example, features associated to proteins/genes could include their expression in some conditions as measured by microarrays, the presence of motifs in their promotor region, information about their structure, etc. A feature vector $x(n_r, n_c)$ can also be associated to each pair of nodes. For example, features directly associated to pairs of proteins could code for the association of the two proteins in another network, their binding in a ChIP-sequencing experiments, etc.

In this context, the problem of supervised network inference can be formulated as follows:

Given a partial knowledge of the adjacency matrix Y of the target network in the form of a learning sample of triplets:

$$LS_p = \{(n_r^k, n_c^k, y_{i_k j_k}) | k = 1, \dots, N_{LS}\},$$

and given the feature representation of the nodes and/or pairs of nodes, find a function $f : U_r \times U_c \rightarrow \{0, 1\}$ that best approximates the unknown entries of the adjacency



Inputs		Output
$[\mathbf{x}(n_1) \quad \mathbf{x}(n_2)]$		$y_{1,2}$
$[\mathbf{x}(n_1) \quad \mathbf{x}(n_3)]$		$y_{1,3}$
\vdots	\vdots	\vdots
$[\mathbf{x}(n_2) \quad \mathbf{x}(n_1)]$		$y_{2,1}$
$[\mathbf{x}(n_2) \quad \mathbf{x}(n_3)]$		$y_{2,3}$
\vdots	\vdots	\vdots
$[\mathbf{x}(n_N) \quad \mathbf{x}(n_{N-1})]$		$y_{N,N-1}$

Figure 2.6: Schematic representation of one of the two main approaches to solve the problem of supervised network inference: the global approach that solves a single supervised learning problem by considering each pair as an object for learning a classifier.

matrix from the feature representation (on nodes or on pairs) relative to these unknown entries.

This problem can be cast as a supervised classification problem, with the peculiarity however that pairs of nodes, and not single nodes, need to be classified. Next, we discuss existing methods to solve this problem.

2.3.2 Network inference methods

Mainly two approaches, global and local, have been investigated in the literature to transform the network inference problem into one or several standard classification problems (Vert, 2010). In this section, we start by briefly presenting these two approaches that make use of existing classification methods, and then talk about more specific approaches that have been proposed for supervised network inference, like output kernel trees.

Global approach

The first, more straightforward, approach, called *pairwise* or *global*, considers each pair as a single object and then apply any existing classification method on these objects (e.g., Takarabe *et al.*, 2012). This approach requires a feature vector defined on pairs. When features on individual nodes are provided, they thus need to be transformed into features on pairs (Tastan *et al.*, 2009). Several approaches have been proposed in the literature to achieve this, ranging from a simple concatenation (**Figure 2.6**) or addition of the feature vectors of the nodes in the pair (Chen and Liu, 2005; Yu *et al.*, 2012) to more complex combination schemes (Yamanishi *et al.*, 2008; Maetschke *et al.*, 2013). Different classification methods have been exploited in the literature: nearest neighbor algorithm (He *et al.*, 2010), support vector machines (Paladugu *et al.*, 2008), logistic regression (Ulitsky *et al.*, 2009), tree-based methods (Wong *et al.*, 2004; Lin *et al.*, 2004; Chen and Liu, 2005; Qi *et al.*, 2006; Tastan *et al.*, 2009; Yu *et al.*, 2012), Markov logic network (Brouard *et al.*, 2013), etc. In particular, in the context of support vector machines, several kernels have been proposed to compare pairs of objects on the basis of individual features defined on these objects and these kernels have been applied for supervised network inference (Ben-Hur and Noble, 2005; Vert *et al.*, 2007; Hue and Vert, 2010; Brunner *et al.*, 2012). We describe several such kernels below.

Special kernels for pairs. In the global approach, feature vectors of both nodes of a pair must be combined to define a feature vector on the pair. We focus here on homogeneous graphs defined on a single type of node. Let us denote an input feature vector for a node n by $\phi(n)$ and an input feature vector for a pair (n_1, n_2) by $\psi(n_1, n_2)$. These feature vectors define respectively a kernel on nodes, i.e. $K(n, n') = \langle \phi(n), \phi(n') \rangle$, and a kernel on pairs of nodes $K_\psi((n_1, n_2), (n_3, n_4)) = \langle \psi(n_1, n_2), \psi(n_3, n_4) \rangle$.

Several solutions have been proposed to derive a feature vector (or kernel) on pairs of nodes from a feature vector (or kernel) on single nodes. Two straightforward solutions include the addition, feature by feature, of the vectors of the two nodes in the pair, i.e. $\psi_{add}(n_1, n_2) = \phi(n_1) + \phi(n_2)$, or their concatenation, i.e., $\psi_{concat}(n_1, n_2) = [\phi(n_1), \phi(n_2)]$. These two feature representations correspond respectively to the following kernels on pairs:

$$\begin{aligned} K_{\psi,add}((n_1, n_2), (n_3, n_4)) &= K(n_1, n_3) + K(n_1, n_4) + K(n_2, n_3) + K(n_2, n_4) \\ K_{\psi,concat}((n_1, n_2), (n_3, n_4)) &= K(n_1, n_3) + K(n_2, n_4), \end{aligned}$$

where K denotes a kernel on nodes. The first kernel considers the pairs as undirected, while the second considers them as directed. Note that in general, these representations are not appropriate for supervised network inference when combined with linear SVM. Indeed, with these representations, the linear scoring functions are respectively:

$$\begin{aligned} f_{add}(n_1, n_2) &= \langle \mathbf{w}, \phi(n_1) \rangle + \langle \mathbf{w}, \phi(n_2) \rangle + b \\ f_{concat}(n_1, n_2) &= \langle \mathbf{w}_1, \phi(n_1) \rangle + \langle \mathbf{w}_2, \phi(n_2) \rangle + b, \end{aligned}$$

With these two functions, the ranking of the candidate partners of a given node n_1 (resp. n_2) is thus the same whatever the node n_1 (resp. n_2), which means for example that the most likely partner of all nodes is a unique node n^* that maximizes $f(n_1, n^*)$, for any node n_1 .

Another solution that overcomes this limitation is the following feature representation of pairs:

$$\psi_{kp}(n_1, n_2) = \phi(n_1) \otimes \phi(n_2)$$

where \otimes is the direct or Kronecker product that produces a vector whose entries are all possible products between one feature from $\phi(n_1)$ and one feature from $\phi(n_2)$. If $\phi(n)$ is of dimension d , then $\psi_{kp}(n_1, n_2)$ will be of dimension d^2 , which can be prohibitive in case of large d . Fortunately, the corresponding kernel, called *Kronecker product pairwise kernel (KPPK)*, can be computed efficiently without needing to compute explicitly the joint feature vector $\psi_{kp}(n_1, n_2)$. We have indeed:

$$\begin{aligned} K_{KPPK}(\{n_1, n_2\}, \{n_3, n_4\}) &= \psi_{kp}(n_1, n_2)^T \psi_{kp}(n_3, n_4) \\ &= K(n_1, n_3)K(n_2, n_4). \end{aligned}$$

Interpreting kernels as similarity measures, this kernel considers two pairs as close to each other if both the first nodes of the two pairs and the second nodes of the two pairs are close to each other. This kernel has been used for example in (Stock *et al.*, 2012) for network inference.

The Kronecker product pairwise kernel considers pairs to be directed (as $K_{KPPK}((n_1, n_2), (n_3, n_4)) \neq K_{KPPK}((n_1, n_2), (n_4, n_3))$). A natural adaptation of this feature representation for undirected pairs is as follows (see Ben-Hur and Noble (2005); Vert (2010) for more details):

$$\psi_{tp}(n_1, n_2) = \phi(n_1) \otimes \phi(n_2) + \phi(n_2) \otimes \phi(n_1),$$

which leads to the kernel:

$$K_{TPPK}(\{n_1, n_2\}, \{n_3, n_4\}) = \psi_{tp}(n_1, n_2)^\top \psi_{tp}(n_2, n_3) \quad (2.22)$$

$$= 2[K(n_1, n_3)K(n_2, n_4) + K(n_1, n_4)K(n_2, n_3)]. \quad (2.23)$$

This kernel is called the *tensor product pairwise kernel (TPPK)*. Vert *et al.* (2007) proposed another solution that combines node feature vectors as follows:

$$\psi_{ml}(n_1, n_2) = (\phi(n_1) - \phi(n_2)) \otimes (\phi(n_2) - \phi(n_1)),$$

which leads to the following kernel:

$$K_{MLPK}(\{n_1, n_2\}, \{n_3, n_4\}) = [K(n_1, n_3) + K(n_2, n_4) - K(n_1, n_4) - K(n_2, n_3)]^2$$

called the *metric learning pairwise kernel (MLPK)*.

Local approach

In the second approach, called *local* (Mordelet and Vert, 2008; Bleakley and Yamanishi, 2009; Vert, 2010; van Laarhoven *et al.*, 2011; Mei *et al.*, 2013), the network inference problem is divided into several smaller classification problems corresponding each to a node of interest and aiming at predicting, from the features, the nodes that are connected to this node in the network (**Figure 2.7**). More precisely, each of these classification problems is defined by a learning sample containing all nodes that are involved in a pair with the corresponding node of interest in LS_p . Interestingly, when trying to make a prediction for a given pair (n_r^i, n_c^i) , one can aggregate the predictions of two classifiers: the one trained for n_r^i and the one trained for n_c^i . Note that it is only possible to train a classifier for a node that is involved in at least one positive and one negative interaction in LS_p . This prevents the use of the local approach to predict interactions for pairs where both nodes do not satisfy this property. Like for the global approach, in principle, any classification method can be used to train each of the classification models, but mainly support vector machines have been investigated in this context (Mordelet and Vert, 2008; Bleakley and Yamanishi, 2009). Global and local approaches are defined more formally in Section 4.2.

From experiments in the literature, there does not seem to be a clear winner between the local and the global approach in terms of predictive accuracy. The global approach is typically more flexible as it can handle any kinds of features and can make prediction for pairs of unseen nodes, but it requires more computing times and resources, given that it aims to infer a network in one step.

Output kernel trees and other approaches

Besides the global and local approaches that make use of standard classification methods, other more specific approaches have also been proposed for supervised network inference.

The Output kernel trees (OK3) algorithm, presented in Section 2.2.4, can be exploited for supervised network inference (Geurts *et al.*, 2007). The approach consists in defining a kernel $K(n, n')$ between the nodes in the network that encodes the presence or absence of direct or indirect connections between the nodes in the training graph. This kernel should be such that two nodes receive a high value of K if they are close in the graph, and a small value of K if they are far away in the graph. An example of a kernel that satisfies this condition is the diffusion kernel (Kondor and Lafferty, 2002) that corresponds to a Gramm matrix $K = \exp(-\beta L)$, where L is the graph Laplacian

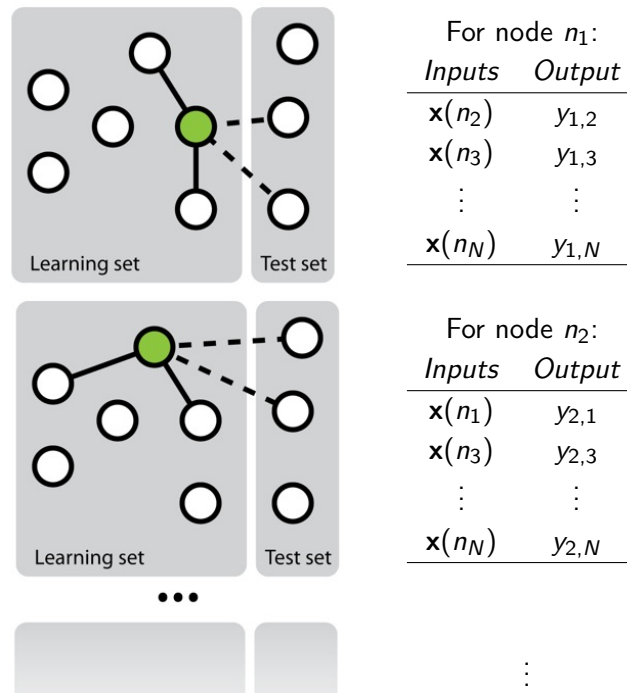


Figure 2.7: Schematic representation of one of the two main approaches to solve the problem of network inference: the local approach that solves several supervised learning problems, each defined by a different node.

and β a parameter. From this kernel and input features defined on the nodes, an output kernel tree can be grown using the algorithm described in Section 2.2.4. From this tree, kernel values can then be predicted between two (new) nodes described by their input feature vectors using (2.21). A high value of this kernel indicates that the two nodes are predicted to be close in the graph and a low value that they are predicted to be far away. Whether there is an edge between two nodes can then be inferred by thresholding these kernel predictions. Examples of network inference results obtained with this algorithm are presented in Section 4.4.4. Note that in order to apply this method, a training network needs to be known completely for a subset of nodes, as a complete kernel matrix is required for training the output kernel tree model. In other words, and anticipating the notations introduced in the next chapter, the method can only be used to make $LS \times TS$ and $TS \times TS$ predictions and not $LS \times LS$ predictions. The method can also be only applied to infer homogeneous and undirected networks.

Along the same idea of output kernel regression, Brouard *et al.* (2011) extended (kernel) ridge regression to learn an approximation of an output kernel and applied it for network inference.

Other approaches exist for supervised network inference. For example, Kato *et al.* (2005) formulate the problem as a matrix completion problem (with input features) and solve it using an expectation-maximization-based approach. The problem has also been formulated as a distance metric learning problem (Vert and Yamanishi, 2005; Yamanishi, 2009): nodes of the graph are embedded into some euclidean space where they are close as soon as they are connected in the training graph and a mapping is then learned from the node feature space to this euclidean space. These approaches are similar with the output kernel regression approach discussed above, with the difference however that in (Vert and Yamanishi, 2005; Yamanishi, 2009), the mapping is explicit

and low-dimensional, while in (Geurts *et al.*, 2007; Brouard *et al.*, 2011), it is carried out implicitly through a kernel.

While our review focused on the inference of the network from node features, it is also possible to solve this problem by exploiting only the network itself. For example, Cheng *et al.* (2012) derive a similarity measure between nodes from the network topology and then use this similarity to infer new interactions. In a hybrid approach, some authors have also included features derived from the (training) network topology in the global approach to improve network inference (Ulitsky *et al.*, 2009). Note that these approaches are limited to the prediction of new interactions between nodes that are already known to interact with other nodes in the network. Again anticipating notations introduced in the next chapter, these methods can only make $LS \times LS$ predictions.

Finally, we only reviewed here methods that have been proposed for the inference of biological networks. The problem of predicting the presence or the absence of an edge between nodes in a network however appears in many domains. This problem is often referred to as the problem of (supervised) link prediction with common applications to social networks (Liben-Nowell and Kleinberg, 2007; Brouard, 2013). In this latter domain, the goal is predict future relationships between participants in a social network from the knowledge of current and past relationships and sometimes also features describing these participants. Another related domain is collaborative filtering or recommender systems (Su and Khoshgoftaar, 2009), where the goal is to connect customers with their potential purchases and can thus be assimilated to the inference of a bipartite graph. To the best of our knowledge, collaborative filtering techniques have however not be very much exploited for the inference of biological networks.

2.4 Unsupervised and semi-supervised biological network inference

This thesis focuses on the problem of supervised network inference, where a model for predicting new interactions is trained from a sample of known interacting and non-interacting pairs of nodes and different measurements/features on these nodes. Unsupervised techniques have also been proposed for the inference of biological networks. By unsupervised inference, we mean any inference methods that does not require a subset of known interacting and non-interacting pairs in order to carry out the inference. Because no model is trained, these techniques require a strong prior biological knowledge of what defines an interaction. This knowledge is exploited to select the most appropriate input features to be measured on the nodes and to devise a model to identify interacting pairs from these features. As a consequence, while supervised techniques can be applied to any kinds of networks, unsupervised methods are network specific.

Several unsupervised methods have been proposed for example for the inference of protein-protein interactions. Marcotte *et al.* (1999) observed that two interacting proteins often have significant similarities, at the sequence level, to a protein in other organisms, i.e., their orthologs are fused in other organisms. They used this property to infer new interactions. Other models of interactions are based on the observation that two proteins are functionally related (in a pathway or in a structural complex) if they have similar phylogenetic profiles or trees (Pellegrini *et al.*, 1999; Huynen *et al.*, 2000), i.e., if they are present or absent in the same species. The explanation is that interacting proteins tend to coevolve. Based on this idea, Huynen *et al.* (2000) first compute the similarity between the phylogenetic profiles of two proteins using mutual information. Then, proteins are clustered according to this similarity and proteins that fall into the smallest clusters are predicted to interact. In a more elaborated approach, Pazos and Valencia (2001) compute the

similarity between two proteins as the linear correlation coefficient between two matrices containing the genetic distances between the proteins and their orthologs as obtained from a phylogenetic tree. Pairs of proteins with the highest values of this correlation coefficient are then predicted to be interacting pairs.

Unsupervised techniques are also very commonly applied for the inference of gene regulatory networks (see De Smet and Marchal (2010); Maetschke *et al.* (2013) for recent reviews). Several data sources can be integrated for inferring regulatory links between genes (De Smet and Marchal, 2010) but the most common source remains microarray expression data. Indeed, one expects that the expression of a gene should be related to the expressions of its transcription factors. Several inference methods have been proposed that are based on this hypothesis. The simplest, but nevertheless efficient, techniques associate a score to every pair of genes based on a comparison of their expression profiles and then infer the existence of a regulatory link between two genes by thresholding this score. Various score measures have been proposed based for example on plain correlation (Langfelder and Horvath, 2008), mutual information (Meyer *et al.*, 2007; Faith *et al.*, 2007b), or partial correlation (de la Fuente *et al.*, 2004). When times series of gene expressions are available, some methods learn a more sophisticated model of the dynamics of the gene expressions for example using differential equations (Quach *et al.*, 2007). The network is encoded in this model through parameters and its inference is carried out by learning these parameters from the data. Between score-based and model-based methods, several researchers have proposed to use supervised regression techniques to predict the expression of one target gene from the expressions of all the other genes. Inferring the transcription factors of the target gene then reduces to the selection of the genes whose expressions are useful to predict the expression of this target gene, which amounts at solving a feature selection problem (in regression). Several feature selection techniques have been exploited in this context, including Random Forests (Huynh-Thu *et al.*, 2010) and L1-based linear regression methods (Haury *et al.*, 2012).

The main advantage of unsupervised methods compared to supervised methods is that they can predict interactions without requiring a partial knowledge of the network to infer. This is especially interesting for “new” networks or networks related to “new” organisms, for which little is known. The drawback of these methods is of course the need for a biological model of interactions, as well as the need to design a new method for every new network. In terms of predictive performance, the superiority of one family of methods over the other for a given network highly depends on the size and the quality of the training network for supervised methods and on the quality of the biological model of interaction for unsupervised methods. Empirical comparisons between the two approaches in the context of gene regulatory networks (Vert, 2010; Tavakkolkhah and Küffner, 2013) and protein-protein interaction networks (Yamanishi and Vert, 2004) nevertheless tends to show that supervised inference methods are more efficient than supervised ones.

To be complete, let us also mention the existence of semi-supervised methods. Standard supervised methods only exploit input features of the nodes that are involved in labeled pairs from the learning sample. In most network inference applications however, all possible nodes (in U_r and U_c) described by their input features are typically known at training time. Semi-supervised techniques try to make use of all the information available at training time to improve the quality of the inferred model. For example, Ernst *et al.* (2008) propose a semi-supervised approach for the inference of gene regulatory networks. A first model is trained using a supervised learning approach, by considering all the unlabeled pairs as negative examples. This model is used to predict an interaction score for all the unlabeled pairs. The pairs that receive a score higher than a given threshold are then labeled

as positive and a new model is learnt from the relabeled learning set. This process is repeated until the labels of the pairs do not change anymore. Brouard *et al.* (2011) propose a semi-supervised extension of their output kernel regression approach for network inference based on kernel ridge regression. Their semi-supervised solution is obtained by introducing a graph-based regularizer that enforces that nodes with similar input features should be mapped to close vectors in the output feature space \mathcal{H} defined by the output kernel. In these two works, the semi-supervised methods are shown to supersede supervised ones.

2.5 Discussion

In the present chapter we explained the concept of biological networks and presented several examples of such networks. Then we discussed the problem of supervised learning and described in some detail the different algorithms that will be used throughout this thesis, tree-based ensemble methods and support vector machines. Then, we provided a formal definition of the problem of supervised network inference and reviewed methods that have been proposed in the literature to address this problem. To be complete, we also briefly discussed unsupervised and semi-supervised network inference methods.

Now that the background has been covered, the rest of the thesis will be devoted:

- in Chapter 3, to a thorough and critical presentation of the main difficulties and pitfalls in the evaluation and the practical application of network inference methods,
- in Chapter 4, to an adaptation of the local and global approaches in the case of tree-based ensemble methods and their systematic comparison with other methods from the literature on several biological networks,
- in Chapter 5, to an application of these methods for the inference of genetic interactions in Yeast.

Chapter 3

Evaluating supervised network inference methods

In this chapter, we examine the assessment of supervised network inference. Supervised inference infers the network from a training sample of known interacting and possibly non-interacting entities and additional measurement data. While these methods are very effective, their reliable validation *in silico* poses a challenge, since both prediction and validation need to be performed on the basis of the same partially known network. Cross-validation techniques need to be specifically adapted to classification problems on pairs of objects. We perform a critical review and assessment of protocols and measures proposed in the literature and derive specific guidelines how to best exploit and evaluate machine learning techniques for network inference. Through theoretical considerations and *in silico* experiments, we analyze in depth how important factors influence the outcome of performance estimation. These factors include the amount of information available for the interacting entities, the sparsity and topology of biological networks, and the lack of experimentally verified non-interacting pairs. Finally, we develop a new approach to compare scores predicted with different models, associated to different performance, and that maximizes the number of true positives.

Contents

3.1	Introduction	46
3.2	Evaluation measures	47
3.3	Evaluation protocols	57
3.4	Lack of negative examples	64
3.5	Impact of heavy-tailed node degree distribution	74
3.6	Merging pair rankings	78
3.7	Conclusion	89

3.1 Introduction

Performance estimation of both unsupervised and supervised inference methods requires a gold standard of experimentally tested interactions, i.e. pairs of entities labeled as interacting or non-interacting. The validation of supervised methods however generally requires special care and the application of cross validation techniques to avoid any sources of bias. Indeed both training and validation need to be performed on the basis of the same partially labeled gold standard. The case of supervised network inference is even more complex as it works on pairs of objects so that the traditional cross validation techniques are not sufficient. In the chapter, we propose a critical review of protocols and measures found in the literature for the validation of supervised network inference methods and derive specific guidelines on how to best exploit machine learning techniques for network inference.

In particular, our discussion of evaluation measures and protocols will account for the following important aspects :

- Most gold standard networks used for inference are very sparse and incomplete, which makes the resulting classification problem extremely imbalanced. This has important consequences on the evaluation metrics, making for example ROC curves mostly inappropriate.
- The predictive performance of a method for a given interaction highly depends on how much the two involved nodes are covered by the gold standard. It is typically much more difficult to predict interactions of regulators that are not contained in the training network. In consequence, one needs to assess performance by distinguishing four families of pairs according to whether or not any of its two nodes are in the training data.
- Cross-validation techniques need to accommodate for the way the training network has been experimentally obtained. At two extremes, one can distinguish sampling of pairs, where available interactions have been uniformly sampled among all possible pairs, and sampling of nodes, where all interactions are known for a subset of nodes.
- It is often the case that no truly non interacting pairs (aka negative examples) are available so that gold standards exclusively consist of interactions tested as positive. As a workaround, usually all previously untested pairs are assumed negative, leading to particular problems at the training and performance estimation stages.
- Topological properties of biological networks apparently allow for good predictions even if no experimental data at all is used. For example, in networks with a heavy-tailed node degree distribution, because of the preferential attachment property, connecting any node to a hub leads to AUC scores that appear better than random guessing. This introduces a positive bias, an overestimation of the true predictive performance, (at training and at test time) that should be taken into account when assessing a given inference method.
- When several sets of objects have been predicted with different models, and are associated to different performance curves, their predicted scores should not be naively compared. They must be gathered cautiously, if we want to obtain the best global precision-recall curve.

The chapter is structured as follows. Section 3.2 discusses common metrics used to evaluate network predictions (that are common to unsupervised and supervised inference methods). Appropriate ways to perform cross-validation in this context are discussed in section 3.3. The impact of

the lack of negative examples in common biological networks is analyzed in Section 3.4. Section 3.5 discusses the positive bias on performance induced by the heavy-tailed degree distribution often met in biological networks. Finally, Section 3.6 presents a new way to merge different sets of predictions by taking into account their different associated performance curves.

3.2 Evaluation measures

In this section, we review and discuss evaluation measures that have been used to quantify the quality of the predictions given by network inference methods. We focus here on statistical measures that compare a predicted network (or subnetwork) with the true one, as in the case of supervised network inference, some part of the true network is supposed to be available for training. In the general context of network inference, other performance measures have been proposed based either on functional annotations shared by genes/proteins or on topological properties of the inferred networks (see Emmert-Streib *et al.*, 2012, for a survey).

The prediction given by a network inference method for a given pair of nodes can typically be of two kinds: a binary (0-1) value, coding for the presence or the absence of an interaction between the two nodes in the predicted network, or a real value, representing some confidence score associated to the pair: the higher the score, the higher the confidence or certainty of the model that there is an interaction between the nodes in the pair. Depending on the supervised network inference method used, this confidence score can have a probabilistic interpretation or not, but we will not assume it is the case. Of course, one can always transform a confidence score into a binary prediction using a decision threshold. The choice of an appropriate threshold is however not an easy problem in practice.

In this section, we assume that we have an adjacency matrix (of a complete or a partial network) and an equivalent matrix of the binary or real scores predicted by a network inference method (**Table 3.1**). In both cases, our goal is to quantify the quality of the predictions with respect to the true network represented by the adjacency matrix. Protocols to obtain these matrices will be discussed in Section 3.3. We first discuss the case of binary predictions and then compare the receiver operating characteristic (ROC) curves and precision-recall (PR) curves that have been predominantly used to evaluate network inference methods that provide confidence scores. We end the section with a brief survey of other measures and a general discussion.

3.2.1 Binary predictions

For binary predictions, a first idea would be obviously to compute the accuracy (the number of correctly predicted pairs divided by the total number of pairs) or equivalently the error rate (one minus the accuracy). However, network inference problems typically correspond to highly imbalanced classification problems as non-interacting pairs often far outnumber interacting ones. Accuracy is not appropriate in such situations because it greatly favors the majority class (high accuracy is given to a model predicting all pairs as non-interacting pairs). Alternative measures require to differentiate between the possible types of errors, that are usually counted and compiled in a confusion matrix. In the case of binary classification, this matrix is a 2×2 matrix where the columns and rows represent respectively the actual and the predicted classes and each cell contains the number of pairs corresponding to these classes (**Table 3.2**).

Table 3.1: Example of adjacency matrices of a 8×5 network. The predicted matrix is composed of confidence scores. By choosing a decision threshold equal to 0.75, the predicted matrix infers perfectly the true one.

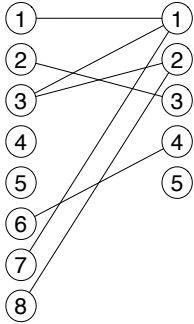
True network	True adjacency matrix	Predicted adjacency matrix
	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.84 & 0.19 & 0.23 & 0.17 & 0.23 \\ 0.44 & 0.31 & 0.92 & 0.43 & 0.18 \\ 0.90 & 0.98 & 0.44 & 0.11 & 0.26 \\ 0.41 & 0.59 & 0.26 & 0.60 & 0.71 \\ 0.22 & 0.12 & 0.30 & 0.32 & 0.42 \\ 0.51 & 0.09 & 0.26 & 0.80 & 0.03 \\ 0.93 & 0.73 & 0.49 & 0.58 & 0.24 \\ 0.46 & 0.96 & 0.55 & 0.52 & 0.23 \end{bmatrix}$

Table 3.2: A confusion matrix for a binary classification. Positive denotes an interaction and negative denotes a non-interaction.

	actual positive (P)	actual negative (N)
predicted positive ($predP$)	true positive (TP)	false positive (FP)
predicted negative ($predN$)	false negative (FN)	true negative (TN)

Several metrics can be then derived from this matrix to evaluate the performance of a model, among which:

- the *true positive rate* (TPR), also called the *sensitivity* or the *recall*, is equal to the number of true positives divided by the number of actual positives: $\frac{TP}{TP+FN}$ or $\frac{TP}{P}$,
- the *true negative rate* (TNR), also called the *specificity*, is equal to the number of true negatives divided by the number of actual negatives: $\frac{TN}{FP+TN}$ or $\frac{TN}{N}$,
- the *false positive rate* (FPR), corresponding to 1 -specificity, is equal to the number of false positives divided by the number of actual negatives: $\frac{FP}{FP+TN}$ or $\frac{FP}{N}$,
- the *false negative rate* (FNR), also called the *miss*, is equal to the number of false negative divided by the number of actual negatives: $\frac{FN}{TP+FN}$ or $\frac{FN}{P}$,
- the *precision* is equal to the number of true positives divided by the number of predicted positives: $\frac{TP}{TP+FP}$.
- the *rate of positive predictions* (RPP) is equal to the number of predicted positive divided by the total number of examples: $\frac{TP+FP}{P+N}$ or $\frac{predP}{P+N}$
- the *F-score* is equal to the harmonic mean of precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Except for the *F-score*, these measures should be combined to give a global picture of the performance of a method, e.g., sensitivity and specificity or precision and recall. In the case of confidence scores,

Table 3.3: Example of confidence scores for a list of 10 objects. Objects are ranked according to the values of their scores. The last three rows report respectively the false positive rate, true positive rate, and precision for increasing values of the confidence threshold.

Confidence scores	0.91	0.86	0.85	0.57	0.54	0.26	0.18	0.16	0.14	0.13
Rank	1	2	3	4	5	6	7	8	9	10
Actual values	1	1	0	1	0	0	1	0	0	0
FPR	0	0	$1/6$	$1/6$	$1/3$	$1/2$	$1/2$	$2/3$	$5/6$	1
TPR/Recall	$1/4$	$1/2$	$1/2$	$3/4$	$3/4$	$3/4$	1	1	1	1
Precision	1	1	$2/3$	$3/4$	$3/5$	$1/2$	$4/7$	$1/2$	$4/9$	$2/5$

Constants: $P = 4$, $N = 6$

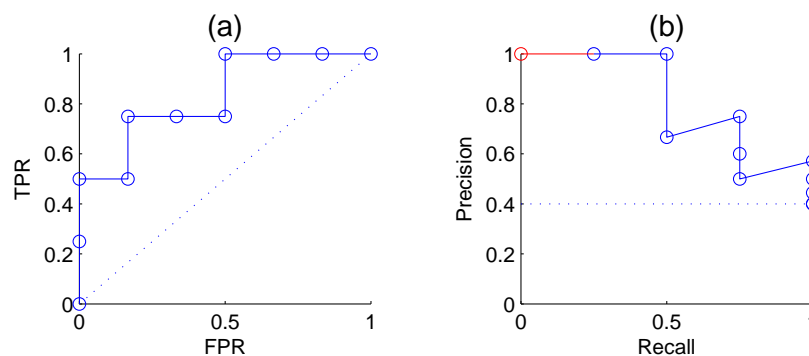


Figure 3.1: ROC curve (A) and precision-recall curve (B) for the predictions in **Table 3.3**.

all these performance measures can be computed for a given threshold on the confidence scores. Nevertheless, often, one would like to measure the performance of a method independently of the choice of a specific threshold. Several curves are used for that purpose that are exposed below.

3.2.2 ROC curves

Receiver Operating Characteristic (ROC) curves plot the TPR as a function of the FPR, when varying the confidence threshold (Fawcett, 2006). In concrete terms, the predictions are sorted from the most confident to the least confident, and the threshold is varied from the maximum to the minimum confidence score. Each value of the threshold corresponds to a different confusion matrix, and thus a different pair of values of the TPR and FPR, and corresponds to a point of the ROC curve.

Varying the threshold of one position in the ranked list of predictions corresponds to predicting one more pair of nodes as interacting, i.e. an increase of $predP$ by 1. If the pair truly interacts, then the TPR increases by $1/P$ (value of one vertical step on the graph), while the FPR does not change. On the contrary, if the pair actually does not interact, then the FPR increases by $1/N$ (value of one horizontal step), while the TPR does not change (**Table 3.3** and **Figure 3.1A**). This results in a jagged curve, that becomes smoother and smoother when the size of the dataset increases, because the sizes of the vertical and horizontal steps decrease respectively when P and N increases.

The two ends of the curve are always the two points $(0, 0)$ and $(1, 1)$, corresponding respectively to $predP = 0$ and $predP = P + N$. A perfect classifier would give the highest values of prediction to the pairs that truly interact, and then would have a corresponding ROC curve passing through the point $(0, 1)$. The curve relative to a random classifier corresponds to the diagonal connecting the two points $(0, 0)$ and $(1, 1)$ (the dotted line in **Figure 3.1A**).

For comparison purposes, it is often convenient to summarize a ROC curve with a single real number. The most common such measure is the area under the ROC curve (AUROC), which is equal to 1 for a perfect classifier and 0.5 for a random one. On the face of it, one typically assumes that the higher the AUROC, the better the predictions.

In many network prediction tasks however, the number of interactions is much lower than the number of non-interactions. It is therefore important to achieve a low *FPR* as even moderate *FPR* can easily lead to much more *FP* predictions than *TP* predictions, and hence a very low precision. To better highlight the importance of small *FPR*, partial AUROC values are sometimes used instead of the full AUROC. For example, Tastan *et al.* (2009) propose statistics like *R50*, *R100*, *R200*, and *R300* that measure the area under the ROC curve until reaching a *FP* equal to 50, 100, 200, and 300 respectively.

Another summary statistic of a ROC curve is the Youden index Fluss *et al.* (2005), which is defined as the maximal value of $TPR - FPR$ over all possible confidence thresholds. It corresponds to the maximal vertical distance between the ROC curve and the diagonal. The Youden index ranges between 0 (corresponding to a random classifier) and 1 (corresponding to a perfect classifier). This statistic was used for example in Hempel *et al.* (2011) to assess gene regulatory network inference methods.

3.2.3 Precision-recall curves

Precision-recall (PR) curves plot the precision as a function of the recall (equal to the TPR), when varying the confidence threshold. As for the ROC curve, varying the threshold of one position in the ranked list of predictions corresponds to predicting one more pair of nodes as interacting. If the pair truly interacts, then the recall increases by $1/p$ while the precision increases by $\frac{FP}{predP(predP+1)}$. On the contrary, if the pair actually does not interact, then the recall does not change and the precision decreases by $\frac{TP}{predP(predP+1)}$. The value of the horizontal step in PR is logically equal to the value of the vertical step in the ROC curve. The value of the vertical step is however not constant (**Table 3.3** and **Figure 3.1B**).

A perfect classifier would give a PR curve passing through the point $(1, 1)$, while a random classifier would have an average precision equal to $\frac{P}{P+N}$ (dotted line in **Figure 3.1B**). All PR curves end at the point $(1, P/(P+N))$ corresponding to predicting all pairs as positive. When all pairs are predicted as negative, recall is 0 but the precision is actually undefined. The coordinates of the first point of the PR curve will therefore be $(1/p, 1)$ if the most likely prediction is actually positive, and $(0, 0)$ otherwise. To make all PR curve defined on the full $[0, 1]$ interval, one sometimes adds a pseudo point to the curve at $(0, 1)$.

The PR curve is also often summarized by the area under the curve (AUPR). The AUPR is sometimes called MAP, for Mean Average Precision Manning *et al.* (2009); Tastan *et al.* (2009). Like for the AUROC, one typically assumes that the higher the AUPR, the better is the classifier, with the AUPR of a perfect classifier equal to 1 and the AUPR of a random classifier close to $P/(P+N)$. In practice, the AUPR can be computed from the curve completed with the additional pseudo-point

Table 3.4: We took the same example as in **Table 3.3**, but triplicated the number of negative objects. The objective is to evaluate the sensitivities of the curves to the class imbalance.

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Actual values	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Constants: $P = 4$, $N = 18$

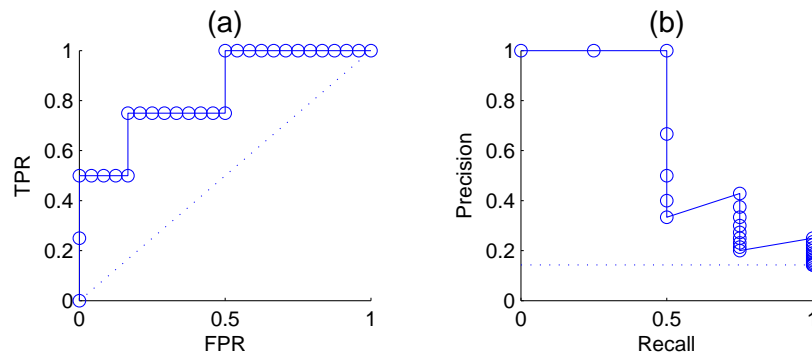


Figure 3.2: ROC curve (A) and PR curve (B) for the scores in **Table 3.4**, where negative examples were triplicated with respect to **Table 3.3**. The comparison with the curves in **Figure 3.1** shows that the ROC curve is unchanged and that the PR curve degrades, as a consequence of tripling the negatives.

or not. In the second case, one can rescale the area by dividing it by $1 - 1/p$ so that its values are equal to 1 for a perfect classifier. Note that it is important to report exactly on which approach was used to compute the AUPR as it can make a significant difference when the number of positives is very small. For example, the AUPR of the PR curve of **Figure 3.1** is equal to 0.81, 0.75, and 0.56 respectively with the pseudo-point, without the pseudo-point but with rescaling, and without the pseudo-point and without rescaling.

3.2.4 Comparison of ROC and PR curves

It is impossible to say which curve, between ROC and PR curves, is a better way to illustrate the performance of a model. Each curve has its own advantages, and the choice of it will depend of the use we want to do of the predictions.

Sensitivity to class imbalance

An important difference between ROC and PR curves is their different sensitivities to the ratio between positives and negatives (class imbalance) among the tested pairs: a ROC curve is independent of the precise value of this ratio, while a PR curve is not. To illustrate this fact, we triplicated every negative examples in the ranked list of predictions of **Table 3.3** and plotted the new ROC and PR curves in **Figure 3.2**. As expected, we obtained exactly the same ROC curves, while the PR curves are different. This happens because, at fixed recall, a large change in FP will lead to no change in the FPR used in ROC curves (because to total number N of negatives will increase in the same proportion), but to a large change in the precision used in PR curves (Davis and Goadrich, 2006).

The independence with respect to the particular content of the test sample in terms of positives and negatives is actually the main advantage of the ROC curve over the PR curve when it comes to compare different classification methods (Fawcett, 2006). ROC curves allow to compare classification methods whatever will be the ratio between positives and negatives expected when practically applying the model. Because of this independence however, ROC curves do not really emphasize a particular intervals of values of this ratio and therefore favor methods that are good for a large range of such values. If one knows for example that the ratio between positives and negatives will be very low when applying the classification model, then one is typically only interested in the bottom-left part of the ROC curve. PR curves, on the other hand, provide a better picture of the performance of a method when the ratio between positives and negatives in the test data is close to the ratio one expects when practically applying the model.

Let us give a concrete numerical example to illustrate this idea. Yu *et al.* (2012) present a ROC curve obtained from their model in which, when the TPR reaches 40%, the FPR is as low as $\sim 2\%$, and when the TPR is 60% the FPR is still as low as $\sim 4\%$. A priori, this results looks good as it means that their model is able to find half of all positive interactions while only selecting a small percentage of negative interactions. Assuming that there is no class imbalance and that the network contains as much positive pairs as negative ones, these two points corresponds to 95% and 94% precisions for respectively 40% and 60% recalls. Now let us assume that there is a string class imbalance and that the network contains 100 times more negative pairs than positive ones. Then precision values fall to 17% and 13% respectively for 40% and 60% recalls. In this latter case, the proposed model seems less good from the perspective of the PR curve than from the perspective of the ROC curve.

The dependence of the PR curve on the ratio between positives and negatives can also be seen as a drawback. First, it means that PR curves (and their associated AUPR) obtained from different datasets can not really be compared when the ratio P/N is very different. This is a limitation if one wants to compare the performance of a method across several networks for example. Second, because of this dependence, it is important that the ratio of positive and negative interactions in the subset of pairs used to validate the method is representative of the final application of the method. Otherwise, the PR curve will not provide a realistic evaluation of the method. Note however that it is possible to adapt a given PR curve to another ratio between positives and negatives than the one adopted to generate it (Hue *et al.*, 2010). Let us assume a PR curve estimated from P_1 positives and N_1 negatives and let us estimate from this curve a new curve corresponding to $P_2 = P_1$ positives and $N_2 \neq N_1$ negatives respectively drawn from the same distribution as the original P_1 positives and N_1 negatives. For a given confidence threshold, the recall (i.e., the proportion of positives greater than the threshold) is unchanged since $P_1 = P_2$. Denoting by TP_1 and TP_2 (resp. FP_1 and FP_2) the number of positives (resp. negatives) with a score greater than the threshold in both cases, we have $precision_1 = \frac{TP_1}{TP_1+FP_1}$ and $precision_2 = \frac{TP_2}{TP_2+FP_2}$. Given that $TP_1 = TP_2$ and $FP_2 = \frac{N_2}{N_1} FP_1$ in average, it is easy to show that $precision_1$ and $precision_2$ are related as follows:

$$precision_2 = \frac{precision_1}{precision_1 + \frac{N_2}{N_1}(1 - precision_1)}. \quad (3.1)$$

Using this formula, one can thus approximate the PR curve for an arbitrary ratio between positives and negatives from the knowledge of at least one PR curve (corresponding to an arbitrary ratio). Note that this PR curve can also be derived from a ROC curve and the knowledge of the actual

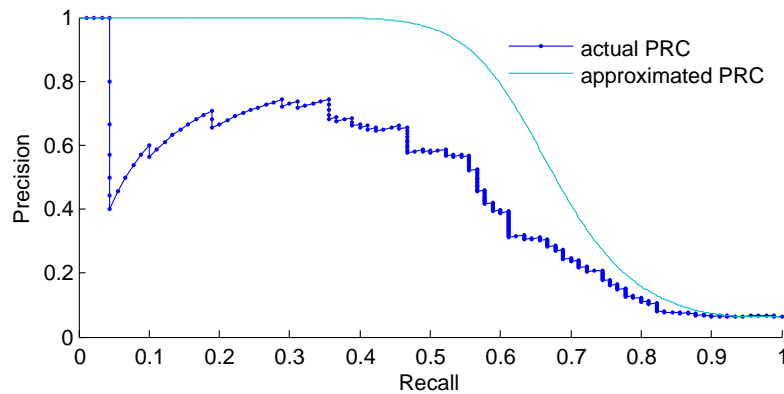


Figure 3.3: Predictions on a small dataset gives an uneven precision-recall curve, which can not be approximated by the method in (Brodersen *et al.*, 2010). The big jumps of the beginning of the curve result from the fact that the classifications of the top ranking are the most weighted.

ratio $\frac{P}{N}$ using the fact that, for any confidence threshold, we have:

$$precision = \frac{TPR}{TPR + FPR \cdot \frac{N}{P}}, \quad (3.2)$$

and that both TPR and FPR are known from the ROC curve.

Unstability of precision-recall curve

Another drawback of the PR curve is the potential unstability of the precision for small recall values. Indeed, for small values of $predP$, the vertical changes of the curve from one confidence threshold to the next can be very huge, independently of the size of the dataset. This is obviously more noticeable when the value of P is small because the horizontal changes are then also relatively large. This unstability makes the estimation of the true PR curve highly imprecise (Brodersen *et al.*, 2010). It is however actually a direct consequence of the stronger focus put by the PR curve on the top of the ranking with respect to the ROC curve.

Brodersen *et al.* (2010) propose to get around this problem by estimating the curve on the basis of a simple distributional assumption about the decision values (see more details in their paper). We tested this model on one small biological network: a drug-protein interaction network in which proteins are nuclear receptors (Yamanishi *et al.*, 2008). There are 54 drugs, 26 proteins and 90 interactions in this network. We performed a 3-fold cross-validation on pairs (see Section 3.3.1), with a global approach and extremely randomized trees. The resulting PR curve and its approximation using Brodersen *et al.* (2010)'s method are shown in **Figure 3.3**. The original PR curve is spiky and uneven. The approximated curve is indeed much smoother but it does not fit at all the original curve. The mismatch between the two curves comes from the assumption made by the method that decision values follow two independent Gaussian distributions. This assumption is clearly violated for this problem, as the two distributions are closer to exponential distributions than to Gaussian distributions (figure not shown).

Davis and Goadrich (2006) present another way to make a PR curve smoother and less spiky. For that purpose, they extended the notion of *achievable curve* from ROC curves to PR curves.

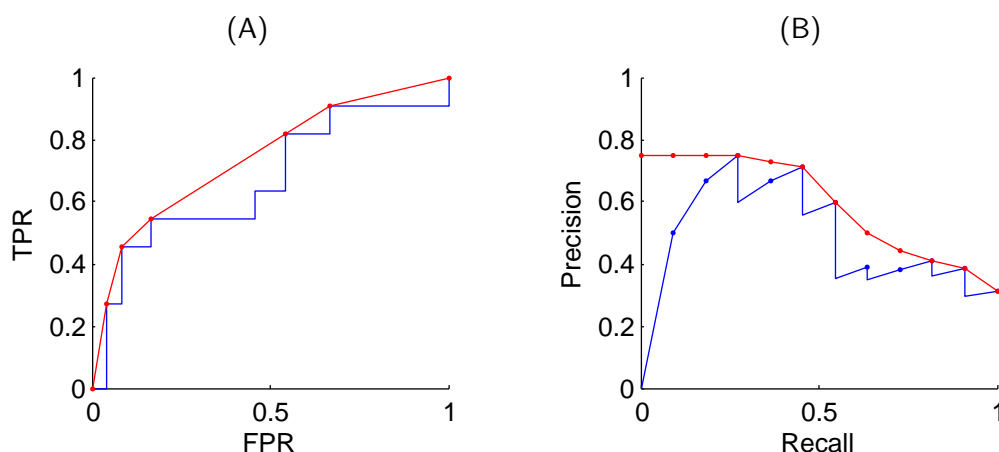


Figure 3.4: Examples of interpolation between points in ROC and precision-recall spaces. The resulting red curves are achievable curves.

In the ROC space, any point on the line segment between two points can actually be achieved by randomly using one of the two classifiers corresponding to these two points, where each of the two classifiers is selected with a probability proportional to its relative distance to the point on the line. From this property, one can deduce that any points on the *convex hull* of a ROC curve can also be achieved and this convex hull is thus the best legal curve that can be built from a set of ROC points (see **Figure 3.4A** for an example).

Using a similar idea, one can also interpolate between two points of a PR curve, by randomly selecting any of the two classifiers corresponding to these two points with a probability ranging from 0 to 1. This idea can be applied to turn any PR curve into some new maximum achievable curve that is smoother than the original curve (see **Figure 3.4B** for an example). Note that, while the interpolated curve between any two points is a straight line in the case of a ROC curve, it is non linear in the case of a PR curve. It can be approximated however by a piecewise linear curve, as in **Figure 3.4B**, where each knot corresponds to a different TP value (see Davis and Goadrich (2006), and also Section 3.6.2 for more details). Note that through this interpolation, the PR curve can always be turned into a monotonically decreasing curve, even if it originally increases at its beginning (as this is the case for the PR curve in **Figure 3.4B**). Indeed, let us suppose that the maximum precision value pr is achieved for a recall value r , and that these precision and recall values correspond to i examples predicted as positives. Then, a precision value pr can be achieved for every recall values lower than r by simply assuming that the order of the i first examples in the ranking is randomized. Every local minimum can also be removed by interpolating the curve between the left and right local maxima that are surrounding this minimum. The decreasing nature of the maximum achievable PR curve will be exploited later in Section 3.6.

Connexion between ROC and PR curves

Despite these differences, it is interesting to note that a deep connection exists between the ROC and the PR spaces, in that a model dominates another model in the ROC space if and only if it dominates the same model in the PR space (Davis and Goadrich, 2006). In practice however, it is often the case that a model does not dominate another model over the whole ROC and PR spaces

and it might thus happen that a method's AUROC is greater than another method's AUROC, while the opposite is true concerning the AUPR.

3.2.5 Other measures and curves

ROC curves and PR curves are the most popular ways to estimate the performance of biological network inference methods, but some other measures and curves can also be found in the literature.

Lift charts

Lift charts (or cumulative lift charts), often used in marketing (Witten and Frank, 2005), plot the TPR, or recall, as a function of the RPP (rate of positive predictions), when varying the confidence threshold. As an illustration, the lift chart relative to the predictions of **Figure 3.1** is shown in **Figure 3.5A**. A perfect classifier would give a curve going through the points $(0, 0)$, $(p/p+n, 1)$ and $(1, 1)$, while a random classifier would be equal to the diagonal connecting the two points $(0, 0)$ and $(1, 1)$.

For example, Geurts (2011) used a lift chart to evaluate the performance of supervised methods for the prediction of regulatory networks, and Yabuuchi *et al.* (2011) for the prediction of compound-protein interactions. Lift charts explicitly shows the number of positive predictions (expressed as a percentage of all possible interactions) that one needs to accept to retrieve a given percentage of all truly positive interactions (recall). This is an important information when one is looking at the experimental validation of the predictions: a method that dominates another in terms of lift chart would require to experimentally test less interactions to achieve a given recall.

Note that when the number of positive examples is much smaller than the number of negative ones, as it often happens in biological networks, there is not much difference between the ROC curve and the lift chart. Indeed, in a lift chart, adding one more pair as predicted positive increases the TPR by $1/P$ and the RPP by $1/P+N$ if this pair truly interacts, and does not affect the TPR and increases RPP by $1/P+N$ if the pair does not interact. In the first case (true positive pair), the slope of curve for this step is equal to $(P+N)/P$, which goes to infinity when $N \gg P$. In the second case (false positive pair), if the ratio P/N gets close to zero, the horizontal step will get close to $1/N$. These values are the same values as the ones obtained in a ROC curve. We can then conclude that when the ratio P/N gets close to zero, the lift chart gets close the ROC curve.

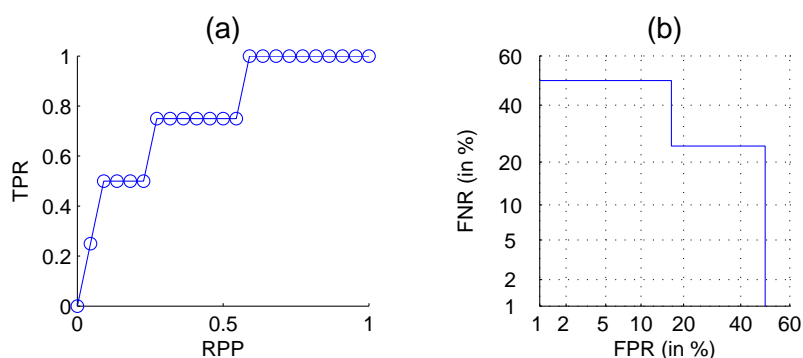


Figure 3.5: From the same example as in **Table 3.4**, we computed a lift chart **(A)** and a DET curve **(B)**.

DET curves

Detection error tradeoff (DET) curves plot the two types of errors versus each other, i.e., FNR as a function of FPR (Martin *et al.*, 1997). In addition, the two axis are log scaled. Without this rescaling, a DET curve would be equivalent to a ROC curve (because $FNR = 1 - TPR$). The interest of the log scale is to expand the lower left part of the curve (which corresponds to the upper left part of the corresponding ROC curve), which as argued in Martin *et al.* (1997) makes the comparison between different methods easier. An example of the DET curve relative to the predictions in **Figure 3** is shown in **Figure 4 (B)**. DET curves were used in Brunner *et al.* (2012) to evaluate classification methods working on pairs of objects.

Correlation coefficient

Several authors (Nijjima *et al.*, 2011; Lapins and Wikberg, 2010; Junaid *et al.*, 2010; Li *et al.*, 2009) use a correlation coefficient for the evaluation of the performance of network inference methods. In this context, the latter is defined as

$$Q^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

where the sum runs over all tested pairs, y_i and \hat{y}_i are the true and predicted value corresponding to the i th pair and \bar{y} is the average value of y_i . Q^2 values vary between 0 and 1, with $Q^2 = 1$ for a perfect classifier.

Average normalized rank

The average normalized rank is another way to compare the performance of different classifiers (Karni *et al.*, 2009; Geurts, 2011). It computes the average rank of all actual positives in the ranking of all pairs according to their confidence score, and then divide it by the total number of pairs. Obviously smaller is the average rank and better is the model.

3.2.6 Discussion

Biological network inference problems, as binary classification problems, are usually very much imbalanced in favor of the negative class, as the proportion of interacting pairs among all possible pairs is very small. Given the discussion in Section 3.2.4, this speaks in favor of the PR curve over the ROC curve. Let us nevertheless consider three typical scenarios related to the use of supervised network inference techniques and discuss the most appropriate use of these measures in each of these scenarios:

- *Development of new supervised network inference methods:* when trying to design a new supervised network inference method, one needs to assess its performance against existing methods, either on a specific target biological network if the method is specialized or on several networks if the method is generic. In this scenario, one has typically no specific application of the method in mind and the combination of both ROC and PR curves can be a good idea. While AUROC and AUPR summary values can be used for comparison purpose, it is always useful to actually report full ROC and PR curves to better characterize the areas of the ROC and PR where the new method dominates competitors.

- *Prioritize interactions for experimental validation:* From a ranking of all the pairs from the most likely to interact to the less likely to interact, a biologist may want to validate experimentally the top-ranked pairs, i.e. the potentially new interacting pairs. More locally, he also may want to find the nodes (e.g., genes/proteins) the most likely to interact with a specific node of special interest for him. In this scenario, the biologist probably wants to find the best tradeoff between the number of true interactions he will find through the experimental validation and the cost associated to this validation. The former is measured by the recall and the latter is typically proportional to the *RPP*, which suggests the use of a lift chart. In addition, if the goal is also to minimize the rate of unsuccessful validation experiments (i.e., the precision), then also looking at the PR curve might be a good idea.
- *Global analysis of the predicted network:* We may want to use the top-ranked pairs to create a new network, or to complete an already known network, for visualization or a more global analysis of its main statistics. These analyzes can be helpful, for example, to discover new therapeutic indication or adverse effect of old drugs in a drug-protein-interaction network (Cheng *et al.*, 2012), to find new pathways in a metabolic network, to highlight clusters of co-expressed genes in a regulatory network, or to find protein sharing similar functions in a protein-protein interaction network. In these cases, we need to find the best possible tradeoff between precision (not to infer wrong things) and recall (to maximize the coverage of the true network). This tradeoff can be found from a PR curve. For example, one could derive from the PR curve the lowest confidence threshold corresponding to a precision greater than 50%.

3.3 Evaluation protocols

Given a learning set LS_p of pairs labeled as interacting or not, the goal of the application of supervised network inference methods is to get a prediction for all pairs not present in LS_p (or a subset of them depending on the application). In addition, one would like to compute an estimate of the quality of these predictions as measured with any of the metrics defined in the previous section. To obtain such estimation, one could rely only on the learning set LS_p as nothing is known about pairs outside this set by construction.

Standard supervised classification methods are typically validated using cross-validation, i.e. leaving part of the examples in the learning sample aside as a test set, training a model from the remaining examples, and testing this model on the test set (and possibly repeat this procedure several times and average). Applying cross-validation in the context of network inference, where we have to classify pairs, needs special care (Park and Marcotte, 2012). Indeed, the predictive performance of a method for a given pair highly depends on the availability in the training data of interactions involving any of the two nodes in the tested pair; It is typically much more difficult to predict pairs with nodes for which no example of interactions are provided in the training network.

As a consequence of this, pair predictions have to be partitioned into four sets, depending on whether the nodes in the pair to predict are represented or not in the learning sample of pairs LS_p . Denoting by LS_c (resp. LS_r) the nodes from U_c (resp. U_r) that are present in LS_p (i.e. which are involved in some pairs in LS_p) and by $TS_c = U_c \setminus LS_c$ (resp. $TS_r = U_r \setminus LS_r$) unseen nodes from U_c (resp. U_r), the pairs of nodes to predict (i.e., outside LS_p) can be divided into the following four families:

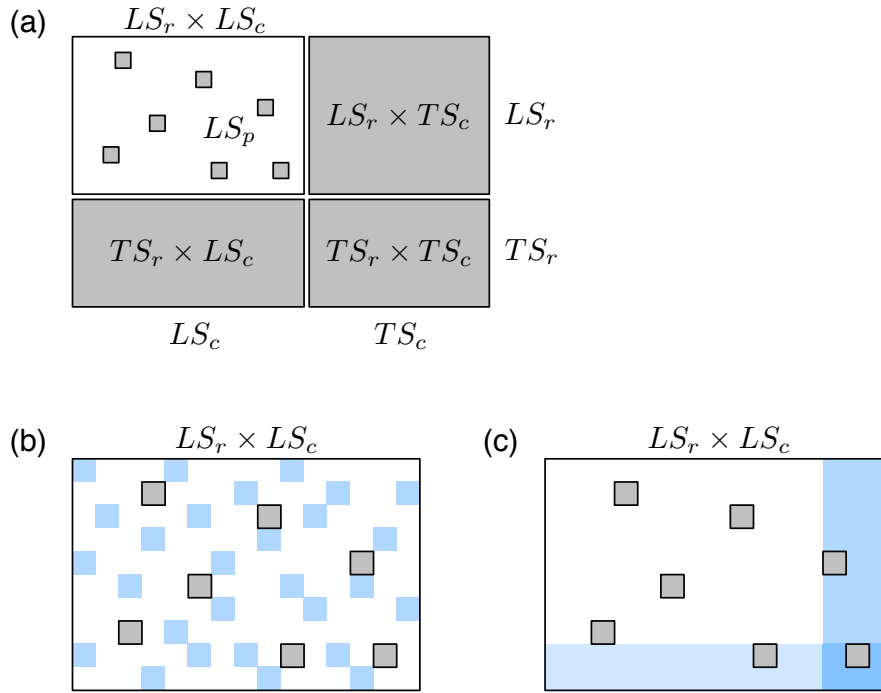


Figure 3.6: Schematic representation of known and unknown pairs in the network adjacency matrix (A) and of the two kinds of cross-validation, cross-validation on pairs (B) and cross-validation on nodes (C). In (A): known pairs (that can be interacting or not) are in white and unknown pairs, to be predicted, are in gray. Rows and columns of the adjacency matrix have been rearranged to highlight the four families of unknown pairs described in the text: $LS_r \times LS_c$, $LS_r \times TS_c$, $TS_r \times LS_c$, and $TS_r \times TS_c$. In (B),(C): pairs from the learning fold are in white and pairs from the test fold are in blue. Pairs in gray represent unknown pairs that do not take part to the cross-validation.

- $(LS_r \times LS_c) \setminus LS_p$: predictions of (unseen) pairs between two nodes which are represented in the learning sample.
- $LS_r \times TS_c$ or $TS_r \times LS_c$: predictions of pairs between one node represented in the learning sample and one unseen node, where the unseen node can be either from U_c or from U_r .
- $TS_r \times TS_c$: predictions of pairs between two unseen nodes.

These pairs are represented in the adjacency matrix in **Figure 3.6A**. In this representation, the rows and columns of the adjacency matrix have been ordered, without loss of generality, in order to make nodes from LS_r and LS_c appear first in the ranking and as a consequence, all four groups define rectangular and contiguous subregions of the adjacency matrix. Such ordering is always possible but the respective sizes of the four groups of pairs that this ordering defines is of course very much problem dependent. Thereafter, we simplify the notations by dropping the subscript r and c and denote the prediction sets as $LS \times LS$, $LS \times TS$, $TS \times LS$ and $TS \times TS$. In the case of an homogeneous undirected graph, only three sets can be defined as the two sets $LS \times TS$ and $TS \times LS$ are obviously confounded.

Typically, one expects different prediction performances for these different kinds of pairs and in particular, that $TS \times TS$ pairs will be the most difficult to predict since less information is available at training about the corresponding nodes. In consequence, we need ways to evaluate the quality of the predictions of these four groups separately. Below, we first present the two main cross-validation procedures that have been proposed in the literature to evaluate supervised network inference methods and discuss which of these four kinds of predictions these procedures are evaluating (Sections 3.3.1 and 3.3.2). We then proceed with some suggestions on how to practically assess network inference methods (Section 3.3.3) and give an illustration on an artificial gene regulatory network (Section 3.3.4).

3.3.1 Cross-validation on pairs

The most straightforward way to generate the groups needed for the cross-validation (i.e. the learning and test sets) is to randomly select pairs from all the known pairs in LS_p (see **Figure 3.6B**). For example, in a specific step of a 10-fold cross-validation, 90% of all the pairs from LS_p are chosen to be in the learning set, while the remaining 10% are then part of the test set. We call such cross-validation *cross-validation on pairs*. Many papers from the literature on supervised network inference only consider this sampling method (see e.g. Qi *et al.*, 2006; Chang *et al.*, 2010; Yabuuchi *et al.*, 2011; Park and Marcotte, 2012).

With CV on pairs, each test set could in principle mix pairs from the four groups aforementioned. If LS_p is relatively dense however (i.e., there are only very few or no pairs in $LS_r \times LS_c \setminus LS_p$), the chance to have a node in a test set pair not present in any learning set pair will be very low. The test set will then be largely dominated by pairs from the $LS \times LS$ group. In this case, one can thus only consider the performance evaluated by CV on pairs as representative of the performance for the $LS \times LS$ pairs. When used to assess the global performance of a method however, CV on pairs will in general give too optimistic estimates.

To obtain an estimate of the four kinds of predictions using CV on pairs, one could partition the pairs in the test fold into the four groups and then estimate the performance for each group separately. The cross-validation scheme proposed in the next section provides however a more natural way to assess the three types of predictions involving the TS . CV on pairs should thus be reserved for the evaluation of $LS \times LS$ pairs. For that purpose, removing pairs in the test folds that do not belong to the $LS \times LS$ group might be useful to obtain a better estimate, especially when the size of LS_p is small with respect to the size of $LS_c \times LS_r$.

3.3.2 Cross-validation on nodes

Instead of sampling pairs, several authors have proposed to sample nodes. In the general case of a bipartite graph, the idea is to randomly split both sets LS_c and LS_r into two sets, respectively denoted LS'_c and TS'_c for LS_c and LS'_r and TS'_r for LS_r . The model is then trained on the pairs in $(LS'_c \times LS'_r) \cap LS_p$ and then evaluated separately on three subsets (see **Figure 3.6C**):

- $(LS'_c \times TS'_r) \cap LS_p$ that gives an estimate of the $LS \times TS$ performance,
- $(TS'_c \times LS'_r) \cap LS_p$ that gives an estimate of the $TS \times LS$ performance,
- $(TS'_c \times TS'_r) \cap LS_p$ that gives an estimate of the $TS \times TS$ performance.

In addition, it might be interesting to evaluate the performance on the union of the three previous subsets of pairs to give an idea of the overall performance of the method. Better estimates could also be obtained by averaging results over k splits instead of one, where the different splits can be obtained either by repeated random resampling or by partitioning the two sets into k -folds and considering each fold in turn as a test set. In this latter case, partitioning LS_c and LS_r into k folds will lead to k^2 candidate (LS'_c, LS'_r) pairs for training and (TS'_c, TS'_r) pairs for evaluation but one could select only k of them arbitrarily to reduce the computational burden. The same approach can obviously also be applied to homogeneous graphs to obtain estimate of the $LS \times TS$ and $TS \times TS$ performances.

Cross-validation on nodes has been applied for example in Vert and Yamanishi (2005); Kato *et al.* (2005); Geurts *et al.* (2007) for evaluating $LS \times TS$ and $TS \times TS$ performances for the prediction of a protein-protein interaction network and an enzyme network. Yamanishi *et al.* (2008) performed 10-fold cross-validation on nodes to evaluate the performance of their method for predicting drug-protein interactions. They separated the pairs in the test sets into 4 classes: new drug versus known protein ($LS \times TS$), known drug versus new protein ($TS \times LS$), new drug versus new protein ($TS \times TS$) and the union of all these three classes.

In some applications, one might be interested only in evaluating $LS \times TS$ or $TS \times LS$ predictions and not $TS \times TS$ predictions. In this case, the same procedure can be applied, except that only one set of nodes has to be split. For example, Mordelet and Vert (2008) uses a local approach to predict the regulatory network of the bacteria *E.coli*, considered as a bipartite graph connecting genes and transcription factors. They evaluated the performance of their algorithm by performing 3-fold cross-validation on the genes only, to measure how well one can predict the presence of an interaction between one transcription factor for which we already know several interactions and one gene for which no interaction is known ($LS \times TS$). As another example, Yamanishi *et al.* (2010) focus on drug-protein interaction predictions and evaluate their performances by 5-fold cross-validation on drugs, hereby assessing how well they can predict interactions between a known protein and a new drug. For the same problem, Bleakley and Yamanishi (2009) evaluated their local method by leave-one-out cross-validation performed separately on drugs and then on proteins to evaluate both $LS \times TS$ and $TS \times LS$ predictions.

3.3.3 Discussion

Cross-validation on pairs provide a natural way to estimate $LS \times LS$ predictions, while cross-validation on nodes provide a natural way to estimate $LS \times TS$, $TS \times LS$, and $TS \times TS$ predictions. A global performance assessment of a method can therefore only be obtained by combining these two protocols. This was done only by a few authors (e.g. Yip and Gerstein, 2008; Bleakley and Yamanishi, 2009; Takarabe *et al.*, 2012). The necessity to evaluate all four groups is however problem dependent. Again, when designing a new supervised network inference method, it is useful to report performances for all families separately, as a method can work well for one family and less good for another. If one is interested in the completion of a particular biological network, then the need for the evaluation will depend, on the one hand, on the content of the learning sample LS_p and, on the other hand, on which kinds of predictions the end user is interested in. Indeed, if all nodes are covered by at least one known interaction in LS_p , then there is no point in evaluating $LS \times TS$ or $TS \times TS$ predictions. If LS_p corresponds to a complete rectangular submatrix of the adjacency matrix (i.e., $LS_p = LS_c \times LS_r$), then there is no point in evaluating $LS \times LS$ predictions.

Also, for some applications, the end-user might not be interested in the extension of the network over one of the two dimensions. For example, when inferring a regulatory network, the user might only be interested by the prediction of new target genes for known transcription factors.

Note that all supervised network inference methods are not able to predict all four groups of prediction. We have already mentioned that local methods can not make predictions for $TS \times TS$ pairs. There are also methods that require a learning sample LS_p containing all possible pairs among a subset of nodes and therefore can not make $LS \times LS$ predictions (Kato *et al.*, 2005; Yamanishi and Vert, 2005; Geurts *et al.*, 2007).

In addition to the four groups previously defined, it is also possible to evaluate independently the predictions related to each individual node (to get for example an idea of the quality of the predictions of new target genes for a given transcription factor). This can be achieved by dividing the test folds according to one of the nodes in the pairs and then to assess performance for each partition so obtained. In practice also, the quality of a prediction depends not only on the fact that the nodes in the pair belong or not to the learning sample, but also on the number of pairs in the learning sample that concern these nodes. We can indeed expect that, for a given node, the more interactions or non-interactions are known in the learning sample for this node, the better will be the predictions for the pairs that involve this node. Assessing each node separately can thus make sense to better evaluate this effect. We will illustrate this idea in Section 3.3.4.

When using k -fold cross-validation to estimate ROC or PR curves, one question we have not addressed so far is how to aggregate the results over the different folds. There are several ways to do that. If one is interested only in AUROC or AUPR values, then one could simply average AUROC or AUPR values over the k folds. If one wants to estimate the whole ROC or PR curves, there are two ways to obtain them: first, by averaging the k curves to obtain a single one, second by merging pairs from the k test folds with their confidence score and building a curve from all these pairs. In the first case, there are several alternative ways to average ROC (and PR) curves. One of them is to sample the x-axis in each curve and then average the k y-axis values corresponding to these points (this is called vertical averaging in Fawcett (2006)). Merging all predictions together is easier to implement but it assumes that the confidence scores obtained from the k different models are comparable, which is not trivially true for all methods. Note that our own practical experience shows that there are only very small differences between these two methods of aggregation and we usually prefer to average the individual ROC curves not to have to address the question of the compatibility of the confidence scores.

Finally, we have seen in section 3.2.4 that PR curves depend on the ratio between positives and negatives. This dependence should be taken into account when performing cross-validation. If CV on pairs and CV on nodes use uniform random sampling, resp. of pairs and of nodes, to define the test folds, then they implicitly assume that the ratio between positives and negatives is the same in the test fold as in the learning sample of pairs. This seems a reasonable assumption in most situations but if one expects a different ratio among the predictions, then the procedure developed in Section 3.2.4 can be used to correct the PR curve accordingly. We will discuss in Section 3.4 one situation that would require such adjustment.

3.3.4 Illustration

In this section, we will illustrate the use of cross-validation with experiments on an artificial network. An artificial network was chosen so that it is possible to accurately estimate performance and

therefore assess the different biases discussed in the chapter. The chosen network is the artificial regulatory network simulated in the context of the DREAM5 network inference challenge (Marbach *et al.*, 2012). This network is an artificial (bipartite) regulatory network, composed of 1565 genes, 178 transcription factors (TF) and 4,012 interactions, corresponding to 1.4% of all the pairs. The network has to be inferred from 804 artificial microarray expression values obtained in various conditions and mimicking typical real microarray compendia. To provide experiments on a homogeneous network as well, we transformed this network into a co-regulatory network composed of 1565 genes and in which there is an interaction between two genes if they are regulated by at least one common TF. The resulting network is composed of 4,191,120 interactions, corresponding to 17.1% of all pairs.

We first carry out a separate evaluation of the different families of pairs, then we evaluate performances for each gene individually, and finally we make an experiment in a more realistic setting to assess the quality of the estimation provided by cross-validation.

Performance over the four families of predictions

We performed a 10-fold cross-validation on both the bipartite and homogeneous networks, with a local approach using Random Forests (Breiman, 2001).

Bipartite network. For the bipartite network, we sample first on pairs, and second on genes and on TFs. The resulting curves and areas under the curves are given in **Figure 3.7AB**.

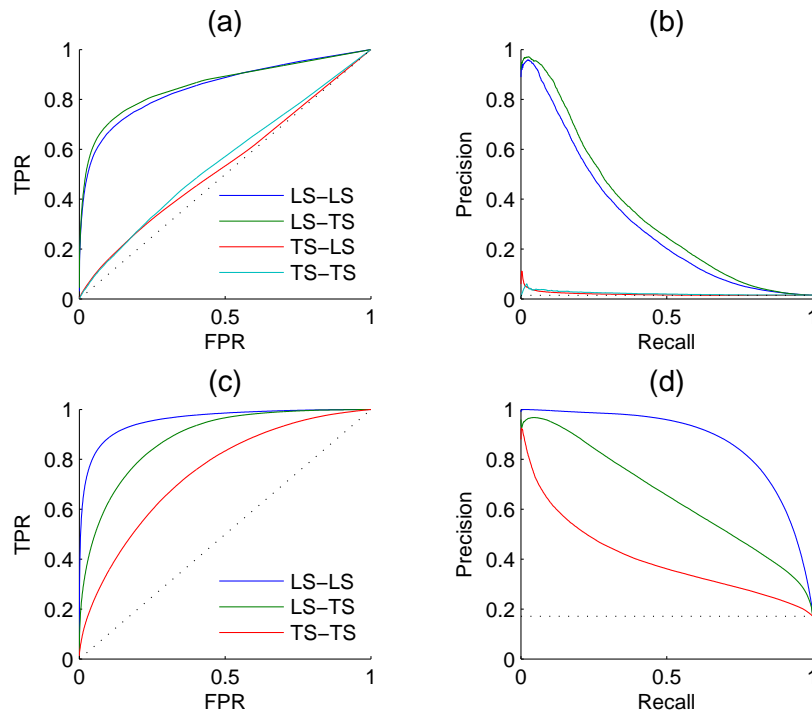
On the other hand, the prediction of pairs involving a TF present in the learning set, and a new gene ($LS \times TS$) gives an AUPR equal to 0.02 and an AUROC equal to 0.53, and the prediction of pairs involving a new TF and a new gene ($TS \times TS$) gives AUPR equal to 0.02 and an AUROC equal to 0.55. These results are barely better than a random classifier (AUPR = 0.01 and AUROC = 0.5). Finding new interactions for a known TF is much easier than finding new interactions for a known gene.

Homogeneous network. For the homogeneous network, we sample first on the pairs and second on the genes. The resulting curves are shown in **Figure 3.7CD**. Prediction of coregulation between two genes belonging to the learning set gives the best AUROC and AUPR. As expected prediction of coregulation between one known gene and one new gene gives less good performance, followed by prediction of coregulation between two new genes.

These two examples clearly highlight the fact that all pairs are not as easy to discover as the others, and that it is thus important to distinguish them during the validation.

Per-node evaluation

As a second experiment, we computed the ROC and PR curves for each of the 178 TFs separately, from the result of the 10-fold cross-validation on genes (bipartite graph). **Figure 3.8** shows the (average) AUROC and AUPR values for all TFs according to their degree. This plot shows that the quality of the predictions differs greatly from one TF to another and that the number of known pairs seems to affect this quality. For low values of degree (lower than about 20), the AUROC globally increases when the degree increases, but for higher values the AUROC does not seem to depend on it. On the other hand, AUPR values globally increase when the degree increases, for all values of TF. We can conclude that the algorithm needs a minimum of known interactions to learn a good



		$LS \times LS$	$LS \times TS$	$TS \times LS$	$TS \times TS$
Bipartite network	(a) AUROC	0.85	0.86	0.53	0.55
	(b) AUPR	0.31	0.34	0.02	0.02
Homogeneous network	(c) AUROC	0.96	0.88	-	0.75
	(d) AUPR	0.88	0.65	-	0.40

Figure 3.7: Top, ROC curves (A) and PR curves (B) for the four groups of predictions obtained by 10-fold cross-validation on the DREAM5 artificial gene regulatory network. The performance of prediction of a pair involving a gene and a TF present in the learning set ($LS \times LS$) is as good as the performance of prediction of a pair involving a gene absent and a TF present in the learning set ($LS \times TS$). On the contrary, predicting an interaction involving a new TF is much more difficult ($TS \times LS$ and $TS \times TS$). Bottom, ROC curves (C) and PR curves (D) obtained by 10-fold cross-validation on the corresponding DREAM5 co-regulatory network. Predictions on pairs involving two genes from the learning set are the best, while predictions on pairs involving two genes from the test set are the worst.

model, and that above a certain threshold, there is no need to increase further the number of known interactions to improve the AUROC.

Note that performance could also be assessed for each gene separately but given the very poor $TS \times LS$ performances observed in the previous section, we expect also very poor results from such experiment.

A more realistic setting

The goal of cross-validation is to estimate, from the training subnetwork, the performance one expects on the prediction of new interactions. We carried out another experiment on the DREAM5 artificial network to evaluate the quality of the estimation obtained by cross-validation in a realistic setting. In this setting, we assume that the known pairs are obtained by first randomly drawing $2/3$ of the genes and $2/3$ of the transcription factors and then randomly drawing $2/3$ of all interacting and non-interacting pairs between these genes and transcription factors (**Figure 3.9**). The resulting training set thus contains $(2/3)^3 \simeq 30\%$ of all possible pairs and the goal is to predict the remaining 70% pairs, which are divided into:

- $LS \times LS$ composed of $2/3 \cdot 2/3 \cdot 1/3 \simeq 15\%$ of all possible pairs
- $LS \times TS$ and $TS \times LS$ composed each one of $2/3 \cdot 1/3 \simeq 22\%$ of all possible pairs
- $TS \times TS$ composed of $1/3 \cdot 1/3 \simeq 11\%$ of all possible pairs

Two validation experiments were performed. First, we evaluated the performance of the (global) Random Forests method by cross-validation across pairs and across nodes on the 30% of known pairs (experiment A). Second, we trained local models based on Random Forests on the known pairs and we evaluated them on the 70% of pairs not used during training (experiment B). Experiment A is therefore supposed to provide a cross-validation estimate of the true performance as estimated by experiment B. The resulting ROC and PR curves obtained from these two experiments for the $LS \times LS$ and $LS \times TS$ families are shown in **Figure 3.10**. As expected, for both kinds of predictions, the curves obtained by the two experiments are very similar, with a very slight advantage to experiment B. This small difference comes from the fact that the number of pairs in the learning set of experiment B is 10% greater than the number of pairs in the learning sets of experiment A (because of 10-fold cross-validation).

3.4 Lack of negative examples

In biological networks, it is common that no truly non interacting pairs are available. Indeed it is often impossible for biologists to experimentally support the lack of an interaction between two nodes. For example you can prove that a specific drug acts on a set of proteins, and you may want to find other proteins being affected by this drug by using machine learning techniques, but you cannot prove that a particular set of proteins is not affected by the drug. This lack of negative examples leads to two main issues for supervised network inference that need to be addressed.

- Standard machine learning methods require both positive (interacting pairs) and negative (non-interacting pairs) examples for training. It is therefore necessary to adopt strategies to cope with this lack of negative examples at the training stage.

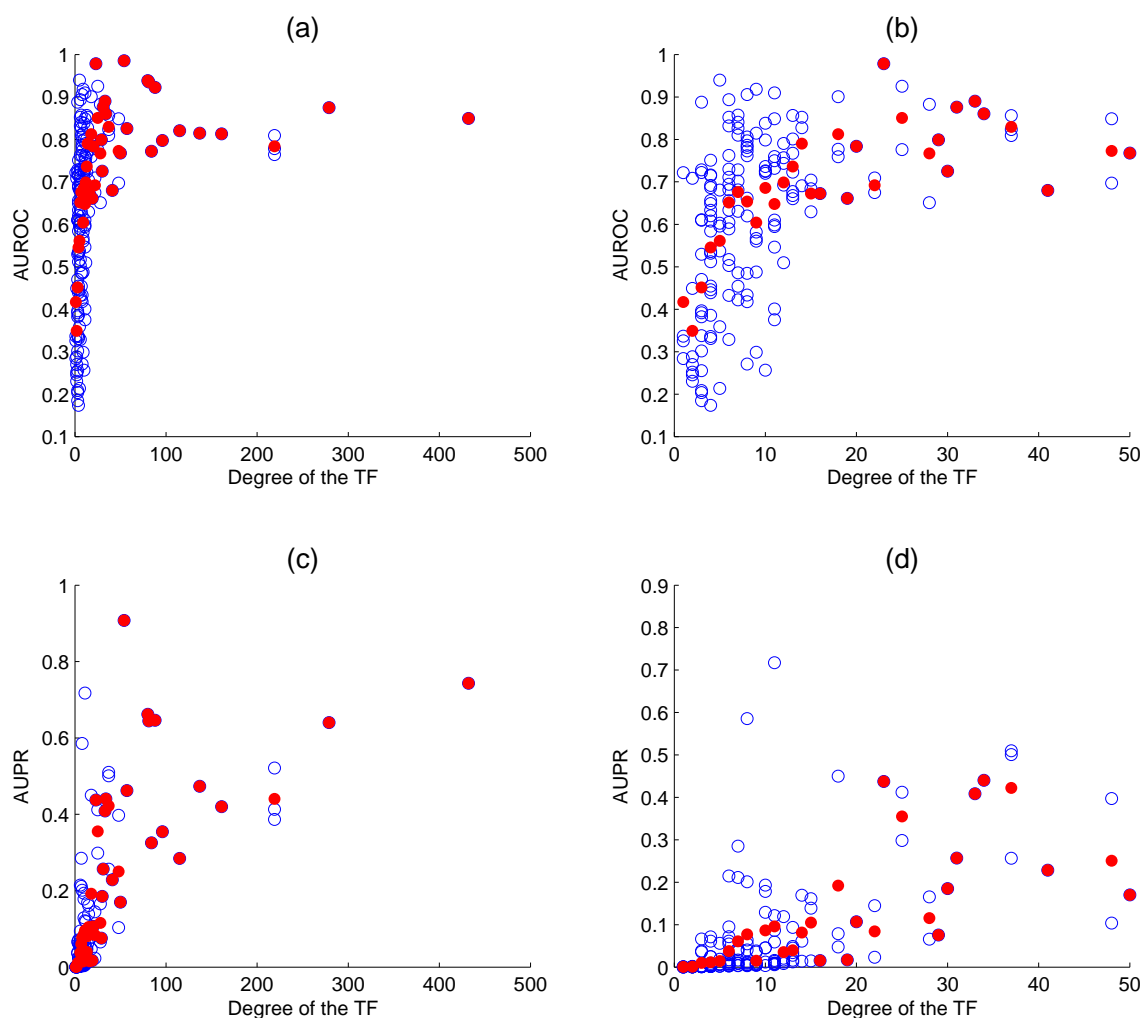


Figure 3.8: AUROC (A,B) and AUPR (C,D) for each transcription factor as a function of its degree (number of targets) on the DREAM5 network. Each value was obtained by 10-fold cross-validation on genes. Each blue point corresponds to a particular TF and plots its average AUC or AUPR value over the 10 folds. Each red point correspond to the average AUC or AUPR values over all TFs of the corresponding degree. Globally, the higher the degree, the higher are the areas under the curve and so the better are the predictions.

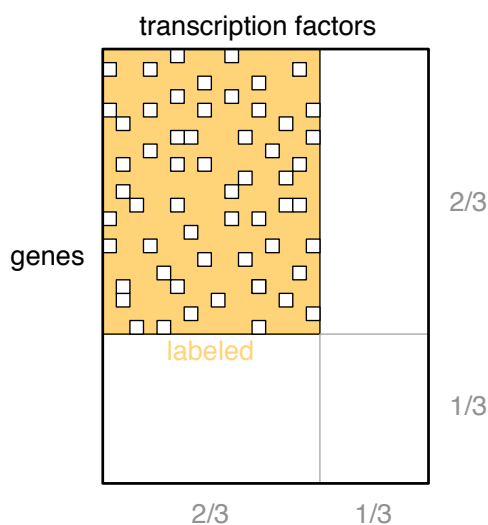


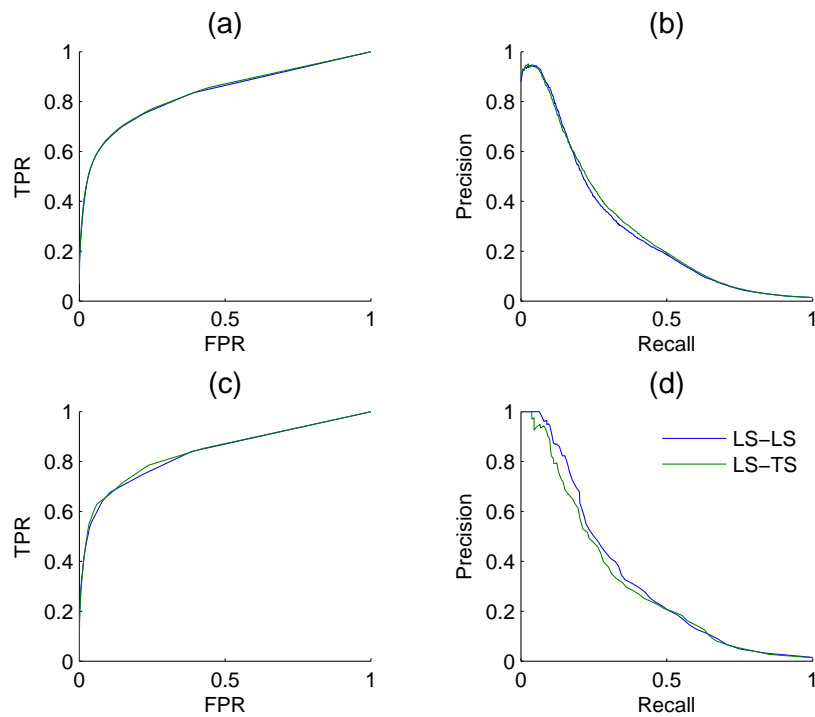
Figure 3.9: To make sure that performance curves computed on labeled data reflect performance that we would obtain by predicting new data, we performed two experiments on the DREAM5 regulatory network, in a more realistic setting: (A) Cross-validations across the considered labeled pairs (orange ones) and (B) prediction of considered unlabeled pairs (white ones) from the labeled ones. Resulting curves are found in **Figure 3.10**.

- Assessing the performance of a method also requires to have access to both true positive and true negative examples, with which to confront the predictions. The impact of the lack of negatives on performance assessment needs to be studied.

We discuss these two issues separately below and conclude with an illustration.

3.4.1 Training a model

Standard supervised machine learning methods require both positive and negative examples for training. The most common way to get around this limitation in the presence of only positive examples is to take as negative examples all, or a subset of, the unlabeled examples, i.e. in our context, considering all or some pairs that have not been measured as interacting as actually non-interacting. This approach has been adopted by most authors in the literature, e.g., in Geurts *et al.* (2007); Mordélet and Vert (2008); Bauer *et al.* (2011); Yip and Gerstein (2008); Yamanishi *et al.* (2008); Takarabe *et al.* (2012); van Laarhoven *et al.* (2011) the authors use all unlabeled pairs as negatives and in Yip and Gerstein (2008); Yabuuchi *et al.* (2011); Hue *et al.* (2010); Chang *et al.* (2010); Yu *et al.* (2012) they use only a subset of them. Although there is a risk that the presence of false negatives in the learning sample will affect the performance of the machine learning method, using only a subset of the unlabeled pairs as negative examples will however reduce very much this risk in the context of biological networks. Indeed, the fraction of positive interactions is expected to be very small in common biological networks, which will lead to only a very small number of false negatives in the learning sample as soon as the size of the negative set is not too large with respect to the size of the positive set. For example, for the protein-protein interaction network of the yeast, it is estimated that 1 pair over 600 is actually interacting (Qi *et al.*, 2006), which corresponds to



		$LS \times LS$	$LS \times TS$
Experiment A	(a) $AUROC$	0.83	0.83
	(b) $AUPR$	0.29	0.31
Experiment B	(c) $AUROC$	0.84	0.84
	(d) $AUPR$	0.33	0.31

Figure 3.10: Comparison of the cross-validation estimates of the $LS \times LS$ and $LS \times TS$ scores, ROC curve in (A) and PR curve in (B), with true score values for the same two families of predictions, ROC curve in (C) and PR curve in (D).

$\sim 0.2\%$ of all the possible pairs. A learning sample composed of 1000 positive and 1000 unlabeled pairs is therefore expected to contain in average only about 2 or 3 false negatives. In addition to the reduction of the number of false negatives, sampling the unlabeled pairs has also the advantage of decreasing the computational cost at the training stage and of improving the class imbalance in the training sample, which affects the performance of some classification methods (Park and Marcotte, 2011; Pandey *et al.*, 2010).

To even further reduce the risk of incorporating false negatives in the training data, one could also replace random sampling from the unlabeled pairs by a selection of a subset of more reliable negative examples using some prior knowledge about the biological interactions of interest. This approach was considered for example in Ben-Hur and Noble (2006) for protein-protein interactions, in Ceccarelli and Cerulo (2009) for gene-transcription factor interactions, and in Yousef *et al.* (2008) for microRNA-gene interactions.

Note that the presence of false negatives is not necessarily detrimental. Elkan and Noto (2008) showed that, under the assumption that the interactions in the learning sample are selected uniformly at random among all interactions, the presence of false negatives in the learning sample will only affect the confidence scores by a constant factor, which will thus leave ROC and PR curves for example unaffected. Although their assumption is quite strong, this nevertheless suggests that the presence of false negatives might not affect too much performances. As an illustration, we run the same experiment as in section 3.3.4 on the DREAM5 regulatory network only turning 10% of positives into negatives when training the model. The AUROC reduces from 0.31 to 0.29 and the AUPR from 0.85 to 0.84, showing that the presence of false negatives only very slightly affects the performance of Random Forests.

One drawback of considering unlabeled pairs as negative pairs for training the model is that the predictions provided by the model for these pairs will be obviously biased towards low confidence scores. One way to obtain unbiased predictions for all unlabeled pairs is to use cross-validation: construct a model using all known positive pairs and a random subset of the unlabeled pairs as negatives, use this model to obtain a prediction for all unlabeled pairs not used during the training stage, and repeat the procedure several times using different subsets of unlabeled pairs until all unlabeled pairs have obtained at least one prediction. Based on this general scheme, Mordelet and Vert (2013) proposed to train several models using small random subsamples of unlabeled pairs, leading to several predictions for each unlabeled pairs that are then aggregated. This approach was applied to the inference of gene regulatory network.

Another approach to deal with the lack of negative examples is to forget about unlabeled examples and exploit machine learning methods, such as one-class SVM (Schölkopf *et al.*, 2001), that can learn a model only from the positive examples. This approach was for example adopted in Yousef *et al.* (2008) to predict miRNA-gene interactions. Machine learning literature also provides several specific algorithms for dealing with positive and unlabeled examples, among which for example Denis *et al.* (2005a); Lee and Liu (2003); Geurts (2011), that could also be used in the context of supervised network inference. Geurts (2011) validated his method for the inference of regulatory networks.

3.4.2 Evaluating a model

The absence of true non interacting pairs in the training data has also an impact on the validation of the model, as the different evaluation measures described in section 3.2 all rely on the availability of a set of known interacting and non-interacting pairs on which to perform the cross-validation.

Like for training, the simplest way to deal with the lack of negatives for validating the model is to consider all unlabeled pairs within the test folds (generated in the context of CV on pairs or CV on nodes) as non-interacting pairs and then estimate ROC or PR curves under this assumption. The presence of false negatives in the gold standard will obviously affect the estimation of the performance.

Effect of false negatives on the curves

Let us try to estimate the effect of false negatives on the PR and ROC curves. For that purpose, let us suppose that the ranking of the examples in a test fold is fixed and then let us compute the change in PR and ROC curve when a proportion x of positives are turned into negatives. The assumption under this model is that false negative examples will get confidence scores distributed similarly as scores of positive examples. We will discuss the relevance of this assumption below.

Given this modification, the error counts in the confusion matrix are modified as follows, in average and for a given confidence thresholds (using the notations of section 3.2):

$$\begin{aligned} P_{new} &= P - P \cdot x = (1 - x)P & N_{new} &= N + P \cdot x \\ TP_{new} &= TP - TP \cdot x = (1 - x)TP & FP_{new} &= FP + TP \cdot x \\ FN_{new} &= FN - FN \cdot x = (1 - x)FN & TN_{new} &= TN + FN \cdot x \end{aligned}$$

From these changes, we can compute the resulting variations in TPR, FPR, and precision that define ROC and PR curves:

$$TPR_{new} = \frac{TP_{new}}{P_{new}} = \frac{(1 - x)TP}{(1 - x)P} = TPR \quad (3.3)$$

$$FPR_{new} = \frac{FP_{new}}{N_{new}} = \frac{FP + TP \cdot x}{N + P \cdot x} > FPR \quad (3.4)$$

$$Prec_{new} = \frac{TP_{new}}{TP_{new} + FP_{new}} = \frac{(1 - x)TP}{(1 - x)TP + FP + TP \cdot x} = (1 - x)Prec < Prec \quad (3.5)$$

(3.4) is valid as soon as the ranking is better than random. Indeed, it can be shown by some straightforward manipulations that (3.4) is equivalent to the following inequality:

$$\frac{TP}{FP} > \frac{P}{N},$$

which is verified for any classifier that is better than random (i.e., a classifier that puts more positives above the confidence threshold than expected at random).

We can thus conclude that the TPR is not influenced by the number of false negatives, that the FPR increases and the precision decreases when the number of false negatives increases. One can thus expect that the introduction of false negatives will systematically degrade both the ROC and the PR curves.

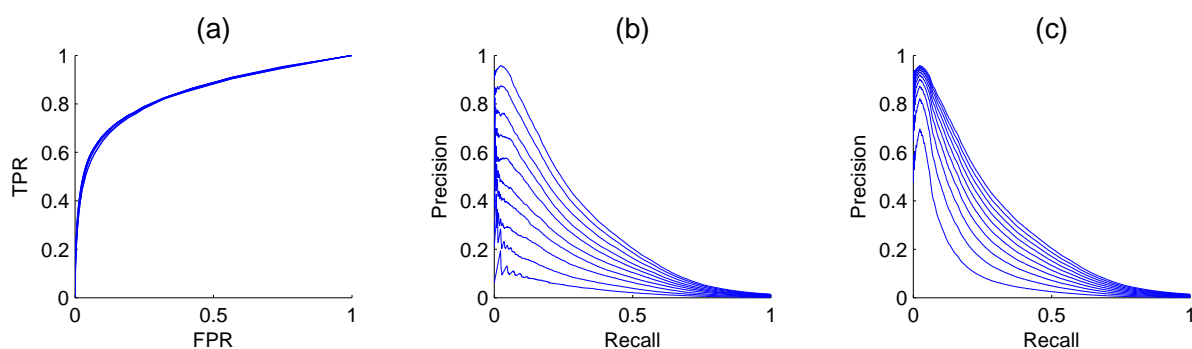


Figure 3.11: Effect of false negatives on ROC and PR curves. We simulated false negatives in the DREAM5 regulatory network, during the testing stage. The ratio of false negatives does not influence the ROC curve (A), but the PR curve (B) decreases while the ratio of positives turned into positives increases. The ratio varies from 0 to 0.9. Curves (C) show the evolution of the PR curve when the ratio P/N is set similarly as in (B). Although the PR curve degrades also in this case, the degradation is not as important as when false negatives are introduced.

As an illustration of the previous discussion, we did some simulations on the DREAM5 regulatory network (see Section 3.3.4). The model was trained with Random Forests with the local approach and we focus our experiment on the $LS \times LS$ pairs. The learning sample was kept unchanged but in each of the 10 CV folds (CV on pairs), we randomly turned a fraction x of positives into negatives, in order to simulate the introduction of false negatives. We tried several proportions $x \in \{0, 0.1, 0.2, \dots, 0.9\}$ and got the curves shown in **Figures 3.11AB**. As expected, the PR curves degrade when the ratio increases. More surprisingly, the ROC curves do not seem to be influenced by the ratio of false negatives. This can be explained by the fact that in equation (3.4), $TP \cdot x$ becomes negligible compared to FP and $P \cdot x$ is negligible compared to N , even for small FPR values as soon as N is large with respect to P .

Actually, there are potentially two effects that play a role in the degradation of the PR curve in **Figures 3.11B**: the introduction of false negatives but also the alteration of class imbalance. Indeed, we have seen in Section 3.2.4 that the PR curve was affected by this ratio. To try to assess both effects separately, we also generated the PR curves obtained from the initial curve by increasing the number of negatives in such a way that the ratio of P/N matches the ratio of P/N in the previous experiment for x ranging from 0 to 0.9. These curves are plotted in **Figures 3.11C**. They are also systematically degraded by the introduction of more negatives but the degradation is not as high as the degradation obtained by the addition of false negatives.

We can conclude from these experiments that PR curves are much more sensitive than ROC curves to false negatives in the true dataset. Interestingly, if we can estimate the ratio x of false negatives, we can modify the PR curve simply by dividing the precision by $1 - x$, to obtain a more realistic PR curve. Note however that this correction only applies under the assumption that false negatives will get scores distributed similarly as positives. This assumption is not unrealistic in practice as we indeed expect that false negatives will be predicted most often as positives (since they are in fact positives). However, it is also possible that for a given biological network, known interactions are the strongest ones (i.e., those with the strongest experimental support) and therefore false negatives will typically correspond to weaker interactions. Their scores, as predicted by network inference

methods, can then be smaller than those of known positives. In this case, the degradation of the PR curve will most probably be somewhere in between curves in **Figures 3.11B** and **Figures 3.11C**. Note that even though PR curves are affected by the introduction of false negatives, this is not really problematic when it comes to compare different inference methods on the same networks, as all methods will be affected in the same way by these false negatives. In this case, correcting the PR curve is not necessary.

Finally, it is useful to remind here that the ratio between positives and negatives used to evaluate PR curves should be as close as possible to the expected ratio in the pairs to predict. Indeed, one could be tempted to estimate performance by cross-validation on pairs on the positives and the selected negatives (randomly or from prior knowledge). The resulting PR curves will be however representative only for the given observed ratio between positives and negatives. If this ratio is different from the expected one, then one should apply the PR curve correction presented in Section 3.2.4.

Effect of false positives on the curves

False negative are common in a dataset with unlabeled data taken as negatives. Unfortunately, in biological networks, pairs can also erroneously be labeled as positive. To evaluate the effect of these false positives on curves, we did the same experiment as here above, but this time converting a fraction of negatives into positives. The modifications that occur in the confusion matrix are then the following:

$$\begin{aligned} P_{new} &= P + N \cdot x & N_{new} &= N - N \cdot x = (1 - x)N \\ TP_{new} &= TP + FP \cdot x & FP_{new} &= FP - FP \cdot x = (1 - x)FP \\ FN_{new} &= FN + TN \cdot x & TN_{new} &= TN - TN \cdot x = (1 - x)TN \end{aligned}$$

From that, we can calculate the resulting variations of the values of the ROC and PR curves:

$$\begin{aligned} TPR &= recall = \frac{TP}{P} & TPR_{new} &= \frac{TP + FP \cdot x}{P + N \cdot x} < TPR \\ FPR &= \frac{FP}{N} & FPR_{new} &= \frac{(1 - x)FP}{(1 - x)N} = FPR \\ Prec &= \frac{TP}{TP + FP} & Prec_{new} &= \frac{TP + FP \cdot x}{TP + FP \cdot x + (1 - x)FP} = Prec + \frac{FP \cdot x}{TP + FP} > Prec \end{aligned}$$

Here the FPR is not influenced by the number of false positives, and the TPR increases and the precision decreases while the number of false positives increases.

To visualize these results, we did again some experiments on the DREAM5 regulatory network. We randomly change a fraction of negatives into positives, in order to simulate a network with a fraction of false positives. We tried several fractions, such that the numbers of erroneous labels are that same as the number of erroneous labeled in the precedent false negative experiment. The resulting curves are presented in **Figure 3.12**. The precision increases, but is counterbalanced by the decreasing of the recall and the PR curve finally decreases, while the number of false positives increases. And as expected, the ROC curve decreases.

Previously we show that it is easy to modify the curves if we know the ratio x of false negatives, because the modifications are linear. Here, it is much more difficult to adapt the curves even if we have an idea of the number of false positives, because such variations are not linear.

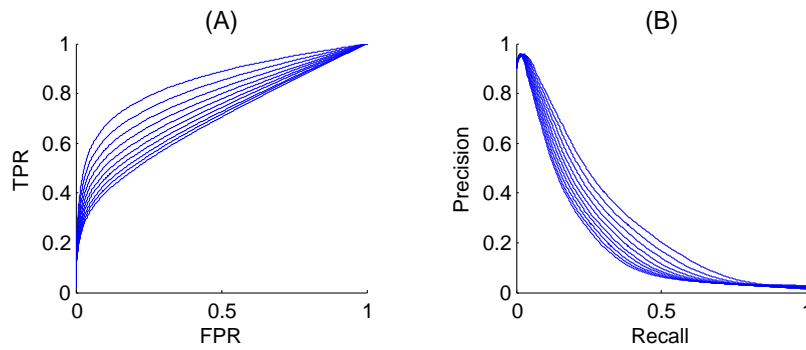


Figure 3.12: False positives. We simulated false positives in the Dream5 regulatory network, during the testing stage. Both the ROC (left) and the PR curves (right) decreases while the ratio of negatives changed into positives increases.

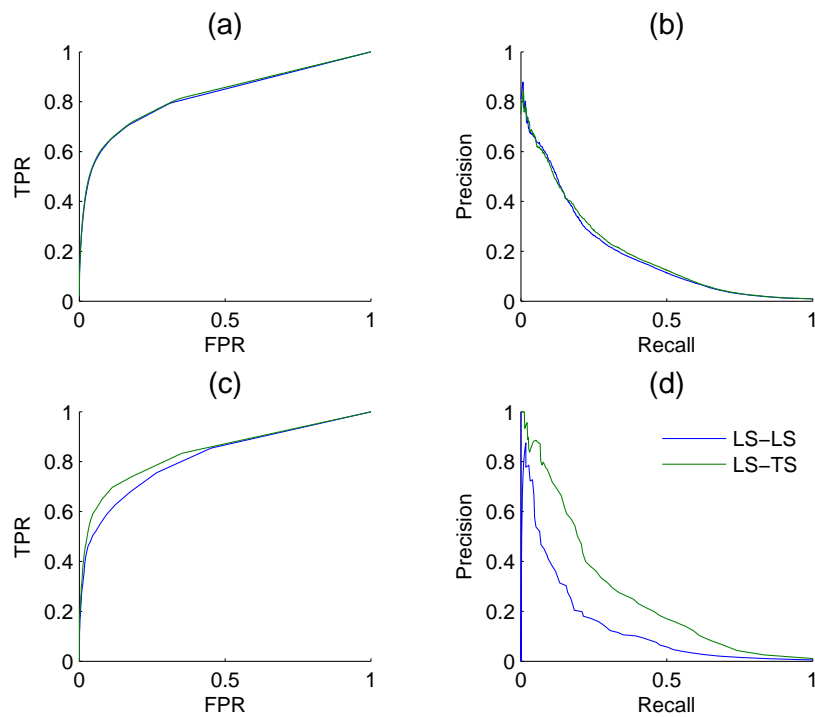
3.4.3 Illustration

To illustrate the practical impact of the absence of negatives on validation, we reproduced the experiment of Section 3.3.4 on the DREAM5 network, this time assuming that only positive (and unlabeled) pairs are available in the training data. More concretely, we again first randomly drew $2/3$ of the genes and $2/3$ of the transcription factors and then randomly drew $2/3$ of the positive pairs existing among these genes and transcription factors. This set of positive pairs then defines our training network and the goal is to find new positive pairs among all the other ones (that are then considered as unlabeled). The positive pairs in the training set were chosen so that they match the positive pairs in the training set in the experiment of Section 3.3.4.

Two validation experiments were performed. First, cross-validation across pairs and nodes was carried out on all pairs between the selected genes ($2/3$) and transcription factors ($2/3$), considering all unlabeled pairs as negative (experiment A). Second, we randomly split the whole set of unlabeled pairs into two subsets. We trained a model on the positive pairs and each of these subsets taken in turn as the set of negative pairs and then used this model to obtain a prediction for the unlabeled pairs in the other subset. The resulting predictions were then evaluated against the true network (experiment B). Experiment A is thus supposed to provide a CV estimate of the true performance as computed by experiment B. The resulting ROC and PR curves obtained from these two experiments are shown in **Figures 3.13** for the $LS \times LS$ and $LS \times TS$ families.

ROC curves and AUROC scores obtained from experiments A and B are very close but noticeable differences appear in PR curves and AUPR scores. Indeed, experiment A gives higher AUPR than experiment B for $LS \times LS$ pairs, but gives lower AUPR for $LS \times TS$ pairs. In other words, cross-validation overestimates the AUPR for $LS \times LS$ pairs and underestimates it for $LS \times TS$ pairs. As discussed above, these differences can be explained, on the one hand, by the presence of false negatives in the test data generated by the cross-validation and, on the other hand, by the differences in the ratio between positives and negatives that exist in the two families of pairs between experiments A and B.

Assuming that both the ratio of false negatives in the training pairs and the ratio of positives and negatives among the unlabeled pairs are known or can be estimated, PR curves and AUPR scores obtained from experiment A can be corrected using results in sections 3.2.4 and 3.4.2, so that they match the conditions of the application of the model in experiment B. Since these quantities



		$LS \times LS$	$LS \times TS$
Experiment A	(a) <i>AUROC</i>	0.82	0.83
	(b) <i>AUPR</i>	0.19	0.20
Experiment B	(c) <i>AUROC</i>	0.82	0.84
	(d) <i>AUPR</i>	0.13	0.26

Figure 3.13: Comparison of the cross-validation estimates of the $LS \times LS$ and $LS \times TS$ scores, ROC curve in (A) and PR curve in (B), with true score values for the same two families of predictions, ROC curve in (C) and PR curve in (D), when only positive and unlabeled pairs are available.

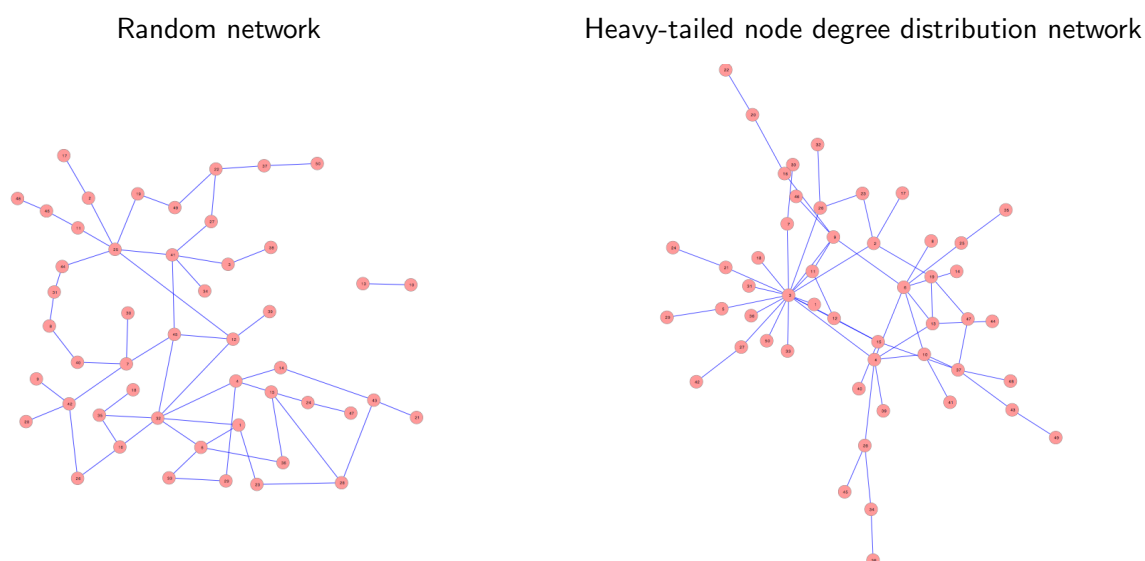


Figure 3.14: Comparison of two networks with 50 nodes and 57 edges each. The left one is a random network and the right one has a heavy-tailed node degree distribution.

are known for our artificial network, we performed these corrections, first adjusting the precision to account for the false negatives and then correcting the curve to account for the different ratio of positives versus negatives. The corrected AUPR are respectively 0.16 and 0.26 for $LS \times LS$ and $LS \times TS$, which are now closer to the value obtained from experiment B.

Note that another factor that could introduce a difference between CV scores and real scores is the composition of the training data in terms of positives and negatives, which might affect learning algorithms. In our experiment however, the ratios of positives versus negatives in the training data are very close ($\sim 0.9\%$ for experiment A and $\sim 1.0\%$ for experiment B).

3.5 Impact of heavy-tailed node degree distribution

Biological networks are typically non-random. In particular, many of them have a heavy-tailed distribution of node degrees: several nodes, called hubs, have degrees greatly higher than the average (Stumpf and Porter, 2012) (**Figure 3.14**). In such networks, a new node, without consideration of its features, is more likely to interact with a hub than with a less connected node. As a consequence, it is possible in such network to obtain better than random interaction predictions without exploiting the node features, by simply connecting any new node with the more connected nodes in the training network.

3.5.1 Experiment on a simple network

More concretely, let us imagine a classifier that, for any $LS \times TS$ pair, outputs a confidence score that is proportional to the degree of the LS node in the training network; pairs involving nodes that are the most connected in the LS will be classified in the top of the ranking, and pairs involving less connected nodes will be classified at the end. This classifier makes sense as soon as the network is actually non-random. Indeed, the more connected is a node in the training network, the higher is the chance for this node to be involved in new interactions.

Table 3.5: Example of a heavy-tailed node degree distribution network. The $LS \times TS$ part is shown in first 10 rows. The degree of each LS nodes in the learning set is given in the last row.

	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0			
	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1		
	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0		
	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0		
Tested network ($LS \times TS$)	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0		
	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		
	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
	1	0	0	1	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0
Degrees in LS	5	4	3	1	2	1	3	1	6	4	4	11	2	6	2	6	3	2	2	2	2	

Constants: $P = 40$, $N = 160$

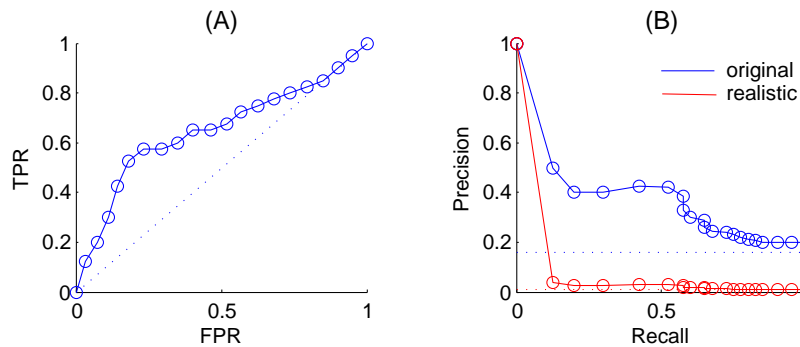


Figure 3.15: The degree of a node can sometimes be a good predictor for new interactions. ROC (A) and PR (B) curves are obtained from predictions based on the degrees of the nodes in the scale-free network from **Table 3.5**, for the original proportion of positives (12.5%, blue curve) and a proportion of 0.1% (red curve).

To see how good the ROC and PR curves of such classifier could be, let us consider the simple example of **Table 3.5**. This small network was randomly generated using the Barabasi-Albert model (Albert and Barabasi, 2002): an algorithm for generating random scale-free networks using a preferential attachment mechanism. It starts with an initial network of m_0 nodes. From that, new nodes are added one by one and are linked each with m existing nodes with a probability proportional to the current degree of the nodes. We performed the algorithm with the parameter $m_0 = 3$ and $m = 2$. The $LS \times LS$ part is not shown in the table, but the degree of each column node is reported below the $LS \times TS$ adjacency matrix to be predicted (LS nodes correspond to columns, TS nodes to rows). For each pair of nodes, we predict a confidence score that is directly proportional to the degree of the LS node. The resulting ROC and PR curves are shown in **Figure 3.15AB**: the blue curve is obtained with the original proportion of positives ($40/200=20\%$). The red PR curve is obtained by correcting the original PR curve for a more realistic ratio of 1% (using Equation (3.1)).

For the ratio of 20% of positives, both ROC and PR curves are much better than the curves obtained from a random classifier. This result confirms that scale-freeness makes it possible to obtain already good predictions without the need to learn a model, and ignoring node features. For

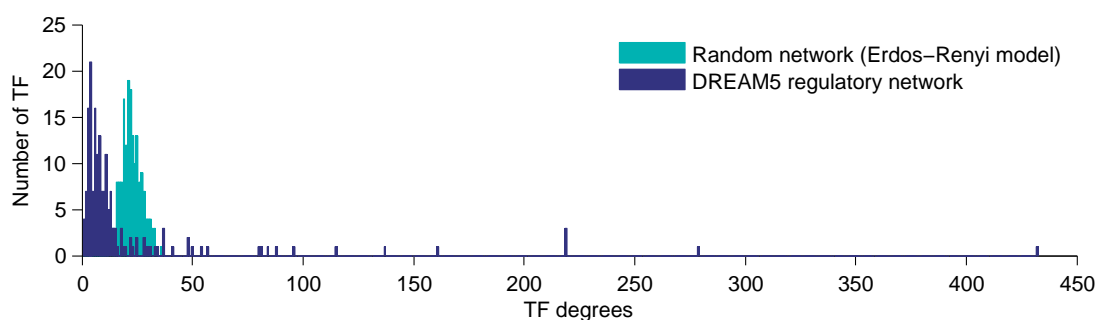


Figure 3.16: DREAM5 regulatory network has a heavy-tailed node degree distribution. It is compared to a random network (computed following an Erdos-Renyi model), which has a gaussian node degree distribution.

the ratio of 1% of positives, the ROC curve remains unchanged but the PR curve is now closer to a random PR curve (especially if one remembers that the first point is fixed arbitrarily to $(0, 1)$). This latter result is related to the fact that, for small P/N ratios, the PR curve is dominated by the top of the ranking. In this example, this top contains all interactions that involve the node of highest degree (12th column) in an arbitrary order and therefore the corresponding precision will decrease if the proportion of negatives increases.

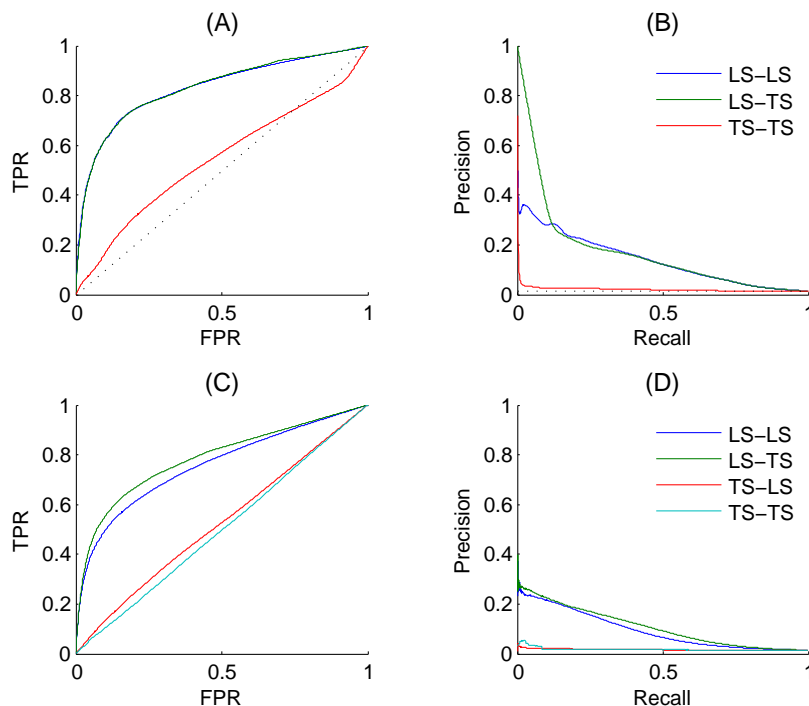
That it appears possible to complete a network based only on preferential attachment shows that using a random classifier as a baseline for assessing the performance of supervised network inference methods is inappropriate. A network inference method that does not perform better than the simple degree-based confidence score, which, for a given regulator, is unable to distinguish between possible targets has to be considered random and generally useless as a predictor. As a consequence, we believe that one should always report the performance of the degree-based confidence score as a baseline for assessing the performance of a supervised network inference method. Note that in the example of **Table 3.5**, the ranking of the interactions within each column of the adjacency matrix is uninformative, since all these interactions share the same confidence score. If one measures separate ROC or PR curves for each column node as discussed in Section 3.3.3 and illustrated in Section 3.3.4, then the appropriate baseline remains the random classifier.

3.5.2 Experiments on the DREAM5 network

Let us illustrate this on the DREAM5 *in-silico* network. The topology of this network is based on known transcriptional regulatory networks of model organisms such as *S.cerevisiae* and *E.coli*. 5% of the TFs collect about 50% of all interactions and it has a heavy-tailed node degree distribution (Figure 3.16).

Predictions using node degrees

Figure 3.17AB shows the ROC and PR curves obtained using the same 10-CV folds as in Section 3.3.4. The $LS \times LS$ pairs are now ranked according to the sum of the degrees of the nodes, computed in the training network, and the $LS \times TS$ and $TS \times LS$ pairs are now ranked according to the degree of the TF and of the gene respectively. The AUROC and AUPR are respectively equal to 0.83 and 0.14 for $LS \times LS$, 0.83 and 0.17 for $LS \times TS$ and 0.54 and 0.02 for $TS \times LS$. We can conclude from these results that the degree of a TF is indeed greatly linked with the probability for



		$LS \times LS$	$LS \times TS$	$TS \times LS$	$TS \times TS$
Using node degrees	(a) <i>AUROC</i>	0.83	0.83	-	0.54
	(b) <i>AUPR</i>	0.14	0.17	-	0.02
Feature permutation	(c) <i>AUROC</i>	0.76	0.78	0.52	0.50
	(d) <i>AUPR</i>	0.09	0.11	0.02	0.02

Figure 3.17: The degree of a node can sometimes be a good predictor for new interactions. ROC curves (A) and PR curves (B) are obtained from predictions made on the DREAM5 dataset using the degree of the nodes in the learning set. ROC curves (C) and PR curves (D) are obtained from predictions made on the DREAM5 dataset when randomly permuting the feature vectors relative to different nodes.

it to interact with a known or a new gene. On the contrary, the degree of a gene does not influence its chance to interact with a new TF. Although better than random, it is important to note however that the degree-based ranking of $LS \times TS$ pairs does not allow to distinguish potential targets of a given TF since they all inherit the degree of the TF.

This experiment confirms that heavy-tailed degree distribution makes it possible to obtain already good predictions without the need to learn a model, and ignoring node features. Using a random classifier as a baseline for assessing the performance of supervised network inference methods is therefore clearly inappropriate. As an illustration, on the DREAM5 network, we obtained with the Random Forests method AUROC values of 0.85 and 0.86 and AUPR values of 0.31 and 0.34 respectively for $LS \times LS$ and $LS \times TS$ pairs (see Section 3.3.4). The AUROC values of 0.85 and 0.86, although very good in absolute values, should be treated cautiously; they are indeed only slightly greater than the 0.83 AUROC of the degree-based ranking. In contrast, the doubling of the more robust AUPR value (from 0.14 and 0.17 for the degree-based random predictor to 0.31 and 0.34

for the trained model) indicates that the Random Forests are able to capture information from the feature vectors and indeed enable reliable predictions.

Random permutation of features

Another way to assess the importance of feature vectors in interaction prediction, is to train the model on new training data obtained by keeping the labels of the pairs unchanged and randomly permuting the feature vectors of the training set nodes (to decorrelate the features from the network). If the performance of the resulting model is close to the performance of a model trained on the original data, then one can conclude that the predictions are uninformative. We carry out this experiment on the DREAM5 artificial regulatory network with Random Forests. Resulting ROC and PR curves (**Figure 3.17CD**) are only slightly worse than those obtained by using the sum of degrees. The AUROC and AUPR are respectively equal to 0.76 and 0.09 for $LS \times LS$, 0.78 and 0.11 for $LS \times TS$, 0.52 and 0.02 for $TS \times LS$ and 0.50 and 0.02 for $TS \times TS$. Nevertheless they are still significantly better than those obtained using a random classifier, showing that the algorithm is able to do predictions only from the known network. Scores under feature permutation can therefore be used as another baseline for assessing the performance of a supervised network inference methods.

3.6 Merging pair rankings

The goal of supervised network inference is basically to predict new interactions from a given set of known interactions. The prediction provided by these methods for a given pair of nodes can be binary, i.e., equal to 1 if this pair of nodes is predicted as an interacting pair and zero otherwise. But most of the time, supervised learning methods provide a quantitative confidence score that reflects how confident the algorithm is about the fact that this pair interacts. This confidence score should be high for pairs that are expected to interact and low for pairs that are expected not to interact. All unlabeled pairs of nodes can then be ranked according to their predicted confidence score from the most likely to interact to the less likely to interact. A biologist can then validate experimentally the top-ranked pairs in this ranking, or use them to complete the already known network (see Section 3.2.6).

As discussed in Section 3.3, unlabeled pairs can be divided into several families: $LS \times LS$, $LS \times TS$, $TS \times LS$ and $TS \times TS$ pairs. These families are typically not equally well predicted, with the $LS \times LS$ pairs better predicted than the $TS \times TS$ pairs for example. Pairs can be ranked separately in each family but in most applications, one is mainly interested in getting a ranking of all unlabeled pairs irrespectively of their family, so as to maximize the chance to have all and only truly interacting pairs in the top of this ranking. The most straightforward way to get this ranking is to rank pairs according to their predicted confidence scores. While very simple, this is probably not the best way to get a unique ranking of all unlabeled pairs, for two reasons. First, given the differences in performance between families, confidence scores should not be treated equally from one family to the other. Indeed, confidence scores will be more reliably predicted for $LS \times LS$ pairs than for $TS \times TS$ pairs and therefore a $LS \times LS$ pair with some confidence score should be higher ranked in the final ranking than a $TS \times TS$ pair with the same confidence score. Second, when using the local approach (see Sections 2.3.2 and 4.2.2), confidence scores for each family of pairs are obtained from different models that are trained from different sets of pairs. In general, the confidence scores predicted by these different models are thus not strictly comparable. Depending on the supervised

learning method used, these scores might be biased differently due to differences in the training set composition. Since the global approach trains a single model, its confidence scores should be more comparable from one family to the other but some bias due to the family could however also exist in this latter approach. The generation of an optimal global ranking of all unlabeled pairs is thus a non trivial task in general.

In this section, we present an approach to merge several individual pair rankings so as to optimize the precision-recall curve of the resulting global ranking. To guide this merging, we assume that a precision-recall curve can be computed or more realistically estimated (e.g., by cross-validation) for each individual ranking. We first formalize the problem in Section 3.6.1 as maximizing the number of true positives for any number of predicted positives. In Section 3.6.2, we introduce and present the algorithm that deal with the considered problem and illustrate it in Section 3.6.3 with data from the DREAM5 coregulatory network. In Section 3.6.3, we perform again an experiment on the same network, but with the more realistic setting of Section 3.3.4. Finally we discuss the pros and cons of our approach in Section 3.6.4.

3.6.1 Objective

To simplify the development, we restrict our discussions here to the merging of two rankings. We will discuss extensions to more than two rankings later. We assume that we have a first ranking of pairs associated to a first precision-recall curve, and a second ranking of pairs associated to a second precision-recall curve (see **Table 3.6** for an illustration). These two rankings are assumed to be defined on two different non-overlapping sets of pairs. Our goal is then to obtain from these two rankings a new global ranking of all pairs in these two rankings that interleaves the two individual rankings (keeping the order of the pairs) in such a way that the merged ranking achieves the best possible precision-recall curve. Considering that $precision = TP/predP$ and $recall = TP/P$, this can be equivalently formulated as follows:

Given two rankings and their associated precision-recall curves, construct a global ranking of all pairs that maximizes the number of truly positive pairs (TP) for any number of predicted positive pairs ($predP$).

Note that since the computation of the precision-recall curves for the two rankings requires to know the true labels of all pairs, the problem might seem practically irrelevant. Although we will indeed assume that the exact precision-recall curves for the two rankings are known in our theoretical developments below, in practice, we will apply the resulting method in the more realistic case where precision-recall curves have been estimated by cross-validation (see Section 3.6.3 for an illustrative experiment).

Before digging into the problem, let us illustrate it with an example based on the DREAM5 coregulatory network. We will assume that we want to merge a set of 1000 $LS \times LS$ predicted pairs (the first ranking) with a set of 700 $TS \times TS$ predicted pairs (the second ranking). Precision-recall curves related to these two families have already been presented in Section 3.3.4 and are reproduced in **Figure 3.18A**. As our goal is formulated as the maximization of the number of true positives TP for any number of positive predictions $predP$, we converted the two precision-recall curves into their corresponding TP - $PredP$ curves, with the TP on the y-axis and $predP$ on the x-axis (see **Figure 3.18B**). These curves were obtained from the precision-recall curves using the following

Table 3.6: Pairs from a first ranking, associated to PR_1 (a first precision-recall curve), are merged with pairs from a second ranking, associated to PR_2 . Pairs from the merged ranking are associated to a final precision-recall curve PR_{tot} , which we want to be the best possible one.

First ranking	1 2 3 4 5 6 7 8 9 10	$\leftrightarrow PR_1$
Second ranking	1 2 3 4 5 6 7	$\leftrightarrow PR_2$
Merged ranking	1 2 1 3 2 3 4 4 5 6 5 7 8 9 6 7 10	$\leftrightarrow PR_{tot}$

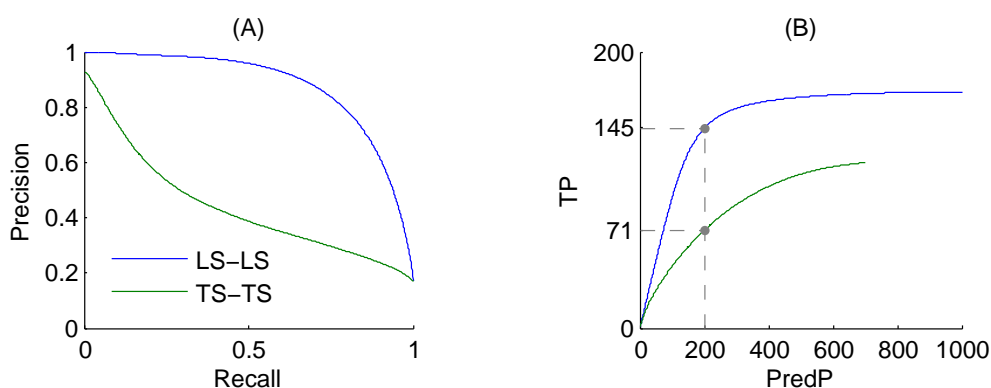


Figure 3.18: (A) Precision-recall curves obtained from a cross-validation on DREAM5 co-regulatory network, with a local approach. Curves relative to $LS \times LS$ and $TS \times TS$ pairs are presented. (B) The corresponding curves plotting the number of true positives according to the number of positive predictions (TP-predP curves).

relations:

$$TP = recall \cdot P$$

$$predP = \frac{TP}{precision}$$

In practice, the number of positives P is unknown. It can however be estimated if one has an idea of the expected percentage of interacting pairs (by multiplying this percentage with the total number of pairs to be predicted). For this network, 17% of all pairs are interacting, which gives respectively $P = 1000 \cdot 0.17$ and $P = 700 \cdot 0.17$ for the $LS \times LS$ and $TS \times TS$ curves. We discuss later the problem of estimating P .

From the TP - $predP$ curves, we can see, for example, that among the 200 top-ranked $LS \times LS$ pairs, 145 are expected to be true positives and that among the 200 top-ranked $TS \times TS$ pairs, 71 are expected to be true positives (gray dots in **Figure 3.18B**). If the top 400 pairs of the merged ranking are composed of the top 200 $LS \times LS$ pairs and the top 200 $TS \times TS$ pairs, then they are expected to include $145 + 71 = 216$ true positives. Whether it is possible to achieve a higher number of true positives by picking in total 400 pairs at the top of the two rankings in different proportion (e.g., 300 from $LS \times LS$ and 100 from $TS \times TS$) is the question that we will address in the next section.

3.6.2 Constructing the optimal merged ranking

In this section, we first address the problem of picking the optimal number of pairs at the top of the two rankings for a given total number $predP$ of positively predicted pairs. From this result, will then follow the solution of the problem of constructing the optimal global ranking.

We assume that precision-recall curves have been estimated (e.g., by cross-validation). Then, we have also an estimated relative TP-predP curve for each of the two rankings. For each pair in the two rankings, the corresponding curve provides the average number of TP to expect when this pair and all its predecessors in the ranking are predicted as positives. From these curves, the objective is to select $predP_1$ pairs in the first ranking and $predP_2$ pairs in the second ranking, so that :

1. they satisfy $predP_1 + predP_2 = predP_{tot}$ for a given $predP_{tot}$,
2. among all possible combinations of $predP_1$ and $predP_2$ that satisfy the first condition, the resulting number of expected TP is maximized.

Before formulating the algorithm that can output the solution to this problem, we will first describe important characteristics of the TP-predP curves, and second explain the conditions that must be met to have an optimal picking.

TP-predP is a piecewise-linear concave-down "curve"

Let us denote the successive points of a TP-predP curve as follows:

$$\{(1, TP_1), (2, TP_2), (3, TP_3), \dots, (N, TP_N)\}.$$

In the development that will follow, we will assume, without loss of generality, that a TP-predP curve satisfies the following two properties:

1. The curve is monotonically increasing, i.e., $TP_{i+1} \geq TP_i, \forall i = 1, 2, \dots, N - 1$.
2. The curve is concave-down, i.e.:

$$TP_{i+1} - TP_i \geq TP_{i+2} - TP_{i+1}, \forall i = 1, \dots, N - 2 \quad (3.6)$$

The first property is trivially met as the number of true positives can not decrease when the number $predP$ of positive predictions increases. Viewing the TP-predP curve as a piecewise-linear curve, the second property means that the slopes of the successive segments are monotonically decreasing as $predP$ increases. In practice, this property is not necessarily met. However, when it is not, it is always possible to construct a new ranking with a strictly better TP-predP curve that satisfies condition (3.6).

Indeed, let us suppose that there is a triplet of points $\{(i, TP_i), (i + 1, TP_{i+1}), (i + 2, TP_{i+2})\}$ violating this condition. Then we flip a coin to decide which one of the pairs ranked at the $i + 1$ and $i + 2$ positions will come first in the ranking. Because of this randomization, the expected TP at point $i + 1$ becomes (see **Figure 3.19**):

$$TP_i + \frac{1}{2}(TP_{i+1} - TP_i) + \frac{1}{2}(TP_{i+2} - TP_{i+1}) = \frac{TP_i + TP_{i+2}}{2},$$

leading to a new (linearly aligned) triplet $\{(i, TP_i), (i + 1, (TP_i + TP_{i+2})/2), (i + 2, TP_{i+2})\}$ now satisfying conditions (3.6) (with equality). The operation also yields a strictly better TP-predP curve

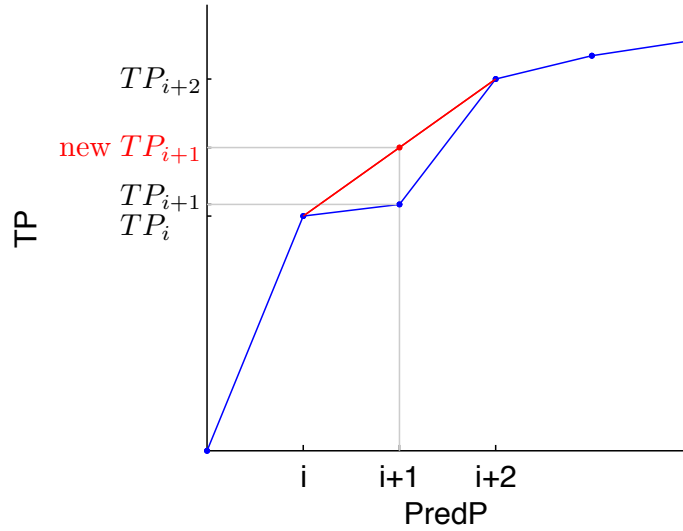


Figure 3.19: Every TP-predP curve can be modified to become concave-down by interpolating their points. It is justified because any level of performance on a new line can be achieved by flipping a weighted coin to decide between the classifiers that the two end points represent.

than the previous one since $(TP_i + TP_{i+2})/2 > TP_{i+1}$ (because conditions (3.6) was not satisfied). A similar randomization can be applied recursively until there are no more triplets violating condition (3.6). This procedure will in fact construct the convex hull of the TP-predP curve (see the discussion in Section 3.2.4 about the convex hull of the ROC curve).

Note that we can reasonably expect from a classification algorithm to produce a concave-down TP-predP curve by itself. Indeed, we expect to find the true interactions at the top of the ranking of pairs (predicted with a high probability to interact) and to find less and less of them when going down through the ranking. In practice we will estimate the TP-predP curve by cross-validation, meaning that they will be average curves over several trials. So they will be very likely already satisfying condition (3.6) for most triplets.

The two characteristics of TP-predP curves that we presented here will now be useful to find the conditions in which two values of $predP_1$ and $predP_2$ are optimal for a given value of $predP_{tot}$.

Necessary and sufficient conditions for an optimal picking

Let us denote by $C_1 = \{(1, TP_1^1), (2, TP_2^1), \dots, (N_1, TP_{N_1}^1)\}$ and $C_2 = \{(1, TP_1^2), (2, TP_2^2), \dots, (N_2, TP_{N_2}^2)\}$ the two TP-predP curves to be merged. The following theorem gives necessary and sufficient conditions for a picking to be optimal.

Theorem 1. *Assuming that the points of these two curves satisfy condition (3.6), then picking the first i_1 pairs from the first ranking and the first i_2 pairs from the second ranking maximizes the number of true positives $TP_{i_1}^1 + TP_{i_2}^2$, for a number of predicted positives $i_1 + i_2$, if and only if these*

two conditions are met:

$$TP_{i_1+1}^1 - TP_{i_1}^1 \leq TP_{i_2}^2 - TP_{i_2-1}^2 \quad (\text{if } i_1 < N_1 - 1 \text{ and } i_2 > 1) \quad (3.7)$$

$$TP_{i_2+1}^2 - TP_{i_2}^2 \leq TP_{i_1}^1 - TP_{i_1-1}^1 \quad (\text{if } i_1 > 1 \text{ and } i_2 < N_2 - 1) \quad (3.8)$$

Proof. Let us first show that (3.7) and (3.8) are two necessary conditions. Indeed, let us assume for example that condition (3.7) is not met (with $i_2 > 1$ and $i_1 < N_1 - 1$). We then have:

$$\begin{aligned} TP_{i_1+1}^1 - TP_{i_1}^1 &> TP_{i_2}^2 - TP_{i_2-1}^2 \\ \Leftrightarrow TP_{i_1+1}^1 + TP_{i_2-1}^2 &> TP_{i_1}^1 + TP_{i_2}^2 \end{aligned}$$

which means that taking the first $i_1 + 1$ pairs from the first ranking and the first $i_2 - 1$ pairs from the second ranking leads to a higher number of true positives than our initial solution, which is impossible since the solution (i_1, i_2) is optimal. Symmetrically, if the second condition is not met, we have (with $i_1 > 1$ and $i_2 < N_2 - 1$):

$$TP_{i_1-1}^1 + TP_{i_2+1}^2 > TP_{i_1}^1 + TP_{i_2}^2$$

which is also impossible since the solution (i_1, i_2) is optimal.

Let us now show by contradiction that if (3.7) and (3.8) are met, then the solution (i_1, i_2) is optimal. Indeed, let us suppose that there exists a better solution (i'_1, i'_2) , i.e., such that $i'_1 + i'_2 = i_1 + i_2$, $i_1 \neq i'_1$, $i_2 \neq i'_2$, and $TP_{i'_1} + TP_{i'_2} > TP_{i_1} + TP_{i_2}$. Then, either we have $i'_1 < i_1$ and $i'_2 > i_2$ or we have $i'_1 > i_1$ and $i'_2 < i_2$. In the first case, we have $TP_{i'_1} \leq TP_{i_1}$ and $TP_{i'_2} \geq TP_{i_2}$ because the TP-predP curves are monotonically increasing. In addition, because condition (3.6) is met, we have two upper bounds for $TP_{i'_1}$ and $TP_{i'_2}$ (see **Figure 3.20**):

$$\begin{aligned} TP_{i'_1} &\leq TP_{i_1} - (i_1 - i'_1)(TP_{i_1} - TP_{i_1-1}) \\ TP_{i'_2} &\leq TP_{i_2} + (i'_2 - i_2)(TP_{i_2+1} - TP_{i_2}) \end{aligned}$$

Summing these two inequalities, one gets:

$$\begin{aligned} TP_{i'_1} + TP_{i'_2} &\leq TP_{i_1} + TP_{i_2} - (i_1 - i'_1)(TP_{i_1} - TP_{i_1-1} - TP_{i_2+1} + TP_{i_2}), \\ &\leq TP_{i_1} + TP_{i_2}, \end{aligned}$$

since $i_1 - i'_1 = i'_2 - i_2$ and condition (3.8) ensures that $(TP_{i_1} - TP_{i_1-1} - TP_{i_2+1} + TP_{i_2}) > 0$. This latter inequality is in contradiction with $TP_{i'_1} + TP_{i'_2} > TP_{i_1} + TP_{i_2}$. The case $i'_1 > i_1$ can be proven similarly from condition (3.7), which proves the sufficient condition. \square

Generating the global ranking

Given the conditions presented in Theorem 1, one can now determine $predP_1$ and $predP_2$ that achieve a specific $predP_{tot} \leq N_1 + N_2$ with maximal TP using the following algorithm:

$$i_1 = 0, i_2 = 0$$

While $(i_1 + i_2 < predP_{tot})$

$$\text{if } i_1 < N_1 \text{ and } TP_{i_1+1} - TP_{i_1} > TP_{i_2+1} - TP_{i_2}$$

$$i_1 = i_1 + 1$$

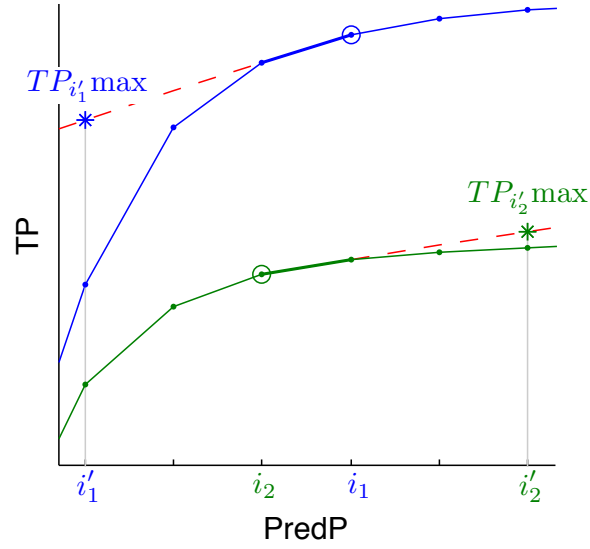


Figure 3.20: The down concavity of TP-predP curve constrains the possible values of $TP_{i'_1}$ and $TP_{i'_2}$.

else

$$i_2 = i_2 + 1$$

return (i_1, i_2)

This algorithm can be shown to be correct because conditions (3.7) and (3.8) are two invariants of the while-loop. This algorithm also shows that for increasing values of $predP_{tot}$ the selected pairs are nested and therefore the algorithm can be trivially modified to generate directly a global ranking of the pairs. The TP-predP curve of the resulting merged ranking can be obtained by summing the TP values corresponding to i_1 and i_2 and a precision-recall curve can be obtained from this curve assuming that the number P of positive pairs can be estimated.

Estimation of the initial TP-predP curves

As already discussed, in practice, the exact TP-predP curve of the two rankings to be merged will be unknown (otherwise the problem would be trivial as then all true labels would be known). We then have to rely on cross-validation techniques to estimate it. Assuming that we have an estimated precision-recall curve, the proposed algorithm requires to derive from this curve an estimated number of TP for any given value of $predP \in \{1, 2, \dots, N\}$ denoting by N the total number of pairs in the ranking. Although there might exist better estimation schemes, we propose to proceed as follows:

- Each $(Precision, Recall)$ pair is transformed into a $(TP, predP)$ pair using the following relations:

$$\begin{aligned} TP &= recall \cdot P \\ predP &= \frac{TP}{precision} \end{aligned}$$

- The resulting curve is sampled with linear interpolation at values of $predP \in \{1, 2, \dots, N\}$

The construction of the TP-predP curve thus requires to have an estimation of the expected number P of positives among our pairs. The value of P can for example be determined by some prior knowledge, or from the proportion of the interactions observed in the training data (assuming for example that there will be a similar proportion of positive pairs in the test data). Interestingly, the final merged ranking will only depend on the ratio P_1/P_2 , not on the absolute values of P_1 and P_2 , where P_1 and P_2 are the expected numbers of positives respectively in the first and in the second ranking. If one expects for example the same proportion of positive pairs in both rankings, then one can construct the merged ranking without even having to estimate this proportion (it can be set arbitrarily).

Dealing with more than two sets of predictions

Theorem 1 and the resulting algorithm can be easily generalized to the merging of three or more pair rankings, for example to merge $LS \times TS$, $TS \times LS$, and $TS \times TS$ predictions as we will do it in Section 5.5. In that case, for K different rankings, the algorithm that determines the optimal number of pairs to pick in each ranking becomes:

$$i_1 = 0, i_2 = 0, \dots, i_K = 0$$

$$\text{While } \left(\sum_{k=1}^K i_k < \text{pred}P_{\text{tot}} \right)$$

$$m = \arg \max_{k: i_k < N_k} (TP_{i_k+1}^k - TP_{i_k}^k)$$

$$i_m = i_m + 1$$

$$\text{return } (i_1, i_2, \dots, i_K)$$

In this algorithm, N_k ($k = 1, \dots, K$) denotes the number of pairs in the k th ranking and $TP_{i_k}^k$ ($k = 1, \dots, K, i_k = 1, \dots, N_k$) denotes the expected number of true positives at position i_k in the k th ranking. At each iteration, the algorithm selects the next pair from the ranking with the greatest slope at its current position.

3.6.3 Illustration

True PR curves

Let us illustrate the proposed algorithm on the example discussed in Section 3.6.1 based on the DREAM5 coregulatory network. We apply the algorithm to merge 1000 $LS \times LS$ pairs and 700 $TS \times TS$ pairs in a global ranking that maximizes the number of TP for any number of $\text{pred}P$. For this experiment, we assume that we have access to the true TP-PredP curves for both rankings. We relax this hypothesis below.

Figure 3.21 illustrates the resulting global ranking. $LS \times LS$ pairs are picked first and the 81 top pairs of the merged ranking are the 81 top $LS \times LS$ pairs. Then the algorithm starts to add $TS \times TS$ pairs. When we reach position 510, we have as many $LS \times LS$ pairs as $TS \times TS$ pairs, i.e. 205 pairs from each ranking. Finally, the end of the ranking, like the beginning, is only composed of $LS \times LS$ pairs.

Precision-recall curves show that $LS \times LS$ pairs are better predicted than $TS \times TS$ pairs. It is then not surprising that the pairs from the first family are found at the top and at the end of the

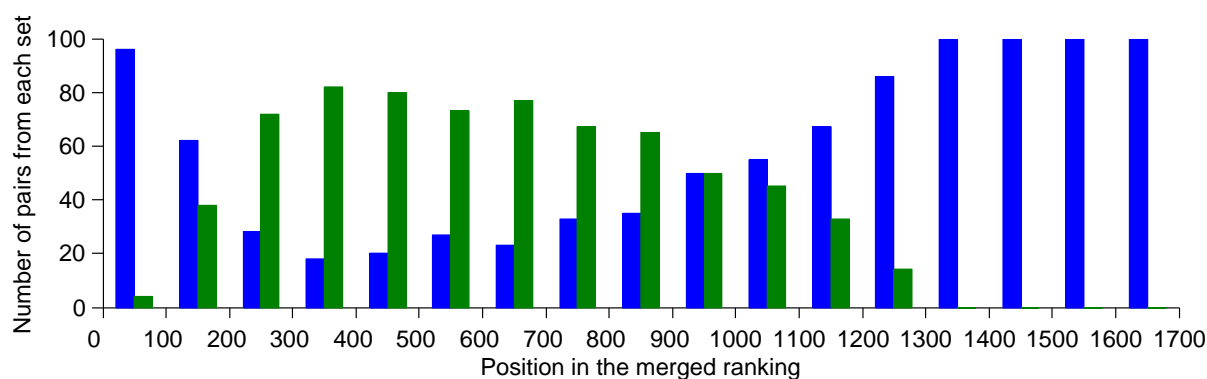


Figure 3.21: The 1700 pairs of the final ranking are divided into bins of 100 pairs. The blue and green bars represent respectively the number of $LS \times LS$ and $TS \times TS$ pairs in each bin.

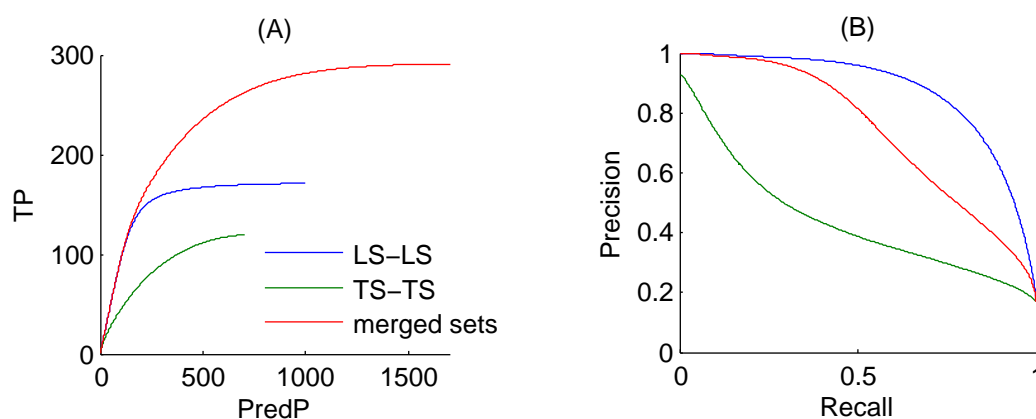


Figure 3.22: (A) TP-predP curves for $LS \times LS$, $TS \times TS$ and the merged ranking. (B) Corresponding precision-recall curves.

global ranking. There are indeed more interacting pairs at the top of the $LS \times LS$ ranking than at the top of $TS \times TS$ ranking and, similarly, more non-interacting pairs at the bottom of the $LS \times LS$ ranking than at the bottom of the $TS \times TS$ ranking.

From the two TP-predP curves and from the obtained final ranking, we can generate the TP-predP curve corresponding to the global merged ranking (**Figure 3.22A**). From this curve, we can generate a new precision-recall curve (**Figure 3.22B**) that will give the performance we can expect from our final ranking. Not surprisingly, the new curves are less good than the $LS \times LS$ curves, but better than the $TS \times TS$ curves.

A more realistic setting

In the previous experiment, we exploited the true precision-recall curves corresponding to the two rankings to be merged. In practice, only an estimated curve (e.g., using cross-validation) will be available. To evaluate the impact of using an estimated precision-recall curve, instead of the true one, to compute the merged ranking, we reproduced the experiment of Section 3.3.4 but on the DREAM5 coregulatory network. We first randomly drew 2/3 of the genes and then randomly drew

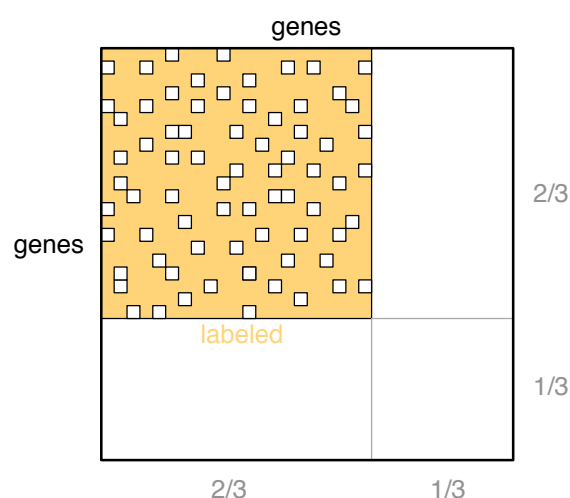


Figure 3.23: To make sure that performance curves evaluated on labeled data reflect the ones that we would obtain by predicting new data, we performed two experiments on the DREAM5 coregulatory network, in a more realistic setting: (A) Cross-validations across the considered labeled pairs (orange ones) and (B) prediction of considered unlabeled pairs (white ones) from the labeled ones. Resulting curves are found in **Figure 3.24**.

2/3 of the pairs existing among these genes (**Figure 3.23**). This set of pairs defines our training network of labeled pairs and the goal is to label and rank all the other pairs. As in Section 3.3.4, about 30% of the pairs represent the training set, 15% the $LS \times LS$ pairs, 22% the $LS \times TS$ and 11% the $TS \times TS$ pairs.

Two validation experiments were performed. In experiment A, cross-validation is performed across the pairs and across the genes of the labeled pairs (orange area) with a local approach. The goal is to estimate the three PR curves corresponding respectively to the prediction of $LS \times LS$ pairs, $LS \times TS$ pairs, and $TS \times TS$ pairs (dark blue, light blue and green curves in **Figure 3.24A**). TP-predP curves are obtained from these PR curves assuming that the percentage of positive pairs is the same in the three families and these curves are merged according to the algorithm of Section 3.6.2. The estimated precision-recall curve of the merged ranking is shown as the red curves in **Figures 3.24A** and **B**.

In experiment B, a model is actually trained on the labeled pairs (with a local approach) to predict labels for the unlabeled pairs. These latter pairs are then ranked, according to their scores and their families, and these rankings are merged according to the optimal procedure of Section 3.6.2 and this time, using the true (and not the cross-validated) TP-predP curves. The purple precision-recall curve in **Figures 3.24B** represents the true curve of the resulting merged ranking, of which the red curve is thus supposed to be an estimate using cross-validation. Both curves are very close, which proves that using cross-validated PR curves to generate the merged ranking works very well on this problem. To highlight the interest of an optimal merging, we also tried merging the predictions directly using the predicted confidence scores, without taking into account performance differences between the families. The resulting (true) precision-recall curve (orange curve in **Figures 3.24B**)

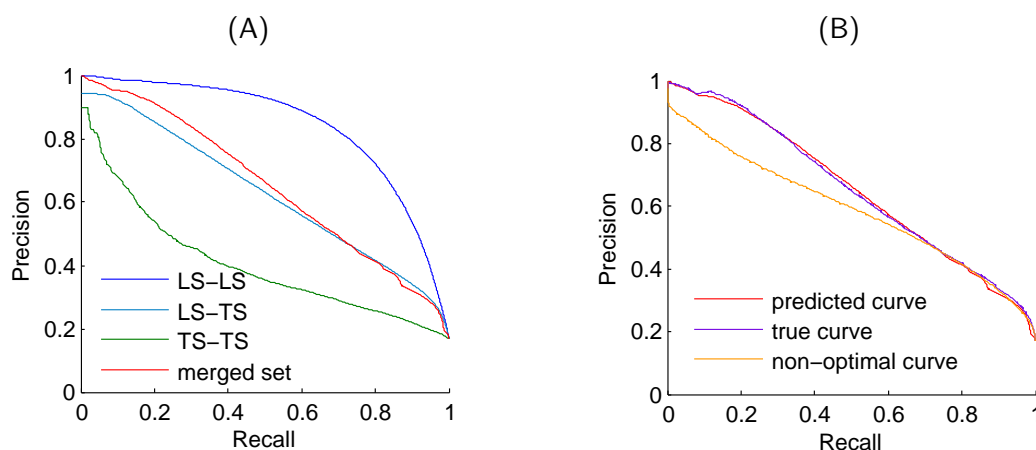


Figure 3.24: (A) Cross-validations on labeled pairs and genes allow us to estimate precision-recall curves of the different families of pairs, and to estimate the curve of the resulting optimal global ranking, assuming the same percentage of positive pairs in each family. (B) The red curve estimated in (A) is very close to the curve obtained when predicting and ranking unlabeled pairs (purple curve). Curve obtained when simply ranking the pairs according to a decreasing order of the scores (orange curve) leads to, as expected, a less good performance.

shows that, as expected, this ranking is worse than the ranking computed by our algorithm from the estimated precision-recall curves.

3.6.4 Discussion

The algorithm presented in this section is useful to merge different rankings belonging to different families of pairs and associated with different performance curves. The results obtained on the DREAM5 coregulatory network are very good: the estimated curve is very close to the true curve, and the curve obtained with the naive ranking (based on the confidence scores) is significantly worse than the optimal one. It has to be noted however that the conditions of this experiment are very idealistic. Indeed, in both experiments A and B, the learning and test sets have been drawn uniformly at random from the available data. In a real scenario, the chance will be high that the learning and test set will not have been drawn uniformly at random but instead with some bias. The curve obtained by the cross-validation of experiment A (which simulates uniform sampling) will thus provide a biased estimate of the true performance obtained in experiment B, which could affect the quality of the merged ranking. It is very difficult to assess the effect of this bias on real datasets, for which the data distribution is unknown. Despite this, we nevertheless believe that the optimal merging procedure proposed in this section will remain beneficial in practice with respect to a simple merging according to confidence scores, in particular when confidence scores are expected not to be comparable from one family to another (e.g., when using the local approach) and when there are important performance differences between the three families.

The solution presented here was applied when the pairs are differentiated according to the family they belong to: $LS \times LS$, $LS \times TS$, $TS \times TS$, i.e. according to the number of nodes they have in the learning and test sets. We showed in Section 3.3.4 (per-node evaluation) that the quality of predictions can also vary from one node to another, according to their degree in the learning set.

Our algorithm could in principle be applied also to merge rankings obtained on individual nodes. Nevertheless, as discussed, the application of the algorithm requires an estimate of the number of positive examples in each ranking, which means in this case an estimate of the degree of each node. Since the degree might vary a lot from one node to another, we believe this assumption actually excludes the application of our algorithm on a per node basis.

3.7 Conclusion

In this chapter, we discussed measures and protocols for the validation *in silico* of supervised methods for the inference of biological networks, i.e. methods that infer a biological network from a training sample of known interacting and non-interacting pairs and a set of features defined on the network nodes (or directly on pairs of nodes). Although this problem is very close to a standard supervised classification problem, it requires to address several important issues related to the need to classify pairs of entities in a candidate interaction and to the nature of biological networks. We carried out a rigorous examination of these issues that we supported by experiments on an artificial gene regulatory network. The main guidelines that can be drawn from this examination are as follows:

- Network inference methods have been assessed mainly using precision-recall (PR) curves and receiver operating characteristic (ROC) curves. The choice of an appropriate metric should be dictated mainly by the application but generally PR curves are more appropriate than ROC curves given the highly imbalanced nature of the underlying classification problem, related to the very sparse nature of most biological networks. PR curves are also less affected by the uninformative predictability due to the heavy-tailed node distribution of biological networks. While PR curves are sensitive to the ratio of positives versus negatives in the test data, we show that it is straightforward to adapt them to a new ratio.
- When validating a model, it is necessary to divide the predictions into four groups, given that the two nodes might either be present or absent in the learning sample of interactions. Indeed, performance is typically very different from one group to another and improves when the number of training interactions involving the nodes in the pairs to be predicted increases. The quality of the predictions for pairs where both nodes have interactions in the training network can be assessed using cross-validation over pairs in the training data. The quality of the predictions for the three other groups of pairs, where at least one node is not represented in the training data, is best assessed by using cross-validation over nodes. Unless the inference problem at hand makes some subgroups of predictions irrelevant, we advocate the joint use of both kinds of cross-validation to get a more detailed assessment of the performance of an inference method.
- We discussed the lack of experimental support for non-interacting pairs in most biological networks. We reviewed several ways to address this problem at training time and showed that the presence of false negatives does not really affect ROC curves but can result in an underestimation of the PR curve. Assuming that the proportion of false negatives in the test data is known and that false negatives are selected randomly among positives, we show that it is possible to correct the PR curve so that it better reflect true performances. The correction is however not necessary when one only wants to compare different methods.

- We showed empirically that the heavy-tailed distribution of node degrees seemingly enables a better than random inference only by exploiting the topology of the training network. As a consequence, random guesses should not be taken as valid baselines for supervised network inference methods, in order not to overestimate the performance. Every validation of a supervised inference method should always be supplemented by a reporting of the performance of the simple degree-based score or a classifier grown from randomly permuted feature vectors.

Thereby, we provided the most comprehensive examination and discussion of issues in the evaluation of supervised inference techniques so far. Given that the examined supervised techniques exploiting prior information on the network are typically superior in performance to unsupervised approaches, a reliable assessment is particularly desirable. Following the guidelines we derived will enable a more rigorous assessment of supervised inference methods, will contribute to an improved comparability of the different approaches in this field and will thus furthermore aid researchers in improving the state of the art methods.

We argued, as others, that predictions within the different pair subgroups should be assessed separately. We have also discussed ways to take into account the resulting information to obtain better global network predictions. Indeed, most methods eventually provide a single ranking of all pairs to be predicted. How to take into account the performance differences between the different groups of pairs to reorganize this ranking into a better one have been developed at the end of this chapter.

Still, there remains several open questions about supervised network inference methods and their validation. First, with a few exceptions, most papers in the domain focus on a given type of biological network. Yet, unlike unsupervised methods that needs some prior knowledge to derive their confidence scores, supervised methods are most of the time generic in that they could be applied to any network without much adaptation. A thorough empirical comparison of these methods on several networks with different characteristics will be presented in Chapter 4, to really understand the advantages and limitations of all these methods.

In this review, we focus on the statistical and *in silico* validation of network inference methods using cross-validation techniques. Such validation helps assess the quality of the predictions and therefore decide on a confidence threshold that best suits application needs. However, even more important is the experimental validation of the predictions provided by network inference techniques. Experimental validation depends on the nature of the biological network at hand and therefore a discussion of these techniques is out of the scope of this chapter. Note nevertheless that experimental validation will be influenced also by the lack of experimental support for non-interacting pairs and that for some (more abstract) networks, experimental validation might be very difficult (e.g., disease-gene networks).

Chapter 4

Tree-based methods for biological network inference

In this chapter, we systematically investigate, theoretically and empirically, the exploitation of tree-based ensemble methods in the context of the global and local approaches for biological network inference. We present these two approaches, extending the later for the prediction of interactions between two unseen network nodes, and discuss their specializations to tree-based ensemble methods, highlighting their interpretability and drawing links with clustering techniques. Extensive computational experiments are carried out with these methods on various biological networks that clearly highlight that these methods are competitive with existing methods.

Contents

4.1	Introduction	92
4.2	Two different approaches	92
4.3	Tree-based ensemble methods	94
4.4	Experiments	97
4.5	Illustration of interpretability of trees	110
4.6	Discussion	124

4.1 Introduction

In this chapter, we would like to systematically investigate, theoretically and empirically, the exploitation of tree-based ensemble methods in the context of the local and global approaches for supervised biological network inference. In Section 2.3.1, we have formalized biological network inference as the problem of classification on pairs, considering in the same framework homogeneous graphs, defined on one kind of nodes, and bipartite graphs, linking nodes of two families. Now we define the general local and global approaches in the context of this formalization, extending in the process the local approach for the prediction of interactions between two unseen network nodes. The chapter discusses in details the specialization of these approaches to tree-based ensemble methods. In particular, we highlight their high potential in terms of interpretability and draw connections between these methods and unsupervised (bi-)clustering methods. Experiments on several biological networks show the good predictive performance of the resulting family of methods. Both the local and the global approaches are competitive with however an advantage for the local approach in terms of compactness of the inferred models. The interpretability of trees and ensembles of trees are illustrated by several experiments on a drug-protein interaction network.

The chapter is structured as follows. Section 4.2 details the global and local approaches and Section 4.3 presents their particularization for tree ensembles. Section 4.4 relates experiments with these methods on several homogeneous and bipartite biological networks. Section 4.5 illustrates the interpretability of tree-based and Section 4.6 concludes and discusses future work directions.

4.2 Two different approaches

In this section, we present the two generic, local and global, approaches we have adopted for dealing with the classification on pairs problem as stated formally in Section 2.3.1. In the presentation of the approaches, we will assume that we have at our disposal a classification method that derives its classification model from a class conditional probability model. Denoting by $f : \mathcal{X} \rightarrow \{0, 1\}$ a classification model (defined on some input space \mathcal{X}), we will denote by $f^p : \mathcal{X} \rightarrow [0, 1]$ (i.e., with superscript p) the corresponding class conditional probability function (with $f(x) = 1(f^p(x) > p_{th})$ for some user-defined threshold $p_{th} \in [0, 1]$).

4.2.1 Global approach

The most straightforward approach for dealing with the problem of supervised network inference is to apply a classification algorithm on the learning sample LS_p of pairs to learn a function

$$f_{glob} : U_r \times U_c \rightarrow \{0, 1\}$$

on the cartesian product of the two input spaces (resulting in the concatenation of the two input vectors of the nodes of the pair). Predictions can then be computed straightforwardly for any new unseen pair from the function. (**Figure 4.1A**)

In the case of a homogeneous graph, the output function y is symmetric, i.e.,

$$y(n_r, n_c) = y(n_c, n_r), \quad \forall n_r, n_c \in U.$$

We will introduce two adaptations of the approach to handle such graphs. First, for each pair (n_r, n_c) in the learning sample, the pair (n_c, n_r) will also be introduced in the learning sample.

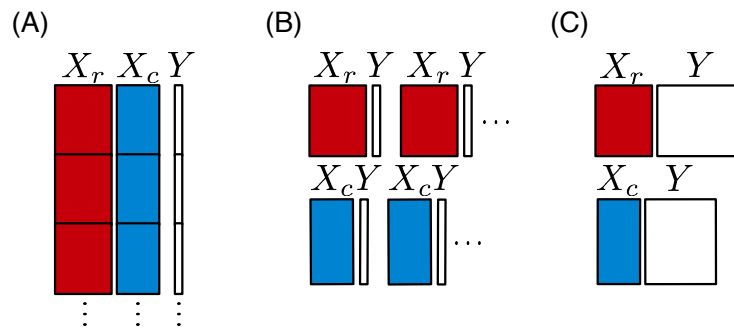


Figure 4.1: Schematic representation of the training data. In the global approach (A) the features vectors are concatenated, in the local approach with single output (B) one function is learnt for each node, and in the local approach with multi-output (C) one function is learnt for one family of nodes and one function for the other one.

Without further constraint on the classification method, this will not ensure however that the learnt function f_{glob} will be symmetric in its arguments. To make it symmetric, we will compute a new class conditional probability model $f_{glob,sym}^P$ from the learned one f_{glob}^P as follows:

$$f_{glob,sym}^P(x_1, x_2) = \frac{f_{glob}^P(x_1, x_2) + f_{glob}^P(x_2, x_1)}{2}.$$

4.2.2 Local approach

The idea of the local approach as first proposed in Bleakley *et al.* (2007), is to build a separate classification model for each node, trying to predict its neighbors in the graph from the known graph around this node. More precisely, for every node $n_c \in LS_c$, a new learning sample $LS(n_c)$ is constructed from the learning sample of pairs LS_p as follows:

$$LS(n_c) = \{ \langle n_r, y(n_r, n_c) \rangle \mid \langle n_r, n_c, y(n_r, n_c) \rangle \in LS_p \}.$$

It can then be used to learn a classification model $f_{n_c} : U_r \rightarrow \{0, 1\}$, which can be exploited to make a prediction for any new pair (n'_r, n'_c) such that $n'_c = n_c$. By symmetry, the same strategy can be adopted to learn a classification model $f_{n_r} : U_c \rightarrow \{0, 1\}$ for each node $n_r \in LS_r$. (**Figure 4.1B**)

These two sets of classifiers can then be exploited to make $LS \times TS$ and $TS \times LS$ types of predictions. For pairs (n_r, n_c) in $LS_r \times LS_c \setminus LS_p$, two predictions can be obtained: $f_{n_c}(n_r)$ and $f_{n_r}(n_c)$. We propose to simply combine them by an arithmetic average of the corresponding class conditional probability estimates:

$$f_{loc}^P(n_r, n_c) = \frac{f_{n_c}^P(n_r) + f_{n_r}^P(n_c)}{2}.$$

As such, the local approach is in principle not able to make directly predictions for pairs of nodes $(n_r, n_c) \in TS \times TS$ (because $LS(n_r) = LS(n_c) = \emptyset$ for $n_r \in TS_r$ and $n_c \in TS_c$). We nevertheless propose to use the following two-steps procedure to learn a classifier for a node $n_r \in TS_r$ (see **Figure 4.2**):

- First, learn all classifiers f_{n_c} for nodes $n_c \in LS_c$,

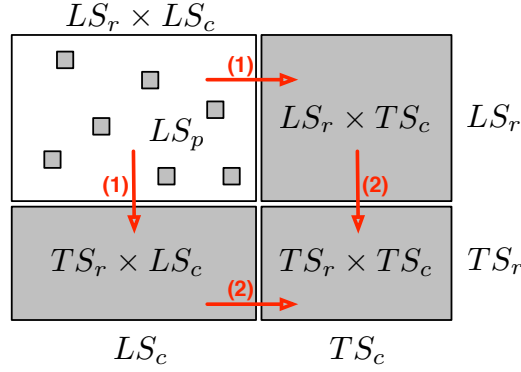


Figure 4.2: The local approach needs two steps to learn a classifier for an unseen node: first, we predict $LS \times TS$ and $TS \times LS$ interactions, and from these predictions, we predict $TS \times TS$ interactions.

- Then, learn a classifier $f_{n_r}^f : U_c \rightarrow \{0, 1\}$ from $LS^f(n_r) = \{\langle n_c, f_{n_c}(n_r) \rangle | n_c \in LS_c\}$, i.e., the predictions given by the models f_{n_c} trained in the first step.

Again by symmetry, the same strategy can be applied to obtain models $f_{n_c}^f$ for the nodes $n_c \in TS_c$. A prediction is then obtained for a pair (n_r, n_c) in $TS \times TS$ by averaging the class conditional probability predictions of both models $f_{n_r}^{f,P}$ and $f_{n_c}^{f,P}$:

$$f_{loc}^P(n_r, n_c) = \frac{f_{n_r}^{f,P}(n_c) + f_{n_c}^{f,P}(n_r)}{2}.$$

Besides averaging, we tried several alternative schemes to merge the two models (such as taking the min, max, or the product of their predictions) but they did not lead to any improvement. Note that building the learning samples $LS^f(n_r)$ and $LS^f(n_c)$ requires to choose a threshold on the class conditional probability estimates. In our experiments, we will set this threshold in such a way that the proportion of edges versus non edges in the predicted subnetworks in $LS \times TS$ and $TS \times LS$ is equal to the same proportion within the original learning sample of pairs. Pahikkala *et al.* (2014a) also proposed recently a two-step procedure to predict interactions between two unseen nodes, with a kernel method.

This strategy can be specialized to the case of a homogeneous graph in a straightforward way. Only one class of classifiers $f_n : U \rightarrow \{0, 1\}$ and $f_n^f : U \rightarrow \{0, 1\}$ are trained for nodes in LS and in TS respectively (using the same two-step procedure as in the asymmetric case for the second). $LS \times LS$ and $TS \times TS$ predictions are still obtained by averaging two predictions, one for each node of the pair.

4.3 Tree-based ensemble methods

Any method could be used as a base classifier for the two approaches. In this chapter, we propose to evaluate the use of tree-based ensemble methods. We then discuss the practical implementation of the two approaches in this context.

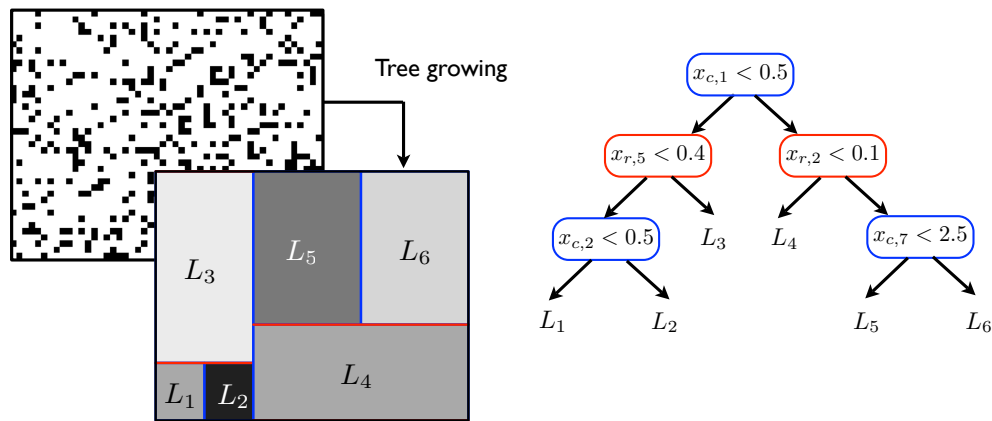


Figure 4.3: In the global approach, the tree growing procedure can be interpreted as interleaving the construction of two trees.

4.3.1 Global approach

The global approach consists in building a tree from the learning sample of all pairs. Each split of the resulting tree will be based on one of the input features coming from either one of the two input feature vectors, $x(n_r)$ or $x(n_c)$. The tree growing procedure can thus be interpreted as interleaving the construction of two trees: one on the row nodes and one on the column nodes. Each leaf of the resulting tree is thus associated with a rectangular submatrix of the graph adjacency matrix $Y(LS_r, LS_c)$ and the construction of the tree is such that the pairs in this submatrix should be, as far as possible, either all connected or all disconnected (see **Figure 4.3** for an illustration).

4.3.2 Local approach

The use of tree ensembles in the context of the local approach is straightforward. We will nevertheless compare two variants. The first one builds a separate model for each row and column nodes as presented in Section 4.2. The second method exploits the ability of tree-based methods to deal with multi-outputs to build only two models, one for the row nodes and one for the column nodes (**Figure 4.1C**). Assuming that the learning sample has been generated by sampling two subsets of objects $LS_r = \{n_r^1, \dots, n_r^{N_r}\}$ and $LS_c = \{n_c^1, \dots, n_c^{N_c}\}$ and that the full adjacency matrix is observed between these two sets, these two models are built from the following learning samples:

$$\begin{aligned}
 LS(n_c) &= \{\langle n_r^i, (y(n_r^i, n_c^1), \dots, y(n_r^i, n_c^{N_c})) \rangle \mid i = 1, \dots, N_r\}, \\
 LS(n_r) &= \{\langle n_c^j, (y(n_r^1, n_c^j), \dots, y(n_r^{N_r}, n_c^j)) \rangle \mid j = 1, \dots, N_c\}.
 \end{aligned}$$

The same multi-output approach can then be applied to build the two models required to make $TS \times TS$ predictions. This approach has the advantage of requiring only four tree ensemble models in total instead of $N_r^U + N_c^U$ models for the single output approach. It can however only be used when the complete submatrix is observed for pairs in $LS \times LS$, since tree-based ensemble method can not cope with missing output values.

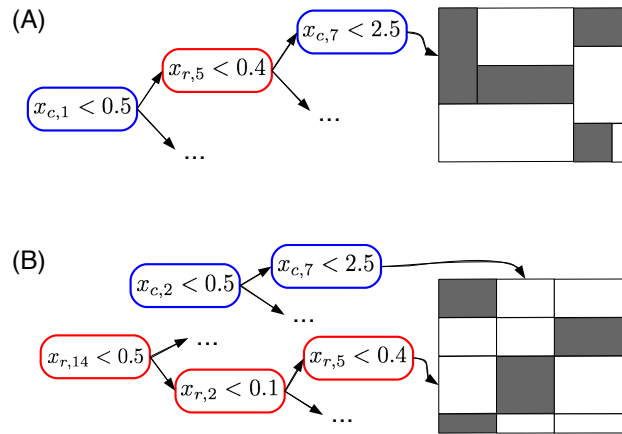


Figure 4.4: Both the global approach (A) and the local approach with multi-output (B) can be interpreted as carrying out a biclustering of the adjacency matrix. Note that in the case of the global approach, the representation is only illustrative. The adjacency submatrices corresponding to the tree leaves can not be necessarily rearranged as contiguous rectangular submatrices covering the initial adjacency matrix.

4.3.3 Interpretability

One main advantage of tree-based methods is their interpretability, directly through the tree structure in the case of single tree models and through feature importance rankings in the case of ensembles (Geurts *et al.*, 2009). Let us to compare both approaches along this criterion.

Tree structure. In the case of the global approach, as illustrated in **Figure 4.4A**, the tree that is built partitions the adjacency matrix $Y(LS_r, LS_c)$ into rectangular regions. These regions are defined such that pairs in each region are either all connected or all disconnect. The region is furthermore characterized by a path in the tree (from the root to the leaf) corresponding to tests on the input features of both nodes of the pair.

In the case of the local multi-output approach, one of the two trees partitions the rows and the other tree partitions the column of the matrix $Y(LS_r, LS_c)$. Each partitioning is carried out in such a way that nodes in each subpartition has a similar connectivity profiles. The resulting partitioning of the adjacency matrix will thus follow a checkerboard structure with also only connected or disconnected pairs in the obtained submatrix, as far as possible (**Figure 4.4B**).

Each submatrix will be furthermore characterized by two conjunctions of tests, one based on row inputs and one based on column inputs. These two methods can thus be interpreted as carrying out a biclustering (Madeira and Oliveira, 2004) of the adjacency matrix where the biclustering is however directed by the choice of tests on the input features. These methods are to biclustering what predictive clustering trees (Blockeel *et al.*, 1998) are to clustering.

In the case of the local single output approach, the partitioning is more fine-grained as it can be different from one row or column to another. However in this case, each tree gives an interpretable characterization of the nodes which are connected to the node from which the tree was built.

Feature importance rankings. When using ensembles, the global approach provides a global ranking of all features from the most to the less relevant. The local multi-output approach provides

two separate rankings, one for the row features and one for the column features and the local single output approach gives a separate ranking for each node. All variants are therefore complementary from an interpretability point of view.

4.3.4 Implementation and computational issues

Global approach. In principle, since tree building is a batch algorithm, the global approach requires to generate the full sample of all pairs, which may be very prohibitive for graphs defined on a large number of nodes (e.g., the PPI network used in our experiments contains about 1000 nodes that lead to about 1 millions pairs described by 650 attributes). Fortunately, since the tree building method goes through the input features one by one, one can separately search for the best split on features relative to nodes in U_r and on features relative to nodes in U_c , which does not require to generate explicitly the full data matrix. This is an important advantage with respect to kernel-based methods that typically requires to handle explicitly a $N_r N_c \times N_r N_c$ Gram matrix. Since tree growing is order $O(N \log(N))$ for a training sample of size N , the computational complexity of the whole procedure however remains $O(N_c N_r (\log(N_c) + \log(N_r)))$. The complexity of the trees (measured by the total number of nodes) is at worst $O(N_c N_r)$ (corresponding to a fully developed tree) but in practice it is related to the number of positive interactions in the training sample, which is typically much lower than $N_c N_r$.

Local approach. The computational complexity of the local approach is the same as the computational complexity of the global approach, i.e. $O(N_c N_r \log(N_r) + N_r N_c \log(N_c))$. Indeed, in the single output approach, N_c and N_r models need to be constructed respectively from N_r samples and N_c samples each. In the multi-output case, only two models are constructed from N_r and N_c samples respectively, but the multi-output variant needs to go through all outputs at each tree node, which multiplies complexity by respectively N_r and N_c for these two models. However, at worst, the complexity of the model is $O(N_c N_r)$ for the single output approach and $O(N_r + N_c)$ for the multi-output approach, which potentially gives an important advantage along this criterion for the multi-output method.

4.4 Experiments

In this section, we carried out a large scale empirical evaluation of the different methods described in Section 4.2 on six real biological networks, three homogeneous graphs and three bipartite graphs. Results on four additional (drug-protein) networks can be found in the supplementary material. Our goal with these experiments is to assess the relative performances of the different approaches and to give an idea of the performance one could expect from these methods on biological networks of different nature. Section 4.4.4 provides a comparison with existing methods from the literature.

4.4.1 Datasets

The first three networks correspond to homogeneous indirect graphs and the last seven to bipartite graphs. The main characteristics of the datasets are summarized in **Table 4.1**.

Table 4.1: Summary of the ten datasets used in the experiments.

	Network	Network size	Number of edges	Number of features
<i>Homogeneous networks</i>	PPI	984×984	2438	325
	EMAP	353×353	1995	418
	MN	668×668	2782	325
<i>Bipartite networks</i>	ERN	154×1164	3293	445/445
	SRN	113×1821	3663	9884/1685
	DPI	1862×1554	4809	660/876
	DPI-E	445×664	2926	445/664
	DPI-I	210×204	1476	210/204
	DPI-G	223×95	635	223/95
	DPI-N	54×26	90	54/26

Protein-protein interaction network (PPI). This network has been compiled from the 2438 high confidence interactions between 984 *S.cerevisiae* proteins highlighted by Mering *et al.* (2002). The input features used for the predictions are a set of expression data, phylogenetic profiles and localization data that totalizes 325 features. This dataset has been used in several studies before (Yamanishi and Vert, 2004; Kato *et al.*, 2005; Geurts *et al.*, 2007).

Genetic interaction network (EMAP). This network (Schuldiner *et al.*, 2005) contains 353 *S.cerevisiae* genes (nodes) connected with 1995 negative epistatic interactions (edges). Inputs consists in measures of growth fitness of yeast cells relative to deletion of each gene separately, and in 418 different environments. (Hillenmeyer *et al.*, 2008).

Metabolic network (MN). This network (Yamanishi and Vert, 2005) is composed of 668 *S.cerevisiae* enzymes (nodes) connected by 2782 edges. There is an edge between two enzymes when these two enzymes catalyze successive reactions. The input feature vectors are the same as those used in the PPI network.

E.coli regulatory network (ERN). This bipartite graph (Faith *et al.*, 2007b) connects transcription factors (TF) and genes of *E.coli*. It is composed of 1164 genes regulated by 154 TF. There is a total of 3293 interactions. The input features (Faith *et al.*, 2007b) are 445 expression values.

S.cerevisiae regulatory network (SRN). This network (Maclsaac *et al.*, 2006) connects TFs and their target genes from *E.coli*. It is composed of 1855 genes regulated by 113 TFs and totalizing 3737 interactions. The input features are 1685 expression values (Hughes *et al.*, 2000; Hu *et al.*, 2007; Chua *et al.*, 2006; Faith *et al.*, 2007a). For genes, we concatenated motifs features (Brohée *et al.*, 2011) to the expression values.

Drug-protein interaction network (DPI). Drug-target interactions (Yamanishi *et al.*, 2011) are related to human and connect a drug with a protein when the drug targets the protein. This network holds 4809 interactions involving 1554 proteins and 1862 drugs. The input features are a binary vectors coding for the presence or absence of 660 chemical substructures for each drug, and the presence or absence of 876 PFAM domains for each protein (Yamanishi *et al.*, 2011).

Four kinds of drug-protein interaction networks (DPI-*). Yamanishi *et al.* (2008) proposed four different drug-protein interaction networks in which proteins belong to four pharmaceutically useful classes: enzymes (DPI-E), ion channels (DPI-I), G-protein-coupled receptors (DPI-G) and nuclear receptors (DPI-N). The input features for proteins are similarity with all proteins in terms of sequence and the input features for drugs are similarity with all drugs in terms of chemical structure (Yamanishi *et al.*, 2008). The number of drugs in these networks are respectively 445, 210, 223 and 54, the number of proteins are 664, 204, 95 and 26 and the number of interactions are 2926, 1476, 635 and 90.

4.4.2 Protocol

Pairs can be divided into four families, according to the number of nodes in the pair that are represented in the learning sample (see Section 3.3). Since prediction performances are expected to differ between the different families of pairs, predictions will then be evaluated separately.

In our experiments, $LS \times LS$ performances in each network are measured by 10 fold cross-validation (CV) across the pairs of nodes, as explained in Section 3.3.1. For robustness, results are averaged over 10 runs of 10 fold CV. $LS \times TS$, $TS \times LS$ and $TS \times TS$ predictions are assessed by performing a 10 times 10 fold CV across the nodes, as explained in Section 3.3.2. The different algorithms return class conditional probability estimates. To assess our models independently of a particular choice of discretization threshold P_{th} on these estimates, we vary this threshold and output in each case the resulting precision-recall curve and the resulting ROC curve. Methods are then compared according to the total area under these curves, denoted AUPR and AUROC respectively (the higher the AUPR and the AUROC, the better), averaged over the 10 folds and the 10 CV runs. For all our experiments, we use ensembles of 100 extremely randomized trees with default parameter setting (Geurts *et al.*, 2006a).

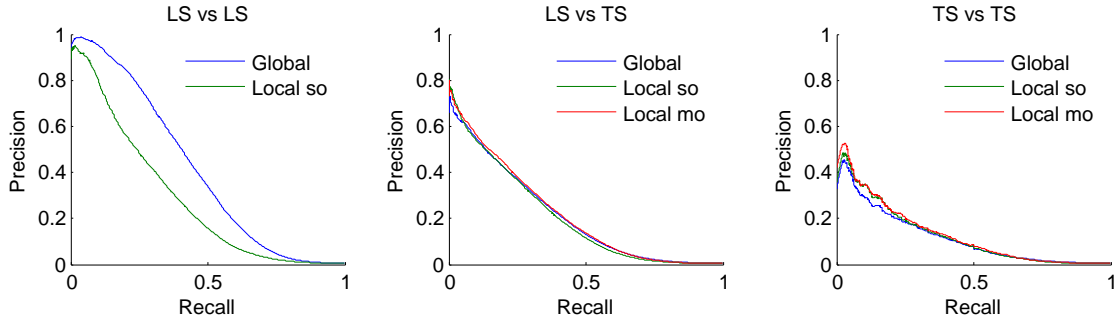
Realistic baseline. As highlighted in Section 3.5, in biological networks, nodes of high degree have a higher chance to be connected to any new node. In our context, this means that we can expect that the degree of a node will be a good predictor to infer new interactions involving this node. We want to assess the importance of this effect and provide a more realistic baseline than the usual random guess performance. To reach this goal, we evaluate the AUROC and AUPR scores when using the sum of the degrees of each node in a pair to rank $LS \times LS$ pairs and when using the degree of the nodes belonging to the LS to rank $TS \times LS$ or $LS \times TS$ pairs. AUROC and AUPR scores will be evaluated using the same protocol as here above. As there is no information about the degrees of nodes in $TS \times TS$ pairs, we will use random guessing as a baseline for the scores of these predictions (corresponding to an AUROC of 0.5 and an AUPR equal to the proportion of interactions among all nodes pairs).

4.4.3 Results

We discuss successively the results on the three homogeneous graphs and then on the seven bipartite graphs.

Table 4.2: Areas under curves for homogeneous networks.

		Precision-Recall (AUPR)			ROC (AUC)		
		LS-LS	LS-TS	TS-TS	LS-LS	LS-TS	TS-TS
<i>PPI</i>	Global	0.41	0.22	0.10	0.88	0.84	0.76
	Local so	0.28	0.21	0.11	0.85	0.82	0.73
	Local mo	-	0.22	0.11	-	0.83	0.72
	Baseline	0.13	0.02	0.00	0.73	0.74	0.50
<i>EMAP</i>	Global	0.49	0.36	0.23	0.90	0.85	0.78
	Local so	0.45	0.35	0.24	0.90	0.84	0.79
	Local mo	-	0.35	0.23	-	0.85	0.80
	Baseline	0.30	0.13	0.03	0.87	0.80	0.50
<i>MN</i>	Global	0.71	0.40	0.09	0.95	0.85	0.69
	Local so	0.57	0.38	0.09	0.92	0.83	0.68
	Local mo	-	0.45	0.14	-	0.85	0.71
	Baseline	0.05	0.04	0.01	0.75	0.70	0.50

**Figure 4.5:** Precision-recall curves for PPI network

Homogeneous graphs

AUPR and AUROC values are summarized in **Table 4.2** for the three methods: global, local single output, and local multiple output. The last row on each dataset is the baseline result obtained as described in Section 4.4.2. **Figures 4.5, 4.6** and **4.7** shows the precision-recall curves obtained by the different approaches on PPI, EMAP and MN, for the three different protocols.

To improve the $TS \times TS$ results, we tried to merge the predictions of the pairs in different ways: instead of taking the average of the two predictions, we tried taking the minimum value, the maximum value or the product of the two values, but the performance did not increase.

In terms of absolute AUPR and AUC values, $LS \times LS$ pairs are clearly the easiest to predict, followed by $LS \times TS$ pairs and $TS \times TS$ pairs. This ranking was expected from previous discussions. Baseline results in the case of $LS \times LS$ and $LS \times TS$ predictions confirm that node degrees are very informative: baseline AUC values are much greater than 0.5 and baseline AUPR values are also significantly higher than the proportion of interactions among all pairs (which are 0.005, 0.03, and 0.01 respectively for PPI, EMAP, and MN), especially in the case of $LS \times LS$ predictions. Nevertheless, the performance of our methods are better than these baselines in all cases. On the EMAP network, the difference in terms of AUC is very slight but the difference in terms of AUPR is important. This is typical of highly skewed classification problems, where precision-recall curves

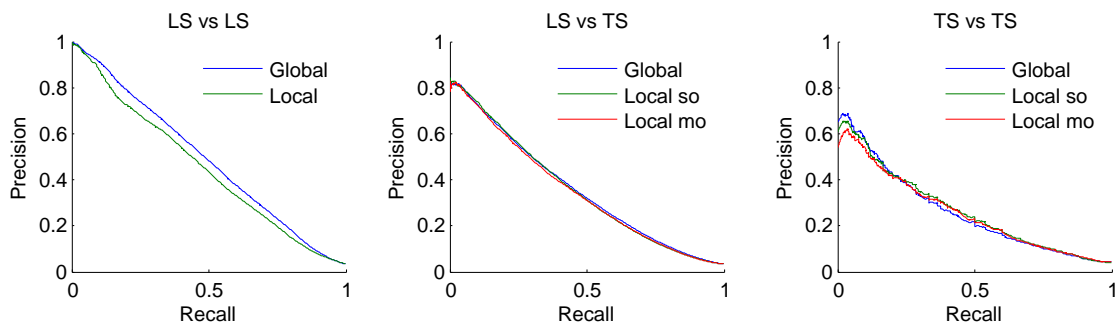


Figure 4.6: Precision-recall curves for EMAP network

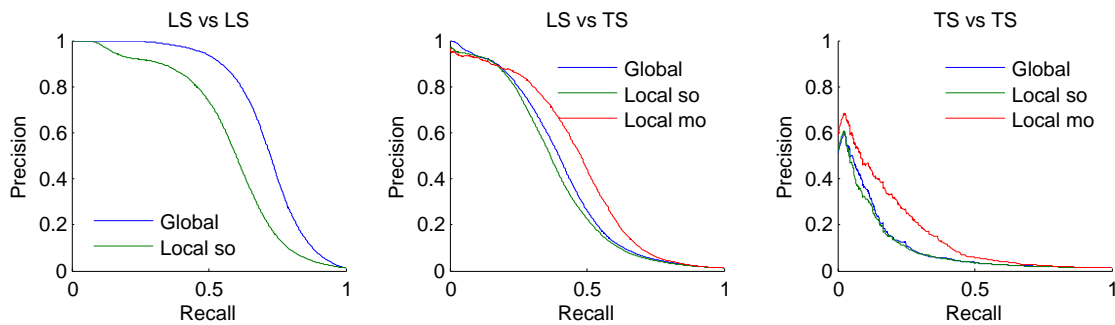


Figure 4.7: Precision-recall curves for metabolic network

are known to give a more informative picture of the performance of an algorithm than ROC curves Davis and Goadrich (2006).

All tree-based approaches are very close on $LS \times TS$ and $TS \times TS$ pairs but the global approach has an advantage over the local one on $LS \times LS$ pairs. The difference is important on the PPI and MN networks. For the local approach, the performance of single and multi-output approaches are indistinguishable, except with the metabolic network where the multi-output approach gives better results. This is in line with the better performance of the global versus the local approach on this problem, as indeed both the global and the local multi-output approaches grow a single model that can potentially exploit correlations between the outputs. Notice that the multi-output approach is not feasible when we want to predict $LS \times LS$ pairs, as we are not able to deal with missing output values in multi-output decision trees.

Bipartite graphs

AUPR and AUROC values are summarized in **Table 4.3**. **Figures 4.8 to 4.14** show the precision-recall curves obtained by the different approaches on ERN, SRN, DPI and the four other kinds of DPI, for the four different protocols. 10 times 10-fold CV was used as explained in Section 4.4.2. Nevertheless, two difficulties appeared in the experiments performed on the DPI network. First, the dataset is larger than the others, and the 10-fold CV was replaced by 5-fold CV to reduce the computational space and time burden. Second, the feature vectors are binary and the randomization of the threshold (in Extra-Tree algorithm) cannot lead to diversity between the different trees of the ensemble. So we used bootstrapping to generate the training set of each tree.

Like for the homogeneous networks, higher is the number of objects of a pair present in the learning set, better are the predictions, i.e., AUPR and AUC values are significantly decreasing from

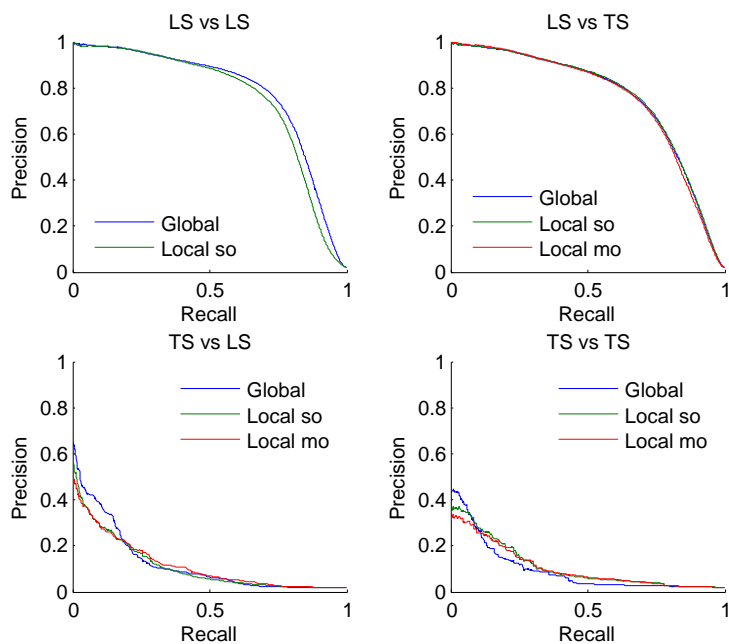


Figure 4.8: Precision-recall curves for E.coli regulatory network (TF vs genes)

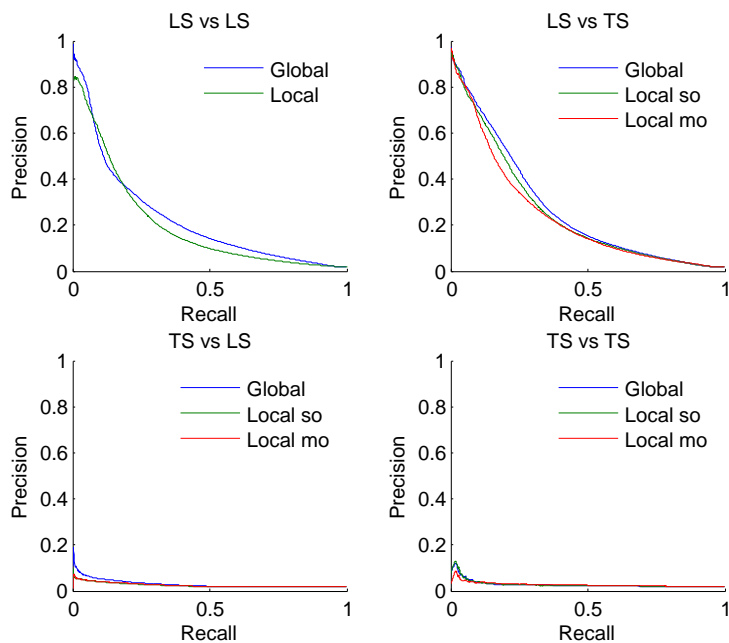


Figure 4.9: Precision-recall curves for S.cerevisiae regulatory network (TF vs genes)

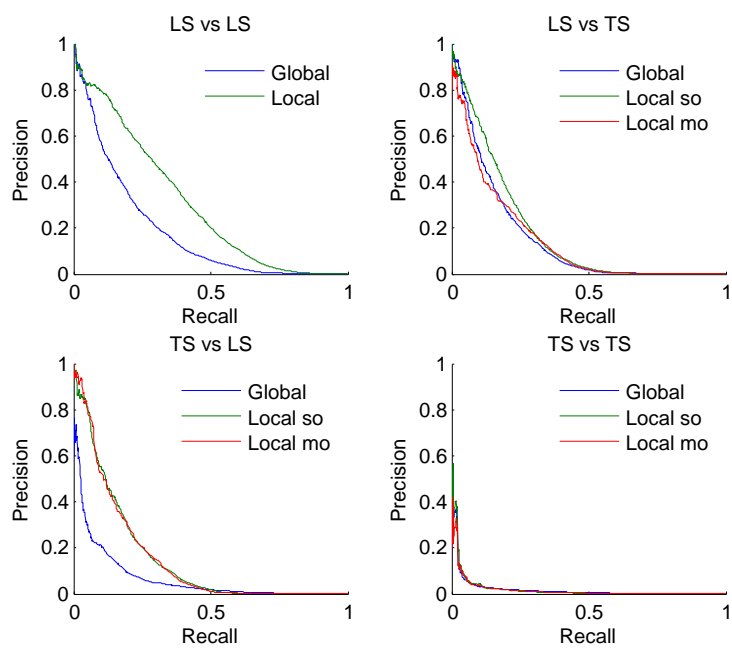


Figure 4.10: Precision-recall curves for drug-protein interaction network

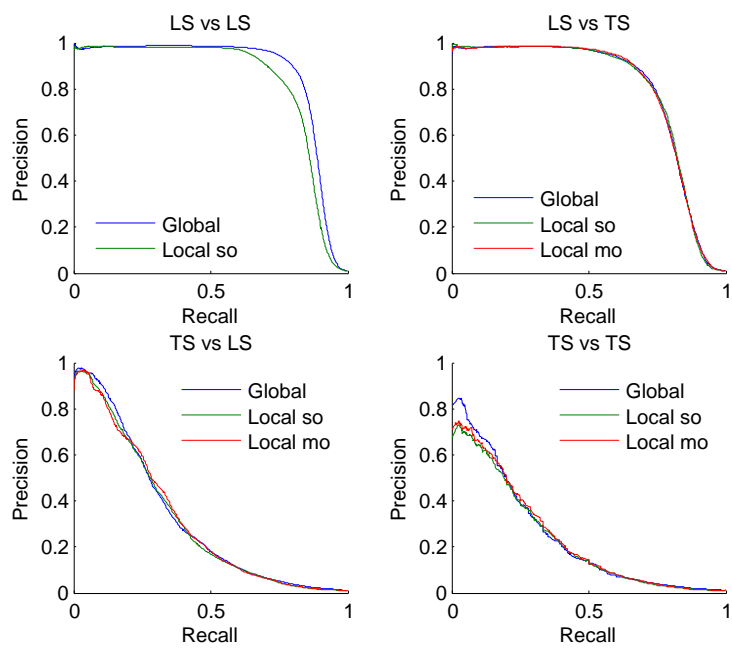


Figure 4.11: Precision-recall curves for drug-protein (enzymes) interaction network

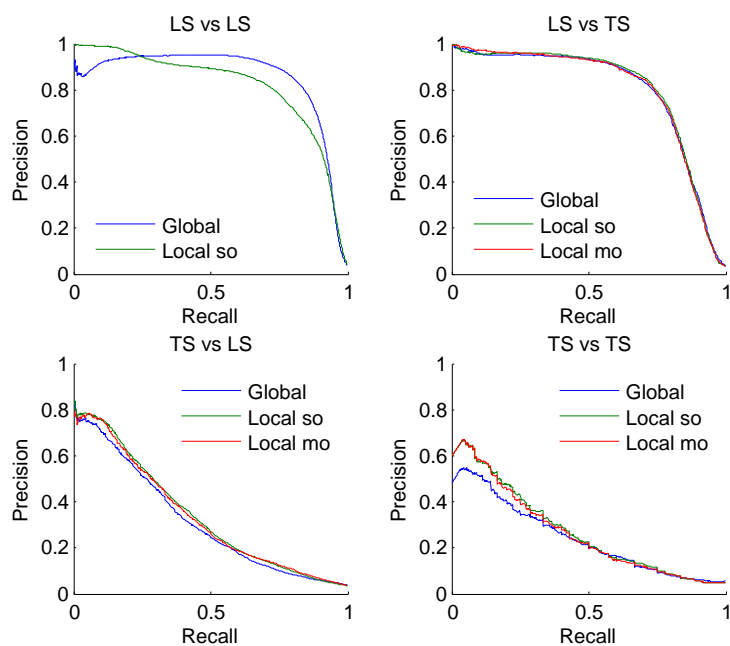


Figure 4.12: Precision-recall curves for drug-protein (ion channels) interaction network

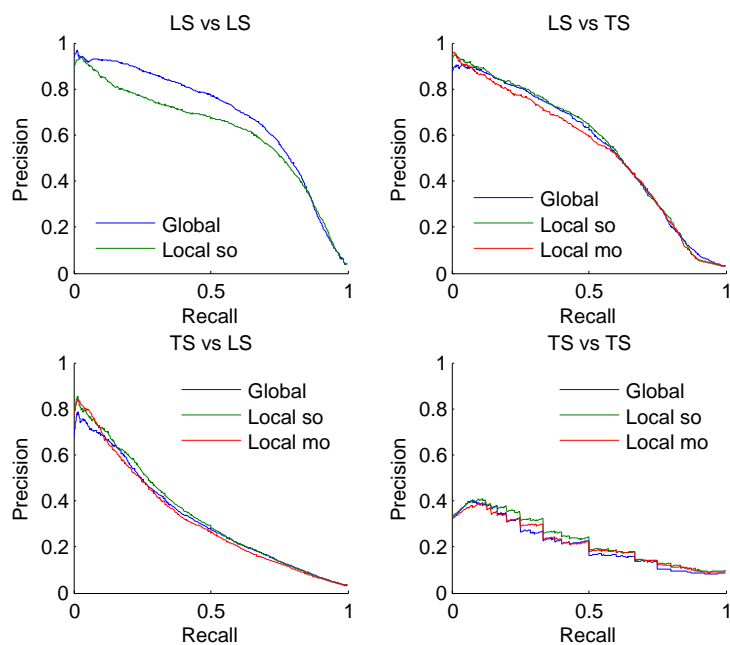


Figure 4.13: Precision-recall curves for drug-protein (GPCR) interaction network

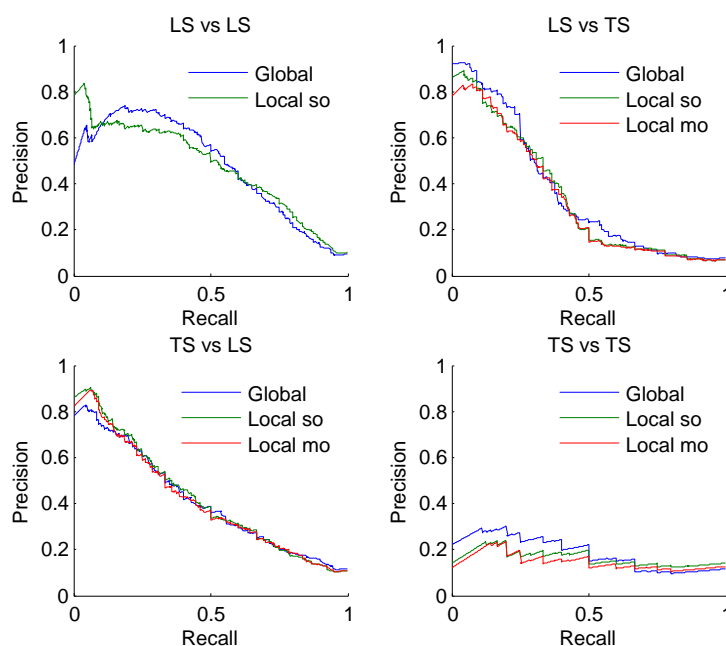


Figure 4.14: Precision-recall curves for drug-protein (nuclear receptors) interaction network

$LS \times LS$ to $TS \times TS$ predictions. On the ERN and SRN networks, performances are very different for the two kinds of $LS \times TS$ predictions that can be defined, with much better results when generalizing over genes (i.e., when the TF of the pair is in the learning sample). On the other hand, on the DPI network, both kinds of $LS \times TS$ predictions are equally well predicted. These differences are probably due in part to the relative numbers of objects of both kinds in the learning sample, as there are much more genes than TFs on ERN and SRN and a similar number of drugs and proteins in the DPI network. Differences are however probably also related to the intrinsic difficulty of generalizing over each object family, as on the four additional DPI networks (see the supplementary material), generalization over drugs is most of the time better than generalization over proteins, irrespectively of the relative numbers of drugs and proteins in the training network. Results are most of the time better than the baselines (based on node degrees for $LS \times LS$ and $LS \times TS$ predictions and on random guessing for $TS \times TS$ predictions). The only exceptions are observed when generalizing over TFs on SRN and when predicting $TS \times TS$ pairs on SRN and DPI.

The three tree-based approaches are very close to each other. Unlike on homogeneous graphs, there is no strong difference between the global and the local approach on $LS \times LS$ predictions: it is slightly better in terms of AUPR on ERN and SRN but worse on DPI. The single and multi-output approaches are also very close, both in terms of AUPR and AUC.

4.4.4 Comparison with related works

In this section, we compare our methods with several other network inference methods from the literature. To ensure a fair comparison and avoid errors related to the reimplementation and tuning of each of these methods, we choose to rerun our algorithms in similar settings as in related papers. All comparison results are summarized in **Table 4.4** and **4.5** and discussed below.

Table 4.4: Comparison with related works on four networks: our results are comparable.

Publication	DB	Protocol	Measures	Their results	Our results
Bleakley <i>et al.</i> (2007)	PPI	$LS \times TS$, 5CV	AUPR	0.25	0.21
	MN			0.41	0.43
Geurts <i>et al.</i> (2007)	PPI	$LS \times TS$, 10CV	AUPR / ROC	0.18 / 0.91	0.22 / 0.84
		$TS \times TS$		0.09 / 0.86	0.10 / 0.76
	MN	$LS \times TS$		0.18 / 0.85	0.45 / 0.85
		$TS \times TS$		0.07 / 0.72	0.14 / 0.71
Mordelet and Vert (2008)	ERN	$LS \times TS$, 3CV	Recall 60 / 80	0.44 / 0.18	0.38 / 0.15
Yamanishi <i>et al.</i> (2011)	DPI	$LS \times LS$, 5CV	AUROC	0.75	0.88
Tabei <i>et al.</i> (2012)	DPI	$LS \times LS$, 5CV	AUROC	0.87	0.88
		$LS \times TS$ & $TS \times LS$		0.74	0.74

These comparisons show that tree-based methods are competitive on all ten networks. Moreover, it has to be noticed that:

- no other method has been tested over all these problems,
- we have not tuned any parameters of the Extra-trees methods. Better performances could be achieved by changing, for example, the randomization scheme Breiman (2001), the feature selection parameter K , or the number of trees.

Homogeneous graphs

Bleakley *et al.* (2007) developed and applied the local approach with support vector machines to predict the PPI and MN networks and show that it was superior to several previous works Yamanishi and Vert (2004); Kato *et al.* (2005). They only try to predict interactions between one protein belonging to the learning set and one protein belonging to the test set ($LS \times TS$ predictions) and used 5-fold CV. Although they exploited yeast-two-hybrid data as additional features for the prediction of the PPI network, we obtain very similar performances with the local multi-output approach: they got an AUPR equal to 0.25 for PPI and equal to 0.41 for MN. Applying a 5-fold CV, we respectively got areas under the curve equal to 0.21 and 0.43 for the prediction of $LS \times TS$ pairs, with the multiple-output local approach.

Geurts *et al.* (2007) used ensembles of output kernel trees (see Section 2.2.4 for an explanation of the algorithm) to infer the MN and PPI networks with the same input data as Bleakley *et al.* (2007). They try to predict $LS \times TS$ and $TS \times TS$ interactions. Applying a (single) 10-fold CV, they got AUPR/AUROC equal to 0.18/0.91 and 0.09/0.86 respectively for the LS-TS and TS-TS of PPI, and equal to 0.18/0.85 and 0.07/0.72 for MN. With the global approach, we got AUPR/AUROC equal to 0.22/0.84 and 0.10/0.76 respectively for the $LS \times TS$ and $TS \times TS$ of PPI (with the global approach). For MN we obtain values equal to 0.45/0.85 and 0.14/0.71 for MN (with the multiple-output local approach). We obtain thus similar or inferior results as Geurts *et al.* (2007) in terms of AUC but much better results in terms of AUPR, especially on the MN data.

Table 4.5: The areas under the curves obtained with our methods are comparable those obtained with methods from the literature, on the four classes of DPI network.

	Method	Precision-Recall			Method	ROC LS-LS
		LS-LS	LS-TS	TS-LS		
<i>DPI-E</i>	KRM ¹	0.83	0.81	0.38	DBSI ³	0.78
	BLM ²	0.83	0.81	0.39	TBSI ³	0.90
					NBI ³	0.97
	Ours	0.87	0.79	0.32		0.97
<i>DPI-I</i>	KRM	0.76	0.81	0.31	DBSI	0.71
	BLM	0.77	0.80	0.32	TBSI	0.90
					NBI	0.98
	Ours	0.85	0.80	0.34		0.97
<i>DPI-G</i>	KRM	0.67	0.62	0.41	DBSI	0.76
	BLM	0.65	0.55	0.38	TBSI	0.75
					NBI	0.94
	Ours	0.68	0.55	0.34		0.95
<i>DPI-N</i>	KRM	0.74	0.61	0.51	DBSI	0.79
	BLM	0.58	0.35	0.40	TBSI	0.53
					NBI	0.84
	Ours	0.48	0.36	0.42		0.86

¹ Yamanishi *et al.* (2008)

² Bleakley and Yamanishi (2009)

³ Cheng *et al.* (2012)

Notice that a ROC curve is generally a less good indicator of the performance than a precision-recall curve, because the biological network are typically very unbalanced. Indeed when the number of negative examples is much higher than the number of positive examples, a large change in the number of false positives can lead to a small change in the false positive rate used in ROC curves, but a large change in the precision use in precision-recall curves Davis and Goadrich (2006).

Bipartite graphs

ERN. Mordelet and Vert (2008) used SVM to predict ERN with the local approach, focusing on the prediction of interactions between known transcription factors and new genes ($LS \times TS$). They evaluated their performances by the precision at 60% and 80% recall respectively, estimated by 3-fold CV. To avoid a bias in their results, they ensure that all genes belonging to a same operon are always in the same fold. Adopting the same evaluation protocol, we got a recall of 37.8% for a prediction of 60% and a recall of 15% for a prediction of 80% (with the multiple-output local approach). Our results are very close although slightly less good.

DPI. The DPI network was predicted in Yamanishi *et al.* (2011) using sparse canonical correspondence analyze (SCCA). This method correlates protein domains to chemical substructures expected to be present in their ligand. Only $LS \times LS$ predictions were considered. They tested their method with a 5-fold CV and obtained an AUROC equal to 75%, while we got an AUROC equal to 88% (single-output local approach). We obtain better results.

Tabei *et al.* (2012) also predicted DPI, by using the global approach and L_1 regularized linear classifiers using as input features all possible products of one drug feature and one protein feature. They differentiate “pair-wise cross validation” (equivalent to our $LS \times LS$ predictions) from “block-wise cross validation” (equivalent to our $LS \times TS$ and $TS \times LS$ predictions). With 5-fold CV, they obtained an AUROC equal to 0.87 for pair-wise CV, and equal to 0.74 for block-wise CV, while we obtained AUROC respectively equal to 0.88 (with the single-output local approach) and 0.74 (with the global approach). The results are thus very similar.

DPI-* Yamanishi *et al.* (2008) and Bleakley and Yamanishi (2009) used SVM to predict the four classes of drug-protein interaction network. The first one used a kernel regression-based method (KRM): a global approach in which they integrated the chemical and genomic spaces into a unified space. The second one used bipartite local models (BLM) and then did not predict $TS \times TS$ interactions. We compared the AUPR of these two methods with ours, in a 10 times 10-fold CV (**Table 4.5**). Extra-Trees (E-T) is comparable to the other methods, sometimes giving better results (for DPI-I) and sometimes giving less good results (for DPI-N).

Cheng *et al.* (2012) developed three different supervised inference methods, which they tested on the four DPI datasets. The methods are drug-based similarity inference (DBSI), target-based similarity inference (TBSI) and network-based inference (NBI). The last one only use network topology similarity to infer new targets for known drugs. NBI gives the best performance of the three but has the disadvantage to only be able to predict $LS \times LS$ pairs. Extra-Trees give better or equal results than these three methods, in terms of AUROC, when doing 10 times 10-fold CV (**Table 4.5**).

4.5 Illustration of interpretability of trees

One advantage of tree-based methods is their interpretability. In this section we will illustrate this interpretability by carrying out some experiments on a drug-protein interaction network. Note that our goal with these experiments is merely to propose and illustrate several semi-automatic procedures to extract interpretable information from tree-based models. We will not try to assess the biological relevance of our findings.

The dataset we will use for the illustration contains 4809 interactions involving 1862 drugs and 1554 proteins, which gives a proportion of interactions equal to 0.17%. Common chemical substructures are often shared by drugs that bind a given protein, and common function sites (e.g., PFAM domains) are often shared by proteins that are bound by a given drug (Yamanishi *et al.*, 2011). Input data used to perform predictions is therefore composed of:

- a binary vector coding for the presence or the absence of 660 chemical substructures for each drug,
- a binary vector coding for the presence or the absence of 876 PFAM domains for each protein.

We first discuss interpretability in the case of single decision trees built using the local or the global approach (Section 4.5.1). We then show how to extract information from ensembles, first in the form of biclusters (Section 4.5.2) and second in the form of pair-based feature rankings (Section 4.5.3).

4.5.1 Interpretability of single decision trees

As discussed in Section 4.3.3, the local approach with multi-output trees yields a checkerboard structured biclustering of the adjacency matrix, while the global approach yields an unconstrained biclustering of the same matrix. In both cases, each bicluster is defined by a subset of row nodes (drugs) and a subset of column nodes (proteins) and by construction is such that pairs defined by these two subsets are either highly connected or highly disconnected. We illustrate below both methods on the DPI network.

Local approach with two multiple-output trees

With the local approach and multi-output trees, two single trees are built: the first one partitions the row nodes (the drugs) and the other partitions the column nodes (the proteins). The drugs that end in a same leaf form a cluster of drugs, and the proteins that end in a same leaf form a cluster of proteins. If we look at the network through its adjacency matrix, each pair of drug and protein clusters define a rectangular subregion of the adjacency matrix and the rectangular regions corresponding to all pairs can be arranged as a checkerboard (see **Figure 4.4B**). Each of these rectangular regions, or biclusters, corresponds to a bipartite subnetwork of the global network. Because of the way the two trees are built, we expect that the resulting subnetworks have either significantly many or significantly few drug-protein interactions. Indeed, drugs (resp. proteins) that fall in the same leaf are assumed to have very similar connectivity profiles with all proteins (resp. drugs), i.e., the corresponding rows (resp. columns) in the adjacency matrix are expected to be similar. Putting together these two constraints (similar rows and similar columns), the rectangular subregion of the adjacency matrix defined by a cluster of drugs and a cluster of proteins should thus either contain many 1s or many 0s. Each cluster of drugs/proteins furthermore corresponds to a path in one of the tree, i.e. a conjunction of tests based on the input features. Given the

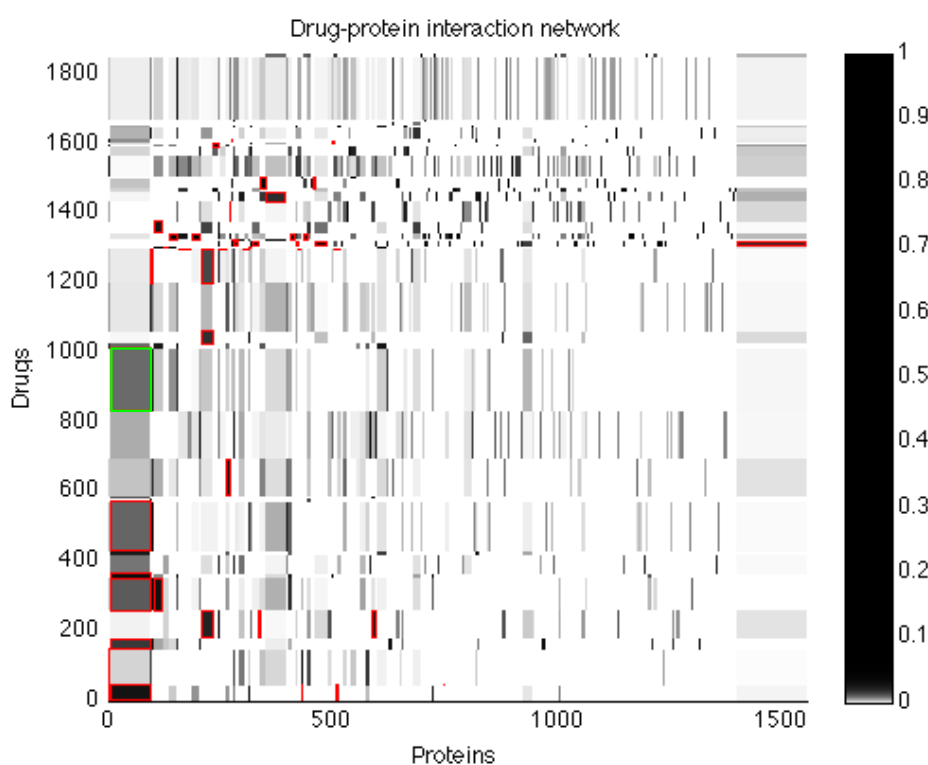


Figure 4.15: The adjacency matrix of the drug-protein interaction network is partitioned into clusters by the two single multi-output trees. Rows represent drugs, while columns represent proteins. Each pair of clusters is expected to be either very connected, or not. The 47 submatrices with p -values lower than 10^{-7} are highlighted in red. In total, they contain a proportion of interactions equal to 2%.

nature of the features for this particular network, clusters of drugs are thus characterized by the absence or presence of some chemical substructures (those that are tested along the tree path), while similarly clusters of proteins are characterized by the absence or presence of some PFAM domains. To understand the model, one should thus look at the subnetworks defined by the two trees that contain a significantly high number of interactions and extract from the tree the chemical substructures and PFAM domains that characterize these subnetworks.

Figure 4.15 shows the checkerboard partition of the adjacency matrix obtained on the drug-protein interaction network. The grey level of each submatrix represents the ratio of interactions among all pairs in the submatrix. To obtain this partition, we arbitrarily stop splitting a node in each tree if the number of drugs or proteins it contains goes below 10% of the total number of drugs or proteins. This corresponds to using a value of $n_{\min} = 186$ for the drugs and $n_{\min} = 155$ for the proteins (see Section 2.2.1). Applying some pruning is necessary to get meaningful clusters. Indeed, if one lets the algorithm grow fully developed trees, most tree leaves, and therefore clusters, will typically contain only one or very few nodes (since most drugs and proteins have different connectivity profiles).

The partition of **Figure 4.15** highlights the very unbalanced nature of the grown trees. If the trees were balanced, given our choice of n_{\min} we would have obtained about 10 clusters/leaves per

dimension, while we have obtained much more leaves, in particular in the tree grown for the proteins (52 for the drugs and 475 for the proteins). This is a consequence of the difficulty of the task: several tree splits only separate a few drugs/proteins from the rest of the drugs/proteins in the corresponding tree node and many splits are thus required to reach the 10% size limit.

We observe also that most subregions contain a large majority of 0s. This is not surprising. Indeed, given the very sparse nature of the network, we did not expect to find subnetworks where all drugs are interacting with all proteins. We nevertheless expect that some of them will contain proportionally more interactions than the global network and some of them proportionally less interactions than the global network. As the goal is mainly to understand what defines the interactions (not the non-interactions), the most interesting subnetworks are those which are proportionally more connected than the global network. To identify them in our illustrative problem, we computed a p -value for each subnetwork that measures whether it is significantly connected or not with respect to a randomly selected subnetwork of the same size. This p -value can be estimated by random permutations: we randomly draw 10^7 windows of the same size (i.e., with the same numbers of drugs and proteins) in the adjacency matrix, and count how many of these windows contain at least the same number of connections as the target window. By dividing the resulting number by 10^7 , we obtain the probability to have a more connected window by chance from the global network. We computed the p -values relative to all submatrices on the drug-protein network. The 47 regions that obtained an estimated p -value lower than 10^{-7} are highlighted in red in **Figure 4.15**. These are thus the subregions for which we did not find any random subregions of the same size with an equal number or more connections. In total, these 47 regions contain a proportion of interactions equal to 2%, which is more than 12 times higher than the proportion of interactions in the global network (i.e., 0.17%).

To illustrate the feature-based characterization of the submatrices, let us analyze the significant submatrix with the highest number of interactions. This submatrix is located in the green rectangle in **Figure 4.15** between the drugs numbered 835 and 1014 and the proteins numbered 6 and 94. **Figure 4.16** shows a zoom of this submatrix, where interactions are represented by black dots. One counts in this submatrix 152 interactions between 180 drugs and 89 proteins, which gives a proportion of interactions almost 6 times greater than in the global network.

The path in the tree relative to drugs that leads to this submatrix is composed of a succession of tests based on the presence or absence of the following drug substructures (in their order of appearance in the path):

1. SC1C(N)CCCC1
2. ≥ 1 Co
3. N-S-C:C
4. ≥ 1 saturated or aromatic nitrogen-containing ring size 8'
5. ≥ 1 unsaturated non-aromatic carbon-only ring size 10' 'C(~Br)(~H)
6. ≥ 1 any ring size 7
7. ≥ 1 **saturated or aromatic carbon-only ring size 6**
8. **C(~C)(~H)(~N)**
9. O-C:C-O-[#1]
10. N-H
11. Nc1cc(Cl)ccc1
12. O-C:C-O

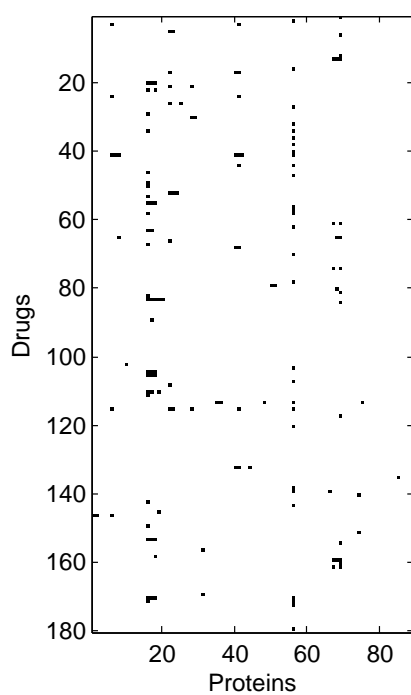


Figure 4.16: The submatrix having the greatest number of interaction, among those with a p -value smaller than 10^{-7} , is characterized in the text.

In this list, the two substructures in bold are the substructures present in the 180 drugs of the subnetwork, while the others are absent in all of them. Similarly, the path in the tree relative to the proteins is composed of a succession of tests based on the presence or absence of the following three PFAM domains:

1. Eukaryotic-type carbonic anhydrase
2. Oestrogen receptor
3. **7 transmembrane receptor (rhodopsin family)**

The bold PFAM domain is the domain present in the 89 proteins of the subnetwork, while the two others are absent in all of them. The features in these two lists, particularly the bold ones, are expected to explain the significantly high proportion of drug-protein interactions within the cluster: a drug with one or more saturated or aromatic carbon-only ring of size 6 and which contains $C(\sim C)(\sim H)(\sim N)$ as a substructure is more likely to interact with a protein from the rhodopsin family than any random protein.

Global approach with one single-output tree

The global approach with single tree can also be interpreted as carrying out a biclustering of the adjacency matrix. Only one tree is built in this case, which classifies drug-protein pairs. Each leaf of this tree corresponds to a subset of pairs from the learning sample. As features are defined either on drugs or on proteins (not on pairs), each leaf also corresponds to a cluster of drugs and a cluster of proteins and all pairs that involve drugs and proteins from these clusters fall into that leaf. Like for the local approach, each leaf thus also delimits a rectangular subregion of the adjacency matrix and this region is therefore characterized by a path in the tree, and consequently by a conjunction of tests based on drug or protein features (see **Figure 4.4A**). Given the way the tree is built, each leaf subregion is also expected to contain either significantly many or significantly few drug-protein interactions.

Figure 4.17 shows the resulting partitioned adjacency matrix, where again the grey level of each submatrix represents the percentage of interactions it contains. To obtain this partition, we arbitrarily stopped splitting a node of the tree if the number of pairs within it went below 1% of the total number of pairs (i.e., $n_{min} = 28935$). As a result, the matrix is divided into 691 rectangular subregions. Note that in the case of the global approach, it is not possible anymore to represent the resulting partitioning of the adjacency matrix as a checkerboard, where each row corresponds to a unique proteins and each column to a unique drug. Indeed, each new split cuts the corresponding subregion of the adjacency matrix either horizontally (when it is based on a drug feature) or vertically (when it is based on a protein feature) but there is no guarantee that the left and right successors of a split will be cut in the same way. However, given the hierarchical nature of the partitioning, the rectangular subregions defined by all tree leaves can be arranged so that they exactly cover the adjacency matrix. This is the representation used in **Figure 4.17**, which is obtained by recursively reordering the rows and columns of the subregions according to the tree splits.

As in the case of the local approach, we are mainly interested in the leaves corresponding to drugs and proteins that are significantly highly connected. We therefore again computed for each submatrix an interaction p -value using the same permutation scheme as for the local approach. Subregions with a p -value smaller than 10^{-7} are highlighted in green in **Figure 4.17**. We only found four regions that meet this criterion, which is much fewer than the 47 regions found with the

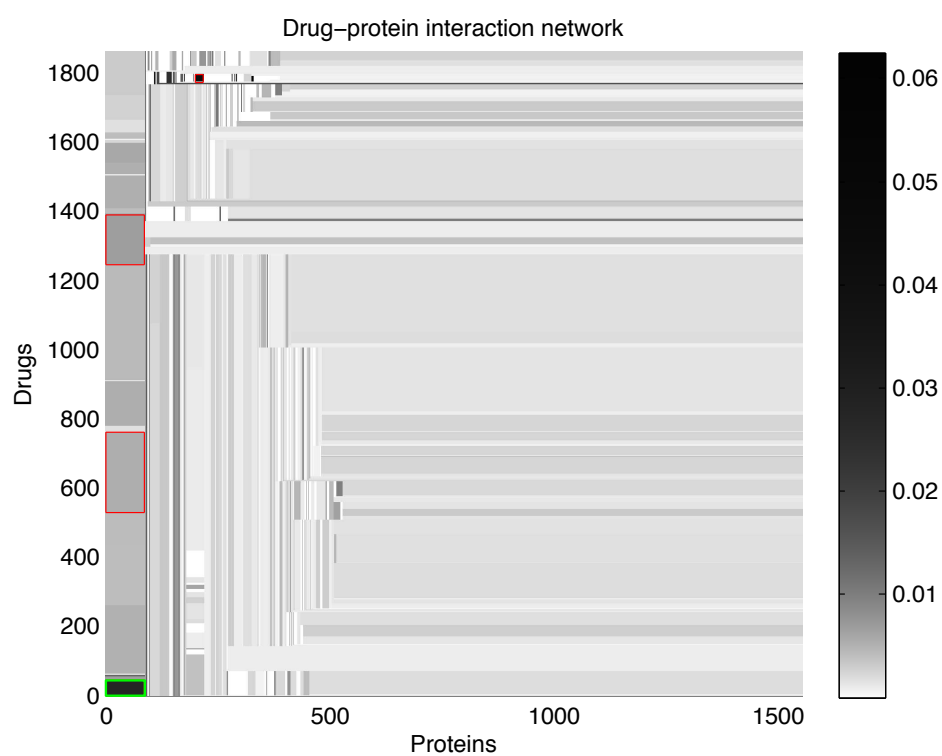


Figure 4.17: The adjacency matrix of the drug-protein interaction network is partitioned into clusters, due to one global single tree. Rows represent drugs, while columns represent proteins. The pairs of each clusters are expected to be either very connected, or not. The four submatrices with p -values lower than 10^{-7} are red surrounded, and have a proportion of interaction equal to 0.9%.

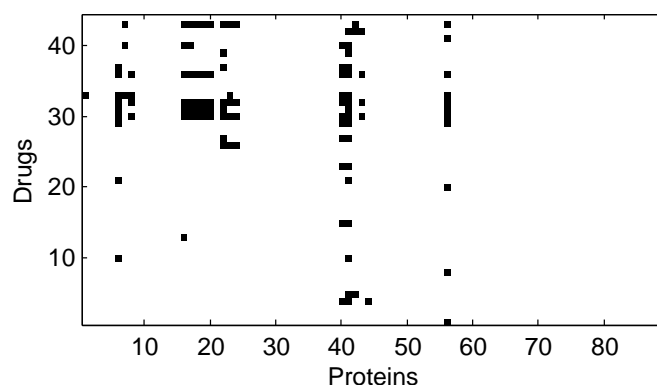


Figure 4.18: One of the four submatrices that got a p -value lower than 10^{-7} , and which is characterized in the text.

local approach. We also observed a very important difference in terms of the number of subregions defined by both methods, despite that fact that their n_{min} values were matched: 691 subregions are found with the global approach and 24700 subregions with the local one. These differences can be explained by the more flexible nature of the global approach that can define unconstrained partitions, while the local approach can only define checkerboard-structured partitions.

These four regions of small p -value contain in total 0.9% of all interactions. One of these regions appears to be found also in the clustering obtained with the local approach, and is characterized here below. It is located in bottom left corner of **Figure 4.17**. A detailed view of this submatrix can be found in **Figure 4.18**. It counts 109 interactions between 44 drugs and 89 proteins, which gives a proportion of interactions 5 times greater than the proportion in the global network. Analyzing the tree path that leads to the corresponding leaf, this subregion is defined by all drugs that contain SC1C(N)CCCC1 as a chemical substructure and all proteins, from the rhodopsin family, that has the 7 transmembrane receptor PFAM domain.

4.5.2 Clustering with ensembles of trees

In Section 4.5.1, we used single trees to partition the adjacency matrix. Yet we know that single trees suffer from high variance and may not be the ideal method to obtain interpretable results. In this section, we propose a way to obtain from an ensemble of trees a clustering of the drugs and proteins as well as a characterization of the corresponding subnetworks in terms of the input features. The procedure is explained and illustrated below using the drug-protein interaction network.

Partitioning the matrix

To obtain a checkerboard partitioning of the adjacency matrix from an ensemble of trees, we propose to proceed as follows. Two ensembles of (100) multi-output trees are grown respectively for the drugs and the proteins using the local approach. From the ensemble relative to the drugs (resp. proteins), one can derive a proximity measure between two drugs (resp. proteins) by counting the number of times the two drugs (resp. proteins) fall in the same leaf over the 100 trees in the ensemble and by normalizing this count by the total number of trees in the ensemble (Breiman, 2001). From this proximity, a distance between drugs/proteins can be computed as one minus the

proximity. Using this distance measure, one can build two distance matrices, one 1862×1862 matrix for the drugs and one 1554×1554 matrix for the proteins and these matrices can be used as inputs of the k -medoids algorithm to obtain a clustering respectively of drugs and proteins. Just as in the local approach with single multi-output trees, these two clusterings then define a checkerboard partitioning of the adjacency matrix. Because the ensemble proximity is such that two drugs (resp. proteins) are close if they have similar connectivity patterns with all proteins (resp. drugs), each subregion of this partitioning should contain either strongly or weakly connected pairs

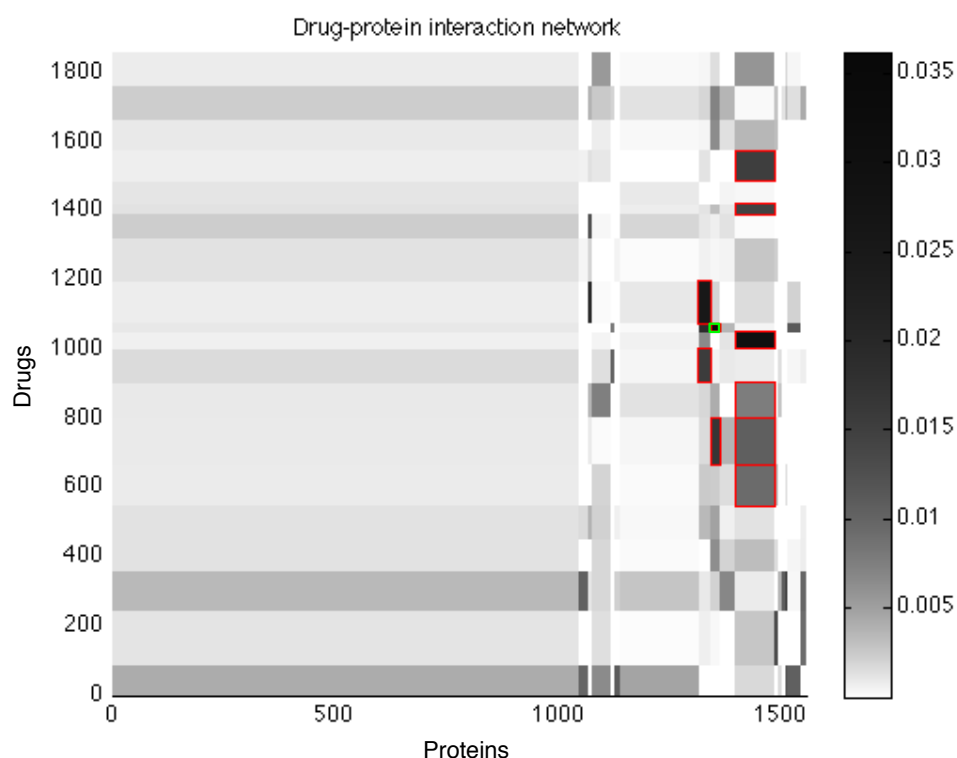


Figure 4.19: The adjacency matrix of the drug-protein interaction network is partitioned into clusters, using a proximity measure derived from ensembles of multi-output trees. The pairs in each clusters are expected to be either very connected or not. The ten regions highlighted in red are the ones with the smallest interaction p -values estimated by random permutations. Together, they contain a proportion of interactions equal to 1.4%

Figure 4.19 shows the checkerboard structure obtained with this procedure using the k -medoids algorithm to find 20 clusters for the drugs and 20 clusters for the proteins. Trees were grown using the same n_{min} thresholds as in the experiment of **Figure 4.15**. The clusters of drugs are almost all of the same size, while clusters of proteins are much more unbalanced in sizes, with some very small (as observed in **Figure 4.15** with single trees) and some very large clusters.

The grey level of each submatrix represents the ratio of interactions within it. To identify significantly connected submatrices, we computed an interaction p -value for each of them as we did in Section 4.5.1. The 10 regions corresponding to p -values smaller than 10^{-7} are highlighted in red

in **Figure 4.19**. In total, they contain a proportion of 1.4% interacting pairs, which is more than 8 times higher than the proportion in the global network.

Figure 4.20 shows a graphical representation of the whole network. The drugs correspond to the black nodes and the proteins to the grey nodes. Interactions that are not included in one of the ten submatrices with lower p -values, are represented by light grey edges. The interactions from the ten submatrices are highlighted with different colors. We can see that, mostly, the edges from a same cluster of interactions are located in a same region of the graph, showing that the drugs and the proteins that make it up are indeed highly connected.

Note that the proximity measure proposed above can be shown to be a dot-product into some euclidean space (Breiman, 2001). Indeed, let us encode the i th drug/protein with a vector of the following form:

$$\mathbf{v}_i = [[l'_{1,1}, l'_{1,2}, \dots, l'_{1,n_1}], [l'_{2,1}, l'_{2,2}, \dots, l'_{2,n_2}], \dots, [l'_{T,1}, l'_{T,2}, \dots, l'_{T,n_T}]]^T \quad (4.1)$$

where T is the number of trees in the ensemble, n_k is the number of leaves in the k th trees, and $l'_{k,l}$ is equal to $1/\sqrt{T}$ if drug/protein i falls into leaf k of the l th tree, 0 otherwise. Then, the proximity between two drugs/proteins is equal to the dot-product $\mathbf{v}_i^T \mathbf{v}_j$ between their corresponding vectors \mathbf{v}_i and \mathbf{v}_j and their distance is the euclidean squared distance between these vectors $\|\mathbf{v}_i - \mathbf{v}_j\|^2$. An alternative to the application of the k -medoids algorithm used above is thus to apply k -means on this vectorial representation. Finally, note that Geurts *et al.* (2006a) have proposed another proximity measure that uses the same encoding as in (4.1) but with $l'_{k,l}$ taken as $1/\sqrt{TN_{k,l}}$ when drug/protein i falls into the k th leaf of the l th tree, where $N_{k,l}$ is the total number of learning sample examples falling into the same leaf. This proximity measure gives more weights to leaves that contain fewer examples. Both measures coincide in the case of fully grown trees with only one example per leaf.

Characterizing the regions

The clustering proposed above is based on the ensemble of trees and thus the fact that a drug/protein belongs to a specific cluster is a direct consequence of its input feature values. For interpretability reason, it is thus interesting to try to characterize each cluster from the point of the view of the input features. Unlike with single trees (see Section 4.5.1), this is not as straightforward as looking at the features that are used along some tree path, because all drugs or proteins in a given cluster do not end automatically in the same leaf in a given tree and also because we now have an ensemble of different trees and not a single tree. Feature importance scores as described in Section 2.2.2 only give a global measure of the importance of a feature, while we are looking here for a more local cluster-specific characterization.

To answer this question, we propose in this section a way to derive local feature importance scores relative to a subset of objects (drugs or proteins). Given an ensemble of trees and a subset of objects S , local feature importance scores are computed as follows:

- For a given object o in S and a tree \mathcal{T} , we compute the importance of a feature X_i , denoted $I(X_i, o, \mathcal{T})$, as the sum of weighted impurity reductions (as computed in Equation 2.7) over all nodes in the path traversed by o in \mathcal{T} where X_i is used as the splitting feature. Impurity reductions are computed on the basis of the training sample from which the ensemble was grown.

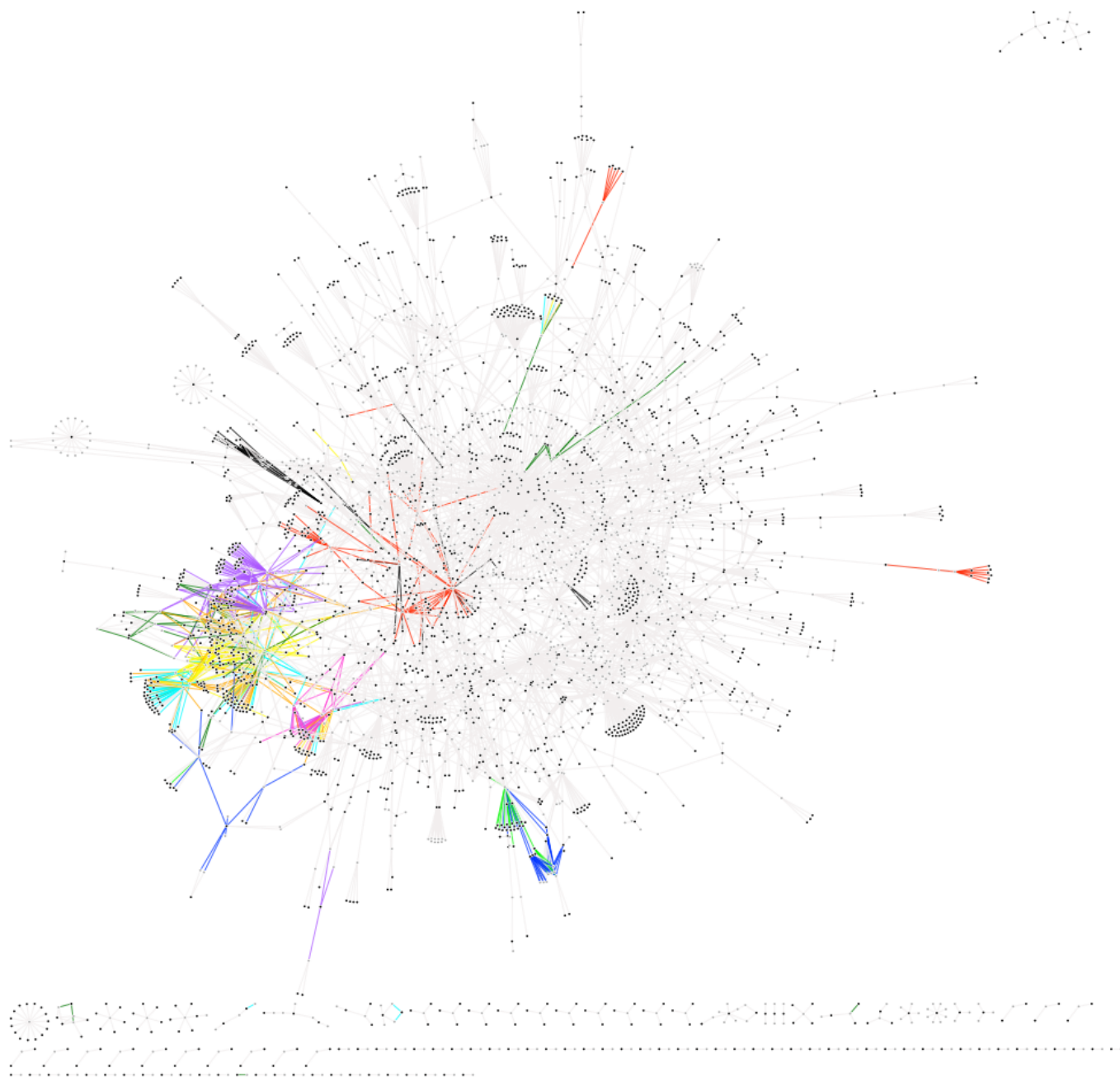


Figure 4.20: A graphical representation of the drug-protein interaction network. The ten subnetworks with the lowest p -values are highlighted by colored edges.

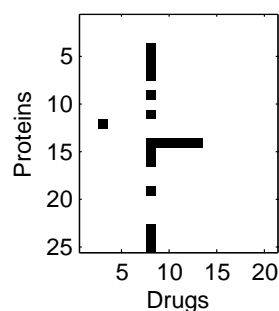


Figure 4.21: The submatrix having the highest proportion of interactions, among those with a p -value lower than 10^{-7} in **Figure 4.19**. This submatrix is characterized in the text.

- These importances are then averaged over all objects in S and all trees in the ensemble to obtain the importance of feature X_i relative to the subset S :

$$I(X_i, S) = \frac{1}{T} \sum_{t=1}^T \frac{1}{|S|} \sum_{o \in S} I(X_i, o, \mathcal{T}_t) \quad (4.2)$$

Intuitively, a feature will be important according to this modified local measure if it is used in a lot of paths traversed by objects from the subset S and if, in these paths, it leads to important weighted impurity reductions. Note that when S is taken as the whole learning sample, this measure does not coincide with the global feature importance score defined in Section 2.2.2. This property could have been obtained by summing unweighted instead of weighted impurity reductions in the first step above but we believe that this would have given too much importance to splits at deep nodes in the tree in the case of small subsets of objects S .

Let us illustrate this cluster-specific feature importance measure on the drug-protein interaction network. We focus on the submatrix with the higher proportion of interactions, among the ten submatrices with a p -value lower than 10^{-7} . This submatrix is highlighted in green in **Figure 4.19** and its edges are in light green in **Figure 4.20**. A detailed view of this submatrix can be found in **Figure 4.21**. It contains 19 interactions between 25 drugs and 21 proteins, corresponding to 3.6% of interactions. Feature importance scores for the drug and protein features were derived respectively for the subsets of 25 drugs and 21 proteins using the procedure above. Given the multi-output setting, we use the average variance over all outputs as our impurity measure (see Equation 2.6).

392 chemical substructures (drug features) obtained an importance higher than zero for the cluster of 25 drugs. We list here only the ten most important ones (in decreasing order of importance):

1. SC1C(N)CCCC1
2. ≥ 1 saturated or aromatic nitrogen-containing ring size 8
3. C(\sim F)(\sim F)
4. N-S
5. S(=O)(=O)
6. O=C-C-C-C-C(=O)-C
7. C(\sim C)(\sim C)(\sim C)(\sim C)

8. Sc1c(N)cccc1
9. C(~Br)(~H)
10. CC1CC(O)CC1

For each substructure in this list, we computed its occurrence frequency in the cluster and in the whole set of drugs. Only one substructure (**C(~C)(~C)(~C)(~C)**, in bold in the list) appears more frequently in the 25 drugs than in the whole set of drugs.

Concerning protein features, 158 PFAM domains have an importance higher than zero for the cluster of 21 proteins. We list here only the ten most important ones (in decreasing order of importance):

1. Eukaryotic-type carbonic anhydrase
2. **Neurotransmitter-gated ion-channel transmembrane region**
3. 7 transmembrane receptor (rhodopsin family)
4. Animal haem peroxidase
5. **Neurotransmitter-gated ion-channel ligand binding domain**
6. Oestrogen receptor
7. Serum albumin family
8. Serotonin (5-HT) neurotransmitter transporter, N-terminus
9. TspO/MBR family
10. Aminotransferase class I and II

The two bold PFAM domains are present in each of the 21 proteins of the cluster, while the 7 other domains are absent in all of them. These two lists of features, and in particular the ones in bold, are those that characterize the most this highly connected region.

Network of features

In (Yamanishi *et al.*, 2011), the authors exploited canonical correlation analysis to find associations between protein domains and chemical substructures that are predictive of drug-protein interactions. Each canonical component highlights several such associations and associations corresponding to all canonical components are compiled into a global (bipartite) network. This network contains 30,668 links between chemical substructures and protein domains that represent 5,3% of all possible links that can exist between the 660 chemical substructures and the 876 protein domains. All these links are expected to govern drug-protein interactions.

Similarly, each submatrix found by our method also associates chemical substructures and protein domains, those that appear in the feature importance ranking obtained for both drug and protein clusters. A network similar to Yamanishi *et al.* (2011)'s network can thus be constructed from tree ensembles by linking chemical substructures and protein domains found in the feature importance lists associated to all submatrices containing a significantly high number of interactions. As noted previously, the chemical structures or protein domains with non zero importance for a given cluster are not always present in the drugs and proteins within the cluster. We thus filter the links so as to only keep the ones that involve features that are more present in the drug or proteins of the cluster than in the whole set of drugs or proteins.

We applied this idea from the partitioning found in **Figure 4.19**. The network built from the ten clusters with the smallest p -values ($< 10^{-7}$) contains 3086 interactions. It corresponds to 0.5% of all possible links between the 660 chemical substructures and the 876 protein domains. From

these 3086 interactions, 992 (32%) can also be found in Yamanishi *et al.* (2011)'s network (see **Figure 4.22**), while 2094 are new. The intersection with Yamanishi *et al.* (2011)'s network is very significant, as assessed with a hypergeometric test ($p\text{-value} < 10^{-100}$). The network can obviously be enlarged by considering clusters with p -values higher than 10^{-7} .

As in (Yamanishi *et al.*, 2011), a weight can be associated to an edge in this network by computing the product of the importance value associated to the chemical substructure and the protein domain that it connects. In **Figure 4.22**, we present the network obtained when keeping only the 17 edges with a weight higher than the arbitrary chosen threshold $6 \cdot 10^{-6}$. From these 17 interactions, 6 can also be found in Yamanishi *et al.* (2011)'s network (intersection $p\text{-value} = 1.43 \cdot 10^{-5}$).

4.5.3 Pair-based feature ranking

A main goal of network inference methods is to rank new pairs from the most likely to the least likely to interact. When using single trees, one gets directly an explanation of the predictions in terms of the features that are tested along the path followed in the tree, either by the pair (in the case of the global approach) or by each of its nodes (in the case of the local approach). With an ensemble of trees, this feature is lost but it is nevertheless possible to obtain a ranking of the features according to their importance for making a specific pair prediction. To obtain such ranking, one can indeed use the local importance measure defined in Equation 4.2 using as the subset S a singleton containing one node from the pair (in the case of the local approach) or one pair (in the case of the global approach).

Let us illustrate this possibility on the drug-protein interaction network. We ranked the drug-protein pairs with a local approach using single-output tree ensembles. To get a prediction for all pairs of the network, we performed a 5-fold cross-validation on pairs. 8 pairs were predicted with the highest probability to interact (i.e. 1), and they are listed here above:

Drug	Protein	Interaction
5,6,7,8-Tetrahydrobiopterin	NOS3	1
6-Mercaptopurine	IMDH1	1
6-Mercaptopurine	PUR1	1
Epothilone B	TBA2	1
Epothilone B	TBA6	1
Hydrochlorothiazide	CAH4	1
Thioguanine	IMDH2	0
Tretinoin	RXRG	1

Only one pair is actually not interacting, or has not been discovered as interacting yet.

We focus on the interaction between 5,6,7,8-Tetrahydrobiopterin and NOS3 (**Figure 4.23**). This interaction allows the production of nitric oxide (NO), which is important in regulation of blood pressure and blood flow. For this particular pair, we ranked both chemical substructures and protein domains according to their local importances for predicting this particular pair. To compute these importance scores, we used the two ensembles (one for 5,6,7,8-Tetrahydrobiopterin and one for NOS3) grown from the learning fold (among the five) that did not contain this particular pair.

The 10 more important chemical substructures are the following:

1. **C(~C)(~C)(~H)(~N)**

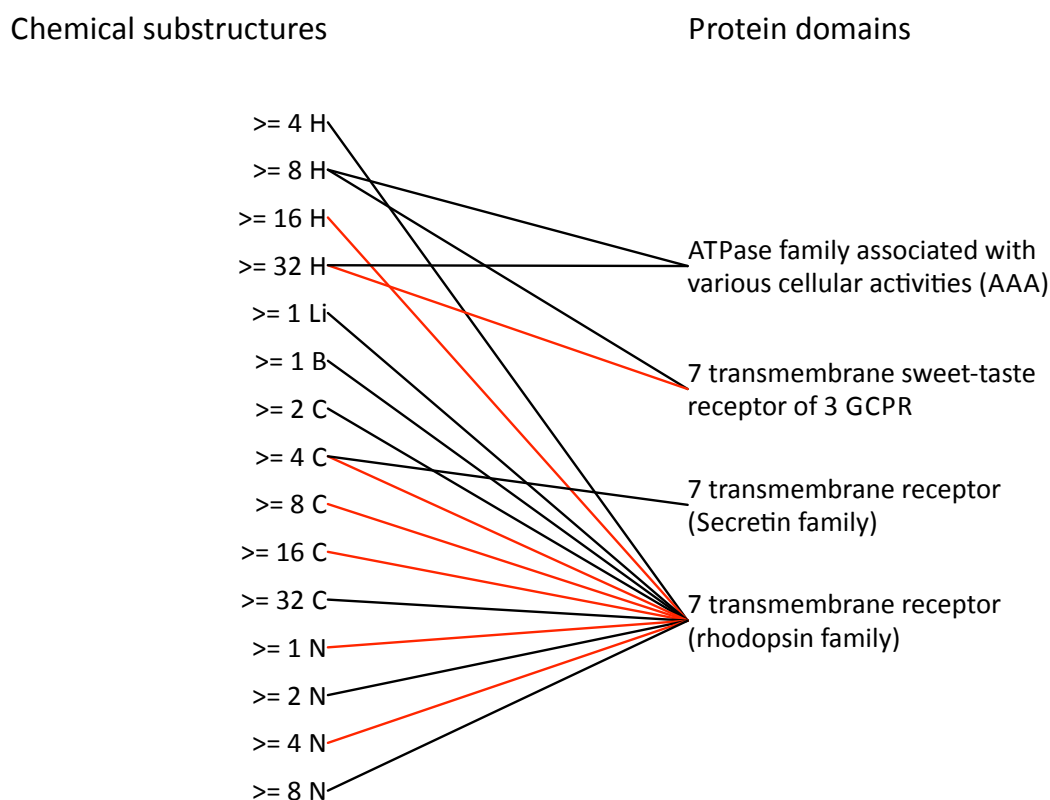


Figure 4.22: Each cluster is associated with a list of drug features and a list of protein features. Only are kept the features more present in the drugs and proteins of the cluster than in other clusters. The result can be represented as a network that connects chemical substructures and protein domains. The network in this figure was constructed from the 10 clusters in **Figure 4.19** that have a p -value lower than 10^{-7} . The weight of an edge is obtained by computing the product of the importance value associated to the chemical substructure and the protein domain that it connects. In the illustrated network, only are kept the 17 edges with a weight higher than the arbitrary chosen threshold $6 \cdot 10^{-6}$. The 6 red lines represent edges shared by our network and a similar network from (Yamanishi *et al.*, 2011), corresponding to more the 35% of the 17 edges of the network.

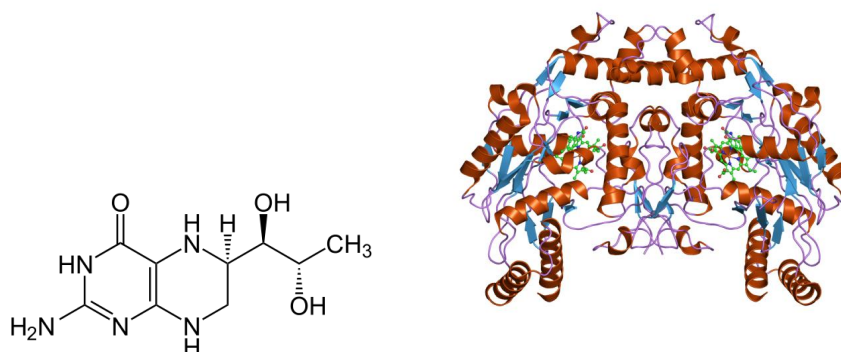


Figure 4.23: The 5,6,7,8-Tetrahydrobiopterin drug (left) and the NOS3 (endothelial nitric oxide synthase) protein (right) have been proved to interact.

2. **O=C-C-N**
3. O=C-C-C-C-C-N
4. [#1]-C-C-N-[#1]
5. C(-C)(-C)(=N)
6. **N-C-C-N-C**
7. N=C-C-C
8. **>= 2 any ring size 6**
9. C(~Br)(:N)
10. C(-C)(=N)

Substructures in bold are the ones present in 5,6,7,8-Tetrahydrobiopterin. The seven PFAM domains for which the importance is higher than zero are the following:

1. PDZ domain (also known as DHR or GLGF)
2. **Nitric oxide synthase, oxygenase domain**
3. ACT domain
4. **FAD binding domain**
5. **Oxidoreductase NAD-binding domain**
6. **Flavodoxin**
7. Biopterin-dependent aromatic amino acid hydroxylase

The domains in bold are the ones present in NOS3. These different features are the ones that led the algorithm to give a high probability to the target pair to interact.

4.6 Discussion

We explored tree-based ensemble methods for biological network inference, both with the local approach, which trains a separate model for each network node (single output) or each node family (multi-output), and with the global approach, which trains a single model over pairs of nodes. We carried out experiments on ten biological networks and compared our results with those from the literature. These experiments show that the resulting methods are competitive with the state of the

art in terms of predictive performance. Other intrinsic advantages of tree-based approaches include their interpretability, through single tree structure and ensemble-derived feature importance scores, as well as their almost parameter free nature and their reasonable computational complexity and storage requirement.

While the local and global approaches are close in terms of accuracy, the most appealing approach in our experiments turns out to be the local multi-output method, which provides less complex models and requires less memory at training time. All approaches remain however interesting because of their complementarity in terms of interpretability. A potential advantage of the global approach that was not explored in this chapter is the possibility to define features on pairs of nodes that might make a difference in some applications (Lin *et al.*, 2004; Qi *et al.*, 2005; Tabei *et al.*, 2012). With the introduction of such features, one would lose however the possibility with tree-based methods of not generating explicitly all pairs when training the model.

As two side contributions, we extended the local approach for the prediction of edges between two unseen nodes and proposed the use of multiple output models in this context. The two-step procedure used to obtain this kind of predictions provides similar results as the global approach, although it trains the second model on the first model's predictions. It would be interesting to investigate other prediction schemes and evaluate this approach in combination with other supervised learning methods such as SVMs. The main benefits of using multiple output models is to reduce model sizes and potentially computing times, as well as to reduce variance, and therefore improving accuracy, by exploiting potential correlations between the outputs. It would be interesting to apply other multi-output or multi-label supervised learning methods (Tsoumakos and Katakis, 2007) within the local approach.

We focused on the evaluation and comparison of our methods on various biological networks. To the best of our knowledge, no other study has considered simultaneously as many of these networks. Our protocol defines an experimental testbed to evaluate new supervised network inference methods. Given our methodological focus, we have not tried to obtain the best possible predictions on each and every one of these networks. Obviously, better performances could be obtained in each case by using up-to-date training networks, by incorporating other feature sets, and by (cautiously) tuning the main parameters of tree-based ensemble methods. Such adaptation and tuning would not change however our main conclusions about relative comparisons between methods.

Our experiments, like others (Tabei *et al.*, 2012), show that the different families of predictions that are defined by the two protocols are not equally well predicted, which justifies their separate assessment, as we have already explained in Section 3.3. These discrepancies in terms of prediction quality should be taken into account when one wants to merge the different families of pairs into a single ranked list of novel candidate interactions from the more to the less confident as predicted by our models. This problem was discussed and solved in Section 3.6. A limitation of our protocol is that it assumes the presence of known positive and negative interactions. Most often in biological networks, only positive interactions are recorded, while all unlabeled interactions are not necessarily true negatives (a notable exception in our experiments is the EMAP dataset). In this chapter, we considered that all unlabeled examples are negative examples. It was discussed in Section 3.4 that this approach is reasonable. It would be interesting nevertheless to design tree-based ensemble methods that explicitly takes into account the absence of true negative examples (e.g., Denis *et al.* (2005a)).

To illustrate the interpretability of trees and ensemble of trees that were explained in this chapter, we performed several illustrative experiments on Yamanishi *et al.* (2011)'s drug-protein interaction

network. We have shown that, with single trees, it was possible to obtain, both from the local and the global approaches, a partitioning of the adjacency matrix into different regions that are expected to be either especially connected or not. These regions can furthermore be characterized by the input features that are tested along the path towards the corresponding leaves. We showed that a similar partitioning, together with a feature-based explanation, can also be obtained from an ensemble of trees by exploiting a tree-based proximity measure and by adapting feature importance scores. Using these tools, a biologist can thus obtain an interpretable explanation about why a particular drug or group of drugs tend to interact with a particular protein or group of proteins. This explanation should help them understanding more deeply the nature of the interactions.

These experimentations have however also highlighted several limitations of the approach that should be addressed as future works. First, while tree-based methods are almost parameter-free, it is not possible to avoid setting several thresholds when it comes to extract interpretable information using these methods (e.g., stop splitting parameters, and threshold on p -values or importance scores). This makes the approach only semi-automatic. Tree-based methods make the hypothesis that it is possible to partition the adjacency matrix into rectangular regions where either all pairs are connected or no pairs are connected. In practice however, as a consequence of a departure with respect to this hypothesis and also of the very sparse nature of the interactions, the partitioning that is obtain contains mostly regions with unconnected pairs and that are very unbalanced in shape and sizes. This makes the interpretation of our results difficult. Finally, each region is characterized by a list of features but it is not always clear which features in these lists are really decisive to define the region and explain the interactions it contains. Indeed, some of these features are mainly used to isolate other regions during the hierarchical tree partitioning and therefore they are not directly characteristic of the region of interest. Our approach when generating for example the network in **Figure 4.22** was to focus on properties that are more significantly present in drugs/proteins in the cluster than in a random set but the lack of some property might also be crucial to explain interactions.

Chapter 5

Predicting genetic interactions in yeast

In this chapter, we try to infer at best the whole genetic interaction network in yeast, with tree-ensemble methods. Genetic interactions correspond to the modification of the action of one gene by the expression of another gene. Several experimental techniques exist to measure them, and we found eleven different subnetworks in the literature, that we use as gold standard to predict the unknown interactions. Additionally, we describe the 22 features sets, related to the genes, that are used as input in our algorithms. We present the different cross-validation techniques that we carry out to assess our methods, and compare the predictive performance to a new baseline based on the bias that occurs between the different subnetworks. We finally predict new genetic interactions, and obtain a global ranking. This ranking, as well as experimentally measured interactions, are then compare to GO terms. Through this comparison, we show that the latter can be cleaned by cross-validation.

Contents

5.1	Introduction	128
5.2	Background	129
5.3	Datasets	136
5.4	Cross-validation experiments	144
5.5	Predictions of unlabeled pairs	163
5.6	Conclusion	170

5.1 Introduction

An important class of biological networks are genetic interaction networks (Mani *et al.*, 2007). There is an interaction between two genes when the effect of a mutation on one gene is modified by a mutation on the other gene. If the effect is modified positively the interaction is called "positive" and if the effect is modified negatively the interaction is called "negative". The knowledge of these interactions is very important to understand the function of the genes and their products, and globally to elucidate functional and organizational principles of biological systems. In addition, a genetic network may provide fundamental insights into the genetic architecture underlying the genotype-phenotype relationship that governs genetic diseases (Costanzo *et al.*, 2013). Experimental techniques exist that allow to quantitatively measure these interactions. Nevertheless, these techniques are laborious and costly, and only about 10% (divided into eleven different subnetworks) of all 18 millions possible pairs between the ~ 6000 genes of the yeast *S.cerevisiae* have been experimentally verified so far. The use of *in silico* network inference techniques is thus very interesting to complete these experimentally confirmed interactions (Wong *et al.*, 2004; Chipman and Singh, 2009).

Building on the experience gained in the previous chapters, our goal here is first to see how well it is possible to complete the genetic interaction network of the yeast *S. cerevisiae* using supervised learning methods, and second to find the best way to apply supervised inference methods to actually infer this network. To achieve this goal, we need first some partial knowledge of the network in the form of a set of experimentally labeled pairs of genes and second, information about genes to use as input features for the inference. *S.cerevisiae* has been extensively studied by biologists and is therefore one of the best-understood eukaryotes at the molecular genetic level. In consequence, plethora of information are available in public datasets to describe yeast genes. Also, as mentioned earlier, about 10% of the possible gene pairs in yeast have been experimentally tested in the context of various studies and are available for training supervised learning models. The yeast genetic interaction network thus constitutes an excellent testbed for the application of supervised network inference methods.

Several papers in the literature (e.g., Wong *et al.*, 2004; Paladugu *et al.*, 2008; Chipman and Singh, 2009; Li and Luo, 2009; Pandey *et al.*, 2010; Li *et al.*, 2011; Zhang *et al.*, 2012; Linden *et al.*, 2011; Alanis-Lobato *et al.*, 2013) have already addressed the problem of the prediction of genetic interactions in yeast. Most of these studies are however limited to synthetic sick and lethal (SSL) interactions, which represent only one type of genetic interactions (the negative ones). Moreover SSL are not quantitative but only indicate the presence or absence of an interaction, without specifying the effect level. Only a few papers have tackled the full problem of predicting both positive and negative interactions in a quantitative way. Among these, Ulitsky *et al.* (2009) and Ryan *et al.* (2010) use supervised learning methods to complete missing values within known subnetworks (i.e., only $LS \times LS$ predictions). Eronen *et al.* (2010) and Linden *et al.* (2011) estimate genetic interactions in a larger scale exploiting matrix approximation techniques. They also focus only on $LS \times LS$ predictions and build their predictions only from the quantitative genetic interaction matrix itself, without incorporating any input features on genes. To our knowledge, no study has been carried out to infer the complete yeast genetic interaction network by combining the full set of all known interactions and a large set of yeast gene features.

This chapter is structured as follows. In Section 5.2, we describe the *S.cerevisiae* specie and its main characteristics. We also define genetic interactions and review the main experimental techniques to measure them. In Section 5.3, we detail the main datasets used in this study. To constitute

our training interaction network, we collected about 4 millions pairs that were experimentally tested in the context of 11 independent studies. As inputs for the genes, we used a total of about 11000 features obtained from 22 different sets of measurements relative to yeast genes. In Section 5.4, we present the results we obtained on these datasets with different cross-validation schemes. We first study separately the performance on the 11 different subnetworks elucidated in the 11 studies and then consider the full network of 4 millions pairs. Finally, Section 5.5 discusses the prediction of the full genetic interaction network, using the best parameter setting identified in the previous section. Following previous works, the quality of these predictions is assessed by checking the enrichment of predicted positive and negative interactions in pairs of genes that participate to a same biological function (as labeled in the Gene Ontology).

5.2 Background

In this section, we first describe the specie that will be studied in this chapter. After that, we define more precisely (positive and negative) genetic interactions and review the different techniques to measure them experimentally.

5.2.1 Yeast *Saccharomyces cerevisiae*

Saccharomyces cerevisiae is a specie of yeast, also called brewer's yeast or baker's yeast. It is a eukaryotic organism which belongs to the Fungi kingdom. It is a single-cell organism, round to ovoid, from 6 to 12 μm long and 6 to 8 μm long. It can survive and grow under two forms: one haploid form (16 chromosomes) and one diploid form (32 chromosomes). The genome of the yeast is composed of 6000 genes and was the first eukaryotic genome to be completely sequenced, in 1996 (Goffeau *et al.*, 1996).

S.cerevisiae is considered as a model organism (Botstein *et al.*, 1997), for different reasons. It can grow on defined media, and its environment can be completely controlled. It grows rapidly, with a 80 minutes generation time. Efficient techniques have been developed that permit any gene to be replaced with a mutant allele or to be deleted from the genome. The yeast shares a common life cycle and cellular architecture with higher eukaryotes, like human and plants (Mell and Burgess, 2002).

Today, the Saccharomyces Genome Database (SGD) (<http://www.yeastgenome.org>) provides information about every yeast gene. Since the yeast genome has been sequenced, the understanding of biological function of genes have considerably increased, and annotation of these functions can frequently be transferable to other species. (Botstein and Fink, 2011)

5.2.2 Genetic interactions

The phenotype of a organism does not only depend on each gene individually. Effects of genes are not independent to each others and dependance can be highlighted by genetic interaction networks. A genetic interaction between two genes A and B is defined by the deviation between:

- the phenotype of an organism where a combination of the two genes were deleted (a double-mutant organism's phenotype)
- and the phenotype that would be expected from the two single-mutant organism's phenotypes (the expected neutral phenotype).

To complete the definition, we need a quantitative phenotypic measure, and a neutrality function (Mani *et al.*, 2007). In the case of yeast genetic interactions, the phenotypic measure is typically the fitness, which corresponds to the growth rate of the colony size relative to the growth rate of the wild type (Schuldiner *et al.*, 2005; Collins *et al.*, 2007; Fiedler *et al.*, 2009; Costanzo *et al.*, 2010). One can then measure an interaction between two genes by comparing the observed colony size of the double mutant to the expected value. If the double mutants grow more slowly or rapidly than expected, the two genes are said to interact with each other.

The expected fitness value of a double-mutant must be computed from the fitness values of the two corresponding single-mutants. Different neutrality functions have been used in literature to compute it (Mani *et al.*, 2007):

- The multiplicative function predicts that double-mutant fitness is the product of the two single-mutant fitness. For a pair of genes (A, B) the expected fitness is:

$$W_{AB} = W_A \cdot W_B,$$

where W_A and W_B are the fitness values of the single-mutants where the genes A and B are deleted, respectively.

- The additive function predicts that the double-mutant fitness is equal to

$$W_{AB} = W_A + W_B - 1.$$

- The log function predicts that the double-mutant fitness is equal to

$$W_{AB} = \log_2 \left((2^{W_A} - 1)(2^{W_B} - 1) + 1 \right)$$

- The minimum function predicts that the double-mutant fitness is equal to the fitness of the less-fit single-mutant:

$$W_{AB} = \min(W_A, W_B)$$

In the case of yeast fitness, the expected double-mutant phenotype is typically modeled with the multiplicative function (Costanzo *et al.*, 2013), and we will use this neutral function in the different examples that follow in this chapter. It has been suggested that this function is the most accurate at identifying functional relationships (Eronen *et al.*, 2010).

Negative interactions

As we already said, a genetic interactions is measured by computing the difference between the measured fitness of the double-mutant and the expected fitness. If the double-mutant grows more slowly than expected, the difference is a negative number and the interaction is called *negative interaction*. In the extreme case where the combination of the two mutations leads to an inviable organism, the interaction is called *synthetic lethal*, otherwise the interaction is called *synthetic sick*. An illustration of such interactions is shown in **Figure 5.1**.

Negative interactions are often found for proteins that work in compensatory pathways, i.e. in parallel pathways that regulate the same function (Kelley and Ideker, 2005). Indeed, in this situation, deletion of either gene is expected to delete the function of one but not both pathway.

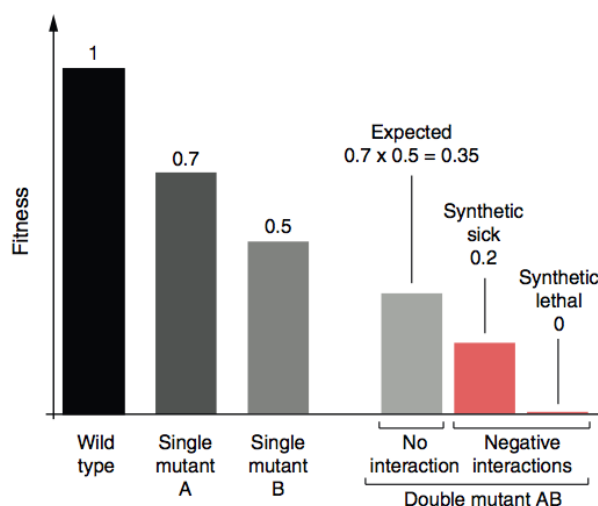


Figure 5.1: Negative genetic interactions happen when the combination of the gene mutations causes a stronger growth defect than expected. In the extreme case of synthetic lethal interactions, the combination of the mutations leads to the death of the organism. Figure taken from Costanzo *et al.* (2011).

Positives interactions

In the contrary case where the double-mutant grows more rapidly than expected, the difference between the measured fitness of the double mutant and the expected fitness is positive and the interaction is called *positive interaction*. They can be classified into different categories according to different biological interpretation (Dixon *et al.*, 2009) (**Figure 5.2**).

A *masking interaction* happens when the fitness of the double mutant is greater than expected, but lower than the fitness of the two single-mutant. It often happens between genes encoding proteins that belong to a same pathway (Fiedler *et al.*, 2009). Indeed, the deletion of a protein disrupts the function, and the deletion of the second protein does not cause a worse fitness.

A *suppression interaction* happens when two genes *A* and *B* interact in such a way that the double-mutant fitness is greater than the lowest single-mutant fitness. It often indicates that the protein encoded by *B* is the suppressor of the function of the protein encoded by *A*, i.e. it is a negative regulator of a pathway associated with *A*. The mutation of the negative regulator *B* leads to hyperactivation and accumulation of a toxic gene product. When the gene *A* is also deleted, it reduces the flux through the pathway, and suppress the toxic effects (Dixon *et al.*, 2009).

A *symmetric interaction* happens when the fitness of the two single mutants and the fitness of the double-mutant are all equal to each other. It is currently observed when the proteins encoded by the genes belong to a same complex.

The different types of negative and positive interactions are illustrated in **Figure 5.3**. These classes of interactions are useful to understand the organization of molecular pathways in an organism (Collins *et al.*, 2010).

Hypothesis about the organization of an organism can also be done by comparing *genetic interaction profiles*. The genetic interaction profile of a gene is a vector that contains the interactions between this gene and all the other genes of the network. If other genes have highly correlated

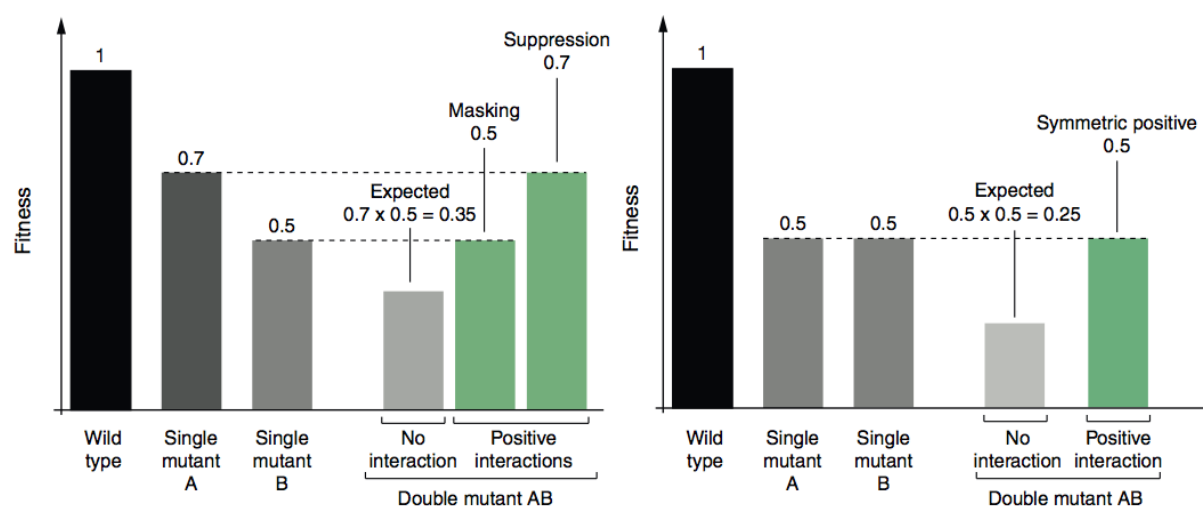


Figure 5.2: Positive interactions happen when the combination of the gene mutations causes a lighter growth defect than expected. They can be divided into three categories (masking, suppression or symmetric interactions), according to fitness of the double-mutant compared to the fitness of the two single-mutant. Figures taken from Costanzo *et al.* (2011).

profiles, it may signify that their encoded proteins act coherently in a biochemical pathway (Collins *et al.*, 2007).

5.2.3 Experimental techniques

In this section, we review the different experimental techniques that are found in the literature to measure genetic interactions in yeast *S.cerevisiae*. These different techniques differ in experimental designs, data pre-processing, and/or interaction scoring schemes (Linden *et al.*, 2011). Our goal in this section is not to be exhaustive or to go into all the details but only to give a brief overview of the techniques behind the genetic interaction datasets we will exploit in the next sections.

Synthetic genetic array (SGA) methodology

The SGA methodology (Tong *et al.*, 2001) is a high-throughput procedure for the systematic construction of double-mutants. In each screen, a query strain containing a mutation is crossed to an array of deletions mutants. The strains are mated and diploid strains (cells that contain two versions of each gene) are generated. Meiosis and sporulation (cellular division that produces haploid cells from diploid cells) give cells carrying zero, one, or two mutations. Several selection steps are then carried out to select haploid cells that carry two mutations. All double-mutant strains corresponding to a given query gene deletion are grown simultaneously on the same plate. Growth rates are monitored by analyzing the colony sizes.

Essential genes are those genes that are necessary for the survival of the organism. They cannot be deleted without leading to the death of the cell. This is problematic to measure interaction involving such genes. To address this issue, a technique called DAmP (decreased abundance by

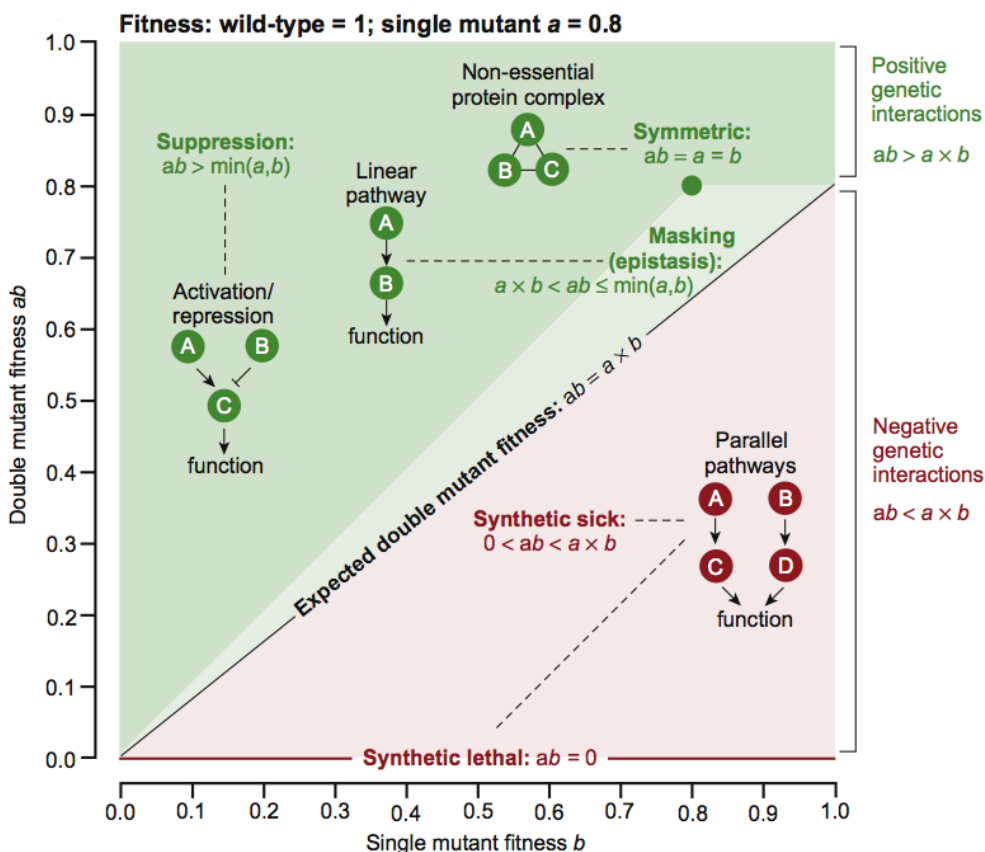


Figure 5.3: Positive and negative interactions happen when the measured double-mutant fitness differs from the expected one. In this example, $W_A = 0.8$, W_B varies from 0 to 1 (x -axis) and $W_{A,B}$ varies from 0 to 1 (y -axis). The dark green zone represents negative suppression interactions, the light green zone represents the positive masking interactions and the red zone represents the positive interactions. Figure taken from Costanzo *et al.* (2013).

mRNA perturbation) is typically used that can decrease the expression of a gene and maintain cell viability. (Schuldiner *et al.*, 2006)

This technique was first proposed to identify synthetic sick or lethal (SSL) interactions by simply monitoring the inviability of the double mutants (Tong *et al.*, 2001, 2004; Tong and Boone, 2006). Used as such, the SGA approach is limited as it does not allow to assess the interactions in a quantitative way (only a binary score is provided for each pair) and it is restricted only to the detection of negative interactions. The next two techniques can be seen as extensions of the original SGA approach to address these two limitations.

Epistatic miniarray profiles (E-MAPs)

The epistatic miniarray profile (E-MAP) approach is a variation of the SGA method that allows to detect both positive and negative interactions and that provides a quantitative measure of the level of interaction for each tested gene pair. Since its introduction in 2005, this approach have been used in several studies that have lead in total to the measurement of more than 1 million gene pairs (Schuldiner *et al.*, 2005; Collins *et al.*, 2007; Wilmes *et al.*, 2008; Fiedler *et al.*, 2009; Jonikas *et al.*, 2009; Aguilar *et al.*, 2010; Zheng *et al.*, 2010; Braberg *et al.*, 2013; Surma *et al.*, 2013).

E-MAPs use the SGA approach to generate systematic double-mutants for a large a priori defined set of genes. To estimate the growth phenotype, digital photographs of the arrays of mutant yeast colonies are taken after a defined period of time. The fitness of each double-mutant is then estimated by measuring the colony area. These raw area values are preprocessed to correct for some artifacts that can appear during the measurements (e.g., uneven image lighting).

To compute an interaction score, the fitness of the double-mutant needs to be compared to the expected fitness, which depends on the fitness of each single-mutant (see Section 5.2.2). To achieve this, the fitness values are normalized in two steps.

First, all fitness values in a given plate (which contain all the double-mutants relative to one query strain) are scaled according to the typical size of the colony on that plate, which is defined as the mean of the colony sizes on the plate ranked between the 40th and 60th percentiles. This scaling has two goals. First, it cancels the difference in growth condition that might exist from one plate to another. Second, and more importantly, it relates the double-mutant fitness to the fitness of the single-mutant corresponding to the query gene. Indeed, because most mutations in the array have little or no growth defect, the typical colony size can be interpreted as an estimation of the fitness of the single-mutant corresponding only to the deletion of the query gene.

The goal of the second normalization is to relate the (scaled) double-mutant fitness to the fitness of the other single-mutant, corresponding to the deletion of the test gene. This latter fitness is estimated by the median of the fitness of all double-mutants arising from the test gene (ie., the double-mutant in each plate that corresponds to this test gene). The measurements are repeated six times and the double-mutant and single-mutant fitnesses are compared to each other using a modified *t*-statistic. The resulting scores, called *S* scores, thus take into account both the strength and the confidence of the interaction (Collins *et al.*, 2006).

Note that each score is actually computed twice because each gene in a pair plays once the role of the query gene and once the role of the test gene. The final score of a pair is thus obtained as the average of the corresponding two scores.

Typically, it has been estimated that an interaction that gets a *S* score lower than -2.5 can be confidently considered as a negative interactions, and an interaction that get a *S* score higher than 2

can be confidently considered as a positive interaction. Note that when the genes are located on the same chromosome and have a low recombination frequency, the corresponding strain is considered incorrect and is therefore removed from the analysis. E-MAPs thus typically contain about 10% missing values (Collins *et al.*, 2010)

In all applications of the E-MAP approach, a set of genes is first selected and all pairs defined by these genes are systematically tested. These genes are not chosen at random but rather they are selected because they belong to a same biological process or pathway. The main motivation of this biased selection is to increase the signal-to-noise ratio by ensuring that the proportion of interactions among the tested gene pairs will not be too small. Indeed, it has been estimated that the chance to have an interaction between two randomly chosen genes is equal to 0.5%, while it can reach 5% when the genes are chosen according to their participation to a specific biological process (Collins *et al.*, 2007).

SGA scores

Another approach based on the SGA method has been adopted in (Baryshnikova *et al.*, 2010; Costanzo *et al.*, 2010) to estimate genetic interactions. Unlike the E-MAP, this approach has been applied in a genome-wide manner to test interactions between one large set of randomly selected query genes and all yeast genes, leading to an asymmetric and rectangular interaction score matrix.

Several systematic biases associated with genome-scale SGA methodology have been identified in (Baryshnikova *et al.*, 2010): one plate may have a different time of incubation, the growth medium may be uneven on the surface, a local competition for nutrient between neighbor mutant strains can happen, or errors from robotic instrument might lead to spurious batch effects. Normalization procedures have been developed to correct for all these biases.

After normalization, a *SGA score* can be computed from two single-mutant fitnesses (W_A and W_B) and one double-mutant fitness (W_{AB}) derived from normalized colony size measurements, assuming a multiplicative expected fitness:

$$\epsilon = W_{AB} - W_A \cdot W_B.$$

Like in the E-MAP approach, single-mutant fitness values are derived computationally from the normalized colony size. All fitness scores W are scaled according to the fitness of a wild type strain and so are always comprised between 0 and 1 (since mutations always decrease the fitness with respect to the wild type). Consequently SGA scores ϵ are always comprised between -1 and 1.

Each double-mutant is replicated four times and fitness values are averaged over these replicates. Unlike in the E-MAP approach, the SGA score does not take into account the variance of the estimated fitness over these replicates but they are instead used to associate a detection p -value to each interaction score. Gene pairs with a p -value higher than 0.05 are considered as missing values. Among measured pairs with a p -value lower than 0.05, one can then distinguish negative, positive, and neutral interactions using two cutoff values. (Costanzo *et al.*, 2010) consider as negative interactions gene pairs with a SGA score $\epsilon < -0.12$, as positive interactions gene pairs with $\epsilon > 0.16$, and as neutral all other pairs.

Genetic interaction mapping (GIM)

Unlike E-MAPs and SGA scores, the GIM (Decourty *et al.*, 2008) approach is not based on the SGA method. Here, the query strain is not crossed to each of the strains of an array individually,

but is mated in a single pool combining all gene deletions of the collection. The double-mutants are selected and grown in rich liquid medium. The tags that mark the deletions are amplified and labeled with dyes. Finally microarrays are used to measure the abundance of double-mutants (signal intensity Q). They are normalized and compared to microarrays obtained with the same procedure, but performed in parallel with a reference deletion, chosen because no significant effect on growth has been observed (signal intensity R). The normalized result is equal to $\log_2(Q/R)$. A pair of genes for which the value $\log_2(Q/R)$ is lower than -1 (in at least one of the performed screens relative to this gene) is considered as negative, included between -1 and 1 is considered as neutral, and higher than 1 is considered as positive.

Like SGA score and unlike the E-MAP approach, the GIM approach is applied with a set of query genes different from the genes in the deletion library, leading in this case to a rectangular matrix of interaction scores.

5.3 Datasets

To train a supervised learning model, we need labeled training data. In our case, training examples are pairs of genes that need to be labeled as positive, negative, or neutral interactions and to be described by input features. In this section, we present the different datasets from which we built our training set: in Section 5.3.1, the genetic interaction subnetworks and in Section 5.3.2, the different gene feature sets.

5.3.1 Training networks

To define our training genetic interaction network, we collected datasets originating from 11 different studies from the literature. These studies were published between 2005 and 2013. Altogether, the resulting training network contains quantitative interaction scores for almost 4 millions gene pairs. Yeast having about 6000 genes, they represent in total about 10% of all possible gene pairs. Note that each pair of genes for which an interaction score has been measured can be classified into one of three classes: positive, negative, and neutral. This is unlike most other biological networks considered in Chapter 4, for which only positive and unlabeled pairs are available (see also the discussion in Section 3.4).

We describe below each of the 11 different subnetworks, focusing on their size and how the tested gene pairs were selected. Details of the different experimental techniques were discussed in Section 5.2.3. We end the section with some global statistics about these subnetworks.

E-MAP

Eight genetic interaction subnetworks were experimentally measured using the E-MAP approach. Each subnetwork corresponds to a set of genes sharing some common biological function and, except for one study, all pairs of genes in the set have been systematically tested, leading to square and symmetric interaction score matrices (with however missing values). Sizes of the gene sets range from 300 to 1500.

The gene sets corresponding to each of the eight E-MAP datasets are defined as follows (in chronological order):

1. 424 genes acting in the early secretory pathway (Schuldiner *et al.*, 2005). The genes in this set are selected because their encoded proteins are located in the Golgi apparatus or in the endoplasmic reticulum.
2. 754 genes involved in chromosome biology, i.e. in functions such as chromatin regulation, transcription, DNA repair or replication, or chromosome segregation (Collins *et al.*, 2007). These genes were selected because they encode a protein from a complex known to be involved into one of the targeted functions.
3. 552 genes involved in one or various RNA-related processes (Wilmes *et al.*, 2008)
4. 483 genes belonging to the phosphorylation network (signaling) (Fiedler *et al.*, 2009). Unlike the previous sets, these genes are not restricted to a specific function but can be involved in several processes in the cell.
5. 356 genes involved in various aspects of plasma-membrane biology (Aguilar *et al.*, 2010)
6. 323 genes encoding general transcription factors and site-specific DNA-binding transcription factors (Zheng *et al.*, 2010).
7. 1487 genes including genes encoding proteins localized in mitochondria and genes encoding proteins acting in the early secretory pathway (Hoppins *et al.*, 2011). Only 481 genes out of the 1487 were used as query, leading to a 1487×481 matrix of genetic interactions. This matrix is called the MITO-MAP.
8. 742 genes involved in lipid metabolism, sorting, post-Golgi trafficking, and other related processes (Surma *et al.*, 2013)

In what follows, we will denote these eight datasets EMAP1 to EMAP8.

Other techniques

In addition to these eight E-MAP networks, we also include in our study three other networks measured with other techniques.

SGA scores. Costanzo *et al.* (2010) apply the SGA score method described earlier to screen 1712 query genes, covering all biological processes. These queries (selected randomly and independently of their function) were crossed to 3885 array strains. This resulted in a large 3885×1712 interaction matrix, with however about 88% missing values.

GIM. To generate this dataset (using the GIM approach), Decourty *et al.* (2008) performed 73 screens with 41 different query mutations (32 duplicates) chosen within genes involved in several RNA metabolism pathways. A subset of 3812 genes were kept in the final dataset. These genes are those that exhibited a signal-to-noise ratio above background in more than half of the experiments.

Table 5.1: The output datasets cover different parts of the 6717×6717 genetic interaction network of the yeast *S.cerevisiae*.

Dataset	Columns	Rows	Negative interactions		Positive interactions		Misses	
EMAP1	424	424	5436	3.02%	2912	1.62%	13116	7.30%
EMAP2	743	743	22942	4.16%	10780	1.95%	187128	39.90%
EMAP3	499	499	6098	2.45%	4290	1.72%	71952	28.90%
EMAP4	451	451	4392	2.16%	2490	1.22%	26036	12.80 %
EMAP5	356	356	2510	1.98%	1042	0.82%	24768	19.54%
EMAP6	300	300	5772	6.41%	1558	1.73%	5990	6.66%
EMAP7	1487	481	18498	2.59%	9141	1.28%	42221	5.9%
EMAP8	741	741	11504	2.10%	3178	0.58%	46976	8.56%
SGA	3840	1673	67245	1.05%	6982	0.11%	5663310	88.15%
GIM	3768	41	1902	1.23%	2079	1.35%	11623	7.52%
ER	322	145	501	1.07%	2007	4.30%	3788	8.11%

ER stress. With the goal of identifying genes contributing to the process of protein folding in the endoplasmic reticulum (ER), Jonikas *et al.* (2009) used the SGA methodology for the construction of 329×152 double-mutants corresponding to genes with a role in ER folding. Here, the phenotype measured to highlight the genetic interaction is not the growth rate, as it was the case in the other networks. Instead, they use the unfolded protein response (UPR) level as a quantitative phenotype, or *ER stress*. The obtained interaction values are called π -scores. A π -score is considered as aggravating when it is smaller than -0.75 and alleviating when it is higher than 0.75 (neutral otherwise).

Statistics about network datasets

One difficulty to integrate these different networks is to match gene names and also to associate them with the feature sets that will be presented in Section 5.3.2. In this goal, we downloaded a list of 6717 genes or open reading frames (ORFs) provided by UCSC (*sacCer2*, June 2008). This genome assembly is based on sequences dated June 2008 from the *Saccharomyces Genome Database* (SGD). The different datasets have then been filtered out to keep only the genes and ORFs that appear in this list.

The main characteristics of the eleven (filtered) datasets are reported in **Table 5.1**, and illustrated in **Figure 5.4**. We also computed the overlap between the different datasets in **Table 5.2**. Some datasets share a quite important part of their pairs with another dataset. For example, EMAP1 shares about 25% of its pairs with EMAP7 and about 25% with EMAP8. ER shares about 20% of its pairs with EMAP1 and 25% with EMAP7.

In total, 3,893,856 pairs have been measured between 5252 genes. We built a graph where there is an edge between two genes if at least one interaction score has been measured between them (including negative, neutral and positive interactions) in one of the eleven output datasets. The adjacency matrix of the resulting graph is represented in **Figure 5.5**. The labeled pairs represent 14.12% of the 5252×5252 matrix (including all genes for which the degree is > 0) and 8.64% of the complete 6713×6713 matrix (including all the genes from the UCSC *sacCer2* list). In this graph, the minimum degree for a gene is equal to 21 and the maximum is equal to 4226 (not taking into

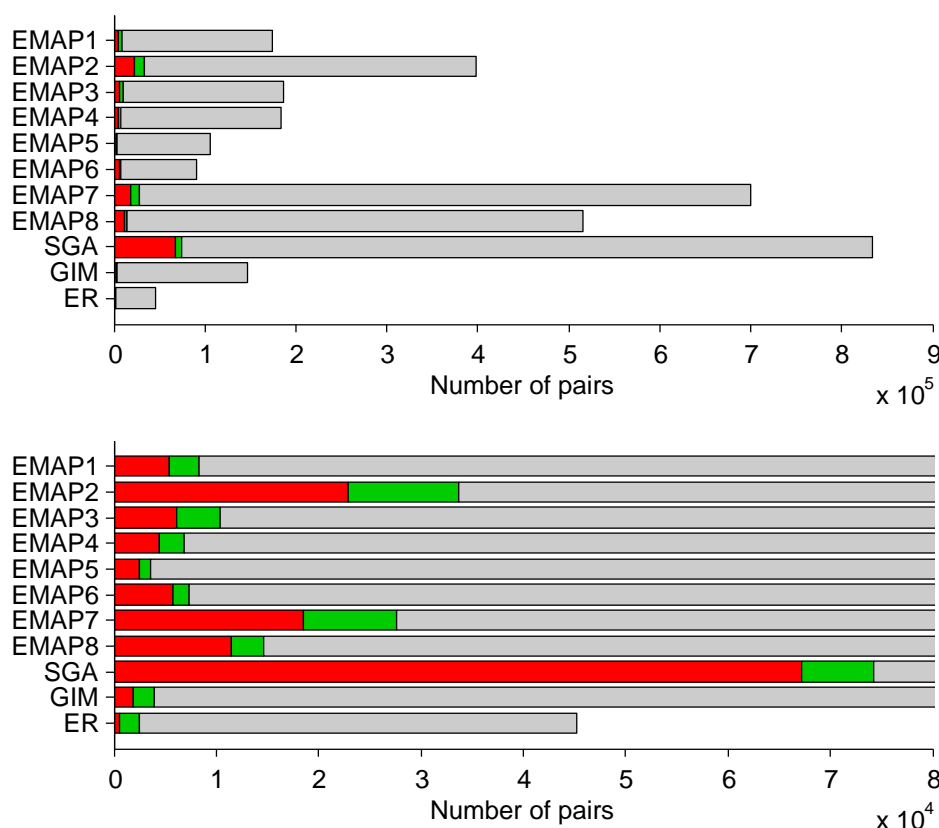


Figure 5.4: The size of the eleven datasets are illustrated in this Figure. The quantities of negative interactions are represented by red bars, positive interactions by green bars and neutral interactions by gray bars. The second graph is a simple zoom-in of the first graph.

Table 5.2: Overlap between the different output datasets. An element (i, j) in the table shows the proportion of pairs in dataset i that are also measured in dataset j .

	EMAP1	EMAP2	EMAP3	EMAP4	EMAP5	EMAP6	EMAP7	EMAP8	SGA	GIM	ER
EMAP1		0.44%	0.02%	0.26%	2.71%	0.02%	24.83%	26.41%	4.30%	0%	4.73%
EMAP2	0.20%		2.86%	3.36%	0.34%	3.67%	2.67%	0.61%	7.67%	1.21%	0.53%
EMAP3	0.02%	5.90%		1.56%	0.05%	1.22%	0.90%	0.06%	4.19%	1.98%	0.09%
EMAP4	0.25%	6.92%	1.56%		2.43%	1.46%	3.87%	2.19%	4.96%	0.92%	0.17%
EMAP5	4.43%	1.22%	0.08%	4.23%		0.05%	11.44%	17.75%	5.56%	0%	0.32%
EMAP6	0.03%	15.95%	2.58%	3.09%	0.06%		1.20%	0.28%	5.40%	0.70%	0.43%
EMAP7	6.20%	1.46%	0.24%	1.03%	1.75%	0.15%		6.66%	4.35%	0%	1.57%
EMAP8	8.76%	0.45%	0.02%	0.77%	3.60%	0.05%	8.84%		3.37%	0%	6.66%
SGA	0.94%	3.67%	0.97%	1.15%	0.74%	0.59%	3.81%	2.22%	0%	0.51%	0.38%
GIM	0%	3.18%	2.51%	1.17%	0%	0.42%	0%	0%	2.78%		0%
ER	18.31%	4.46%	0.36%	0.71%	0.77%	0.84%	24.43%	8.84%	6.72%	0%	

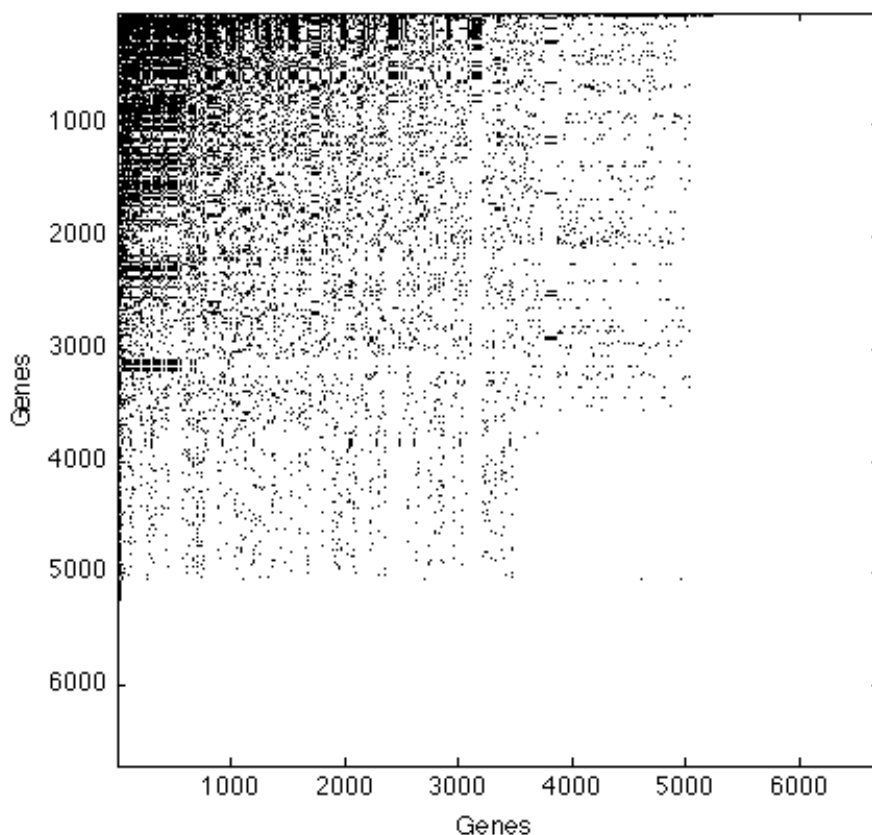


Figure 5.5: 6713×6713 adjacency matrix of the genetic interaction network of the yeast *S.cerevisiae*. A black point corresponds to a pair for which an interaction score (negative, neutral or positive) has been measured in at least one of the eleven datasets. To place most labeled pairs in the top left corner of the graph, genes are sorted on the axes in decreasing order of the number of measured pairs in which they are involved (i.e., their degree).

account the set of unlabeled genes). Although 4 Millions pairs have already been experimentally checked, these statistics show that a huge amount of pairs still have to be labeled.

In addition to the network of all labeled pairs, we also computed the two networks of pairs interacting negatively and positively. We consider that a pair interacts negatively if it is labeled as negative in at least one dataset, without being labeled positively in another one. We got a total of 179,482 negatively interacting pairs, which represent 4.61% of the labeled pairs. In the same manner, we considered that a pair interacts positively if it is labeled as positive in at least one dataset, without being labeled negatively in another one. We got a total of 54,107 positively interacting pairs, which represent 1.39% of the labeled pairs. Pairs that are labeled as negative in one dataset and positive in another one are considered as neutral. We found 1264 pairs with such conflicting measurements.

Table 5.3: Summary of all input datasets

Nr	Descriptions	Types of data	Nb of genes	Nb of variables	% of misses
1	Chemo-gen. (homoz.)	Real	4715	418	9.49
2	Chemo-gen. (heteroz.)	Real	5289	726	3.26
3	Fitness (homoz.)	Real	4625	14	0
4	Fitness (heteroz.)	Real	5711	11	0
5	Chemo-gen. (Costanzo)	Real	6004	11	2.57
6	Morphol. data (SCMD)	Real	4693	501	0
7	Express. (deleted genes)	Real	6157	300	0.44
8	TF deletion (1)	Real	5970	269	1.31
9	TF deletion (2)	Real	6171	212	12.41
10	Expression	Real	6245	904	0
11	Histone modifications	Real	6100	44	40.24
12	Protein abundances	Real	4146	6	45.8
13	Localization	Binary	6214	23	0
14	Upstream motifs	Binary	5564	6816	0
15	Phylogenetic profiles	Binary	6299	145	0
16	eQTL	Binary	4715	41	0
17	Natural variations	Real	5832	38	0
18	Gene essentiality	Binary	6717	1	0
19	Transcription network	Binary	4351	157	0
20	Kinase network	Binary	1331	87	0
21	Protein complexes	Binary	2668	547	0
22	Chaperons	Binary	3813	128	0
23	RBP network	Binary	4476	46	0

5.3.2 Input datasets

The set of input features used for the inference is composed of 22 datasets, for a total of 11319 features (4503 when excluding the upstream motifs). These datasets are summarized in **Table 5.3** and discussed separately below. We divide the 22 sets into four main categories: measurements obtained following gene deletions (6 datasets), gene expression and protein abundance measurements (6 datasets), genetic information (6 datasets), and networks (5 datasets).

Measurements obtained after gene deletions

Chemical genomics. Hillenmeyer *et al.* (2008) performed chemical genomic assays on the yeast whole-genome heterozygous and homozygous deletion collections. They quantified the growth fitness of gene deletion strain, in the presence of chemicals or environmental stress conditions. The goal of this experiment was to uncover phenotypes for genes of the yeast. We can distinguish two different datasets:

- The homozygous dataset (both alleles are deleted) that contains 418 variables that represent the different small molecules and environmental stresses. They are related to 4715 genes, all non-essential since their homozygous deletion would be lethal.

- The heterozygous dataset (one allele is deleted) that contains 726 variables and is related to 5289 genes.

		environments			
		x1	x2	x3	...
deleted genes	g1				
	g2				
	⋮				

Fitness. In the study of Steinmetz *et al.* (2002), heterozygous diploid and homozygous diploid deletion strains were quantitatively measured and monitored in parallel in different medium conditions. We can again distinguish two datasets:

- the homozygous dataset, containing 14 variables related to 4625 genes.
- the heterozygous dataset, containing 12 variables related to 5711 genes.

Chemical genomics (Costanzo). These chemical genomic assays were performed by Costanzo *et al.* (2010) as described previously in (Hillenmeyer *et al.*, 2008). The obtained fitness defect scores were mean normalized by genes and by experiment. The dataset contains 11 variables and is related to 6004 genes.

Morphological data (SCMD). This dataset comes from the assays in (Ohya *et al.*, 2005). They deleted genes and obtained quantitative morphological data of yeast mutant cells, like cell shape or nuclear morphology of cells at a specific stage of the cell cycle. The dataset contains 501 variables and is related to 4693 genes.

Gene expression and protein abundance

Expression with deleted genes. This dataset comes from (Hughes *et al.*, 2000). They constructed a reference database of gene expression profiles corresponding to 300 diverse mutations and chemical treatments in yeast. The dataset contains 300 variables and is related to 6157 genes.

TF deletion. These data come from the two references (Hu *et al.*, 2007) and (Chua *et al.*, 2006).

- In the first paper, the authors grew up 269 transcription factor knockout strains and measured gene expression of each of these strains using microarrays. The dataset contains 269 variables and is related to 5970 genes.
- In the second paper, microarray profiling was performed in replicate in dye reversal format for each transcription factor experiment (overexpression or deletion) and the dataset shows each replicate separately. There are 55 overexpressed TFs and 51 deleted TFs. Then the dataset contains 212 variables and is related to 6171 genes.

Expression data (mRNA). This dataset comes from (Faith *et al.*, 2007a). It collects Affymetrix expression compendia for *Saccharomyces cerevisiae*, totaling 904 variables related to 6245 genes.

Histone modifications. Histone are proteins found in the nuclei of eukaryotic cells. DNA winds around them and histones allow, or not, the DNA to be transcribed. This dataset comes from (O'Connor and Wyrick, 2007). They warehoused genome-wide microarray data mapping patterns of 22 histone modifications and other chromatin features. The dataset contains 44 variables related to 6100 genes (half of the variables is relative to promoters and the other half is relative to ORF).

Protein abundances and variation. These data come from (Newman *et al.*, 2006). They measure protein abundance by flow cytometry. The first two variables are the abundance measurements for strains grown in a rich and a minimal medium. The third and fifth columns are the coefficient of variation values for the two media, and the fourth and sixth columns are the running median of CV values for the two media. The dataset contains then 6 variables and is related to 4146 genes.

Genetic information

Localization. The vector of features in this case consists of 23 binary values coding for the presence/absence of the protein in a given intracellular location. This data was obtained from the experiment in (Huh *et al.*, 2003). The dataset contains 23 variables and is related to 6214 genes.

Upstream motifs. This dataset comes from (Brohée *et al.*, 2011). The dataset was collected using the method in (Janky and van Helden, 2008). It contains over-represented spaced motifs (dyads) in the upstream non-coding sequences of genes in 19 species of the class Saccharomycetes. A total of 33,276 dyads have been collected for 5564 genes. However, we limited ourself to motifs that appear in the upstream sequences of at least two of the 2010 genes that appear in at least one of the first five E-MAPs. This reduces the number of motifs from 33,276 down to 6,816.

Phylogenetic profile. The existence of orthologs of a given gene in a set of species is potentially an important source of information for the prediction of biological networks. In our experiments, we use the phylogenetic profiles gathered by Yamanishi and Vert (2005). They were obtained from the orthologous clusters in KEGG. Only fully sequenced genomes are taken into account. Each gene is described by a vector of 145 binary values, each one coding for the presence or the absence of an orthologous gene in a given organism. The dataset contains 145 variables and is related to 6299 genes.

eQTL. These data come from (Smith and Kruglyak, 2008). eQTL stands for 'expression quantitative trait loci' and are genomic loci that regulate genetic expression levels. Smith and Kruglyak (2008) performed linkage analysis on the transcript level within glucose and ethanol conditions, and 3997 and 3489 linkages were observed in glucose and ethanol respectively. They also determined loci that show gene-environment interactions (gxeQTL) and found 1555 linkages. They divided the genome into 10-centimorgan-sized bins and counted the number of linkages observed in glucose, ethanol, or gxeQTL that fell within each of the bins. The majority of linkages fell into bins with a significant excess of linkages. Significant bins that were located immediately next to each other were merged into a single peak to form 13, 13, and 15 peaks for glucose, ethanol, and gxeQTL respectively. Each of the 41 variables of this dataset correspond to these peaks and are related to 4715 genes.

Natural variations. These data come from (Liti *et al.*, 2009). They effectively estimate the copy number of each gene in 38 different strains. The dataset contains then 38 variables and is related to 5832 genes.

Gene essentiality. This dataset contains only one binary feature indicating for each gene whether it is essential or not. The list of essential genes was obtained from (Giaever *et al.*, 2002) and is composed of 1115 genes.

Networks

Transcription network. This dataset comes from Balaji *et al.* (2006) and contains 12,722 interactions between transcription factors and genes. There is an interaction when the former regulates the expression of the latter. The dataset contains one variable for each of the 157 transcription factors and concerns 4351 genes.

Kinase network. These data come from (Ptacek *et al.*, 2005) that identified 4064 phosphorylation events involving 1331 proteins. The dataset contains 87 variables (protein kinases) and is related to 1331 genes.

Protein complexes. These data come from (Krogan *et al.*, 2006). A Markov clustering algorithm organized these interactions into 547 protein complexes averaging 4.9 subunits per complex. Then the dataset contains 547 binary variables involving 2668 genes.

Chaperons. Chaperones are class of cellular proteins which assist folding of proteins to functional conformation correctly and efficiently. The dataset come from ChaperoneDB¹, which is a collection of yeast chaperone interactions. It contains 128 binary variables (chaperons) related to 3813 genes.

RNA-binding proteins network. These data come from (Hogan *et al.*, 2008). RNA-binding proteins (RBP) play a role in the regulation of many post-transcriptional steps in gene expression. They searched for the RNA targets of 46 proteins. They chose a 1% false discovery rate as a criterion to distinguish targets from non-targets, for most proteins. The dataset contains 46 variables (proteins) and is related to 4476 genes.

5.4 Cross-validation experiments

In this section, we investigate the exploitation of extremely randomized trees (Geurts *et al.*, 2006a) and support vector machines in the context of the global and the local approaches (see Section 4.2 for an explanation of these approaches) for the supervised inference of the genetic interactions of the yeast. To predict negative interactions, we consider that our "positive" examples are negative interactions and that our "negative" examples are the ensemble of both positive and neutral interactions. Symmetrically, to predict positive interactions, we consider that our "positive" examples are positive interactions and that our "negative" examples are the ensemble of negative and neutral interactions. In both cases, unlike what we did for most networks in Chapter 4, there is no need

¹<http://chaperonedb.ccb.utoronto.ca>

to convert the unlabeled (or missing) pairs into “negative” examples given the availability of the neutral interactions as well as the interactions of the opposite sign. This is an ideal setting as it avoids the introduction of false negatives in the training data (see Section 3.4).

Our ultimate goal is of course to provide predictions of interactions for pairs that have not been measured in any of the eleven genetic interaction subnetworks. Before addressing this question in Section 5.5, we evaluate in this section the performance of our predictive models by using cross-validation. In Section 5.4.1, we first perform cross-validation experiments across the 11 individual subnetworks separately. In the context of these experiments, we also carry out a first assessment of the relevance of the 22 input datasets for predicting genetic interactions and we compare the performance of Extra-Trees and support vector machines. In Section 5.4.2, we take benefit from the overlap between the different subnetworks to see how well the interaction scores provided in the context of one study can predict the interactions measured in the context of another study. This performance provides a baseline with which we compare the results obtained in Section 5.4.1 with supervised inference methods. Finally, in Section 5.4.3, we carry out cross-validation experiments on the set of all measured gene pairs.

5.4.1 Cross-validation across individual networks

We first perform cross-validation on pairs (to evaluate $LS \times LS$ predictions) across individual networks. We start with EMAP1 and consider every of the 23 datasets in turn to predict both positive and negative interactions. We then carry out cross-validation experiments on all other subnetworks. In this section also, we compare extremely randomized trees with support vector machines and evaluate the inclusion of the genetic interaction profile among the inputs to improve $LS \times LS$ predictions.

Performance of the different input datasets

The goal of our first experiment is to assess the ability of the different input datasets to predict genetic interactions. For that purpose, we performed 10-fold cross-validation on pairs ($LS \times LS$) across EMAP1, using as input every of the 23 datasets in turn. We used the Extra-Trees algorithm with the global approach and predicted separately negative and positive interactions. For the ten binary input datasets, we used bootstrapping to generate the training set of each tree (while Extra-Trees are normally grown from the full dataset). Indeed, split randomization as done in the Extra-Trees algorithm do not lead to enough diversity with binary features, which yield poor performance. For each input dataset, we restricted our training data to the set of EMAP1 genes for which a measurement of the tested input features was available. This filtering reduced the size of the EMAP1 network from 424×424 originally to sizes ranging from 94×94 (for the kinase network input dataset) to 424×424 (gene essentiality). By doing so, we avoid having to deal with missing values and thus get a better assessment of each feature set. It means however that our AUROC and AUPR scores are not strictly comparable from one dataset to another, since they are not computed from the exact same set of pairs.

Areas under ROC curves (AUROC) and precision-recall curves (AUPR) that we obtained are shown in **Figure 5.6**. They allow to rank the different input datasets from the best to the worst for predicting genetic interactions. The rankings derived from the AUROC and AUPR are very similar. The best (resp. the worst) dataset in terms of AUROC is also the best (resp. the worst) in terms

Table 5.4: Comparison of AUPR values resulting from the prediction of negative and positive interactions of EMAP1, with homozygous chemo-genomic dataset. When the positive AUPR is adapted to take into account the difference between the ratios of the two networks (in the last column), it increased but stay lower than the negative AUPR.

	Negative int.	Positive int.	Positive int.
Ratio of interactions	3.20%	1.87%	3.20%
AUPR	0.48	0.21	0.29

of AUPR. The difference in performance between the datasets are nevertheless more important in AUPR than in AUROC.

The vertical lines in these plots represent two baselines:

1. Plain lines represent the AUROC and AUPR obtained with random predictions. These values are equal to 0.5 for AUROC and are equal to the proportion of interactions for AUPR (i.e., 0.03 for the negative interactions and 0.02 for the positive ones).
2. Dotted lines represent the AUROC and AUPR obtained when using only node degrees to predict interactions. This second baseline, which is motivated in Section 3.5, is more realistic. For negative interactions, the AUROC and AUPR of this baseline are equal respectively to 0.85 and 0.27. For positive interactions, they are respectively equal to 0.86 and 0.11.

An input dataset that does not allow to reach the level of these baselines is not useful to predict genetic interactions on its own (although it might still be useful when combined with another dataset).

For negative interactions, only 9 out of the 23 datasets give better AUPR than those obtained with node degrees (the second baseline). Among these 9, the homozygous chemo-genomic dataset outperforms clearly the others. For positive interactions, only 4 out of the 23 datasets give AUPR scores better than those obtained with node degrees, and no dataset gives better AUROC. In this case, ROC curves is then unable to detect performances that are better than the baseline. Consequently and according to conclusions of Chapter 3, we chose to focus on precision-recall curves in the rest of this chapter. For positive interaction prediction, the homozygous chemo-genomic dataset is again the most predictive.

Datasets from the first category ('measurements obtained after gene deletion') are the most predictive. This is not surprising since genetic interaction scores are also measured by performing gene deletions. Among this category, heterozygous deletion collections give less good results. Measurements with only one deleted allele does not seem to be sufficient to be exploited to predict genetic interactions. Among the three other dataset families, some expression datasets are also more informative than the baselines but no dataset stand out from the last two categories, genetic information and networks, although the natural variation data and the protein complex network reach much higher AUPR than the others.

When analyzing the different AUPR values of **Figure 5.6**, negative interactions appear to be much easier to predict than positive interactions. However, given that the ratio of interactions versus non-interactions is not the same in the two networks, their precision-recall curves can not really be compared (but ROC curves can). **Table 5.5** presents AUPR values adapted using the formula presented in Section 3.2.4 to make them comparable: all the values have been modified to

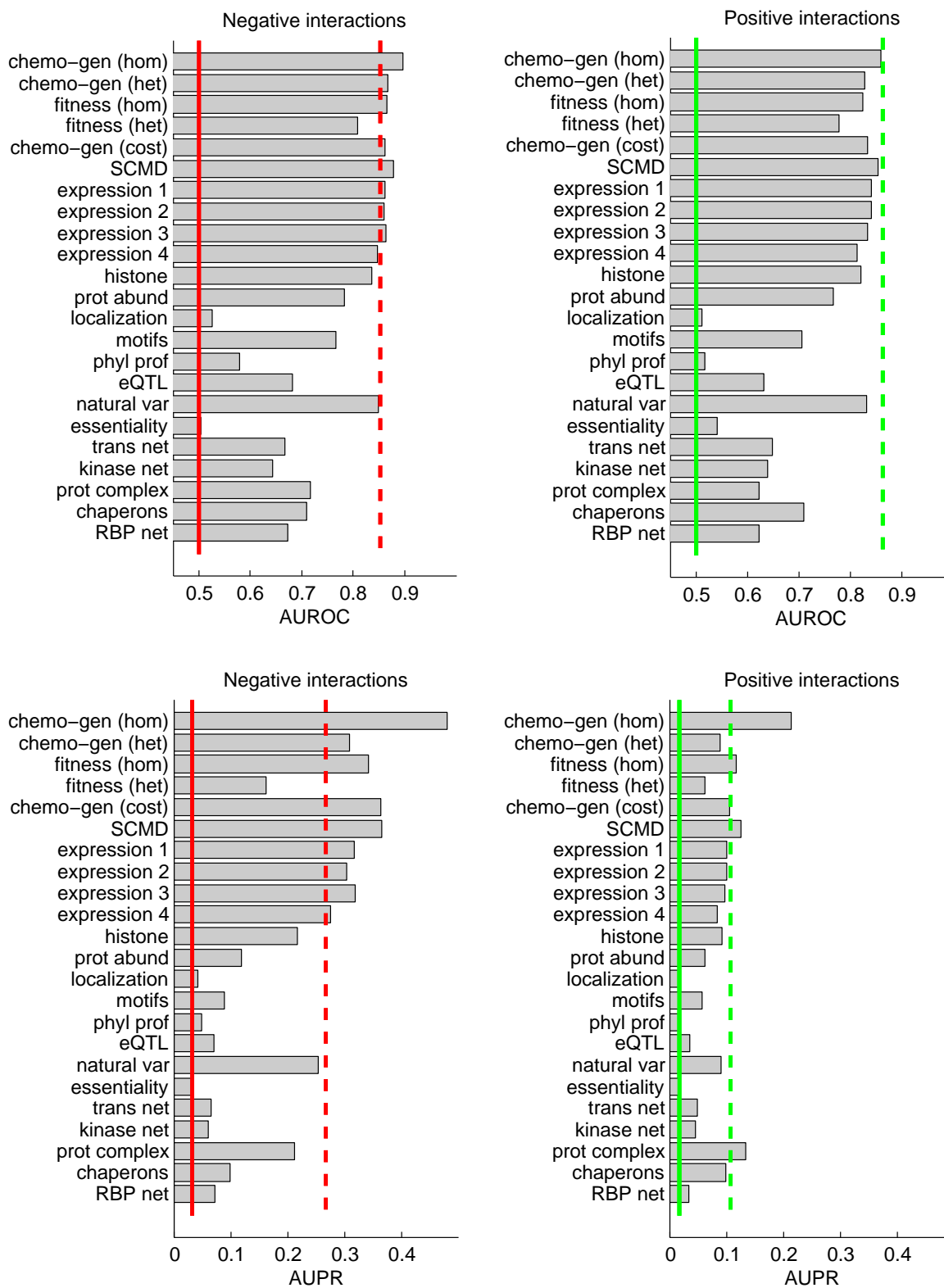


Figure 5.6: Area under the ROC (AUROC) and precision-recall curves (AUPRC) as a function of the 23 input datasets, for $LS \times LS$ predictions in EMAP1.

Table 5.5: As in **Figure 5.6**, we present the AUPR that illustrate the predictive performance of the 23 feature sets in the prediction of EMAP1. Moreover, we adapted the values to make them comparable between each other, using the formula presented in Section 3.2.4.

Nr	Descriptions	Negative interactions		Positive interactions	
		AUPR	Adapted AUPR	AUPR	Adapted AUPR
1	Chemo-gen. (homoz.)	0.48	0.46	0.21	0.19
2	Chemo-gen. (heteroz.)	0.31	0.31	0.09	0.15
3	Fitness (homoz.)	0.34	0.34	0.12	0.17
4	Fitness (heteroz.)	0.16	0.15	0.06	0.10
5	Chemo-gen. (Costanzo)	0.36	0.35	0.11	0.16
6	Morphol. data (SCMD)	0.37	0.34	0.13	0.18
7	Express. (deleted genes)	0.32	0.30	0.10	0.16
8	TF deletion (1)	0.30	0.29	0.10	0.16
9	TF deletion (2)	0.32	0.31	0.10	0.16
10	Expression	0.28	0.26	0.09	0.13
11	Histone modifications	0.22	0.21	0.09	0.15
12	Protein abundances	0.12	0.11	0.06	0.10
13	Localization	0.04	0.04	0.02	0.03
14	Upstream motifs	0.09	0.08	0.06	0.09
15	Phylogenetic profiles	0.05	0.05	0.02	0.03
16	eQTL	0.07	0.07	0.02	0.03
17	Natural variations	0.25	0.24	0.09	0.15
18	Gene essentiality	0.03	0.03	0.02	0.03
19	Transcription network	0.07	0.06	0.05	0.07
20	Kinase network	0.06	0.06	0.05	0.07
21	Protein complexes	0.21	0.14	0.13	0.16
22	Chaperons	0.10	0.08	0.04	0.05
23	RBP network	0.07	0.06	0.03	0.05

be representative of a network with 3.02% of pairs interacting, i.e. the complete EMAP1 network of negative interactions. These results confirm that negative interactions are indeed easier to predict than positive ones. Nevertheless, there are four exceptions concerning the feature sets number 14, 19, 20 and 21 where adapted AUPR are slightly better for positive interactions than for negative ones.

We combined several input datasets to check whether it is possible to improve the performances obtained with the homozygous chemo-genomic dataset. We add to this dataset the three next best predictive datasets, and performed the same cross-validation experiment as before. For the prediction of negative interactions, the three additional datasets are SCMD, chemo-genomic (Costanzo) and fitness (homozygous). For positive interactions, they consist in protein complex network, SCMD and fitness (homozygous). We use as the training network the subnetwork of 353 genes among the 424 genes in EMAP1 that have values in the homozygous chemo-genomic dataset. For the three additional datasets, we had to complete the missing values. For numerical features, we replaced

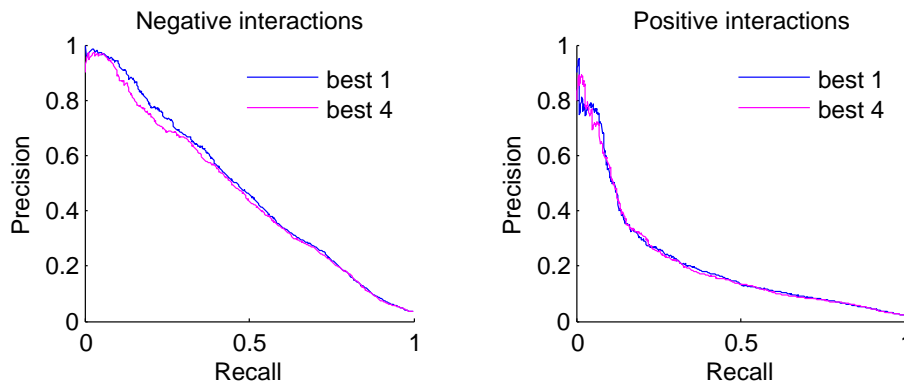


Figure 5.7: Precision-recall curves obtained by cross-validation on pairs of genes from EMAP1, for predicting negative and positive interactions. Blue curves are obtained when using homozygous chemo-genomic input dataset (the best predictive one) and magenta curves are obtained when combined the four best predictive input datasets. Not improvement happened when adding some input set to the most predictive one.

missing values by the average feature value computed over non-missing genes. For binary features, we filled missing values with zeroes. The resulting precision-recall curves are shown in **Figure 5.7**. They show that there is no improvement when combining the best four datasets.

Performances on the different output datasets

We performed the same cross-validation experiments on the EMAP2, EMAP3, EMAP4, EMAP5 and SGA subnetworks (see **Figure 5.8** for SGA) and obtained similar results as on EMAP1. Whatever the subnetwork, the homozygous chemo-genomic dataset always gives the best performance for predicting genetic interactions and no significant improvement can be gained by combining several datasets. We will therefore always use this specific dataset as input in the experiments that we will perform in the rest of this chapter.

Figure 5.9 shows precision-recall curves obtained by cross-validation on pairs on the 11 output datasets with both the (single output) local and global approaches and the Extra-Trees algorithm. These results shows the performance at predicting $LS \times LS$ pairs, i.e., the performance we would obtained when predicting missing values within each network. Note that the ratio of missing values is very different from one network to the other. It varies from 6% in EMAP7 to 88% in SGA (see Table 5.1). From these curves, it is clear that the global approach gives always slightly better results than the local approach. This result is consistent with results of Chapter 4, where we already noticed that the global approach was better than the local one for $LS \times LS$ predictions on nine out of ten biological networks. Note that we could not test the global approach on the SGA network because of memory problems due to the large size of this network.

On all EMAPs, negative interactions are much easier to predict than the positive ones, confirming our previous results on EMAP1. On GIM, both kinds of interactions are equally well predicted and on the ER network, positive interactions are better predicted than negative ones. Note that this last network is the only one where the ratio of positive interactions (4.3%) is higher than the ratio of negative interactions (1.1%), which might explain this difference.

Among the different EMAPs, EMAP4 seems to be the most difficult to predict. The reason why the predictions on EMAP4 are not as good as on the other EMAPs could be explained from

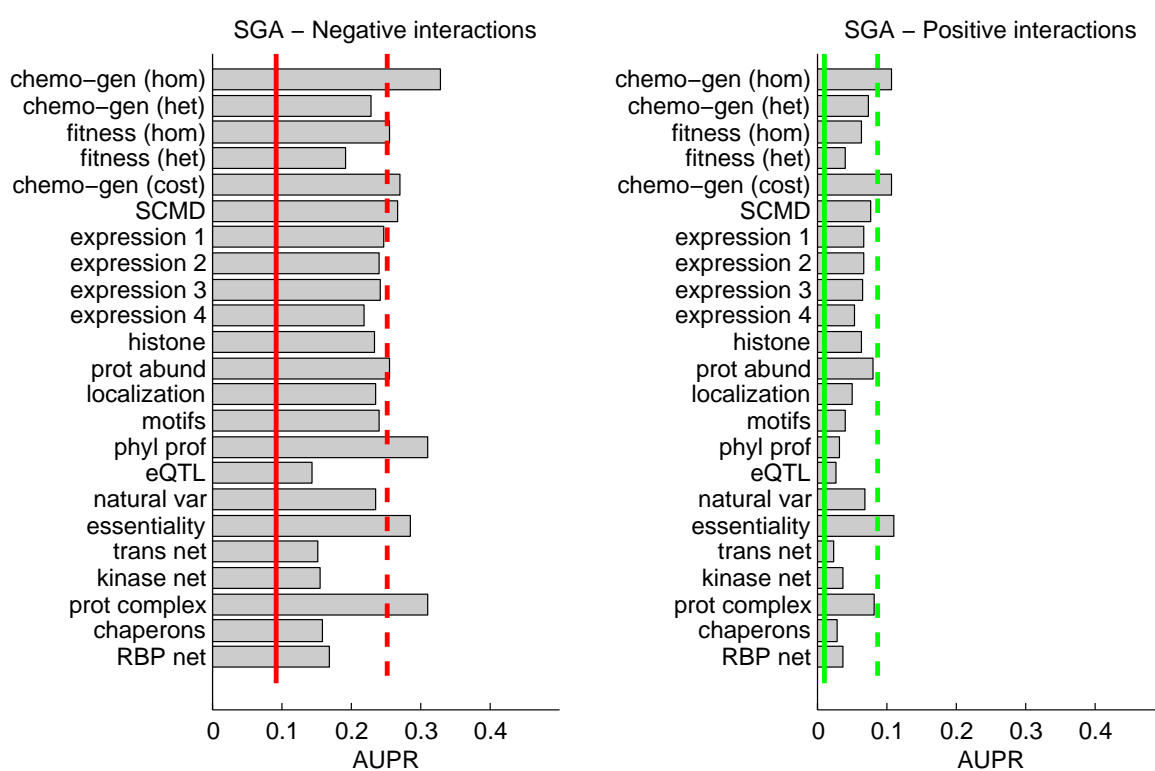


Figure 5.8: Area under the precision-recall curves (AUPRC) as a function of the 23 input datasets, for $LS \times LS$ predictions in SGA.

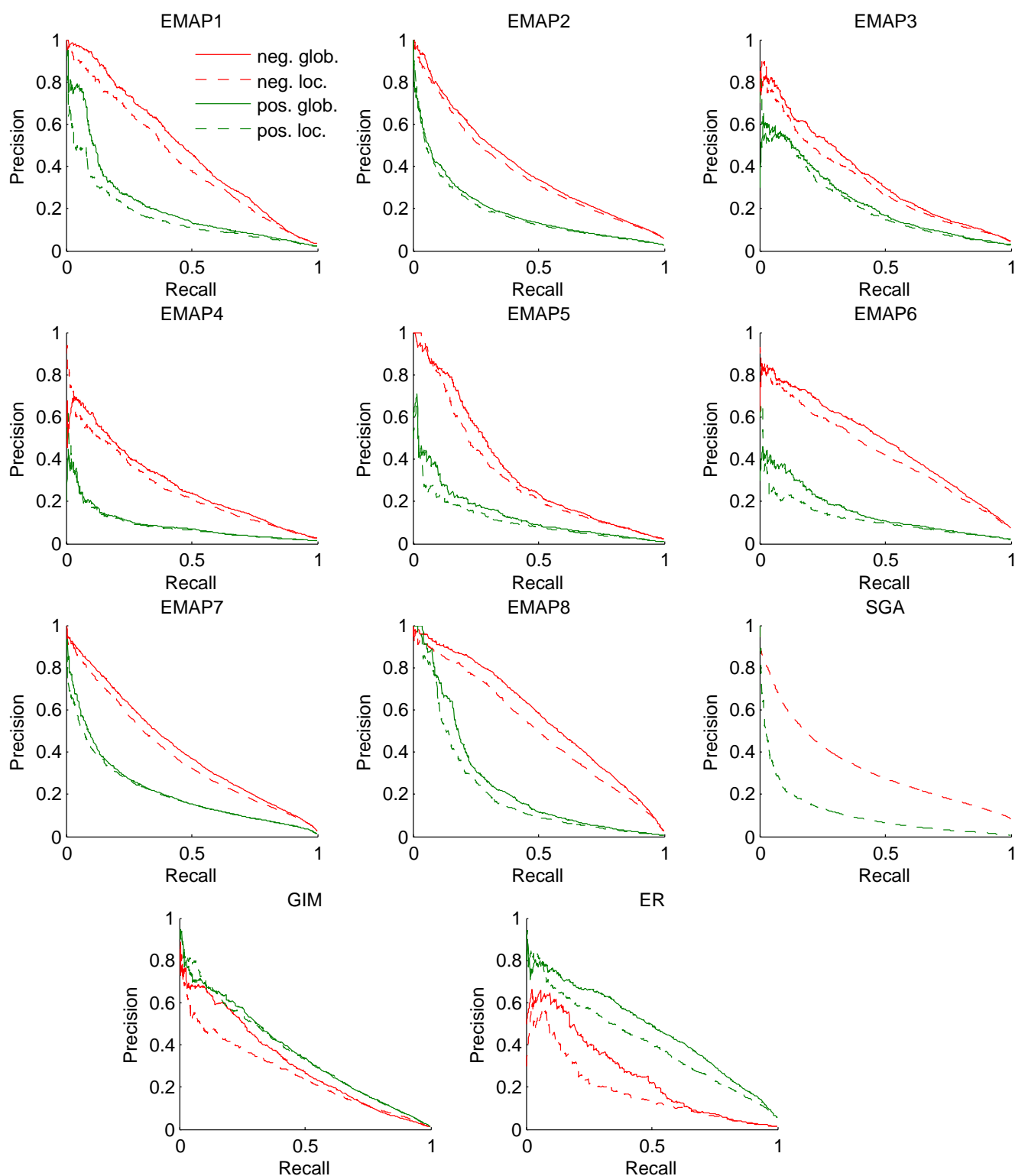


Figure 5.9: Precision-recall curves obtained by cross-validation on pairs of genes from each of the 11 output interaction networks, for predicting negative (red curves) and positive (green curves) interactions. Full lines represent the global approach and dotted lines the local approach. Negative interactions seem to be much easier to predict than positive ones, except on GIM and ER, and the global approach works slightly better than the local one, whatever the network. Due to memory constraints, the global approach could not be applied on the larger SGA network.

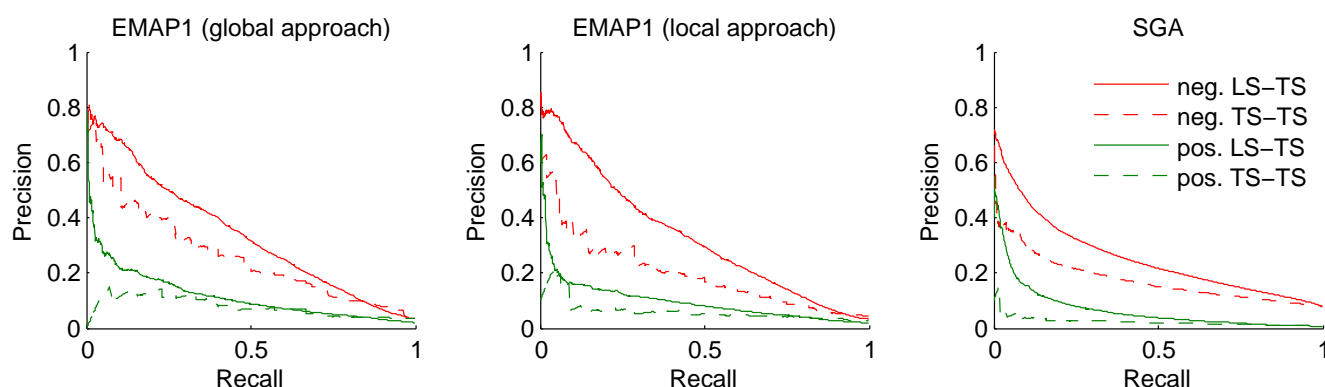


Figure 5.10: Precision-recall curves obtained by cross-validation on genes from EMAP1 and SGA, for predicting negative (red curves) and positive (green curves) interactions. Full lines represent $LS \times TS$ predictions and dotted lines $TS \times TS$ predictions.

the particular nature of this dataset. While in all other E-MAPs, genes are selected because of their participation to some specific biological process, in EMAP4, genes have been selected because of their participation in the phosphorylation network. They are thus potentially involved into very different biological processes and thus do not necessarily share common interaction profiles.

Cross-validation on genes

To measure the ability to predict $LS \times TS$ and $TS \times TS$ pairs, we repeated previous experiments with (10-fold) cross-validation on genes instead of pairs (cfr Chapter 3). We performed this cross-validation on EMAP1 to EMAP5, ER, GIM and SGA (see **Figure 5.10** for EMAP1 and SGA). The conclusions of this experiment are as expected. First $LS \times TS$ pairs are more difficult to predict than $LS \times LS$ pairs, and even more so for $TS \times TS$ pairs. Second, negative interactions are better predicted than positive ones, except for GIM and ER.

Comparison with support vector machines

We also carried out some experiments comparing extremely randomized trees with support vector machines (SVM) in the context of the local approach. For these experiments, we used LIBSVM Chang and Lin (2001) with a gaussian kernel and default parameter setting². As previously, we performed 10-fold cross-validation on pairs to estimate $LS \times LS$ performance and 10-fold cross-validation on genes to estimate $LS \times TS$ and $TS \times TS$ performance. The resulting AUPR scores on EMAP1 to EMAP5 are presented in **Table 5.6** both for negative and positive interactions. As already observed on other networks in Chapter 4, Extra-Trees and SVM are indistinguishable on the four EMAPs and whatever the nature, positive or negative, of the interactions.

Use of genetic interaction profile

In the local approach, a prediction for a $LS \times LS$ pair (g_a, g_b) is computed as the average of the prediction of the model trained for g_a and the model trained for g_b . These two models are trained

²Different parameter settings were manually explored on EMAP1 but no improvement could be obtained with respect to the default setting.

Table 5.6: Area under precision-recall curves for the first five EMAPs for the local approach with Extra-Trees and support vector machines. Two cross-validations were tried: CV on pairs and CV on genes. Both methods are very close to each other.

	Negative interaction						Positive interaction					
	$LS \times LS$		$LS \times TS$		$TS \times TS$		$LS \times LS$		$LS \times TS$		$TS \times TS$	
	E-T	SVM	E-T	SVM	E-T	SVM	E-T	SVM	E-T	SVM	E-T	SVM
EMAP1	0.43	0.43	0.34	0.35	0.19	0.18	0.17	0.15	0.12	0.08	0.03	0.03
EMAP2	0.37	0.41	0.27	0.31	0.14	0.13	0.15	0.18	0.11	0.14	0.06	0.06
EMAP3	0.32	0.32	0.25	0.24	0.13	0.12	0.20	0.20	0.13	0.14	0.11	0.09
EMAP4	0.26	0.27	0.20	0.21	0.12	0.11	0.08	0.08	0.06	0.06	0.03	0.03
EMAP5	0.32	0.28	0.23	0.22	0.13	0.14	0.12	0.09	0.07	0.07	0.05	0.04

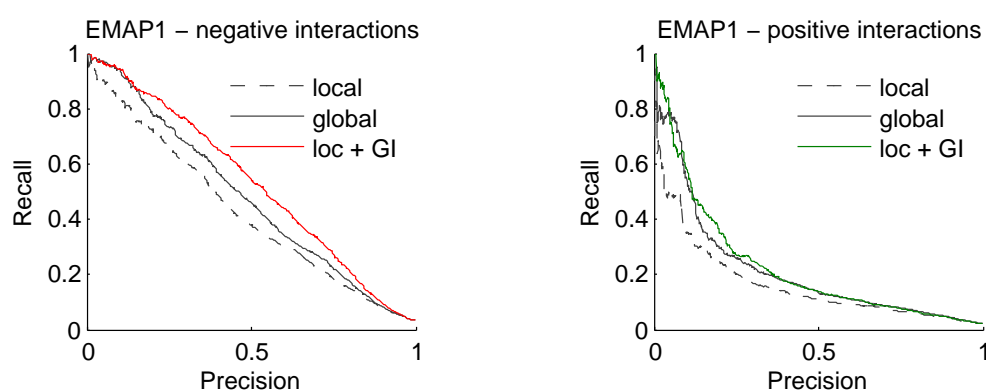


Figure 5.11: Precision-recall curves obtained by cross-validation on pairs for predicting negative and positive interactions. Full gray lines represent the global approach and dotted gray lines the local approach. Red and green curves represent the local approach in which we add GI profiles to the chemogenomic feature set. This latter method outperforms the other ones in term of AUPR.

only from known interactions involving g_a and g_b respectively and therefore their training ignores all other interactions. This is in contrast with the global approach that consider all known interactions at once during the training stage. This might explain why the global approach outperforms the local one on $LS \times LS$ pairs. As a way to exploit more information when training the local model, we propose to add the genetic interaction (GI) profile of each gene to the input features when training the local model. When training a local model for a gene g_a , the genetic interaction profile of a given gene $g \neq g_a$ in the training set is constructed as a vector where each component corresponds to a gene $g' \neq g, g_a$ and is equal to the interaction score of the pair (g, g') if this pair was measured and zero otherwise. This vector is concatenated to the chemogenomic features before training the local model. Note that this trick can only be applied for $LS \times LS$ pairs, because by definition a TS gene does not have any available GI profile.

We experimented with this extension on several networks (see **Figure 5.11** for results on EMAP1) and found that incorporating the GI profile typically improves predictive performance. Actually, it even leads to better results than the global approach in a number of cases.

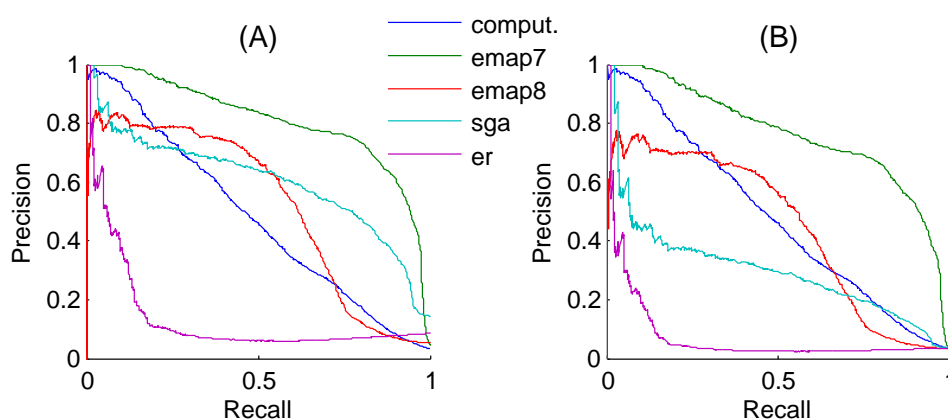


Figure 5.12: (A) Precision-recall curves when using four networks separately to predict EMAP1, on their respective common pairs. (B) The same curves as in (A), but corrected according to the different ratios of interactions. These curves must be compared with the blue curve that was obtained by cross-validating across pairs of EMAP1 (“comput.”).

5.4.2 Computational versus experimental predictions

We showed in **Table 5.2** that several networks have pairs in common. But labels attributed to these common pairs are not systematically the same in the different networks, due to technical variation in biological experiments. A comparison of two experimental techniques applied on the same pairs is interesting because it will give us another baseline with which to compare the performance of our computational predictions. To perform such comparison for two subnetworks with an intersection, we propose to rank the common pairs according to their interaction scores in one of the two subnetworks and then to plot the precision-recall curves obtained when considering the binary labeling of each pair in the second subnetwork as the true label. The resulting curve will show how close the two networks are along their common pairs.

We performed this experiment four times to predict the negative interactions in EMAP1 from interaction scores in the four networks with which EMAP1 shares at least 3% of its pairs. These networks are EMAP7 (25%), EMAP8 (26%), SGA (4%), and ER (5%). Resulting precision-recall curves are shown in **Figure 5.12A**. For these curves to be strictly comparable, we also corrected them to take into account the difference in the proportion of interactions in each of the tested sets of pairs, by using the formula presented in Section 3.2.4. Indeed, for example, the ratio of interactions in the pairs from EMAP1 that are shared by EMAP7 is equal to 5%, and in the pairs that are shared with SGA it is equal to 14%. We adapted the curves to a common ratio equal to 3%, which is the ratio of negative interactions in EMAP1. The modified curves are shown in **Figure 5.12B**.

Among the four networks, EMAP7 appears to be the closest to EMAP1, followed by EMAP8, then SGA, and finally ER. These comparisons must be interpreted carefully due to the fact that the set of compared pairs is not the same in every experiment. Nevertheless, this ranking was expected: the two closest networks were generated by using the same experimental approach as EMAP1 (Epistatic Mini-Array Profiles), and the two furthest networks were created by using a different experimental approach (SGA approach). Moreover, the furthest network, ER, defines its interactions on the basis of a different phenotypic measure (ER stress instead of growth rate). These results show that interactions established by biological experiments are very dependent on the type of experimental method.

Table 5.7: SGA pairs were used to predict pairs from the ten other networks, along the genes they share with SGA, for negative and positive interactions. Resulting AUPR (“Experim.” columns) are almost always lower than AUPR obtained by cross-validating across pairs of the different networks (“Comput.” columns), meaning that computational techniques can do better predictions than biological experiments.

	Negative interactions		Positive interactions	
	Experim.	Comput.	Experim.	Comput.
EMAP1	0.30	0.48	0.09	0.21
EMAP2	0.37	0.40	0.13	0.19
EMAP3	0.27	0.35	0.16	0.23
EMAP4	0.29	0.28	0.12	0.09
EMAP5	0.23	0.35	0.07	0.14
EMAP6	0.36	0.48	0.09	0.15
EMAP7	0.26	0.42	0.08	0.21
EMAP8	0.26	0.57	0.06	0.25
GIM	0.16	0.31	0.04	0.36
ER	0.03	0.25	0.04	0.48

Comparison with cross-validation over one subnetwork

It is interesting to compare these PR curves with the curve we obtained by cross-validation over pairs on EMAP1. We included this curve in **Figure 5.12** (“comput”). It corresponds to the curve obtained with the global approach in the first graph of **Figure 5.9**. A comparison of this curve with the other network curves show how computational techniques compare with experimental techniques. Globally, looking at corrected curves, the computational predictions are better than the experimental predictions of EMAP8, SGA, and ER. Only EMAP7 can predict EMAP1 interactions better than the computational technique.

To confirm this result, we reproduced the same experiment with the SGA network, which is the largest network and covers all biological processes. Given its size, this network shares a non negligible number of pairs with the other networks (between 3% and 8% of pairs in every of them, as shown in **Table 5.2**). In our experiment, SGA interaction scores are used to rank the pairs that are common with SGA in each network separately and a precision-recall curve is computed using the binary labels predicted by the other networks as the true labels. Resulting AUPR scores are shown in **Table 5.7**. They are compared with AUPRs computed by cross-validation on pairs applied on each of the ten networks separately, i.e. areas under the curves in **Figure 5.9**.

These results confirm the results on EMAP1. Computational techniques give almost always better predictions than biological experiments and this is true for both kinds of interactions, positive and negative. Although our computational predictions are not perfect, this experiment nevertheless suggests that computational techniques can reach the same level of accuracy as experimental techniques. Note however that we are only comparing experimental techniques with $LS \times LS$ predictions. $LS \times TS$ and $TS \times TS$ predictions are typically less good as shown for example in **Table 5.6**.

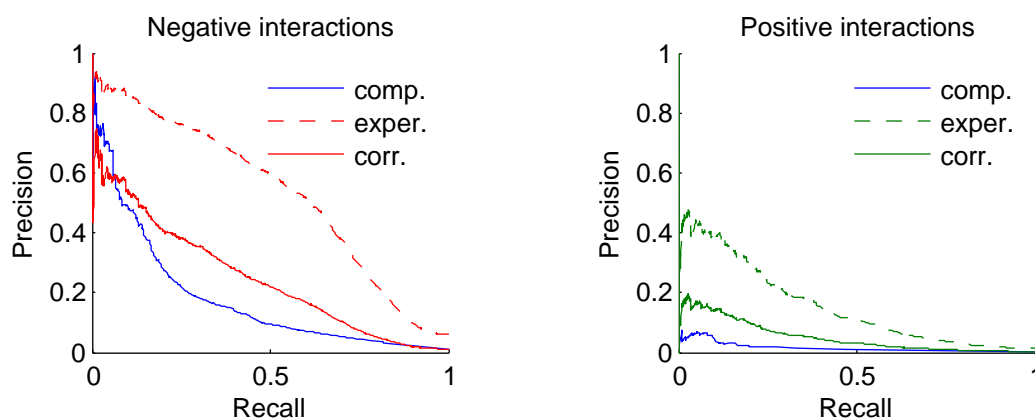


Figure 5.13: Precision-recall curves when we train a model on the ten oldest subnetworks, and we predict the pairs of the newest subnetwork (blue curves). These curves must be compared to those obtained when we compare two experimental techniques (red and green curves).

Comparison with the prediction of the newest subnetwork from the oldest ones

As explained in Section 5.3.1, genes related to a given EMAP are used to share a common function. So when we perform a cross-validation across a given EMAP, we evaluate the ability of the method to predict interactions involving genes that share the same function than the genes in the learning set. To represent a more realistic problem, i.e. to evaluate the prediction of interactions involving genes that do not share the function of the genes in a training subnetwork, we propose to learn a model using the ten oldest subnetwork and to evaluate its predictive performance using the interactions in the newest subnetwork (i.e. EMAP8). The method used here is the local approach in which we add GI profiles to the chemogenomic feature set.

The pairs of EMAP8 are divided into two subsets:

1. the 38,860 pairs that are also present in the ten oldest subnetworks (subset A),
2. and the 46,952 pairs that have never been measured (subset B).

We use the subset A to compare two experimental techniques applied on the same pairs. The precision-recall curves obtained for negative and positive interactions (dotted red and green curves in **Figure 5.13**) are used as a baseline with which we compare the performance of the predictions done on the subset B (blue curves). For these curves to be strictly comparable, we also corrected them to take into account the difference in the proportion of interactions in each of the tested sets of pairs, by using the formula presented in Section 3.2.4. The baseline curves become then the red and green full curves.

Now, computational techniques do not outperform experimental techniques anymore. They remain however close to each other, while computational techniques are obviously much faster and less expensive.

Comparison with predictions on the exact same pairs.

A limitation of the last experiment is that predictive performance is compared over two completely distinguished sets of pairs. As an additional experiment, we trained a model on the ten oldest

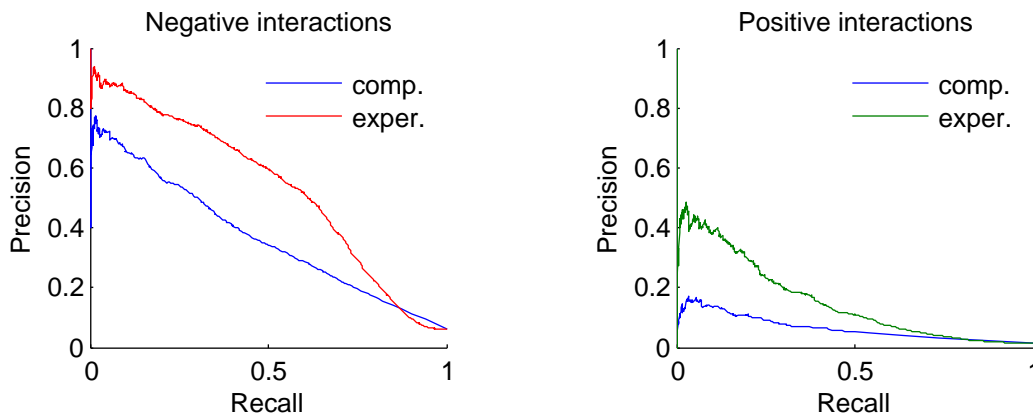


Figure 5.14: We did the same experiment as the one performed in **Figure 5.14**, but this time removing the subset A (see text) from the training set and using this precise subset as the test set to compute the PR curve.

subnetworks, but removing from them the pairs belonging to the subset A. We then trained the model on the subset A, and obtain the blue curves in **Figure 5.14**. We can compare these curves with the baseline which is computed on the same subset A, as in the previous experiment (red and green curves). Experimental techniques perform better than computational techniques to predict new interactions, but there are nevertheless again rather close to each other.

5.4.3 Cross-validation across the ensemble of all networks

In this section, we put the eleven networks together to form a global consensus genetic interaction network, and perform cross-validations on pairs and on genes on this network with various input feature sets. Our goal is two-fold: first, to confirm the main conclusions drawn in the previous section from experiments on individual subnetworks and, second, to assess the performance we can expect when making predictions with a model trained on all known interactions, as we will do it in Section 5.5.

Construction of the consensus network. As explained in Section 5.4.2, some pairs are shared by several networks, and they are not always labeled with the same values in every of them. A label will be attributed to such a pair according to the following rule. If the pair is labeled as negative in at least one of the networks without being labeled as positive in another one, it will be labeled as negative in the final network. In the same way, if the pair is labeled as positive in at least one of the networks without being labeled as negative in another one, it will be labeled as positive in the final network. Finally, if a pair is labeled as negative in a network and positive in another, it will not be labeled in the final network and will be treated as a missing value. The global genetic interaction network contains finally 90,379 negative interactions (4.64% of all pairs) and 27,659 positive interactions (1.42% of all pairs) between 5252 genes.

Cross-validation on pairs with different input sets. We first performed 10-fold cross-validation on pairs on this global network, using as inputs each of the 22 feature sets in turn. Area under precision-recall curves are presented in **Figure 5.15** for all feature sets and for negative and positive interactions. For negative interactions, feature sets that are more predictive than the baselines

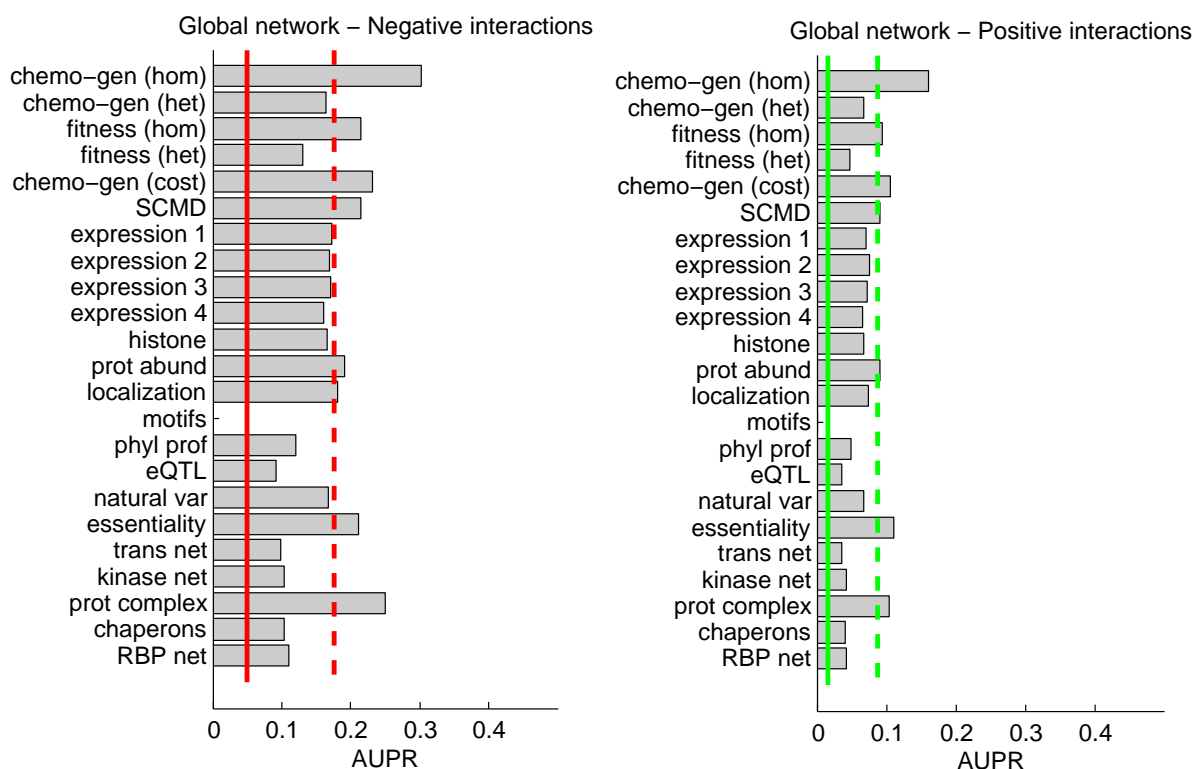


Figure 5.15: Area under the precision-recall curves (AUPRC) as a function of the 23 input datasets, for $LS \times LS$ predictions in the ensemble of all networks.

are homozygous chemo-genomic, homozygous fitness, chemo-genomic (Costanzo), SCMD, protein abundance, localization, essentiality, and protein complex network. The good predictive performance of the protein complex network suggests that genetic interactions tend to appear between proteins belonging to a same complex. For positive interactions, feature sets that are more predictive than the second baseline are the same sets, except localization and essentiality. In a second step, we tried to combine these input datasets to check whether it is possible to improve performance with respect to the use of only the most predictive feature set (homozygous chemogenomic). Like in Section 5.4.1, we filled in missing values by using the average over non-missing values for numerical features and zeroes for binary features. Like in previous experiments, the precision-recall curves in **Figure 5.16** show that we can not get any improvement by combining several feature sets.

As illustrated in Section 4.5, one advantage of tree-based ensemble methods like Extra-Trees is their ability to provide a measure of importance for each input variable. With the local approach, we computed the importance of each feature in the homozygous chemo-genomic input set from the ensemble of trees related to each gene. We then averaged all these importances for each feature over all ensembles to finally get one importance value for each feature. Importance values for all features, ranked from the most to the less important, are plotted in **Figure 5.17** for positive and negative interactions. Importance scores are smoothly decreasing from the most important to the less important feature and no group of features stands out from the crowd. This suggests that the information is spread rather uniformly over all features. The 10 most important features, both for positive and negative interactions, are nevertheless reported in **Table 5.8** for information purpose.

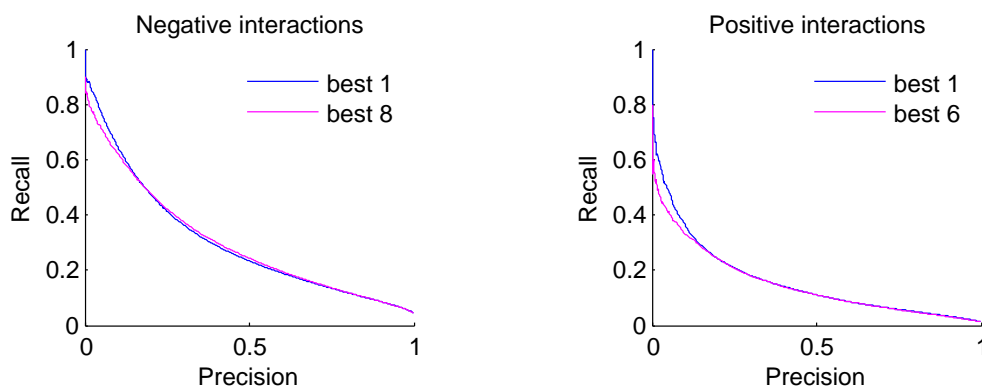


Figure 5.16: Precision-recall curves obtained by cross-validation on pairs from the ensemble of all networks, for predicting negative and positive interactions. Blue curves are obtained when using homozygous chemo-genomic input dataset (the best predictive one) and magenta curves are obtained when combined all input datasets that perform better than the second baseline. Not improvement happened when adding some input set to the most predictive one.

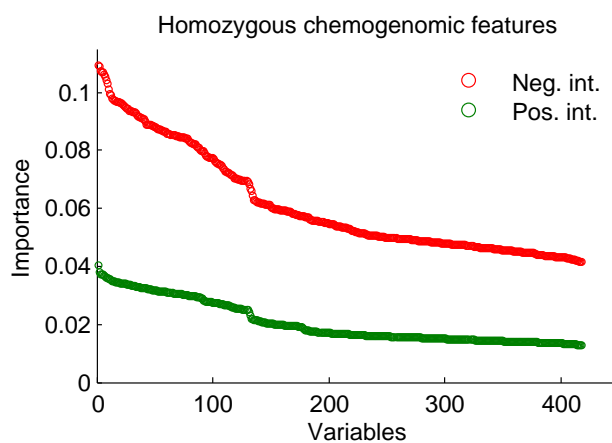


Figure 5.17: We computed the importance of each variable of the chemogenomic feature set with the local approach. No group of variables seem to distinguish clearly from the others, showing that all conditions in which the fitnesses of the mutant cells are measured may be important to predict genetic interactions.

Table 5.8: Variables of chemogenomic dataset represent conditions in which the growth fitness of the gene deletion collection is measured. These conditions are the presence of chemicals or environmental stress conditions. Here are presented the ten conditions that are the most important to predict negative and positive interactions.

Negative interactions Top 10 chemogenomic variables	Positive interactions Top 10 chemogenomic variables
bleomycin: 1.13 $\mu\text{g}/\text{ml}$	myriocin: 0.2 $\mu\text{g}/\text{ml}$
papuamide B: 0.7 $\mu\text{g}/\text{ml}$	04 08 04 02 minimal media
bleomycin: 1.13 $\mu\text{g}/\text{ml}$	03 06 06 01 minimal media
04 05 18 02 bleomycin: 1.7 $\mu\text{g}/\text{ml}$	latrunculin: 3 μm
04 05 18 01 bleomycin: 1.7 $\mu\text{g}/\text{ml}$	LiCl: 100 mm FK506: 1 $\mu\text{g}/\text{ml}$
myriocin: 0.2 $\mu\text{g}/\text{ml}$	cantharidin: 100 μm
latrunculin: 0.78 μm	benzaldehyde: 0.003%
pH7.5 FK506: 1 $\mu\text{g}/\text{ml}$	03 06 06 06 minimal media
LiCl: 100 mm FK506: 1 $\mu\text{g}/\text{ml}$	nocodazole: 10 $\mu\text{g}/\text{ml}$
latrunculin: 3 μm	bleomycin: 1.13 $\mu\text{g}/\text{ml}$

Cross-validation on genes. We then performed cross-validations on pairs and on genes on this global network, to estimate the quality of the $LS \times LS$, $LS \times TS$, and $TS \times TS$ predictions (we will exploit these results in Section 5.5). We use as inputs the homozygous chemogenomic dataset (which reduced the size of the global network to 4689×4689 because of missing genes in this input dataset) and we restricted our experiment to the local approach. This approach gave previously less good results than the global one, but the latter approach needs too much memory space to be applied on this network due to its size. To improve $LS \times LS$ predictions, we nevertheless added the genetic interaction profile of each gene among the inputs, as explained in Section 5.4.1.

Figure 5.18 presents the resulting precision-recall curves for negative and positive interactions. The curves can be compared with those obtained with degree prediction for $LS \times LS$ and $LS \times TS$, and can be compared to those obtained with random prediction for $TS \times TS$. Our results are always significantly better than baselines, proving that our method can learn from the chemogenomic input dataset. GI profiles significantly improve performance of $LS \times LS$ predictions, even more so than what we already observed on EMAP1. This may come from the fact that GI profiles are here much larger, and therefore potentially more informative, in the consensus network than in the single EMAP1.

$LS \times LS$ graphs illustrate the performance that we can expect for the prediction of the ~ 8.5 millions pairs that are still unlabeled between the 4473 genes represented in the eleven networks. $LS \times TS$ graphs illustrate the performance we can expect for the prediction of the ~ 1 million pairs that are still unlabeled between the 4473 genes represented in the eleven networks and the 216 genes that are not represented. $TS \times TS$ graphs illustrate the performance that we can expect when predicting the 23,220 pairs that are still unlabeled between the 216 genes not represented in the eleven networks. These three groups of pairs and their relative sizes are illustrated in **Figure 5.19**. Almost 90% of the unlabeled pairs belong to the $LS \times LS$ family.

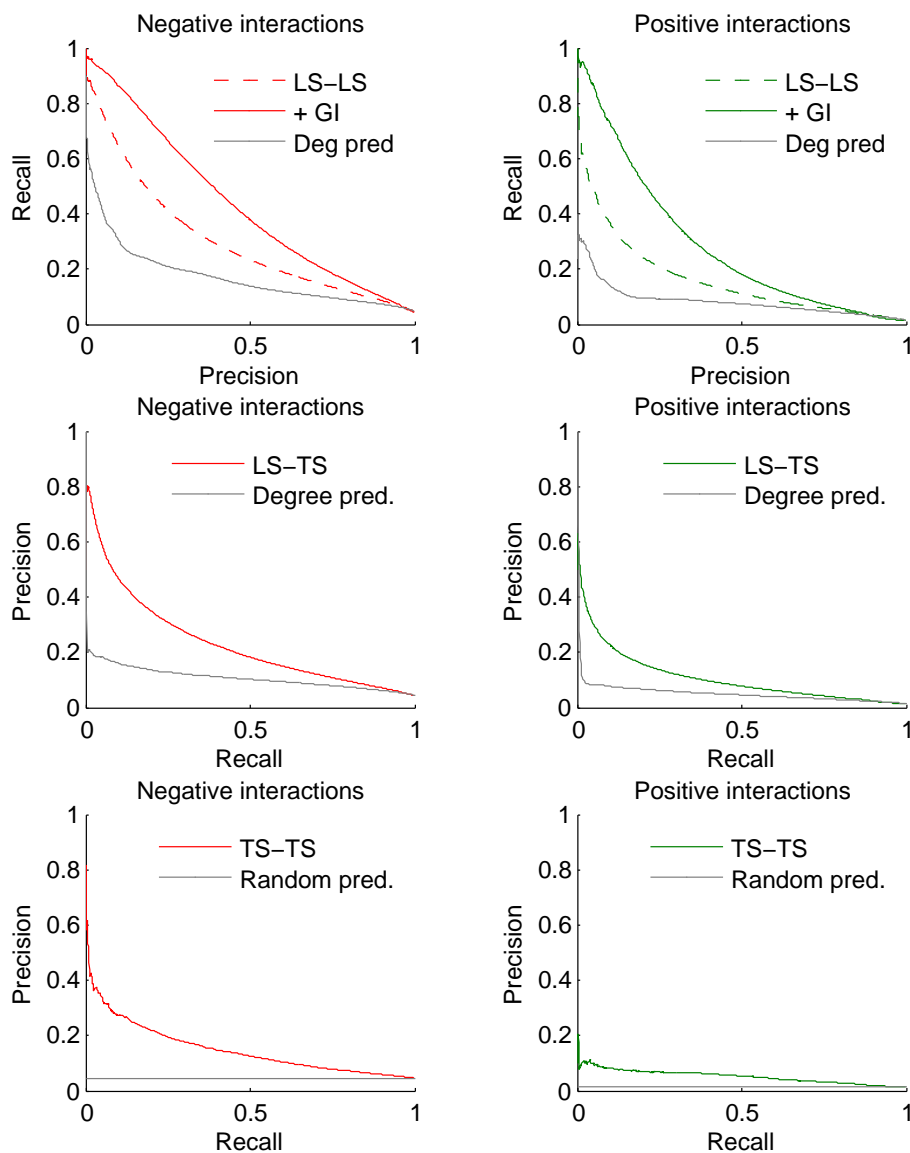


Figure 5.18: Precision-recall curves obtained by cross-validation on pairs and on genes on a global network that groups the eleven single networks. We used Extra-Trees with a local approach. Baselines for $LS \times LS$ and $LS \times TS$ are predictions from node degrees, and baseline for $TS \times TS$ is random prediction.

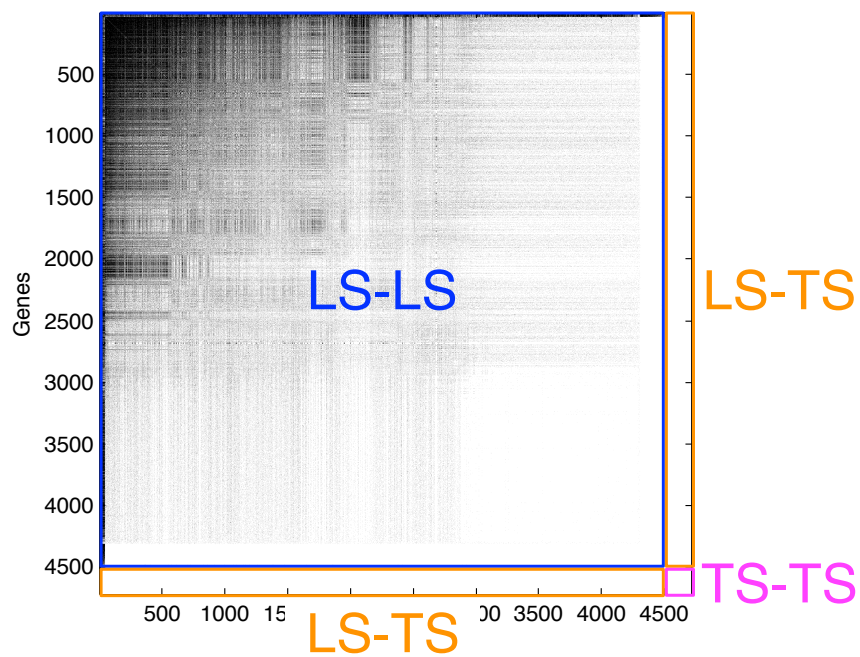


Figure 5.19: The unlabeled pairs of the adjacency matrix of the global network (white areas) are divided into three groups: $LS \times LS$ pairs between two known genes, $LS \times TS$ pairs between one known gene and one unknown gene and $TS \times TS$ pairs between two unknown genes. The first group covers almost 90% of the unlabeled pairs.

5.5 Predictions of unlabeled pairs

The primary goal of the application of supervised inference methods is to predict new interactions. We demonstrated by cross-validation in Section 5.4.3 that the method is able to predict new genetic interactions significantly better than a simple approach that would give predictions proportional to the node degrees. For the $LS \times LS$ pairs that form largely the majority in the global network, one might even expect better predictions than by using experimental techniques as shown in Section 5.4.2. Supervised methods are thus clearly able to extract useful information about genetic interactions from the chemogenomic input dataset. In this section, we first discuss how to use lessons learned in previous sections (and chapters) to obtain a final ranking of all unlabeled gene pairs in yeast from the most likely to the less likely to (positively or negatively) interact. In the absence of an experimental validation of this ranking, we then perform an independent validation by confronting this ranking with functional annotation of yeast genes available in the Gene Ontology.

5.5.1 Global ranking of the unlabeled pairs

To predict positive and negative interactions among gene pairs that are still unlabeled, we learnt a model on pairs that were already labeled in one of the eleven networks, i.e. the consensus network that was used in the cross-validation experiments of Section 5.4.3. The input dataset used in this experiment is the homozygous chemogenomic set that led to the best performance overall. The local approach was chosen because the global one leads to memory problems due to the size of the learning set. We also added GI profiles to the features of the chemogenomic dataset because as shown in **Figure 5.18**, it significantly improves $LS \times LS$ predictions. Two models are trained, one for positive interactions and one for negative interactions.

With the resulting models, one eventually can get two numerical predictions for each unlabeled pair, which estimate the probability for this pair to correspond to a positive and a negative interactions respectively. All unlabeled pairs can then be ranked from the most likely to the less likely to positively or negatively interact directly using these two predictions. From Section 3.6, we know however that a better ranking, in terms of precision-recall curve, could be obtained by taking into account the difference in performance between the three families of pairs, $LS \times LS$, $LS \times TS$, and $TS \times TS$. Indeed, for example, a $TS \times TS$ pair receiving a prediction of 0.8 must not necessarily be ranked higher than a $LS \times LS$ pair receiving a prediction of 0.7 given that the $LS \times LS$ precision-recall curve dominates the $TS \times TS$ curve (see **Figure 5.18**). To obtain our final ranking, we therefore applied the method developed in Section 3.6 that gives the ranking associated to the best precision-recall curve. The (estimated) precision-recall curves of the two merged rankings are shown in **Figure 5.20**.

A prediction of the interaction network can then be obtained by selecting the pairs at the top of these rankings. Note that $LS \times TS$ pairs are only introduced in these rankings respectively at position 48210 for negative interactions (i.e. after 0.5% of all predicted pairs) and at position 21760 for positive interactions (i.e. after 0.2% of all predicted pairs). As an illustration, the ten top pairs in the optimal rankings for positive and negative interactions are shown in **Table 5.9** for the predictions using only the chemogenomic feature set, and in **Table 5.10** when we added the GI profiles for $LS \times LS$ predictions. There is no common pairs in the top 10 between **Table 5.9** and **Table 5.10** but the top 100 pairs however share 12 pairs for negative interactions and 2 pairs for positive interactions, the top 1000 pairs share 20% of the pairs for negative interactions and 5%

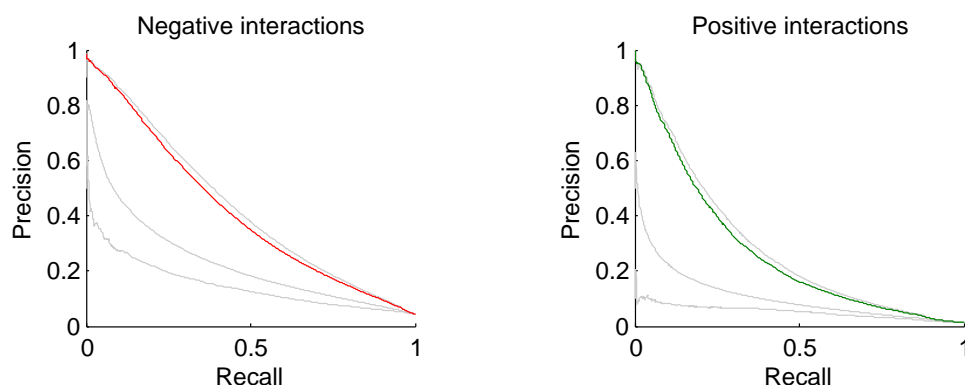


Figure 5.20: Red and green curves are respectively the estimated precision-recall curves of the two merged rankings of negative and positive interactions. Their AUC are equal to 0.41 and 0.27. They are computed from the six gray curves that estimate the performance of $LS \times LS$, $LS \times TS$ and $TS \times TS$ (respectively from the highest to the lowest curves). These gray curves correspond to those found in **Figure 5.18**.

for positive interactions and the top 10,000 pairs share 26% for negative interactions and 18% for positive interactions.

5.5.2 Validation with Gene Ontology

With cross-validation, we can evaluate the quality of the predictions provided for the unlabeled pairs by our supervised learning models. However, by splitting the training set into random folds, cross-validation assumes that the unlabeled pairs are distributed similarly as the training pairs. In case of violation of this assumption, cross-validation error estimates might not reflect faithfully the true performance of the method. It is thus always interesting to validate the predictions by other means. In this Section, we propose to validate our predictions but exploiting the fact that interacting genes are often involved into the same biological processes (Baryshnikova *et al.*, 2010). The top pairs of our ranking should thus be enriched in pairs of genes that share similar functions and the level of this enrichment can then be used as a measure of the quality of the ranking. Before computing this enrichment, we first describe the Gene Ontology, which is used as reference database of gene functional annotations.

Gene Ontology Gene Ontology (GO) (Ashburner *et al.*, 2000) is a bioinformatics project, the goal of which is to structure the description of genes and their product. GO is not limited to *S.cerevisiae*, but is common to a wide variety of species. Indeed biological role of a protein in one organism can often be transferred to other organisms.

The GO database is composed of terms annotating the behavior of gene products. These terms are divided into three groups: *cellular component* refers to the place in the cell where a gene product is active, *biological process* refers to a biological objective to which a gene product contributes and *molecular function* refers to the biochemical activity of a gene product. These three terms (parent terms) represent general concepts and are divided into terms representing more specific concepts (children terms). These terms are themselves divided into more and more specific terms. They are structured into a directed acyclic graph, which forms a kind of hierarchy. For example, the term

Table 5.9: From all the known genetic interactions, we built a model to predict new interactions. The list of the 10 pairs with the highest probability to interact is presented for negative and positive interactions.

Top 10 pairs predicted negative		Top 10 pairs predicted positive	
ARL3	VPS63	MSE1	MAK10
YNL198C	CLC1	RPS17A	MAK10
VPS51	ARL1	MSE1	ATP7
VPS63	ERV14	UAF30	MAK10
CTK1	BCY1	YNL170W	MRPL24
IRS4	SYS1	ATP7	GLO3
YJL024C	SYS1	MRPL24	MAK10
SYS1	ANP1	RPS17A	YKL053W
VPS63	IRS4	MSE1	YKL053W
CTK1	CLC	YNL170W	PPA2

Table 5.10: As in **Table 5.9** we built a model to predict new interactions, but adding GI profiles to the chemogenomic feature set.

Top 10 pairs predicted negative		Top 10 pairs predicted positive	
COG8	VPS1	PEX3	PEX10
RAV2	RGP1	MGA2	LOC1
JNM1	CPR6	RPS17A	MGA2
RAD50	NUP84	QRI8	RPS17A
ASE1	ARP1	HLJ1	RPS11A
HOC1	OPI3	SPT8	RIC1
ARL1	VMA2	LOC1	RPN4
YME1	RPO41	GEF1	AIR1
MDM39	COG7	CUE1	RPS17A
RGP1	TFP1	RPS17A	SEC22

"cellular component" is the parent term of the "extracellular" and "intracellular" terms. This latter term has children terms like "nucleus" or "chromosome".

Yeast genes have been annotated by GO terms. From this annotation, one can define that two genes are functionally similar if they share some common terms in a reference list of GO terms. For the similarity measure to be relevant, this reference list should however contain neither too general nor too specific terms. Myers *et al.* (2006) proposed a list of GO terms established by experts that can be used to measure functional similarity between genes. From this list, one can derive a graph where each node corresponds to a gene and each edge connects two genes that share a common GO term from the reference list. For yeast, this graph contains 513,562 edges relative to 3970 genes, which means that 2.28% of all possible gene pairs share a function.

Functional enrichment of labeled pairs. Before assessing our predictions, we first analyzed the labeled pairs in the eleven subnetworks from the point of view of their functional enrichment. Among all available labeled pairs, 6.63% actually share a GO term from the reference list. Given that 2.28% of all pairs of genes share a GO term, this clearly indicates that pairs in the eleven datasets were not chosen at random. Indeed, if the 3,893,796 labeled pairs would have been selected at random, the probability to observe a proportion of pairs with similar GO terms equal to 6.63% or higher would have been less than 10^{-16} . This result is not surprising since for example all tested genes in the EMAPs have been precisely selected on the basis of their function. Among the labeled pairs, a proportion of 11.31 % of the positive interactions and 12.12% of the negative interactions appear to share a common GO term. The corresponding p -values (measuring the probability of observing higher or equal proportions if the positive and negative pairs would have been selected randomly among all labeled pairs) are both less than 10^{-16} . This indeed confirms that interacting genes are more likely to share a function than random genes.

To get a finer picture, we modified and reproduced an experiment carried out in (Baryshnikova *et al.*, 2010) to validate the SGA dataset. We ranked the labeled pairs according to their interaction scores (in normal order for the positive interactions and in reverse order for the negative interactions) and computed a precision-RPP curve from the resulting ranking by considering a pair as positive if its genes share a GO term and as negative otherwise. RPP is the rate positive prediction and we chose this metric instead of the recall because we found it more interpretable in this situation (see the next paragraph). Since the genetic interaction scores are not comparable from one network to the other, we had to rescale them to produce the merged ranking of labeled pairs. For this rescaling, we took into account the thresholds used on each dataset to separate negative, neutral, and positive interactions, i.e., -2.5 and 2 for the EMAPs, -0.12 and 0.16 for SGA, -1 and 1 for GIM, and -0.75 and 0.75 for ER. Interactions scores for the EMAP were left unchanged while negative scores for SGA, GIM, and ER were respectively multiplied by $2.5/0.12$, 2.5 and $2.5/0.75$, and positive scores were respectively multiplied by $2/0.12$, 2 and $2/0.75$. The resulting precision-recall curves are shown in **Figure 5.21**. Note that following Baryshnikova *et al.* (2010), we use a logarithmic scale for the x-axis to better emphasize the beginning of the curve.

From these curves, positive interactions appear to be less functionally related than negative interactions, or at least to involve relations more complex than those identified in GO. For negative interactions, the enrichment seems significant. For example, for a RPP equal to $1.7 \cdot 10^{-4}$, we got a precision equal to 0.5 (the blue dot in the figure). This means that among the 331 pairs with the lowest negative scores, 50% involve genes sharing a GO term, which has to be compared with 6.63% if these pairs were selected at random from the labeled pairs. This analysis is consistent with the

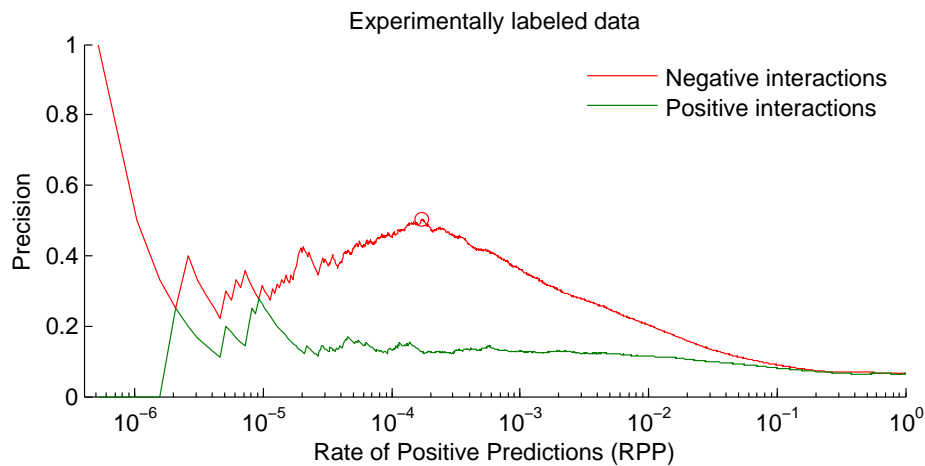


Figure 5.21: Pairs labeled in the eleven datasets were compared with a list of pairs sharing a same GO term (blue curves). GO terms were used as true values, and genetic interaction scores were used as predictions, to build precision-recall curves. These curves estimates the functional utility of true genetic interactions.

similar analysis done in (Baryshnikova *et al.*, 2010) on only the labeled pairs from the SGA dataset. For positive interactions, the enrichment is much less important and the precision-recall curve is very close to the random one, except at its very beginning.

Functional enrichment of predicted pairs. To assess the quality of our predictions for the unlabeled pairs, we plotted the same precision-RPP curve as before, this time for the unlabeled pairs ranked according to their predicted confidence score. We computed the precision-recall curves for both positive and negative interactions separately, and for the model trained with the GI interaction profile and without. The resulting four curves are shown in **Figure 5.22**.

As in the case of labeled pairs, negative interactions give better precision-RPP curves than positive interactions, with or without GI. Results with GI seems to be slightly better than results without GI. Precision-recall curves are better than random but the enrichment in GO terms is nevertheless much less important for our predictions than for the experimentally verified pairs. Note that the proportion of unlabeled pairs that share a GO term (1.57%) is lower than the proportion of labeled pairs that share a GO term (6.63%). The random baseline is therefore lower here than for the curve in **Figure 5.22**.

Now we make the assumption that the proportion of interacting pairs is the same in the labeled set than in the unlabeled set (i.e. equal to 4.64% for negative interactions and 1.42% for positive interactions). We then select, among the unlabeled pairs, the 861,350 with the highest probability to interact negatively and the 276,930 with the highest probability to interact positively, and consider them as interacting. 3.25% of these pairs predicted negative actually share a GO term, as well as 4.28% of the pairs predicted positive. The corresponding p -values if the negative and positive pairs would have been selected at random among unlabeled pairs are both less than 10^{-16} . This means that pairs predicted to interact are more likely to share a function than random unlabeled pairs.

Cleaning the experimental predictions. Results in Section 5.4.2 clearly show that experimental techniques are not perfect as they can produce incoherent scores when applied to the same pairs.

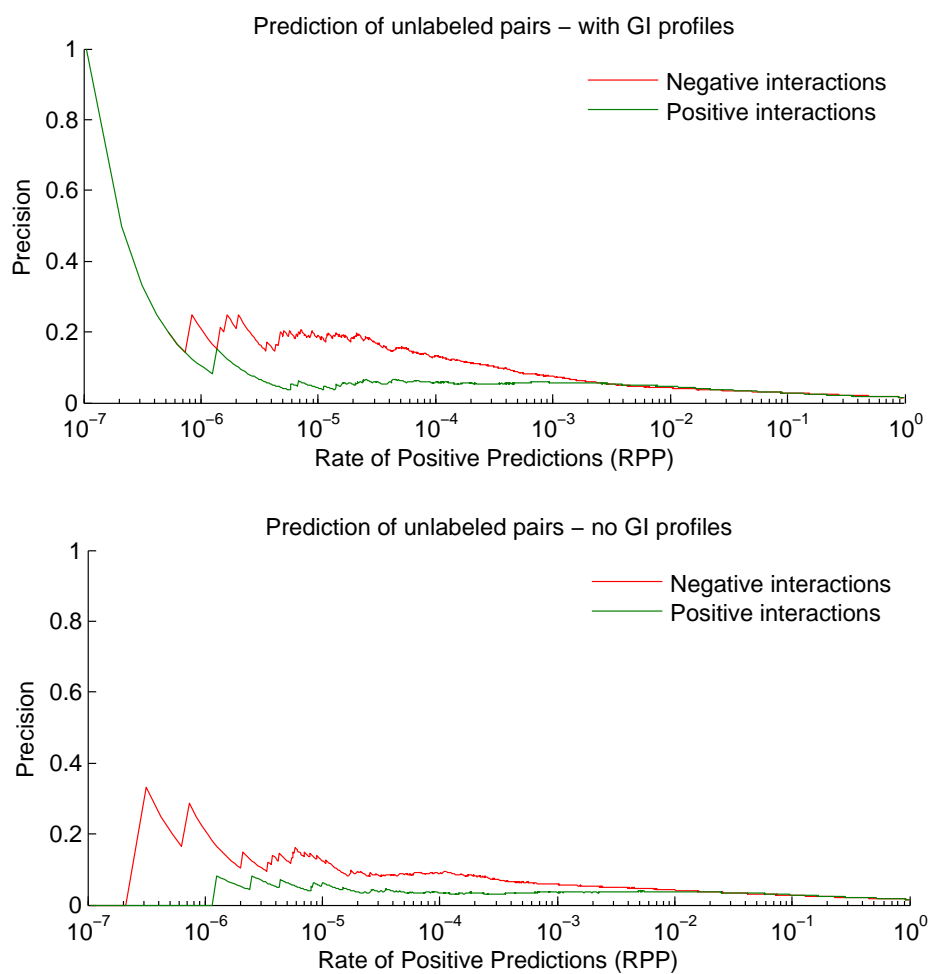


Figure 5.22: Unlabeled pairs, predicted by our approach, were compared with a list of pairs sharing a same GO term, in the same way as in **Figure 5.23**.

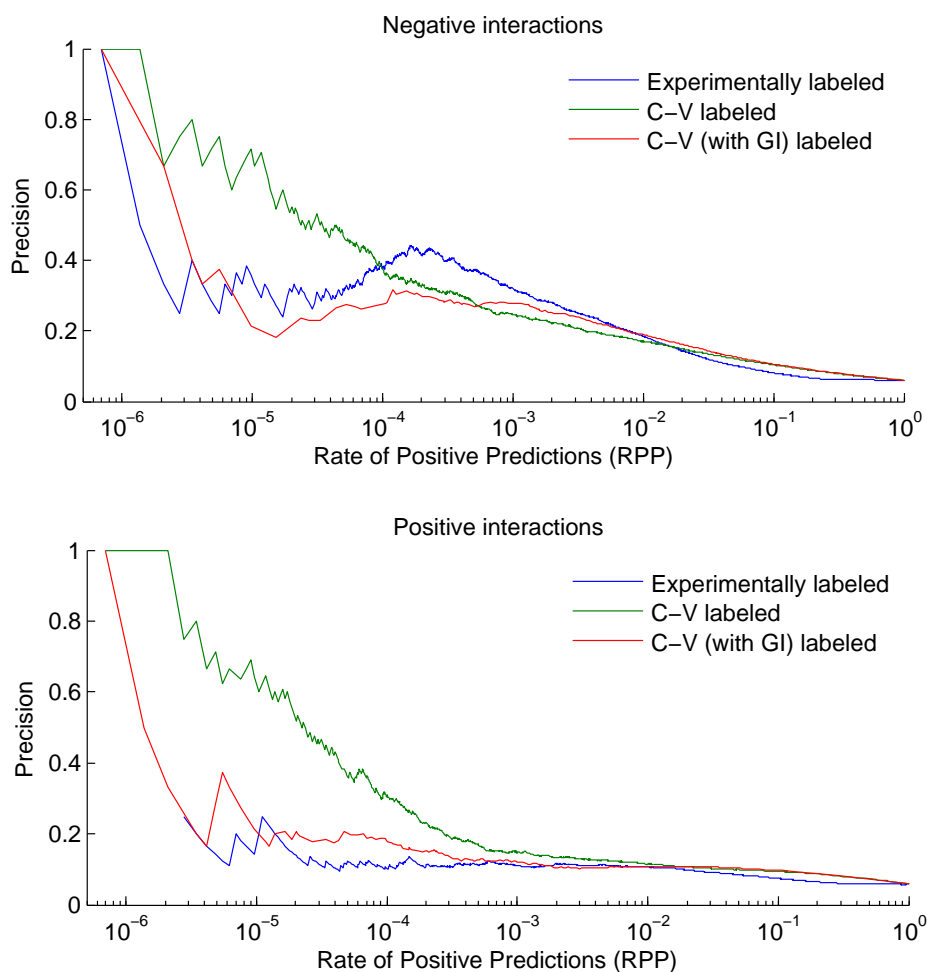


Figure 5.23: Pairs labeled in the eleven datasets were compared with a list of pairs sharing a same GO term (blue curves). The same experiment was performed on the exactly same group of pairs, but with predictions obtained by cross-validation without (green curves) and with (red curves) GI profiles added to the feature set.

In addition to providing predictions for unlabeled pairs, supervised techniques can also produce a relabeling of the labeled pairs through cross-validation: each pair in a fold gets a prediction from the model trained from the other folds. As a last experiment, we wanted to see whether cross-validation could be used to somehow clean the experimental predictions. To evaluate this, we reproduce in **Figure 5.23** the curves previously plotted in Figure 5.21 (becoming blue curves in this new Figure), to compare them with two new curves obtained by replacing the original (rescaled) interaction scores of the labeled pairs by the predicted confidence scores obtained by 10-fold cross-validation on pairs. The green curve is obtained using only the chemogenomic features as inputs while the red curves is obtained when adding the GI profiles.

For negative interactions, the CV score with GI (red curve) has the highest AUPR (8.87%), followed by the CV score without GI (green curve, 8.56%) and the original score (blue curve, 7.31%). For small recall values however, the CV score without GI clearly dominates the two other scores. Results are similar for positive interactions. the CV score without GI clearly dominates the two other scores. CV score with GI is less good but it also dominates the original score. These results show that cross-validation can indeed improve the functional enrichment among the top ranked pairs and

therefore suggest that it could also improve the prediction of interactions. This latter conclusion should however be taken with caution since two genes sharing a function do not necessarily interact and vice-versa. Supervised learning might introduce a bias towards functionally related genes for some reason, without necessarily improving the detection of genetic interactions. It is interesting to note that adding the GI profile seems to deteriorate the functional enrichment for small recalls, while it typically always improves the quality of the $LS \times LS$ predictions in our experiments. More work is needed to understand this contradiction.

5.6 Conclusion

In this chapter, our goal was to predict at best genetic interactions in yeast *S.cerevisiae* with supervised learning algorithms, and to evaluate the quality of our predictions. There is a genetic interaction between two genes if a mutation in one gene modifies the effect of a mutation in the other gene. Typically, genetic interactions are highlighted by the observation of some phenotype. In most networks presented in this chapter, the measured phenotype is the fitness of the cell. If the fitness resulting from the mutation of a gene g_a is aggravated (resp. improved) by a mutation on a gene g_b , there is a negative (resp. positive) interaction between g_a and g_b . If the fitness is unaffected, we say that the pair of genes is neutral. Biologists have measured several genetic interaction subnetworks in yeast, involving between 300 and more than 3000 genes each. We gathered all these subnetworks to constitute our training network. Together, these subnetworks totalize about 4 millions pairs that represent $\sim 10\%$ of all possible pairs in yeast. To use as input features for the inference, we collected 22 different sets of features defined on yeast genes, totalizing about 11000 individual features.

We first performed several cross-validation experiments on the eleven subnetworks using extremely randomized trees with both the local and the global approaches. The main conclusions of these experiments are as follows:

- In terms of methods, the global and the local approaches with trees are close to each other, with however an advantage to the global approach. The latter is however too computational demanding to be applied on the bigger subnetworks or on their union. A comparison of SVMs and extremely randomized trees in the context of the local approach shows that both methods are comparable.
- The 22 feature sets were evaluated individually and the best results (in terms of AUPR) are clearly obtained with the homozygous chemo-genomic dataset that measures the growth fitness of deletion strains in various stress conditions. Combining several feature sets does not significantly improve with respect to using the chemo-genomic dataset alone.
- Overall, our results show that we are able to predict new interactions with a reasonable accuracy (in all cases better than the baselines defined in Section 3.5). As expected, $LS \times LS$ pairs are the easiest to predict and the $TS \times TS$ pairs are the most difficult to predict. Positive interactions, which are less common than negative ones in most of the 11 subnetworks, are more difficult to predict than negative interactions.
- Adding the genetic interaction profile within the input features allow to improve quite significantly the performance when predicting $LS \times LS$ pairs.

- We observed similar trends on the 11 subnetworks (and when taking their union) but some subnetworks are nevertheless easier to predict than others.

Some pairs belong to several subnetworks and they are not necessarily labeled similarly in all these subnetworks due to technical variations in the experimental process. In a second step, we exploited these intersections to compare the quality of our computational predictions with the quality of the experimental predictions of one study compared with another on their intersection. Very interestingly, this comparison showed that in some settings computational techniques are competitive and sometimes even better than experimental techniques. This suggests that the performance of computational techniques is actually rather good.

Finally, we took the union of the eleven known subnetworks and we computed the predictions of a model trained from this union for all remaining unlabeled pairs. We then ranked them from the more to the less likely to (positively or negatively) interact as predicted by this model. We evaluated the quality of this global ranking by confronting it with functional annotations of yeast genes available in the Gene Ontology. More precisely, we compared the functional enrichment of gene pairs at the top of this ranking to the functional enrichment of gene pairs at the top of the ranking of the labeled pairs from the eleven subnetworks (again to compare computational and experimental predictions). We found out that top (positive and negative) predictions are indeed significantly enriched in pairs of gene sharing some function, with a stronger enrichment in the case of negative pairs. Experimental predictions are however more enriched than computational predictions. Interestingly, we also showed that re-predicting experimental predictions using cross-validation improved the level of functional enrichment among the labeled pairs.

Overall, we believe our results are quite good, especially given the apparent high noise in the experimental techniques from which we obtained our training network. An obvious next step from a biological point of view would be now to experimentally validate the most confident predicted interactions to confirm their relevance and the precision values estimated by cross-validation. There remain also several possibilities to improve the quality of our predictions. Globally, positive interactions seem to be more difficult to predict. This can be explained partly by the smaller number of examples of such interactions in the training network. Also, the biological factors that underly positive interactions have not been the subject of as many systematic studies as those behind negative interactions (Ryan *et al.*, 2010). A better understanding of these factors could help improving their predictions and can be an interesting future work direction. Although the global approach gave better results than the local one, we could not use it on the complete training network of 4 millions pairs due to computer memory problems. It would be interesting to work on the implementation of this approach to make it applicable on the complete set of labeled pairs. It would be also interesting to compare the results we obtained with our supervised learning approach with those obtained by Eronen *et al.* (2010); Linden *et al.* (2011) using matrix approximation models, and maybe to combine these methods with ours.

Chapter 6

Conclusion

6.1 Main findings and conclusions

Networks are ubiquitous in biology. They are very important to represent and understand all cellular activities and their elucidation is one of the main challenges of systems biology. The work presented in this manuscript is related to the problem of the supervised inference of biological networks, i.e., the computational inference of these networks by applying supervised machine learning methods on a training sample of known interacting and non-interacting pairs. We structured our contributions on this topic around three main questions:

- How to **evaluate** supervised network inference methods in a fair and unbiased way and, as a corollary, how to best apply them to truly infer a real network?
- How to best exploit **tree-based ensemble methods** for supervised network inference and how do these methods compare with existing methods from the literature?
- How well can we predict **genetic interactions in yeast** using supervised network inference methods?

These three questions were addressed respectively in Chapters 3, 4, and 5 of this manuscript. Our main findings about these three questions are briefly summarized below. We refer the reader to the conclusions of the separate chapters for more detailed discussions.

Performance evaluation. We performed in Chapter 3 an in-depth and comprehensive examination and analysis of measures and protocols to evaluate the performance of supervised network inference methods and models. We first reviewed metrics that have been used in the literature to quantify the quality of the predictions provided by inference methods. We concluded from this review that precision-recall curves are more appropriate than ROC curves given the very sparse nature of most biological networks. PR curves can nevertheless be usefully complemented by other measures depending on the targeted application. We then showed that when estimating these measures by cross-validation, it is important to differentiate the tested pairs according to the presence or not of its two nodes in the learning set, as this presence has an important impact on generalization performance. This differentiation defines four groups of pairs, called respectively $LS \times LS$, $LS \times TS$ or $TS \times LS$, and $TS \times TS$, and we argued that a separate evaluation should be performed for each of these four groups if one wants a detailed assessment of the performance of an inference method.

We showed that pairs from the $LS \times LS$ group can be assessed using a cross-validation over the pairs in the training data, while pairs from the three other groups can be assessed using a cross-validation over the nodes.

We discussed two other issues raised during the evaluation of inference methods that are linked to specificities of biological networks. First, the distribution of node degrees in biological networks often has a heavy tail, which means that several nodes have a degree much higher than the average. We showed that because of this property, it is possible to obtain better than random predictions without exploiting the node input features. We therefore suggested two simple baselines to replace random guesses when evaluating network inference methods: a classifier built from randomly permuted input features and a classifier that scores each pair by the sum of the degrees of its two nodes in the training network. A second issue in several biological networks is the lack of experimentally verified non-interacting pairs. We showed that the common solution, i.e. considering all non positive pairs as negative pairs, leads to an underestimation of the PR curve. Under certain conditions however, it is possible to correct PR curves to avoid such underestimation.

Finally, we proposed a simple algorithm to merge rankings of disjoint sets of pairs, when each of these rankings is associated with a separate precision-recall curve. When applied to merge the individual rankings corresponding to the four groups of pairs defined earlier, this algorithm is shown, theoretically and empirically, to produce a better global ranking of all pairs, in terms of the resulting PR curve, than a naive merging of these rankings according to the confidence score predicted for each pair.

Overall, we believe that following the guidelines provided in this chapter will enable a more rigorous assessment of supervised inference methods, will contribute to an improved comparability of the different approaches in this field, and will thus furthermore aid researchers in improving the state-of-the-art methods.

Supervised network inference with tree-based methods. In Chapter 4, we systematically investigated the exploitation of tree-based methods for supervised network inference. The main goal of this investigation was to understand the advantages and limitations of these methods when applied for network inference. We considered the application of these methods both within the global and local approaches, where the latter was extended in the process to make $TS \times TS$ predictions and to handle multi-output methods. Through extensive experiments on 10 different biological networks, we showed that the global, local single output and local multi-output approaches with trees are very close to each other in terms of predictive performance. They are also very competitive with existing approaches from the literature. All approaches have the same computational complexity but the local multi-output method nevertheless provides less complex models and require less memory at training time.

Finally, we studied the interpretability of these methods. We first showed that both the global and local multi-output methods with single trees partitions the adjacency matrix into rectangular subregions of highly or weakly connected pairs, where each subregion is defined by a conjunction of tests based on the input features. The partition takes the form of a checkerboard in the case of the local approach and is unconstrained in the case of the global approach. With tree ensembles, we showed how to obtain a similar partition of the adjacency matrix by exploiting a similarity measure derived from the ensemble. In this case, the subregions are described by a ranked list of features.

Overall, we believe results in this chapter clearly show the relevance and competitiveness of tree-based methods for network inference. They compare favorably with other methods on a wide range of

networks, they are essentially parameter-free (no parameters were actually tuned in our experiments), and they can be exploited in various ways to provide interpretable results.

Predicting genetic interactions in Yeast. In Chapter 5, we built on the experience gained in previous chapters to try to predict at best the genetic interaction network in yeast *S.cerevisiae*. For that purpose, we collected a large dataset, assembling 4 millions gene pairs experimentally tested in the context of 11 different studies and 23 sets of measurements to use as gene input features for the inference. Through several cross-validation experiments, we showed that it was indeed possible to predict genetic interactions with significantly better performance than with the baselines. We observed that negative interactions are easier to predict than positive ones and that, as expected, $LS \times LS$ pairs are better predicted than $TS \times TS$ pairs. We also found that the homozygous chemo-genomic feature set, that quantifies the growth fitness of deletion strains in the presence of stress conditions, was by far the most informative feature set. By exploiting the overlaps between the 11 subnetworks of experimentally tested pairs, we were able to compare the predictions of one experimental technique against another (that might differ due to technical variations in the experimental process). Very interestingly, we concluded from this comparison that some experimental techniques do not provide better predictions, when compared to another, than our computational network inference approach, when compared to the same technique, in particular for the $LS \times LS$ pairs.

Finally, we trained a model using the full training set of 4 millions pairs and predicted a confidence score for positive and negative interactions for each of the remaining unlabeled gene pairs in Yeast. As an indirect validation of this ranking, we showed that the pairs of genes with the highest predicted probability to interact are more likely to share a common function (as derived from gene annotation in the Gene Ontology) than randomly selected gene pairs.

Overall, the experiments carried out in this chapter show that predicting genetic interactions in Yeast is indeed possible to some extent and also suggest that the accuracy of computational methods is not very far from the accuracy of experimental techniques.

6.2 Limitations and future research directions

There are several limitations to our research that open interesting directions for future research. We first discuss potential extensions of network inference methods and then open questions more related to biological applications.

6.2.1 Methods

The local approach offers the most interesting directions for future developments. A priori, this approach was not adapted to predict edges between two unseen nodes (i.e., $TS \times TS$ predictions) because no model can be trained for any of the two nodes. We made this adaptation possible by proposing to learn a second model on the predictions obtained from a first model. Despite the fact that the second model is trained on potentially inaccurate predictions, this two-step procedure brings the local approach to the same performance as the global approach for $TS \times TS$ predictions. It would be now interesting to more deeply analyze this approach and also to apply it in combination with other supervised learning methods. Recently, Pahikkala *et al.* (2014a) proposed a very similar two-step procedure that they applied with kernel methods. In this context, they were able to show

a strong relationship between the two-step local approach and the global approach using the tensor product pairwise kernel (see Equation 2.22). It would be interesting to find a similar relationship between the global and the local approaches with trees, although given the non-parametric nature of tree-based models, finding such connection might turn out to be much more difficult. The application of multi-output trees in the local approach is very successful, as it reduces model size without any loss in accuracy. Similarly, it would be very interesting to integrate other more sophisticated multi-output or multi-label methods (Tsoumakas and Katakis, 2007) within the local approach. To the best of our knowledge, these methods have not been used for biological network inference.

We saw in Section 2.3.2 that there exist several elegant ways to design (implicit) vectorial representations for pairs of nodes, through kernels. In our experiments with the global approach, we only used as inputs for a pair the concatenation of the feature vectors of the two nodes in the pair. While this simple representation is inappropriate in the case of linear models (see Section 2.3.2), it worked well in our experiments with tree-based methods that train non-linear models. It would be nevertheless interesting to explore (ideally automatic) ways to incorporate more complex features on pairs in tree-based methods. Also concerning input features, we showed in Chapter 5 that adding the genetic interaction profiles among the input features significantly improved the quality of $LS \times LS$ predictions. It should be interesting to investigate more systematically this idea in the context of the networks of Chapter 4. Note that incorporating outputs among the inputs is reminiscent of some multi-label approaches such as classifier chains (Read *et al.*, 2009), that could be also explored here.

Most biological network inference problems are intrinsically semi-supervised, or even transductive, problems, since features values are known for all nodes involved in the unlabeled pair during the training stage. Following some work with kernel methods (Brouard *et al.*, 2011), it would be interesting to design semi-supervised approaches for network inference based on tree methods, in order to exploit all available data. Semi-supervised extensions of tree-based methods are however very sparse in the literature (Leistner *et al.*, 2009).

In the case of genetic interactions, experimental techniques allow to really detect non-interacting (neutral) pairs. In most of the networks considered in Chapter 4, we only have positive and unlabeled pairs. We got around this problem by considering (wrongly) all the unlabeled pairs as negative examples and discussed in Section 3.4 the consequences of this choice. Nevertheless, several methods have been proposed in the literature to take into account the lack of negative examples (Ben-Hur and Noble, 2006; Elkan and Noto, 2008; Geurts, 2011; Mordelet and Vert, 2013), and in particular there exist adaptations of tree-based methods to this setting (Denis *et al.*, 2005b). It would be interesting to consider the application of these methods in the context of network inference.

6.2.2 Biological applications

This thesis presents an engineer's perspective on the problem of biological network inference and it therefore focused more on methodological aspects. Several problems highlighted throughout the thesis would deserve to be analyzed more thoroughly from a biologist's perspective.

We illustrated the interpretability of trees on a drug-protein interaction network in Section 4.5. We have shown that tree-based methods define clusters of drugs and proteins with either many or few connections between them and that each cluster is furthermore characterized by specific lists of features on drugs and on proteins. It would be very important in the future to analyze the relevance of these lists from a biological point of view. We would like to verify, for a given cluster of pairs, if the associated PFAM domains list (protein features) can be linked to the associated chemical

substructures list (drug features) and how their simultaneous presences can explain the high rate of interactions in the cluster.

Concerning the prediction of the genetic interaction network in yeast, a very important and obvious next step would be to experimentally validate the pairs that are predicted by our methods to be the most likely to interact, positively or negatively, in order to check the precision of our method. This validation should be done in close collaboration with biologists.

There remain also several potential directions to improve the quality of our predictions. For example, positive interactions have been found to be more difficult to predict than negative ones. The reason of this difference is not clear and deserves to be studied more thoroughly. A better understanding of this difference could help improving the quality of the predictions of these interactions. A potential improvement could be to address the problem of predicting genetic interactions as a regression problem, instead of a classification problem, by trying to predict the quantitative score associated to each pair. This could be particularly interesting for positive interactions. Indeed, we explained in Section 5.2.2 that positive interactions can be divided into masking, suppression, and symmetric interactions and quantitative predictions could help distinguishing these different families.

In chapter 5, we collected a very large number of datasets of gene features and evaluated their individual and joint predictive powers, concluding that the homozygous chemo-genomic feature set was the most informative. Although our coverage of potential gene features is already very large, we believe that some improvement could nevertheless still be obtained by integrating new feature sets, defined on genes or directly on pairs of genes. In particular, features that measure effects related to gene knockouts seems to work particularly well for predicting genetic interactions. For example, we would like to test the dataset recently produced in (Kemmeren *et al.*, 2014) that measures the expression of all yeast genes resulting from the knockout of 1484 genes.

Finally, in this thesis and in most existing works in the literature, we are considering separately the problem of the inference of several networks (protein-protein interactions, genetic interactions, GRN...) for several organisms (*S. cerevisiae*, *E. coli*, *H. sapiens*...). These networks and organisms are however strongly related and there is no doubt that better predictions could be obtained by jointly analyzing these networks for several species or strains. Such joint analysis would require however non trivial methodological developments to adapt existing multi-task and transfer machine learning approaches to the network inference problem.

Bibliography

- Aguilar, P. *et al.* (2010). A plasma-membrane e-map reveals links of the eisosome with sphingolipid metabolism and endosomal trafficking. *Nature Structural and Molecular Biology*, **136**, 952–963.
- Alanis-Lobato, G., Cannistraci, C. V., and Ravasi, T. (2013). Exploitation of genetic interaction network topology for the prediction of epistatic behavior. *Genomics*, **102**, 202–208.
- Albert, R. and Barabasi, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, **74**, 47–97.
- Almaas, E., Vázquez, A., and Barabási, A.-L. (2008). Scale-free networks in biology. In F. K. ed., editor, *Biological networks*. World Scientific.
- Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., , and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, **25**(1), 25–29.
- Bachmaier, C., Brandes, U., and Schreiber, F. (2013). Biological networks. In *Handbook of Graph Drawing and Visualization*. CRC Press.
- Balaji, S., Babu, M., Iyer, L., Luscombe, N., and Aravind, L. (2006). Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *J Mol Biol*.
- Baryshnikova, A., Costanzo, M., Kim, Y., Ding, H., Koh, J., Toufighi, K., Youn, J.-Y., Ou, J., Luis, B.-J. S., Bandyopadhyay, S., Hibbs, M., Hess, D., Gingras, A.-C., Bader, G. D., Troyanskaya, O. G., Brown, G. W., Andrews, B., Boone, C., and Myers, C. L. (2010). Quantitative analysis of fitness and genetic interactions in yeast on a genome scale. *Nature Methods*, **7**, 1017–1025.
- Bauer, T., Eils, R., and Konig, R. (2011). Rip: the regulatory interaction predictor—a machine learning-based approach for predicting target genes of transcription factors. *Bioinformatics*, **27**, 2239–2247.
- Ben-Hur, A. and Noble, W. S. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, **21**, i38–i46.
- Ben-Hur, A. and Noble, W. S. (2006). Choosing negative examples for the prediction of protein-protein interactions. *BMC bioinformatics*, **7**(Suppl 1), S2.
- Ben-Hur, A., Ong, C. S., Sonnenburg, S., Scholkopf, B., and Ratsch, G. (2008). Support vector machines and kernels for computational biology. *Plos Computational Biology*, **4**(10).
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Bleakley, K. and Yamanishi, Y. (2009). Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics*, **25**(18), 2397–2403.

- Bleakley, K., Biau, G., and Vert, J.-P. (2007). Supervised reconstruction of biological networks with local models. *Bioinformatics*, **23**, i57–i65.
- Blockeel, H., De Raedt, L., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of ICML 1998*, pages 55–63.
- Botstein, D. and Fink, G. R. (2011). Yeast: An experimental organism for 21st century biology. *Genetics*, **189**, 695–704.
- Botstein, D., Chervitz, S. A., and Cherry, J. M. (1997). Yeast as a model organism. *Science*, **277**(5330), 1259–1260.
- Braberg, H., Jin, H., Moehle, E. A., Chan, Y. A., Wang, S., Shales, M., Benschop, J. J., Morris, J. H., Qiu, C., Hu, F., Tang, L. K., Fraser, J. S., Holstege, F. C., Hieter, P., Guthrie, C., Kaplan, C. D., and Krogan, N. J. (2013). From structure to systems: High-resolution, quantitative genetic analysis of rna polymerase ii. *Cell*, **154**, 775–788.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**, 123–140.
- Breiman, L. (2001). Random forests. *Machine learning*, **45**(1), 5–32.
- Breiman, L., Friedman, J., Olsen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth International.
- Brem, R. B., Yvert, G., Clinton, R., and Kruglyak, L. (2002). Genetic dissection of transcriptional regulation in budding yeast. *Science*, **296**, 752–755.
- Brodersen, K., Ong, C., Stephan, K., and Buhmann, J. (2010). The binormal assumption on precision-recall curves. In *20th International Conference on Pattern Recognition (ICPR)*, pages 4263–4266.
- Brohée, S., Janky, R., Abdel-Sater, F., Vanderstocken, G., André, B., and van Helden, J. (2011). Unraveling networks of co-regulated genes on the sole basis of genome sequences. *Nucleic Acids Res.*, **39**(15), 6340–6358.
- Brouard, C. (2013). *Inférence de réseaux d'interaction protéine-protéine par apprentissage statistique*. Ph.D. thesis, Université d'Evry Val d'Essonne.
- Brouard, C., D'Alche-Buc, F., and Szafranski, M. (2011). Semi-supervised penalized output kernel regression for link prediction. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 593–600, New York, NY, USA. ACM.
- Brouard, C., Vrain, C., Dubois, J., Castel, D., Debily, M.-A., and d'Alché Buc, F. (2013). Learning a markov logic network for supervised gene regulatory network inference. *BMC Bioinformatics*, **14**, 273.
- Brunner, C., Fischer, A., Luig, K., and Thies, T. (2012). Pairwise support vector machines and their application to large scale problems. *Journal of Machine Learning Research*, **13**, 2279–2292.
- Butte, A. J., Tamayo, P., Slonim, D., Golub, T. R., and Kohane, I. S. (2000). Discovering functional relationships between rna expression and chemotherapeutic susceptibility using relevance networks. *PNAS*, **97**(22).
- Ceccarelli, M. and Cerulo, L. (2009). Selection of negative examples in learning gene regulatory networks. In *Bioinformatics and Biomedicine Workshop, 2009. BIBMW 2009. IEEE International Conference on*, pages 56–61.
- Cerulo, L., Elkan, C., and Ceccarelli, M. (2010). Learning gene regulatory networks from only positive and unlabeled data. *BMC Bioinformatics*, **11**, 228.

- Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Chang, D. T.-H., Syu, Y.-T., and Lin, P.-C. (2010). Predicting the protein-protein interactions using primary structures with predicted protein surface. *BMC Bioinformatics*, **11**(Suppl 1).
- Chen, X.-W. and Liu, M. (2005). Prediction of protein-protein interactions using random decision forest framework. *Bioinformatics*, **21**, 4394–4400.
- Cheng, F., Liu, C., Jiang, J., Lu, W., Li, W., Liu, G., Zhou, W., Huang, J., and Tang, Y. (2012). Prediction of drug-target interactions and drug repositioning via network-based inference. *PLoS Computational Biology*, **8**(5).
- Chipman, K. C. and Singh, A. K. (2009). Predicting genetic interactions with random walks on biological networks. *BMC Bioinformatics*, **10**(17).
- Chua, G., Morris, Q. D., Sopko, R., Robinson, M. D., Ryan, O., Chan, E. T., Frey, B. J., Andrews, B. J., Boone, C., and Hughes, T. R. (2006). Identifying transcription factor functions and targets by phenotypic activation. *PNAS*, **103**, 12045–12050.
- Collins, S. R., Schuldiner, M., Krogan, N. J., and Weissman, J. S. (2006). A strategy for extracting and analyzing large-scale quantitative epistatic interaction data. *Genome Biol.*, **7**(7).
- Collins, S. R., Miller, K. M., Maas, N. L., Roguev, A., Fillingham, J., Chu, C. S., Schuldiner, M., Gebbia, M., Recht, J., Shales, M., Ding, H., Xu, H., Han, J., Ingvarsdottir, K., Cheng, B., Andrews, B., Boone, C., Berger, S. L., Hieter, P., Zhang, Z., Brown, G. W., Ingles, C. J., Emili, A., Allis, C. D., Toczyski, D. P., Weissman, J. S., Greenblatt, J. F., and Krogan, N. J. (2007). Functional dissection of protein complexes involved in yeast chromosome biology using a genetic interaction map. *Nature*, **446**, 806–810.
- Collins, S. R., Roguev, A., and Krogan, N. J. (2010). Quantitative genetic interaction mapping using the e-map approach. *Methods Enzymol.*, **470**, 205–231.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**, 273–297.
- Costanzo, M., Baryshnikova, A., Bellay, J., Kim, Y., Spear, E., Sevier, C., Ding, H., Koh, J., Toufighi, K., and Mostafavi, S. (2010). The genetic landscape of a cell. *Science*, **327**(5964), 425.
- Costanzo, M., Baryshnikova, A., Myers, C. L., Andrews, B., and Boone, C. (2011). Charting the genetic interaction map of a cell. *Current Opinion in Biotechnology*, **22**, 66–74.
- Costanzo, M., Baryshnikova, A., VanderSluis, B., Andrews, B., Myers, C. L., and Boone, C. (2013). Genetic networks. In *Handbook of Systems Biology: Concepts and Insights*, chapter 6, pages 115–135. Academic Press.
- Craig, R. A. and Liao, L. (2007). Phylogenetic tree information aids supervised learning for predicting protein-protein interaction based on distance matrices. *BMC Bioinformatics*, **8**(6).
- Cunningham, P., Cord, M., and Delany, S. J. (2008). Supervised learning. In M. Cord and P. Cunningham, editors, *Machine Learning Techniques for Multimedia Case Studies on Organization and Retrieval*, pages 21–49.
- Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. *Proceedings of the 23rd International Conference on Machine Learning*.
- de la Fuente, A., Bing, N., Hoeschele, I., and Mendes, P. (2004). Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, **20**(18), 3565–3574.

- De Smet, R. and Marchal, K. (2010). Advantages and limitations of current network inference methods. *Nature Reviews Microbiology*, **8**(10), 717–729.
- Debré, P. and Gall, J.-Y. L. (2014). Séquençage des génomes et médecine personnalisée : perspectives et limites. *Bulletin de l'Académie nationale de médecine*.
- Decourty, L., Saveanu, C., Zemam, K., Hantraye, F., Frachon, E., Rousselle, J.-C., Fromont-Racine, M., and Jacquier, A. (2008). Linking functionally related genes by sensitive and quantitative characterization of genetic interaction profiles. *PNAS*, **105**(15), 5821 – 5826.
- Denis, F., Gilleron, R., and Letouzey, F. (2005a). Learning from positive and unlabeled examples. *Theoretical Computer Science*, **348**(1), 70–83.
- Denis, F., Gilleron, R., and Letouzey, F. (2005b). Learning from positive and unlabeled examples. *Theoretical Computer Science*, **348**(1), 70–83.
- Dixon, S. J., Costanzo, M., Baryshnikova, A., Andrews, B., and Boone, C. (2009). Systematic mapping of genetic interaction networks. *Annu. Rev. Genet.*, **43**, 601–625.
- Eisen, M., Spellman, P., Patrick, O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, **95**, 14863–14868.
- Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *KDD '08 Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220.
- Emmert-Streib, F., Glazko, G. V., Altay, G., and de Matos Simoes, R. (2012). Statistical inference and reverse engineering of gene regulatory networks from observational expression data. *Frontiers in Genetics*, **3**, 1–15.
- Ernst, J., Beg, Q. K., Kay, K. A., Balazsi, G., Oltvai, Z. N., and Bar-Joseph, Z. (2008). A semi-supervised method for predicting transcription factor-gene interactions in escherichia coli. *Plos Computational Biology*, **4**(3).
- Eronen, V.-P., Linde, R. O., Lindroos, A., Kanerva, M., and Aittokallio, T. (2010). Genome-wide scoring of positive and negative epistasis through decomposition of quantitative genetic interaction fitness matrices. *PLoS ONE*, **5**(7), e11611.
- Faith, J., Driscoll, M., Fusaro, V., Cosgrove, E., Hayete, B., Juhn, F., Schneider, S., and Gardner, T. (2007a). Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. *Nucleic Acids Research*, **36**, 866–870.
- Faith, J. J., Hayete, B., Thaden, J. T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J. J., and Gardner, T. S. (2007b). Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol*, **5**(1), e8.
- Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, **27**(8), 861 – 874.
- Fernandez, M., Ahmad, S., and Sarai, A. (2010). Proteochemometric recognition of stable kinase inhibition complexes using topological autocorrelation and support vector machines. *J. Chem. Inf. Model.*, **50**, 1179–1188.
- Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *JMLR*, **15**, 3133–3181.
- Fiedler, D., Braberg, H., Mehta, M., Chechik, G., Cagney, G., Mukherjee, P., Silva, A. C., Shales, M., Collins, S. R., and van Wageningen, S. (2009). Functional organization of the s. cerevisiae phosphorylation network. *Cell*, **136**(5), 952–963.

- Fluss, R., Faraggi, D., and Reiser, B. (2005). Estimation of the Youden Index and its associated cutoff point. *Biometrical journal. Biometrische Zeitschrift*, **47**(4), 458–472.
- Gama-Castro, S., Jiménez-Jacinto, V., Peralta-Gil, M., Santos-Zavaleta, A., Peñaloza-Spinola, M., Contreras-Moreira, B., Segura-Salazar, J., Muñoz-Rascado, L., Martínez-Flores, I., Salgado, H., Bonavides-Martínez, C., Abreu-Goodger, C., Rodríguez-Penagos, C., Miranda-Ríos, J., Morett, E., Merino, E., Huerta, A., Treviño-Quintanilla, L., and Collado-Vides, J. (2008). Regulondb (version 6.0): gene regulation model of *Escherichia coli* K-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. *Nucleic Acids Res*, **36**(Database issue), D120–4.
- Geurts, P. (2002). *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. Ph.D. thesis, University of Liège.
- Geurts, P. (2011). Learning from positive and unlabeled examples by enforcing statistical significance. In *JMLR: Workshop and Conference Proceedings*, volume 15.
- Geurts, P., Ernst, D., and Wehenkel, L. (2006a). Extremely randomized trees. *Machine Learning*, **63**(1), 3–42.
- Geurts, P., Wehenkel, L., and d'Alché-Buc, F. (2006b). Kernelizing the output of tree-based methods. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 345–352. ACM.
- Geurts, P., Touleimat, N., Dutreix, M., and d'Alché Buc, F. (2007). Inferring biological networks with output kernel trees. *BMC Bioinformatics*, **8**(Suppl 2), S4.
- Geurts, P., Irtuthum, A., and Wehenkel, L. (2009). Supervised learning with decision tree-based methods in computational and systems biology. *Molecular BioSystems*, **5**, 1593–1605.
- Giever, G., Chu, A., Ni, L., Connelly, C., Riles, L., Véronneau, S., Dow, S., Lucau-Danila, A., Anderson, K., André, B., Arkin, A., Astromoff, A., El-Bakkoury, M., Bangham, R., Benito, R., Brachat, S., Campanaro, S., Curtiss, M., Davis, K., Deutschbauer, A., Entian, K., Flaherty, P., Foury, F., Garfinkel, D., Gerstein, M., Gotte, D., Güldener, U., Hegemann, J., Hempel, S., Herman, Z., Jaramillo, D., Kelly, D., Kelly, S., Kötter, P., LaBonte, D., Lamb, D., Lan, N., Liang, H., Liao, H., Liu, L., Luo, C., Lussier, M., Mao, R., Menard, P., Ooi, S., Revuelta, J., Roberts, C., Rose, M., Ross-Macdonald, P., Scherens, B., Schimmack, G., Shafer, B., Shoemaker, D., Sookhai-Mahadeo, S., Storms, R., Strathern, J., Valle, G., Voet, M., Volckaert, G., Wang, C., Ward, T., Wilhelmy, J., Winzeler, E., Yang, Y., Yen, G., Youngman, E., Yu, K., Bussey, H., Boeke, J., Snyder, M., Philippsen, P., Davis, R., and Johnston, M. (2002). Functional profiling of the *Saccharomyces cerevisiae* genome. *Nature*, **418**, 387–391.
- Gillis, J. and Pavlidis, P. (2011). The impact of multifunctional genes on "guilt by association" analysis. *PLoS ONE*, **6**(2).
- Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., Galibert, F., Hoheisel, J. D., Jacq, C., Johnston, M., Louis, E. J., Mewes, H. W., Murakami, Y., Philippsen, P., Tettelin, H., and Oliver, S. G. (1996). Life with 6000 genes. *Science*, **274**(5287), 546–567.
- Goh, K.-I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., and Barabási, A.-L. (2007). The human disease network. *PNAS*, **104**, 8685–90.
- Gottlieb, A., Stein, G. Y., Ruppín, E., and Sharan, R. (2011). Predict: a method for inferring novel drug indications with application to personalized medicine. *Molecular Systems Biology*, **7**(496).
- Greiner, M., Pfeiffer, D., and Smith, R. (2000). Principals and practical application of the receiver operating characteristic analysis for diagnostic tests. *Preventive Veterinary Medicine*, **45**, 23–41.

- Han, J.-D. J., Bertin, N., Hao, T., Goldberg, D. S., Berriz, G. F., Zhang, L. V., Dupuy, D., Walhout, A. J. M., E. Cusick, M., Roth, F. P., and Vidal, M. (2004). Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, **430**(6995), 88–93.
- Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.
- Hauray, A.-C., Mordelet, F., Vera-Licona, P., and Vert, J.-P. (2012). Tigrass: Trustful inference of gene regulation using stability selection. *BMC Systems Biology*, **6**, 145.
- He, Z., Zhang, J., Shi, X.-H., Hu, L.-L., Kong, X., Cai, Y.-D., and Chou, K.-C. (2010). Predicting drug-target interaction networks based on functional groups and biological features. *Plos One*, **5**, e9603.
- Hempel, S., Koseska, A., Nikoloski, Z., and Kurths, J. (2011). Unraveling gene regulatory networks from time-resolved gene expression data – a measures comparison study. *BMC Bioinformatics*, **12**(1), 292.
- Hillenmeyer, M. *et al.* (2008). The chemical genomic portrait of yeast: Uncovering a phenotype for all genes. *Science*, **320**, 362–365.
- Hogan, D. J., Riordan, D. P., Gerber, A. P., Herschlag, D., and Brown, P. O. (2008). Diverse rna-binding proteins interact with functionally related sets of rnas, suggesting an extensive regulatory system. *Plos biology*, **6**, 2297–2313.
- Hoppins, S., Collins, S. R., Cassidy-Stone, A., Hummel, E., DeVay, R. M., Lackner, L. L., Westermann, B., Schuldiner, M., Weissman, J. S., and Nunnari, J. (2011). A mitochondrial-focused genetic interaction map reveals a scaffold-like complex required for inner membrane organization in mitochondria. *The Journal of Cell Biology*, **195**(2), 323–340.
- Hu, Z., Killion, P. J., and Iyer, V. R. (2007). Genetic reconstruction of a functional transcriptional regulatory network. *Nature genetics*, **39**, 683–687.
- Hue, M. and Vert, J.-P. (2010). On learning with kernels for unordered pairs. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel.
- Hue, M., Riffle, M., Vert, J.-P., and Noble, W. S. (2010). Large-scale prediction of protein-protein interactions from structures. *BMC Bioinformatics*, **11**, 144.
- Hughes, T., Marton, M., Jones, A., Roberts, C., Stoughton, R., Armour, C., Bennett, H., Coffey, E., Dai, H., He, Y., Kidd, M., King, A., Meyer, M., Slade, D., Lum, P., Stepaniants, S., Shoemaker, D., Gachotte, D., Chakraburttu, K., Simon, J., Bard, M., and Friend, S. (2000). Functional discovery via a compendium of expression profiles. *Cell*, **102**, 109–126.
- Huh, W., Falvo, J., Gerke, C., Carroll, A., Howson, R., Weissman, J., and O'Shea, E. (2003). Global analysis of protein localization in budding yeast. *Nature*, **425**, 686–691.
- Huynen, M., Snel, B., Lathe, W., and Bork, P. (2000). Predicting protein function by genomic context: Quantitative evaluation and qualitative inferences. *Genome Research*, **10**, 1204–1210.
- Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *Plos One*, **5**(9), e12776.
- International Human Genome Mapping Consortium (2004). Finishing the euchromatic sequence of the human genome. *Nature*, **431**, 931–945.
- Ito, T., Tashiro, K., Muta, S., Ozawa, R., Chiba, T., Nishizawa, M., Yamamoto, K., Kuhara, S., and Sakaki, Y. (2000). Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations of between the yeast proteins. *Proc. Natl. Acad. Sci.*, **97**, 1143–1147.

- Janky, R. and van Helden, J. (2008). Evaluation of phylogenetic footprint discovery for predicting bacterial cis-regulatory elements and revealing their evolution. *BMC Bioinformatics*, **9**.
- Jarvinen, A. P., Hiissa, J., Elo, L. L., and Aittokallio, T. (2008). Predicting quantitative genetic interactions by means of sequential matrix approximation. *PLoS ONE*, **3**(9).
- Jonikas, M. C., Collins, S. R., Denic, V., Oh, E., Quan, E. M., Schmid, V., Weibezahn, J., Schwappach, B., Walter, P., Weissman, J. S., and Schuldiner, M. (2009). Comprehensive characterization of genes required for protein folding in the endoplasmic reticulum. *Science*, **27**(5922), 1693–1697.
- Junaid, M., Lapins, M., Eklund, M., Spjuth, O., and Wikberg, J. E. S. (2010). Proteochemometric modeling of the susceptibility of mutated variants of the hiv-1 virus to reverse transcriptase inhibitors. *Plos One*, **5**(12).
- Kaneshiha, M., Goto, S., Kawashima, S., Okuno, Y., and Hattori, M. (2004). The kegg resources for deciphering the genome. *Nucleic Acids Res.*, **32**, 277–280.
- Karni, S., Soreq, H., and Sharan, R. (2009). A network-based method for predicting disease-causing genes. *Journal of computational biology*, **16**(2), 181–189.
- Kato, T., Tsuda, K., and Kiyoshi, A. (2005). Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, **21**(10), 2488–2495.
- Kelley, R. and Ideker, T. (2005). Systematic interpretation of genetic interactions using protein networks. *Nature Biotechnology*, **23**(5).
- Kemmeren, P., Sameith, K., van de Pasch, L. A., Benschop, J. J., Lenstra, T. L., Margaritis, T., O'Duibhir, E., Apweiler, E., van Wageningen, S., Ko, C. W., van Heesch, S., Kashani, M. M., Ampatzidis-Michailidis, G., Brok, M. O., Brabers, N. A., Miles, A. J., Bouwmeester, D., van Hooff, S. R., van Bakel, H., Sluiter, E., Bakker, L. V., Snel, B., Lijnzaad, P., van Leenen, D., Koerkamp, M. J. G., and Holstege, F. C. (2014). Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors. *Cell*, **157**(3), 740–752.
- Kim, M.-S. et al. (2014). A draft map of the human proteome. *Nature*, **509**, 575–581.
- Kondor, R. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proc. of the 19th International Conference on Machine Learning*, pages 315–322.
- Krogan, N. J., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., Li, J., Pu, S., Datta, N., and Tikuisis, A. P. (2006). Global landscape of protein complexes in the yeast *saccharomyces cerevisiae*. *Nature*, **440**(7084), 637–643.
- Langfelder, P. and Horvath, S. (2008). Wgcna: an R package for weighted correlation network analysis. *BMC Bioinformatics*, **9**(559).
- Lapins, M. and Wikberg, J. E. (2010). Kinome-wide interaction modelling using alignment-based and alignment-independent approaches for kinase description and linear and non-linear data analysis techniques research article. *BMC Bioinformatics*, **11**.
- Lee, W. and Liu, B. (2003). Learning with positive and unlabeled examples using weighted logistic regression. *Proceedings of the International Conference on Machine Learning*, **20**(1), 448.
- Leistner, C., Saffari, A., Santner, J., and Bischof, H. (2009). Semi-supervised random forests. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 506–513.
- Li, B. and Luo, F. (2009). Predicting yeast synthetic lethal genetic interactions using protein domains. In *Bioinformatics and Biomedicine, 2009. BIBM '09. IEEE International Conference on*, pages 43–47.

- Li, B., Cao, W., Zhou, J., and Luo, F. (2011). Understanding and predicting synthetic lethal genetic interactions in *saccharomyces cerevisiae* using domain genetic interactions. *BMC Systems Biology*, **5**(73).
- Li, S., Xi, L., Wang, C., Li, J., Lei, B., Liu, H., and Yao, X. (2009). A novel method for protein-ligand binding affinity prediction and the related descriptors exploration. *J Comput Chem*, **30**, 900–909.
- Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, **58**(7), 1019–1031.
- Lin, N., Wu, B., Jansen, R., Gerstein, M., and Zhao, H. (2004). Information assessment on predicting protein-protein interactions. *BMC Bioinformatics*, **5**, 154.
- Linden, R. O., Eronen, V.-P., and Aittokallio, T. (2011). Quantitative maps of genetic interactions in yeast - comparative evaluation and integrative analysis. *BMC Systems Biology*, **5**(45).
- Liti, G., Carter, D. M., Moses, A. M., Warringer, J., Parts, L., James, S. A., Davey, R. P., Roberts, I. N., Burt, A., and Koufopanou, V. (2009). Population genomics of domestic and wild yeasts. *Nature*, **458**(7236), 337–341.
- Louppe, G. (2014). *Understanding Random Forests*. Ph.D. thesis, University of Liège.
- Louppe, G., Wehenkel, L., Suter, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 431–439.
- Maclsaac, K. D., Wang, T., Gordon, B., Gifford, D. K., Stormo, G. D., and Fraenkel, E. (2006). An improved map of conserved regulatory sites for *saccharomyces cerevisiae*. *BMC Bioinformatics*.
- Madeira, S. and Oliveira, A. (2004). Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **1**(1).
- Maere, S., Heymans, K., and Kuiper, M. (2005). BiNGO: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, **21**, 3448–3449.
- Maetschke, S. R., Madhamshettiwar, P. B., Davis, M. J., and Ragan, M. A. (2013). Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Briefings in Bioinformatics*.
- Mani, R., St.Onge, R. P., Hartman, J. L., Giaever, G., and Roth, F. P. (2007). Defining genetic interaction. *PNAS*, **105**(9), 3461–3466.
- Manning, C., Raghavan, P., and Schütze, H. (2009). *An introduction to information retrieval*. Cambridge University Press.
- Marbach, D., Costello, J., Küffner, R., Vega, N., Prill, R., Camacho, D., K.R., A., Consortium, T. D., Kellis, M., Collins, J., and G., S. (2012). wisdom of crowds for robust network inference. *Nature methods*, **9**(8), 794–804.
- Marcotte, E. M., Pellegrini, M., Ng, H.-L., Rice, D. W., Yeates, T. O., and Eisenberg, D. (1999). Detecting protein function and protein-protein interactions from genome sequences. *Science*, **285**(5428), 751–753.
- Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The det curve in assessment of detection task performance. In *Proc. Eurospeech*, pages 1899–1903.
- McKusick, V. (1998). *Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders*. Johns Hopkins University Press, Baltimore, MD, 12th edn. edition.

- Mei, J.-P., Kwoh, C.-K., Yang, P., Li, X.-L., and Zheng, J. (2013). Drug–target interaction prediction by learning from local information and neighbors. *Bioinformatics*, **29**, 238–245.
- Mell, J. C. and Burgess, S. M. (2002). Yeast as a model genetic organism. *Encyclopedia of Life Science*.
- Mercier, G., Berthault, N., Meyrand, M., Touleimat, N., Képès, F., Fourel, G., Gilson, E., and Dutreix, M. (2005). A haploid-specific transcriptional response to irradiation in *s. cerevisiae*. *Nucleic Acid Research*, **33**, 6635–6643.
- Mering, C. V., Krause, R., Snel, B., Cornell, M., Oliver, S. G., Fields, S., and Bork, P. (2002). Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, **417**, 399–403.
- Meyer, P. E., Kontos, K., Lafitte, F., and Bontempi, G. (2007). Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J. Bioinformatics Syst. Biol.*, **2007**, 8–8.
- Mordelet, F. and Vert, J.-P. (2008). Sirene: supervised inference of regulatory networks. *Bioinformatics*, **24**(16), i76–i82.
- Mordelet, F. and Vert, J.-P. (2013). A bagging svm to learn from positive and unlabeled examples. *Pattern Recognition Letters*, (in press).
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.
- Myers, C. L., Barrett, D. R., Hibbs, M. A., Huttenhower, C., and Troyanskaya, O. G. (2006). Finding function: evaluation methods for functional genomic data. *BMC Genomics*, **7**(187).
- Newman, J. R. S., Ghaemmighami, S., Ihmels, J., Breslow, D. K., Noble, M., DeRisi, J. L., and Weissman, J. S. (2006). Single-cell proteomic analysis of *s. cerevisiae* reveals the architecture of biological noise. *Nature*, **441**(7095), 840–846.
- Niculescu, C. and Persson, L.-E. (2004). *Convex Functions and their Applications: A Contemporary Approach*. Springer.
- Nijijima, S., Yabuuchi, H., and Okuno, Y. (2011). Cross-target view to feature selection: Identification of molecular interaction features in ligand-target space. *J. Chem. Inf. Model.*, **51**, 15–24.
- O'Connor, T. R. and Wyrick, J. (2007). Chromatindb. <http://www.bioinformatics2.wsu.edu/cgi-bin/ChromatinDB/cgi/About.pl>.
- Ohya, Y., Sese, J., Yukawa, M., Sano, F., Nakatani, Y., Saito, T. L., Saka, A., Fukuda, T., Ishihara, S., Oka, S., Suzuki, G., Watanabe, M., Hirata, A., Ohtani, M., Sawai, H., Frayssé, N., Latgé, J.-P., François, J. M., Aebi, M., Tanaka, S., Muramatsu, S., Araki, H., Sonoike, K., Nogami, S., and Morishita, S. (2005). High-dimensional and large-scale phenotyping of yeast mutants. *PNAS*, **102**, 19015–19020.
- Pahikkala, T., Stock, M., Airola, A., Aittokallio, T., De Baets, B., and Waegeman, W. (2014a). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. In T. Calders, F. Esposito, E. Hullermeier, and R. Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8725 of *Lecture Notes in Computer Science*, pages 517–532. Springer Berlin Heidelberg.
- Pahikkala, T., Stock, M., Airola, A., Aittokallio, T., Baets, B. D., and Waegeman, W. (2014b). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. Submitted manuscript (2014).
- Paladugu, S. R., Zhao, S., Ray, A., and Raval, A. (2008). Mining protein networks for synthetic genetic interactions. *BMC Bioinformatics*, **9**, 426.

- Pandey, G., Zhang, B., Chang, A. N., Myers, C. L., Zhu, J., Kumar, V., and Schadt, E. E. (2010). An integrative multi-network and multi-classifier approach to predict genetic interactions. *PLoS Computational Biology*, **6**(9).
- Park, Y. and Marcotte, E. M. (2011). Revisiting the negative example sampling problem for predicting protein-protein interactions. *Bioinformatics*, **27**, 3024–3028.
- Park, Y. and Marcotte, E. M. (2012). Flaws in evaluation schemes for pair-input computational predictions. *Nature Methods*, **9**(12), 1134–1136.
- Pazos, F. and Valencia, A. (2001). Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering*, **14**(9), 609–614.
- Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D., and Yeates, T. O. (1999). Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA*, **96**, 4285 – 4288.
- Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, **44**, 203–231.
- Ptacek, J. *et al.* (2005). Global analysis of protein phosphorylation in yeast. *Nature*, **438**, 679–684.
- Qi, Y., Klein-seetharaman, J., Bar-joseph, Z., Qi, Y., and Bar-joseph, Z. (2005). Random forest similarity for protein-protein interaction prediction. *Pac Symp Biocomput*, **2005**, 531–542.
- Qi, Y., Bar-Joseph, Z., and Klein-Seetharaman, J. (2006). Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins*, **63**(3), 490–500.
- Quach, M., Brunel, N., and Alché Buc, F. (2007). Estimating parameters and hidden variables in non-linear state-space models based on odes for biological networks inference. *Bioinformatics*, **23**(23), 3209–3216.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 254–269, Berlin, Heidelberg. Springer-Verlag.
- Ryan, C., Greene, D., Cagney, G., and Cunningham, P. (2010). Missing value imputation for epistatic maps. *BMC Bioinformatics*, **11**, 197.
- Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, **13**(7), 1443–1471.
- Schrynemackers, M., Küffner, R., and Geurts, P. (2013). On protocols and measures for the validation of supervised methods for the inference of biological networks. *Frontiers in Genetics*, **4**(262).
- Schrynemackers, M., Wehenkel, L., Babu, M. M., and Geurts, P. (2014). Classifying pairs with trees for supervised biological network inference. *CoRR*, **abs/1404.6074**.
- Schuldiner, M., Collins, S., Thompson, N., Denic, V., Bhamidipati, A., Punna, T., Ihmels, J., Andrews, B., Boone, C., Greenblatt, J., Weissman, J., and Krogan, N. (2005). Exploration of the function and organization of the yeast early secretory pathway through an epistatic miniarray profile. *Cell*, **123**, 507–519.

- Schuldiner, M., Collins, S., Weissman, J., and Krogan, N. (2006). Quantitative genetic analysis in *saccharomyces cerevisiae* using epistatic miniarray profiles (e-maps) and its application to chromatin functions. *Methods*, **40**, 344–352.
- Shannon, P., Markiel, A., Ozier, O., Baliga, N., Wang, J., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **13**(11), 2498–2504.
- Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.
- Smith, E. N. and Kruglyak, L. (2008). Gene-environment interaction in yeast gene expression. *Plos Biology*, **6**(4), e83.
- Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**(12), 3273–3297.
- St Onge, R., Mani, R., Oh, J., Proctor, M., Fung, E., Davis, R., Nislow, C., Roth, F., and Giaever, G. (2007). Systematic pathway analysis using high-resolution fitness profiling of combinatorial gene deletions. *Nature Genetics*, **39**(2), 199–206.
- Steinmetz, L. M., Scharfe, C., Deutschbauer, A. M., Mokranjac, D., Herman, Z. S., Jones, T., Chu, A. M., Giaever, G., Prokisch, H., and Oefner, P. J. (2002). Systematic screen for human disease genes in yeast. *Nature Genetics*, **31**(4), 400–404.
- Stock, M., Pahikkala, T., Airola, A., Salakoski, T., Baets, B. D., and Waegeman, W. (2012). Learning monadic and dyadic relations: Three case studies in systems biology. In *ECML Workshop on Learning and Discovery in Symbolic Systems Biology*.
- Stumpf, M. P. H. and Porter, M. A. (2012). Critical truths about power laws. *Science*, **335**, 665–666.
- Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, **2009**, 4:2–4:2.
- Surma, M. A., Klose, C., Peng, D., Shales, M., Mrejen, C., Stefanko, A., Braberg, H., Gordon, D. E., Vorkel, D., Ejsing, C. S., Robert Farese, J., Simons, K., Krogan, N. J., and Ernst, R. (2013). A lipid e-map identifies *ubx2* as a critical regulator of lipid saturation and lipid bilayer stress. *Molecular Cell*, **51**, 519–530.
- Tabei, Y., Pauwels, E., Stoven, V., Takemoto, K., and Yamanishi, Y. (2012). Identification of chemogenomic features from drug-target interaction networks using interpretable classifiers. *Bioinformatics*, **28**, i487–i494.
- Takarabe, M., Kotera, M., Nishimura, Y., Goto, S., and Yamanishi, Y. (2012). Drug target prediction using adverse event report systems: a pharmacogenomic approach. *Bioinformatics*, **28**(18), i611–i618.
- Tastan, O., Qi, Y., Carbonell, J. G., and Klein-Seetharaman, J. (2009). Prediction of interactions between hiv-1 and human proteins by information integration. *Pacific Symposium on Biocomputing*, **14**, 516–527.
- Tavakkolkhah, P. and Küffner, R. (2013). Extending partially known networks. In A. de la Fuente, editor, *Gene Network Inference*, chapter 6, pages 87–105. Springer-Verlag Berlin Heidelberg.
- Tong, A. and Boone, C. (2006). Synthetic genetic array analysis in *saccharomyces cerevisiae*. *Methods Mol Biol.*, **313**.

- Tong, A., Lesage, G., Bader, G., Ding, H., Xu, H., Xin, X., Young, J., Berriz, G., Brost, R., Chang, M., Chen, Y., Cheng, X., Chua, G., Friesen, H., Goldberg, D., Haynes, J., Humphries, C., He, G., Hussein, S., Ke, L., Krogan, N., Li, Z., Levinson, J., Lu, H., Ménard, P., Munyana, C., Parsons, A., Ryan, O., Tonikian, R., Roberts, T., Sdicu, A., Shapiro, J., Sheikh, B., Suter, B., Wong, S., Zhang, L., Zhu, H., Burd, C., Munro, S., Sander, C., Rine, J., Greenblatt, J., Peter, M., Bretscher, A., Bell, G., Roth, F., Brown, G., Andrews, B., Bussey, H., and Boone, C. (2004). Global mapping of the yeast genetic interaction network. *Science*, **303**(5659).
- Tong, A. H. Y., Evangelista, M., Parsons, A. B., Xu, H., Bader, G. D., Page, N., Robinson, M., Raghibizadeh, S., Hogue, C. W. V., Bussey, H., Andrews, B., Tyers, M., and Boone, C. (2001). Systematic genetic analysis with ordered arrays of yeast deletion mutants. *Science*, **294**(5550), 2364–2368.
- Touleimat, N., Zehraoui, F., Dutreix, M., and d'Alché Buc, F. (2008). Semi-automated inference of regulatory pathways using perturbed data. In *JOBIM 2008*.
- Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, **3**(3), 1–13.
- Uetz, P., Giot, L., Cagney, G., Mansfield, T., Judson, R., Knight, J., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., and Rothberg, J. (2000). A comprehensive analysis of protein-protein interactions in *saccharomyces cerevisiae*. *Nature*, **403**, 623–627.
- Ulitsky, I., Krogan, N., and Shamir, R. (2009). Towards accurate imputation of quantitative genetic interactions. *Genome Biology*, **10**(12), R140.
- van Laarhoven, T., Nabuurs, S. B., and Marchiori, E. (2011). Gaussian interaction profile kernels for predicting drug–target interaction. *Bioinformatics*, **27**, 3036–3043.
- Vert, J.-P. (2010). Reconstruction of biological networks by supervised machine learning approaches. In *Elements of Computational Systems Biology*, chapter 7, pages 165–188. John Wiley & Sons, Inc.
- Vert, J.-P. and Yamanishi, Y. (2004). Supervised graph inference. *Advances in Neural Information Processing Systems*, **17**, 1433–1440.
- Vert, J.-P. and Yamanishi, Y. (2005). Supervised graph inference. In *Advances in Neural Information and Processing system*, pages 1433–1440.
- Vert, J.-P., Qiu, J., and Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, **8**(Suppl 10), S8.
- von Mering, C., Jensen, L., Snel, B., Hooper, S., Krupp, M., Foglierini, M., Jouffre, N., Huynen, M., and Bork, P. (2005). STRING: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Res.*, **1**(33).
- Wade, C., Umbarger, M., and McAlear, M. (2006). The budding yeast rna and ribosome biosynthesis (rrb) regulon contains over 200 genes. *Yeast*, **23**(4), 293–306.
- Wang, C., Ding, C., Meraz, R., and Holbrook, S. (2006). Psol: a positive sample only learning algorithm for finding non-coding rna genes. *Bioinformatics*, **22**(21), 2590–2596.
- Wilmes, G. M. *et al.* (2008). A genetic interaction map of RNA-processing factors reveals links between sem1/dss1-containing complexes and mrna export and splicing. *Molecular Cell*, **32**, 735–746.
- Witten, I. H. and Frank, E. (2005). *Data mining: Practical Machine Learning Tools and Techniques, Second edition*. Morgan Kaufmann.

- Wong, S. L., Zhang, L. V., Tong, A. H. Y., Li, Z., Goldberg, D. S., King, O. D., Lesage, G., Vidal, M., Andrews, B., Bussey, H., Boone, C., and Roth, F. P. (2004). Combining biological networks to predict genetic interactions. *PNAS*, **101**, 15682–15687.
- Yabuuchi, H., Nijjima, S., Takematsu, H., Ida, T., Hirokawa, T., Hara, T., Ogawa, T., Minowa, Y., Tsujimoto, G., and Okuno, Y. (2011). Analysis of multiple compound-protein interactions reveals novel bioactive molecules. *Mol Syst Biol*, **7**(472).
- Yamanishi, Y. (2009). Supervised bipartite graph inference. *Adv. Neural Inform. Process. Syst*, **21**, 1841–1848.
- Yamanishi, Y. and Vert, J.-P. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, **20**, i363–i370.
- Yamanishi, Y. and Vert, J.-P. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, **21**, i468–i477.
- Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., and Kanehisa, M. (2008). Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, **24**(13), i232–i240.
- Yamanishi, Y., Kotera, M., Kanehisa, M., and Goto, S. (2010). Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, **26**, i246–i254.
- Yamanishi, Y., Pauwels, E., Saigo, H., and Stoven, V. (2011). Extracting sets of chemical substructures and protein domains governing drug-target interactions. *Journal of Chemical Information and Modeling*, page 110505071700060.
- Yang, P., Yang, Y. H., Zhou, B. B., and Zomaya, A. Y. (2010). A review of ensemble methods in bioinformatics. *Current Bioinformatics*, **5**.
- Yip, K. Y. and Gerstein, M. (2008). Training set expansion: an approach to improving the reconstruction of biological networks from limited and uneven reliable interactions. *Bioinformatics*, **25**, 243–250.
- Yousef, M., Jung, S., Showe, L. C., and Showe, M. K. (2008). Learning from positive examples when the negative class is undetermined- microRNA gene identification. *Algorithms Mol Biol*, **3**(1), 2.
- Yu, H., Chen, J., Xu, X., Li, Y., Zhao, H., Fang, Y., Li, X., Zhou, W., Wang, W., and Wang, Y. (2012). A systematic prediction of multiple drug-target interactions from chemical, genomic, and pharmacological data. *PLoS ONE*, **7**(5).
- Zhang, Y., Li, B., Srimani, P. K., Chen, X., and Luo, F. (2012). Predicting synthetic lethal genetic interactions in *saccharomyces cerevisiae* using short polypeptide clusters. *Proteome Science*, **10**(Suppl 1).
- Zheng, J., Benschop, J., Shales, M., Kemmeren, P., Greenblatt, J., Cagney, G., Holstege, F., Li, H., and Krogan, N. (2010). Epistatic relationships reveal the functional organization of yeast transcription factors. *Molecular Systems Biology*, **6**(420).
- Zhong, W. and Sternberg, P. W. (2006). Genome-wide prediction of *c. elegans* genetic interactions. *Science*, **311**(5766), 1481–1484.