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

PhD dissertation

Machine learning-based feature ranking:
Statistical interpretation and gene network inference

by Vân Anh Huynh-Thu

Advisor:
Prof. Louis Wehenkel

Co-advisor:
Prof. Pierre Geurts

December 2011

Jury members

Rodolphe SEPULCHRE, Professor at the University of Liège, President;

Louis WEHENKEL, Professor at the University of Liège, Advisor;

Pierre GEURTS, Associate Professor at the University of Liège, Co-Advisor;

Kristel VAN STEEN, Associate Professor at the University of Liège;

Yvan SAEYS, Ph.D at the Ghent University;

Christophe AMBROISE, Professor at the University of Evry-Val-d'Essonne (FRANCE);

Diego DI BERNARDO, Research Assistant Professor at the University of Naples "Federico II" (ITALY);

Robert KÜFFNER, Group Leader at the Ludwig-Maximilians-Universität München (GERMANY)

Summary

Machine learning techniques, and in particular supervised learning methods, are nowadays widely used in bioinformatics. Two prominent applications that we target specifically in this thesis are biomarker discovery and regulatory network inference. These two problems are commonly addressed through the use of feature ranking methods that order the input features of a supervised learning problem from the most to the less relevant for predicting the output. This thesis presents, on the one hand, methodological contributions around machine learning-based feature ranking techniques and on the other hand, more applicative contributions on gene regulatory network inference.

Our methodological contributions focus on the problem of selecting truly relevant features from machine learning-based feature rankings. Unlike the p-values returned by univariate tests, relevance scores derived from machine learning techniques to rank the features are usually not statistically interpretable. This lack of interpretability makes the identification of the truly relevant features among the top-ranked ones a very difficult task and hence prevents the wide adoption of these methods by practitioners. Our first contribution in this field concerns a procedure, based on permutation tests, that estimates for each subset of top-ranked features the probability for that subset to contain at least one irrelevant feature (called CER for "conditional error rate"). As a second contribution, we performed a large-scale evaluation of several, existing or novel, procedures, including our CER method, that all replace the original relevance scores with measures that can be interpreted in a statistical way. These procedures, which were assessed on several artificial and real datasets, differ greatly in terms of computing times and the tradeoff they achieve in terms of false positives and false negatives. Our experiments also clearly highlight that using model performance as a criterion for feature selection is often counter-productive.

The problem of gene regulatory network inference can be formulated as several feature selection problems, each one aiming at discovering the regulators of one target gene. Within this family of methods, we developed the GENIE3 algorithm that exploits feature rankings derived from tree-based ensemble methods to infer gene networks from steady-state gene expression data. In a second step, we derived two extensions of GENIE3 that aim to infer regulatory networks from other types of data. The first extension exploits expression data provided by time course experiments, while the second extension is related to genetical genomics datasets, which contain expression data together with information about genetic markers. GENIE3 was best performer in the DREAM4 *In Silico Multifactorial* challenge in 2009 and in the DREAM5 *Network Inference* challenge in 2010, and its extensions perform very well compared to other methods on several artificial datasets.

Résumé

L'apprentissage automatique supervisé est largement utilisé de nos jours en bio-informatique. Deux problèmes importants auxquels nous nous intéressons dans cette thèse sont la découverte de biomarqueurs et l'inférence de réseaux de régulation génétique. En général, ces deux problèmes sont abordés via l'utilisation de méthodes permettant de classer les variables d'un problème d'apprentissage supervisé, selon leur pertinence pour la prédiction de la sortie. Cette thèse présente, d'une part, des contributions méthodologiques s'articulant autour de techniques de classement de variables par apprentissage automatique et, d'autre part, des contributions plus applicatives portant sur l'inférence de réseaux de régulation génétique.

Plus précisément, nos contributions méthodologiques se focalisent sur le problème de sélection de variables pertinentes à partir de classements obtenus par apprentissage automatique. Au contraire des p-values retournées par les tests univariés, les scores de pertinence calculés par les méthodes d'apprentissage automatique pour classer les variables ne sont généralement pas interprétables d'un point de vue statistique. Ce manque d'interprétabilité rend très difficile l'identification des variables pertinentes parmi celles qui sont les mieux classées et fait que ces méthodes ne sont pas largement adoptées par les praticiens. Notre première contribution dans ce domaine est le développement d'une procédure basée sur des tests de permutation, estimant, pour chaque sous-ensemble de variables les mieux classées, la probabilité que ce sous-ensemble contienne au moins une variable non pertinente (appelée CER, pour "conditional error rate"). Comme deuxième contribution, nous avons réalisé une étude comparative de plusieurs procédures, incluant notre méthode du CER, qui remplacent les scores de pertinence originaux par des mesures pouvant être interprétées de façon statistique. Ces procédures, qui ont été évaluées sur plusieurs jeux de données synthétiques et réels, diffèrent grandement en termes de temps de calcul, ainsi qu'en termes de faux positifs et faux négatifs qu'elles retournent. Nous expériences montrent également qu'utiliser la performance du modèle de prédiction comme critère pour sélectionner les variables est souvent contre-productif.

Le problème d'inférence de réseaux de régulation génétique peut être formulé comme plusieurs problèmes de sélection de variables, le but de chacun étant d'identifier les régulateurs d'un gène cible. Au sein de cette famille de méthodes, nous avons developpé GENIE3, un algorithme exploitant des classements des variables obtenus par apprentissage automatique pour inférer des réseaux génétiques à partir de données d'expression "steady-state" de gènes. Dans un deuxième temps, nous avons developpé deux extensions de GENIE3 ayant pour but d'inférer des réseaux de régulation à partir d'autres types de données. La première extension exploite des données

d'expression provenant de séries temporelles, tandis que la second extension exploite des bases de données génomiques et génétiques, contenant des données d'expression ainsi que de l'information sur des marqueurs génétiques. GENIE3 est arrivé en première position au challenge DREAM4 *In Silico Multifactorial* en 2009 et au challenge DREAM5 *Network Inference* en 2010. Ses extensions sont également compétitives comparées à d'autres méthodes sur plusieurs réseaux artificiels.

Acknowledgments

This dissertation would have never seen the light without the help and support of some people that I would like to thank deeply.

My first thanks go to Pierre Geurts, co-advisor of this thesis, for his great guidance and his constant availability. He was present every time I felt lost and needed to discuss. To me, he is an outstanding advisor.

I am grateful to Louis Wehenkel, advisor of this thesis, who gave me the opportunity to work as a PhD student in his nice and very interesting research group. I thank him for his wise advices and for the time he has taken out of his busy schedule and devoted to this thesis.

I thank Alexandre Irrthum for his kind help on various works and for being patient with me while answering to all my (stupid) questions about biology.

I also thank Yvan Saeys for his nice collaboration on GENIE3 and for providing me useful comments and advices on the evaluation of the feature selection methods.

Thank you to Gustavo Stolovitzky, Daniel Marbach, James Costello, and Robert Küffner for allowing me to include results of their experiments around the DREAM5 *Network Inference* challenge in this thesis.

I would like to express my sincere appreciation to the people who kindly accepted to reread parts of this thesis: Fabien Heuze, Raphaël Marée, Olivier Stern, and Benjamin Stevens.

The four years spent in the GIGA bioinformatics offices have been wonderful, thanks to colleagues who created and maintained a very pleasant working environment (not to mention the lunch times): Vincent Botta, Pierre-Yves Gilson, Fabien Heuze, Florence Lemahieu, Gilles Louppe, Raphaël Marée, Loïc Rollus, Marie Schrynemackers, Yannick Schutz, Olivier Stern, and Benjamin Stevens.

I gratefully acknowledge Alain Empain, Raphaël Marée, Giuseppe Saldi, Olivier Stern, and the SEGI team for providing and maintaining computing resources. A single computer would have taken 21268 days to run all the jobs I submitted, and I would not have finished my PhD before 2069.

I am grateful to all the members of the jury for their interest in this thesis

and for taking time to read and evaluate this dissertation.

I thank all my family and friends for giving me their support and love, without really knowing or understanding my research topics (but I agree that this is entirely my fault).

Johan, thank you for accepting to share your life with a researcher and for supporting me whatever the road that I choose to take.

Contents

Part I

Background

1 Introduction

The work presented in this thesis is related to two well-known open problems in the field of computational biology, namely feature selection and gene regulatory network inference. We tackled both problems using machine learning algorithms, and more particularly tree-based ensemble methods. This chapter is organized as follows. Section 1.1 first provides an introduction to the regulation of gene expression. Section 1.2 is related to microarray gene expression data, which is the main type of data analyzed in the context of our research. Sections 1.3 and 1.4 respectively introduce the problems of feature selection and network inference, while Section 1.5 provides some motivations for the choice of tree-based ensemble methods to deal with these problems. Section 1.6 shortly describes the different contributions of this thesis. Finally, Section 1.7 depicts the organization of this document.

Contents

1.1 Regulation of gene expression

Proteins are the central biochemical compounds that allow living organisms to develop and interact with their environment. Most proteins are produced in the cell from stretches of DNA contained in chromosomes, that are called *protein-coding genes*. Two steps are needed for each of these genes to produce a protein. The first step is the *transcription* of the DNA of the gene into a messenger RNA molecule (mRNA), and the second step is the *translation* of this mRNA into a protein. Note that in many species, a large number of RNA molecules are actually non-coding, i.e. they are not translated into a protein. Many non-coding molecules are nevertheless involved in biological functions, e.g. the regulation of translation.

The *expression* of a gene indicates the process by which the information contained in this gene is used to obtain a protein or a functional RNA molecule. The differences that exist between cells in an organism are mostly due to the action of diverse mechanisms of regulation across many stages, some of them being able to increase or inhibit gene expression, depending notably on external signals coming from the environment.

One of the main mechanisms of regulation of gene expression operates at the level of transcription and is governed by the action of proteins called *transcription factors* (Maston *et al.*, 2006). To activate or inhibit the transcription of a gene into a mRNA, transcription factors bind to specific sites that are located in a particular region of the DNA called *promoter*. The interaction of transcription factors with the promoter is illustrated in Figure 1.1.

Besides transcription factors, other types of regulators exist. In particular, *microRNAs* are small molecules of non-coding RNA that regulate gene expression at the post-transcriptional level (Kloosterman and Plasterk, 2006). By binding to a mRNA, a microRNA can stimulate its degradation or inhibit its translation into a protein.

Hundreds of transcription factors and microRNAs have been discovered in the human genome so far. However the complete deciphering of the complex system induced by the different mechanisms of regulation remains one of the major challenges of systems biology.

1.2 Microarrays

Nowadays, a widely used technique to measure the expression levels of a large number of genes simultaneously is microarray analysis. Microarrays are DNA chips that allow to measure the quantity of mRNAs in a cell and

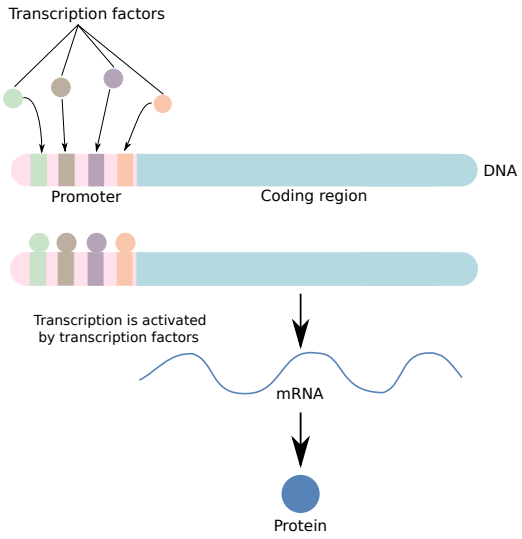


Figure 1.1: **Regulation of transcription by transcription factors.** To regulate the transcription of a gene into a mRNA, transcription factors bind to specific sites located in the promoter region of the gene.

therefore to establish its gene expression profile.

Each spot of a microarray contains millions of copies of a *probe*, i.e. a single stranded segment of DNA that is complementary to a known part of a mRNA. mRNAs are extracted from the cell to be analyzed, tagged with a fluorescent label, and spread over the array. Each mRNA sequence binds (or *hybridizes*) to a corresponding probe. The array is then washed to eliminate the unbound sequences and scanned with a laser. Each probe that is hybridized generates a fluorescent signal and the intensity level of a spot depends on the number of copies of the corresponding probe that are hybridized. The expression level of a mRNA can therefore be determined by measuring the fluorescence intensity of its spot. A bright spot indicates a highly expressed mRNA while a dark spot indicates a lowly expressed mRNA. Figure 1.2 shows an example of Affymetrix®[1] microarray.

Unfortunately, microarray analyses are expensive. For that reason, the number of samples in expression datasets is usually much lower than the number of genes. Datasets typically contain tens to hundreds of samples while up to several thousands of genes can be screened in one single experiment.

In our research, we analyzed microarray expression data for two tasks: feature selection and network inference. These problems are introduced in

[1] http://www.affymetrix.com/

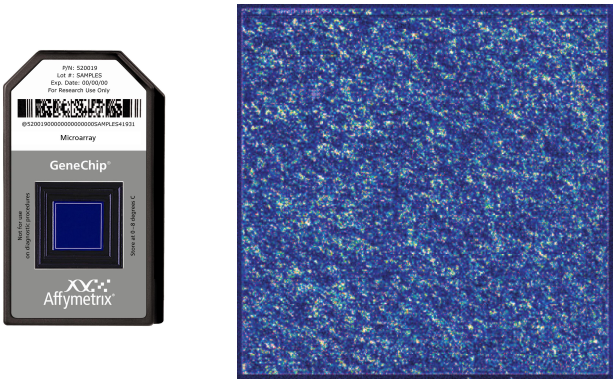


Figure 1.2: **GeneChip® microarray from Affymetrix and its output.** A microarray analysis results in a fluorescent image, where a bright spot indicates a highly expressed mRNA and a dark spot indicates a lowly expressed mRNA. Images courtesy of Affymetrix.

the two following sections.

1.3 Feature selection

Our research focuses on feature selection in the context of supervised learning. We thus assume that we have at our disposal a dataset containing samples annotated with an output value that can be discrete or continuous. A typical example from biology is a gene expression dataset where samples come from patients that are either healthy or affected by a particular disease.

Given a (typically very large) set of candidate variables[2], we define feature selection as the identification of the *maximal* subset of *relevant* variables, i.e. variables that convey information about the output variable, *either alone or in combination with other relevant variables*. The notion of relevance, which may remain imprecise at this stage, is formalized in Section 3.2 of Chapter 3.

Feature selection can have several objectives (see e.g. the reviews of Guyon and Elisseeff, 2003 and Saeys *et al.*, 2007):

- To reduce the data in order to limit computational requirements and increase the algorithmic speed;

[2]Although in some works, raw "variables" are distinguished from "features" that are constructed from the original variables, these two words are considered synonymous in this thesis and hence are alternatively used without any distinction.

- To reduce the number of features in order to reduce the costs of future collections of data;
- To build a predictive model of higher accuracy;
- To gain insights into the problem under study.

In this work, we mainly focus on the last objective, i.e. the understanding of the problem. In the example of the expression dataset, the goal of the feature selection task would be to discover all the genes whose expression is useful to distinguish between healthy and affected patients and therefore that are likely to be implied in the disease under study. In the field of biology, the genes that are selected are called *biomarkers* and a set of biomarkers is said to be a *signature*.

Feature selection algorithms are traditionally divided into three categories, according to the way they are combined with the predictive model: filter methods, wrapper methods, and embedded methods (Guyon and Elisseeff, 2003).

Filter methods perform feature selection by looking at the intrinsic properties of the data. They are usually ranking methods, i.e. methods that order the features using a relevance score. An example of filter method is the statistical hypothesis testing, such as the t-test. Filter methods are simple, fast, and scale easily to high-dimensional data. However, they do not interact with any learning algorithm and therefore the selection of the features is performed independently of the optimization of the predictive model.

Wrapper methods consist in training a predictive model to evaluate a specific subset of features, by using a quality criterion which is usually the performance of the model. The learning algorithm is used as a black box in an iterative search procedure exploring the space of possible feature subsets. Unlike filter approaches, wrapper methods have the advantage to interact with the learning algorithm. They are also multivariate as they evaluate groups of features. Their main drawback is their high computational cost.

Finally, embedded methods incorporate the selection of the features in the learning of the predictive model. Tree-based methods are examples of embedded methods and the predictive models that are learned by these algorithms can be directly used to compute relevance measures for the variables. Like wrapper methods, embedded methods have the advantages to be multivariate and to interact with the learning algorithm, while being typically less computationally intensive.

Independently of the category of the feature selection method, when the sole information available about the problem under consideration is limited to a dataset containing a finite number of samples, it is not possible to exactly

identify the maximal subset of relevant variables. Thus, any feature selection algorithm is at risk of either missing some sought features (false negatives) or of erroneously selecting some truly non desired ones (false positives), and a choice among different possible sensitivity/specificity compromises has always to be made.

1.4 Network inference

Networks are commonly used in biological research to represent information. Various kinds of networks exist. For example, metabolic networks represent chemical reactions between metabolites as well as the enzyme proteins that catalyze these reactions. Another example is the protein-protein interaction network that connects all proteins that can physically interact.

In this thesis, network inference refers to the task of recovering gene *regulatory* networks. A regulatory network represents regulatory interactions among genes that happen at the level of transcription, through transcription factors. It often offers a simplified view of gene regulation, as it does not take into account several key players such as microRNAs or other non-coding regulatory elements. Also, regulatory interactions actually involve DNA molecules, mRNAs, and proteins, and these different elements are merged to obtain a simple network representing only genes. Therefore regulatory networks are usually represented by graphs where each node corresponds to a gene. In such graphs, an edge directed from one gene to another gene indicates that the first gene codes for a transcription factor that regulates the rate of transcription of the second gene. An example is shown in Figure 1.3.

Edges in regulatory networks can be directed or undirected. An undirected edge connecting two genes indicates that there exists a transcriptional regulatory interaction between these two genes, while a directed edge means furthermore that the source gene regulates the expression of the target gene. Edges can also be signed. When a gene is connected to another gene, a positive (resp. negative) sign indicates that the former is an activator (resp. repressor) of the latter. Our research focuses on directed and unsigned networks. The targeted networks are thus graphs with p nodes, where an edge directed from one gene i to another gene j indicates that gene i (directly) regulates, either positively or negatively, the expression of gene j $(i, j = 1, \ldots, p)$.

The problem of the inference of regulatory networks has been studied for many years in the literature and many algorithms already exist. De Smet and Marchal (2010) proposed a categorization of these methods, shown in Figure 1.4. First, they distinguish supervised from unsupervised methods.

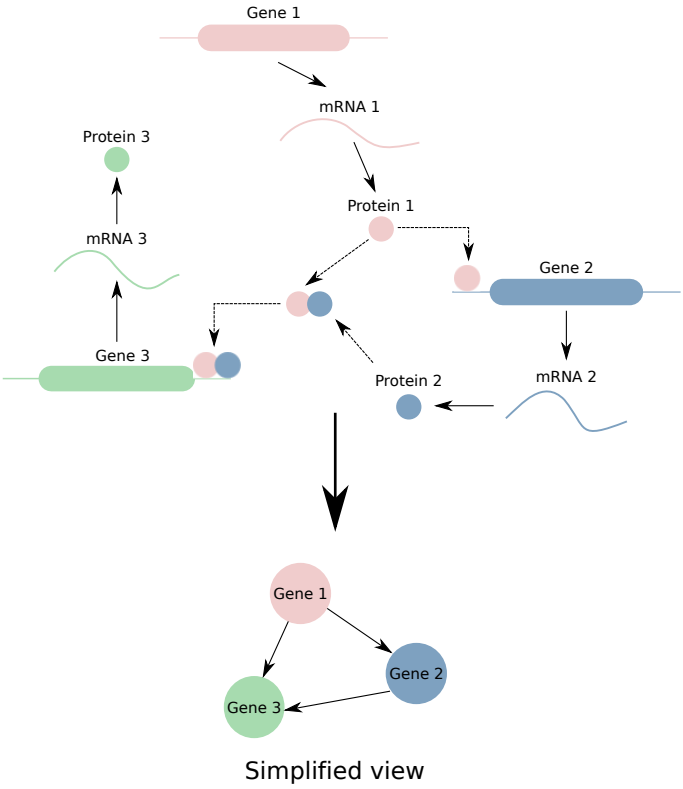


Figure 1.3: **Example of a regulatory network.** Regulatory interactions involving DNA molecules, mRNAs, and proteins are simplified to a simple network that only represents genes. Figure modified from Hecker *et al.* (2009).

Supervised methods exploit prior partial knowledge of the network to guide the network inference, while unsupervised methods do not assume any prior knowledge. There are also integrative and non-integrative methods. Non-integrative methods only use expression data for the inference. Given the dynamic and combinatorial nature of genetic regulation, measurements of different kinds can be obtained, including steady-state expression profiles resulting from gene knockouts or time series measurements resulting from random perturbations. On the other hand, integrative methods also use other kinds of information besides expression data, e.g. counts of sequence motifs that serve as binding sites for transcription factors. Finally, direct methods consider only individual interactions while module-based methods search for sets of genes that are regulated by the same transcription factors. In regard to these categories, GENIE3, a network inference algorithm that we developed, is a direct, unsupervised, and non-integrative method.

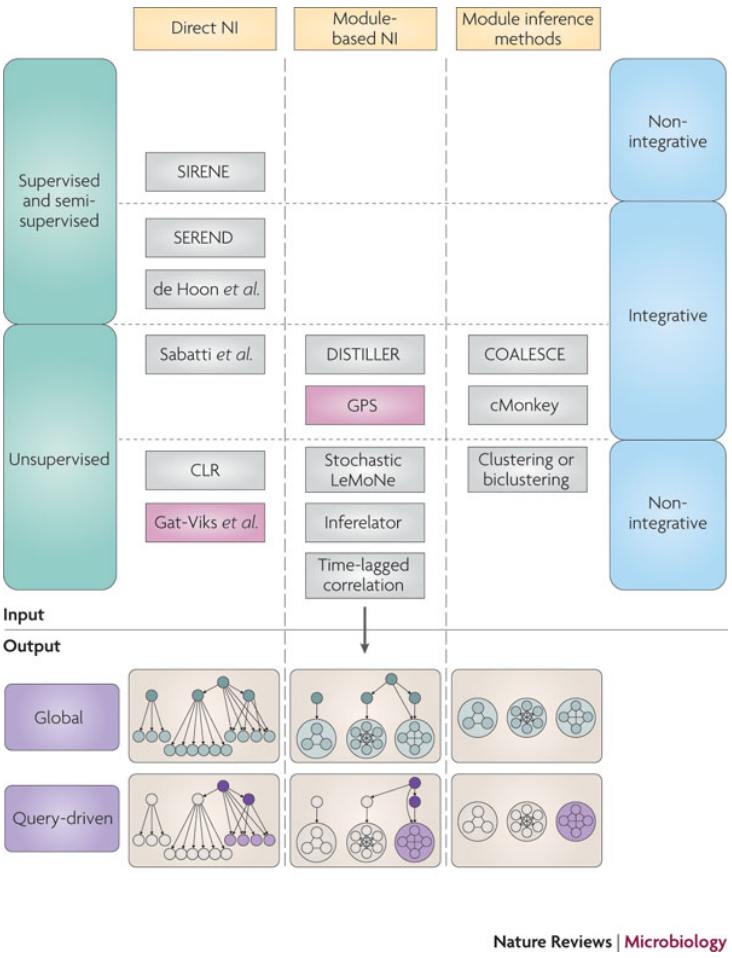


Figure 1.4: **Categorization of the network inference algorithms.** In regard to this categorization, our GENIE3 algorithm is a direct, unsupervised, and non-integrative method. Figure taken from De Smet and Marchal (2010).

Many network inference algorithms work first by providing a ranking of the potential regulatory links from the most to the less significant. A practical network prediction is then obtained by setting a threshold on this ranking. Our research focuses only on the first task and the question of the choice of an optimal confidence threshold, although important, is left open. A network inference algorithm is thus defined in this thesis as a procedure that assigns weights $w_{i,j} \geq 0, (i, j = 1, \ldots, p)$ to putative regulatory links from any gene i to any gene j, with the aim of yielding larger values for weights that correspond to actual regulatory interactions.

1.5 Tree-based ensemble methods

Our different contributions to the tasks of feature selection and network inference are based on rankings. These may be either rankings of variables (for the feature selection task) or rankings of regulatory links (for the network inference task). To compute these rankings, we chose to resort to variable importance scores that are provided by supervised learning algorithms, and more particularly by tree-based ensemble methods. Several reasons motivated the choice of tree-based methods (see Geurts *et al.*, 2009 for a review). First, they are able to detect multivariate interacting effects between features. They thus constitute an interesting alternative to standard univariate statistical tests for the feature selection task, and are also appealing for the inference of regulatory networks, as the regulation of the expression of one gene is expected to be combinatorial, i.e. to involve several regulators. Tree-based methods have also the advantage to be non-parametric. They thus do not make any assumption about the nature of interactions between the variables. As their computational complexity is typically at most linear in the number of features, they can deal with high-dimensionality, a characteristic usually encountered in gene expression datasets. They are also flexible as they can handle both continuous and discrete variables. They are fast to compute, highly scalable, and essentially parameter-free.

While the different algorithms developed in the context of this thesis essentially rely on tree-based ensemble methods, we also performed, for the sake of comparison, some experiments with another popular supervised learning algorithm, namely the linear support vector machine (SVM). Like tree-based methods, SVMs are able to deal with high-dimensionality and in some application domains, finely-tuned SVMs can yield higher levels of prediction performance than tree-based ensemble methods (Ben-Hur *et al.*, 2008; Geurts *et al.*, 2009). However, tuning the meta-parameters of SVMs requires a lot of human intervention, which makes these methods not easy to use for non-specialists.

Tree-based methods and linear SVMs are described in details in Chapter 2.

1.6 Contributions

Univariate hypothesis testing is widely used to solve the feature selection problem, especially in the context of "biomarker discovery" in bioinformatics. A classic procedure consists in applying a statistical test, such as a t-test, in order to compute a p-value for each variable of the considered problem and

selecting the variables that have a p-value lower than a chosen threshold. Univariate tests are simple and fast, but they can only identify variables that provide a significant amount of information about the output variable in isolation from the other inputs. As mentioned in the previous section, when one seeks for multivariate interacting effects between features, one can nowadays resort to relevance scores provided by machine learning techniques, such as tree-based ensemble methods. However, unlike the p-values returned by univariate tests, these relevance scores are usually not statistically interpretable. This lack of interpretability prevents the wide adoption of these methods by practitioners and also makes the identification of the truly relevant variables among the top-ranked ones, i.e. the determination of a relevance threshold, a very difficult task in practice. Hence two of the main contributions of the present thesis focus on the task of replacing the variable relevance scores with measures that can be interpreted in a statistical way and that help to determine a relevance threshold, such as p-values, false discovery rates (FDRs) or family wise error rates (FWERs). The first contribution, which is described in Chapter 3, concerns a procedure, based on permutation tests, that computes for each subset of top-ranked variables the *conditional error rate* (CER), that estimates the probability for that subset to contain at least one irrelevant feature (Huynh-Thu *et al.*, 2008). Our experiments show that the CER-based procedure allows reliable identifications of relevant features. We also show that the direct extension of the classic approach based on permutation tests for estimating FDRs of univariate variable scoring procedures does not extend very well to the case of importance measures derived from multivariate approaches. We performed a large-scale evaluation of the CER- and FDR-based methods, including their comparison to other, existing and novel, procedures that select relevant features from rankings derived from machine learning techniques (Huynh-Thu *et al.*, 2012). This large-scale evaluation of different methods constitutes our second contribution within the field of feature selection, and is presented in Chapter 4. All the evaluated methods, which were assessed on several artificial and real datasets, help to determine a relevance threshold, but they differ greatly in terms of computing times and the tradeoff they achieve in terms of false positives and false negatives. Our experiments also clearly highlight the fact that using model prediction performance as a criterion for feature selection is often counter-productive.

Besides these two contributions to feature selection, this thesis also comprises contributions related to the application of feature selection/ranking techniques to the problem of gene regulatory network inference. The problem of recovering the regulatory interactions occurring among p genes can indeed be viewed as p feature selection problems, the goal of each being to retrieve the regulators of one of the p genes. This framework is exploited in

GENIE3, an algorithm that we developed for the inference of gene regulatory networks from steady-state expression data (Huynh-Thu *et al.*, 2010). In this algorithm, the expression pattern of each gene (target gene) is predicted from the expression patterns of all the other genes (input genes), using tree-based ensemble methods. The importance of an input gene in the prediction of the target gene expression pattern is then taken as an indication of a putative regulatory link. The algorithm, which is presented in Chapter 6, is simple and generic, making it adaptable to other types of data and interactions. We hence developed two extensions of GENIE3, described in Chapter 8, that aim to infer regulatory networks from other types of data besides steady-state expression data. The first extension exploits expression data provided by time course experiments. In this procedure, the weight of a regulatory link directed from one gene to another gene in the predicted network is given by the importance of the expression of the first gene at a time point t for the prediction of the expression of the second gene at time point $t + h$, where $h > 0$ denotes a given time horizon. The second extension of GENIE3 that we performed is related to "genetical genomics" datasets, which contain expression data together with information about genetic markers. The idea of this extension is to use the genetic markers as input variables, either alone or together with the gene expression patterns of the input genes, when predicting the expression of a target gene. We show that GENIE3 and its extensions perform very well compared to other methods in the DREAM challenges.

1.6.1 Publications

- Publications related to our contributions to feature selection:
  - V. A. Huynh-Thu, L. Wehenkel, and P. Geurts (2008) Exploiting tree-based variable importances to selectively identify relevant variables. *JMLR: Workshop and Conference proceedings*, **4**:60-73.
  - V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, and P. Geurts (2012) Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. *Bioinformatics*, **28**(13):1766-1774.
- Publications related to one of our contributions to network inference (GENIE3):
  - V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, **5**(9):e12776.

  - D. Marbach, J. C. Costello, R. Küffner, N. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, the DREAM5 consortium (including P. Geurts, V. A. Huynh-Thu, A. Irrthum, Y. Saeys, and L. Wehenkel), M. Kellis, J. J. Collins, and G. Stolovitzky (2011) Wisdom of crowds for robust gene network inference. *Nature Methods*, **9**:796-804.

- During the course of the PhD, we also worked, in collaboration with biologists of the GIGA center at the University of Liège, on studies involving joint analyses of microRNA and mRNA expression patterns. One of these studies led to the following publication:

  - N. Garbacki, E. Di Valentin, V. A. Huynh-Thu, P. Geurts, A. Irrthum, C. Crahay, T. Arnould, C. Deroanne, J. Piette, D. Cataldo, and A. Colige (2011) MicroRNAs profiling in murine models of acute and chronic asthma: a relationship with mRNAs targets. *PLoS ONE*, **6**(1): e16509.

1.7 Thesis outline

This document is divided in four parts and is organized in the following way. Part I is intended to give the reader some background on our work, Part II is dedicated to the problem of feature selection, Part III is related to the network inference task, and finally Part IV contains the appendices.

Part I: Background After the present introductory chapter, Chapter 2 closes the first part and provides precise descriptions of the main machine learning algorithms that we used in our work, i.e. tree-based methods and linear SVMs.

Part II: Feature selection This part covers Chapters 3 to 5. Chapter 3 concerns two statistical procedures based on permutation tests for selecting relevant variables from a ranking derived from a tree-based ensemble method. In Chapter 4, we perform a large-scale evaluation of these two procedures, including their comparison to other, existing and novel, feature selection methods. Chapter 5 concludes the part.

Part III: Network inference This part comprises 4 chapters. Chapter 6 focuses on GENIE3, an algorithm for the inference of gene regulatory networks from static steady-state expression data, that uses tree-based ensemble

methods. Chapter 7 summarizes different results that were obtained in the context of the network inference challenge of DREAM5. Chapter 8 describes extensions of the GENIE3 method to other types of data: expression data provided by time series experiments and genetical genomics data. Finally, Chapter 9 concludes the part.

Part IV: Appendices This part contains some supplementary information related to our different contributions.

2

Machine learning background

In the previous chapter, we motivated the use of machine learning algorithms, especially tree-based ensemble methods, to deal with the problems of feature selection and network inference. The present chapter provides a description of these methods, which form the basis for the major contributions of this thesis. Section 2.1 introduces notions of supervised learning, while Sections 2.2 and 2.3 are respectively dedicated to tree-based ensemble methods and support vector machines. In particular, we show how these algorithms can be used to derive importance measures for variables. Finally, Section 2.4 describes the main metrics that we used to evaluate the performances of feature selection and network inference algorithms.

Contents

2.1 Supervised learning

Machine learning is a branch of artificial intelligence whose goal is to extract knowledge from observed data. In particular, supervised learning is the machine learning task of inferring a model $f : \mathcal{X} \to \mathcal{Y}$ that predicts the value of an output variable $Y \in \mathcal{Y}$, given the values of m inputs $(X_1, X_2, \ldots, X_m) = \mathbf{X} \in \mathcal{X}$. The model f is learned from N instances (also called *samples*) of input-output pairs that are drawn from the (usually unknown) joint distribution of the variables:

$$LS = \{(\mathbf{x}_k, y_k)\}_{k=1}^{N}. \tag{2.1}$$

The set of instances is called *learning sample*. Depending on whether the output is discrete or continuous, the learning problem is a *classification* or a *regression* problem respectively.

Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function that, given an instance $(\mathbf{x}, y)$, measures the difference between the value $f(\mathbf{x})$ predicted by the model f from the input $\mathbf{x}$, and the observed value y of the target variable. For a classification problem, a typical loss function is given by:

$$L(y, f(\mathbf{x})) = I(y \neq f(\mathbf{x})) \tag{2.2}$$

$$= \begin{cases} 0, & \text{if } f(\mathbf{x}) = y, \\ 1, & \text{if } f(\mathbf{x}) \neq y, \end{cases} \tag{2.3}$$

while for a regression problem, a widely used loss function is the squared error:

$$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \tag{2.4}$$

The goal of supervised learning is to find, from a learning sample LS, a model f that minimizes the *generalization error*, i.e. the expected value of the loss function, taken over different instances randomly drawn from the joint distribution of the input/output pairs:

$$E_{\mathbf{x},y}\left[L(y, f(\mathbf{x}))\right]. \tag{2.5}$$

Supervised learning algorithms typically work by minimizing the *training error*, which is the average prediction error of the model over the instances of the learning sample:

$$\frac{1}{N}\sum_{k=1}^{N} L(y_k, f(\mathbf{x}_k)). \tag{2.6}$$

As the training error is calculated on the same samples that were used to learn the predictive model, it typically underestimates the generalization error, as shown in Figure 2.1. The training error typically decreases when

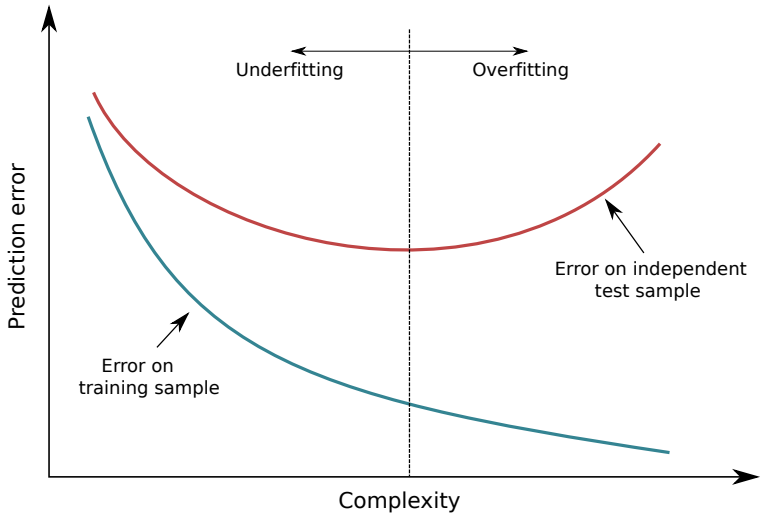


Figure 2.1: **Overfitting and underfitting.** The blue (resp. red) curve plots, for varying levels of complexity of the predictive model, the average value of the loss function over the instances of the learning sample (resp. of an independent test sample). Overfitting occurs when the model is too complex and underfitting occurs when the model is not complex enough.

the complexity of the model is increased, i.e. when the model is allowed to fit more closely the training data. If the complexity is too high, the model may also fit the noise contained in the data and thus will have a poor generalization performance. In this case, we say that the model *overfits* the training data. On the other hand, if the model has a too low complexity, it *underfits* the data and will also have a high generalization error. Hence there is an optimal model complexity that leads to the minimal generalization error.

An unbiased way of estimating the generalization error of a model is to compute its prediction error on an independent set of instances. However, such an independent set is usually not available. In this case, a widely used technique to estimate the generalization error is *k-fold cross-validation*. In this procedure, the instances of the learning sample are divided into k (non-overlapping) parts and k predictive models are learned. Each model is learned using the instances of $k-1$ parts and tested on the instances of the remaining part. The generalization error is then estimated by the average prediction error over the k parts. The special case where k is set to the number N of instances in the original learning sample is called the *leave-one-out* procedure.

For more details, the reader is invited to refer to the books of Hastie *et al.*

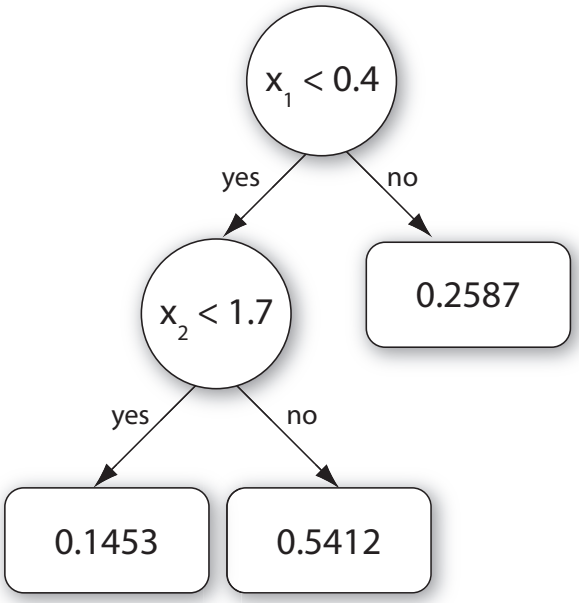


Figure 2.2: **Example of a regression tree.** Each interior node of a tree is a test on one input variable and each terminal node contains a predicted value for the output variable.

(2001) and Bishop (2006). We describe below the two supervised learning techniques used in this thesis: tree-based ensemble methods and support vector machines.

2.2 Tree-based ensemble methods

2.2.1 Classification and regression trees

Classification and regression trees (Breiman *et al.*, 1984) solve the supervised learning problem by developing tree structured models. The basic idea of this method is to recursively split the learning sample with binary tests based each on one input variable, trying to reduce as much as possible the uncertainty about the output variable in the resulting subsets of samples.

Figure 2.2 shows the structure of a tree. In this example, all variables are continuous and the learning sample is split based on two input variables X_1 and X_2.

Finding the smallest tree that minimizes the training error in Equation (2.6) is known to be a NP-complete problem (Hyafil and Rivest, 1976).

In practice, tree-based methods are thus based on greedy algorithms. A classification or regression tree is typically constructed top-down, starting from a root node corresponding to the whole learning sample. At each interior node $\mathcal{N}$, the best input variable and the best test on this variable are chosen, i.e. the variable and the test that most reduce the uncertainty about the output variable in the resulting subsets of samples, by maximizing:

$$\#S.U_Y(S) - \#S_t.U_Y(S_t) - \#S_f.U_Y(S_f), \tag{2.7}$$

where S denotes the set of samples that reach node $\mathcal{N}$, S_t (resp. S_f) denotes its subset for which the test is true (resp. false), # denotes the cardinality of a set of samples, and $U_Y(\cdot)$ is the uncertainty about the output in a subsample. The samples of S are then split into two subsamples following the optimal test and the same procedure is applied on each of these subsamples. A node becomes a terminal node if the uncertainty about the output variable, over the samples reaching that node, is equal to zero. Each terminal node contains a predicted value for the output, corresponding to the majority class (for classification) or the mean value of the output (for regression) taken over the samples that reach that node.

In the case of classification, the uncertainty $U_Y(\cdot)$ about the output in a subsample S can be measured by the entropy $H_Y(\cdot)$ of the class frequencies, e.g. the Shannon entropy:

$$H_Y(S) = -\sum_{i=1}^{C} p(c_i).\log_2 p(c_i), \tag{2.8}$$

where C is the number of classes and $p(c_i)$ denotes the proportion of samples in S belonging to class c_i. Alternatively to the Shannon entropy, the class impurity can also be measured by the Gini index:

$$H_Y(S) = 1 - \sum_{i=1}^{C} p(c_i)^2. \tag{2.9}$$

For a regression problem, we consider as measure of uncertainty the empirical variance of the output variable Y in the subsample:

$$\mathrm{Var}_Y(S) = \frac{1}{\#S}\sum_{k=1}^{\#S}(y_k - \bar{y})^2, \tag{2.10}$$

where $\bar{y}$ is the mean value of Y in the subsample S.

However, a fully grown tree typically overfits the training data. Overfitting can be avoided by performing a *pruning* of the tree, i.e. by removing

some of its subtrees. Two types of pruning exist: pre-pruning and post-pruning. In a pre-pruning procedure, a node becomes a terminal node instead of a test node if it meets a given criterion, such as:

- The number of samples reaching the node is below a threshold $N_{\min}$;
- The uncertainty about the output variable, over the samples reaching the node, is below a threshold $U_{\min}$;
- The optimal test is not statistically significant, according to some statistical test.

On the other side, the post-pruning procedure consists in fully developing a first tree $\mathcal{T}_1$ from the learning sample and then computing a sequence of trees $\{\mathcal{T}_2, \mathcal{T}_3, \ldots\}$ such that $\mathcal{T}_i$ is a pruned version of $\mathcal{T}_{i-1}$. The prediction error of each tree is then calculated on an independent set of samples and the tree that leads to the lowest prediction error is selected (Breiman *et al.*, 1984).

The main drawback of the post-pruning procedure is that an independent set of samples is needed, while the main drawback of pre-pruning is that the optimal value of the parameter related to the chosen stop-splitting criterion ($N_{\min}$, $U_{\min}$, the significance level) is dependent on the considered problem.

Besides pruning, ensemble methods constitute another way of avoiding overfitting. These methods are described in the following section.

2.2.2 Ensemble methods

Single trees are usually very much improved by ensemble methods, which aggregate the predictions of several trees, either by an average (for a regression problem) or by a majority vote (for a classification problem). The goal of ensemble methods is to use diversified models to reduce the variance of a learning algorithm. In the case of a tree, the variance comes mostly from the choices, made at each split node, of the input variable and the cut-point used for the test. The tree-based ensemble methods that we used in our work rely on randomization to generate diversity among the different models. These methods are Bagging (Breiman, 1996), Random Forests (Breiman, 2001), and Extra-Trees (Geurts *et al.*, 2006a).

Bagging In the Bagging (for "Bootstrap AGGregatING") algorithm, each tree of the ensemble is built from a bootstrap replica, i.e. a set of samples obtained by N random samplings with replacement in the original learning sample. The choices of the variable and of the cut-point at each test node are thus implicitly randomized via the bootstrap sampling.

Random Forests This method adds an extra level of randomization compared to the Bagging. In a Random Forests ensemble, each tree is built from a bootstrap sample of the original learning sample and at each test node, K variables are selected at random (without replacement) among all candidate attributes before determining the best split.

Extra-Trees In the Extra-Trees (for "EXTremely RAndomized Trees") method, each tree is built from the original learning sample but at each test node, the best split is determined among K random splits, each determined by randomly selecting one input variable (without replacement) and a cut-point.

2.2.3 Parameters

Tree-based (ensemble) methods have several parameters whose values must be defined by the user:

- The parameter related to the chosen stop-splitting criterion, such as N_{min}, the minimal number of samples required for a node to become a test node. Increasing the value of N_{min} results in smaller trees and hence models with a higher bias and a lower variance. Its optimal value depends on the level of noise contained in the learning sample. The noisier the data, the higher the optimal value of N_{min}. Usually, N_{min} is fixed to 1 for ensemble methods, so that each tree of the ensemble is fully developed.

- K, the number of input variables that are randomly chosen at each node of a tree. This parameter thus determines the level of randomization of the trees. A smaller value of K results in more randomized trees. The optimal value of K is problem-dependent, but empirical validations performed by Geurts *et al.* (2006a) show that $K = \sqrt{m}$ and $K = m$, where m is the number of input variables, are near-optimal on classification and regression problems respectively.

- T, the number of trees in an ensemble. It can be shown that the higher the number of trees, the lower the generalization error (Breiman, 2001). Therefore, the chosen value of T is a compromise between model accuracy and computing times.

2.2.4 Variable importance measures

One interesting characteristic of tree-based methods is the possibility to compute from a tree an importance score for each input variable. This score measures the relevance of a variable for the prediction of the output. Several variable importance measures have been proposed in the literature for tree-based methods. For a classification problem, we considered in our work a measure that, at each test node $\mathcal{N}$, computes the total reduction of the class entropy due to the split (Breiman *et al.*, 1984; Friedman, 2001):

$$I(\mathcal{N}) = \#S.H_Y(S) - \#S_t.H_Y(S_t) - \#S_f.H_Y(S_f). \tag{2.11}$$

In the case of regression, the importance measure that we used computes the total reduction of the variance of the output at each test node $\mathcal{N}$ (Breiman *et al.*, 1984):

$$I(\mathcal{N}) = \#S.\mathrm{Var}_Y(S) - \#S_t.\mathrm{Var}_Y(S_t) - \#S_f.\mathrm{Var}_Y(S_f). \tag{2.12}$$

For a single tree, the overall importance s_i of one variable X_i is then computed by summing the I values of all tree nodes where this variable is used to split:

$$s_i = \sum_{k=1}^{p} I(\mathcal{N}_k).f(\mathcal{N}_k, X_i), \tag{2.13}$$

where p is the number of test nodes in the tree and $\mathcal{N}_k$ denotes the kth test node. $f(\mathcal{N}_k, X_i)$ is a function that is equal to one if X_i is the variable selected at node $\mathcal{N}_k$ and zero otherwise. The features that are not selected at all thus obtain an importance value of zero and those that are selected close to the root node of the tree typically obtain high scores. Attribute importance measures can be easily extended to ensembles, simply by averaging importance scores over all trees of the ensemble. The resulting importance measure is then even more reliable because of the variance reduction effect resulting from this averaging.

In the context of the Random Forests method, Breiman (2001) proposed an alternative procedure to compute the importance of a variable. For each tree that was learned, the procedure consists in computing the prediction accuracy of the tree on the out-of-bag samples (i.e. the training instances that were not present in the bootstrap sample used to build the tree), before and after randomly permuting the values of the corresponding variable in these samples. The reduction of the tree accuracy that is obtained after the permutation is then computed, and the importance of the variable is given by the average accuracy reduction over all the trees of the ensemble. While this procedure has some advantages with respect to the entropy

and variance reduction-based measures (Strobl *et al.*, 2007), it gives in most practical applications very similar results while being much more computationally demanding. Furthermore, it does not extend to methods that do not consider bootstrap sampling, like the Extra-Trees.

A property of measure (2.11) is its additivity: the sum of entropy reductions brought by the different splits of a classification tree $\mathcal{T}$ is equal to the reduction of the class entropy provided by the tree, defined as the total mutual information $I_Y^{\mathcal{T}}$ brought by the tree about the classification variable (Wehenkel, 1998):

$$\sum_{k=1}^{p} I(\mathcal{N}_k) = N.H_Y(LS') - N.H_{Y|\mathcal{T}}(LS') = N.I_Y^{\mathcal{T}}, \tag{2.14}$$

where LS' is the learning sample from which $\mathcal{T}$ was built (i.e. the original learning sample for the Extra-Trees method and a bootstrap sample for the Bagging and Random Forests methods), N is the size of LS', and $H_{Y|\mathcal{T}}(LS')$ is the residual entropy of the tree in LS', defined by:

$$H_{Y|\mathcal{T}}(LS') = \sum_{k=1}^{q} P(S_k|LS')H_Y(S_k), \tag{2.15}$$

where q is the number of terminal nodes in $\mathcal{T}$ and S_k is the subset of LS' that reaches the kth terminal node. Given the definition (2.13) of the variable importance, we have the following equalities:

$$\sum_{i=1}^{m} s_i = \sum_{k=1}^{p} I(\mathcal{N}_k) = N.I_Y^{\mathcal{T}}. \tag{2.16}$$

In the case of unpruned trees (as they are normally in the case of Bagging, Random Forests, and Extra-Trees), $H_{Y|\mathcal{T}}(LS')$ is usually very close to zero[1] and hence the mutual information $I_Y^{\mathcal{T}}$ is very close to the initial total entropy of the classification variable in LS'. Therefore, we have:

$$\sum_{i=1}^{m} s_i \approx N.H_Y(LS'). \tag{2.17}$$

Similarly, the importance measure used in the case of regression and based on variance reduction (2.12) is such that the sum of importances of all variables is very close to the total initial variance of the output:

$$\sum_{i=1}^{m} s_i \approx N.\mathrm{Var}_Y(LS'). \tag{2.18}$$

[1] $H_{Y|\mathcal{T}}(LS')$ is not strictly equal to zero because of the noise that may be contained in the training data. It might indeed happen that some samples with exactly the same values of input variables have different output values.

The sum of the importances, for a single tree as well as for an ensemble of trees, is therefore usually almost constant for a given output distribution (and learning sample size).

2.3 Support vector machines

The support vector machine (SVM) is a supervised learning method from the family of kernel-based algorithms (Boser *et al.*, 1992). A kernel is a function $K(\mathbf{x}_k, \mathbf{x}_{k'})$ that measures the similarity between two samples $\mathbf{x}_k$ and $\mathbf{x}_{k'}$, by associating a real number to this pair of samples. In our research, we considered the kernel only in its simplest form, which is the *linear* kernel, defined by the scalar product of the vectors representing the two samples:

$$K(\mathbf{x}_k, \mathbf{x}_{k'}) = \mathbf{x}_k^\top \mathbf{x}_{k'}. \tag{2.19}$$

We call *linear SVM* the SVM based on the linear kernel. The rest of this section describes this algorithm, first in the case of a classification problem, then in the case of regression.

2.3.1 Linear SVMs for binary classification

Hard-margin SVM

Let us assume that the learning sample is given by

$$LS = \{(\mathbf{x}_k, y_k)\}_{k=1}^N, \tag{2.20}$$

where $y_k \in \{-1, 1\}$ and $\mathbf{x}_k = (x_k^1, \ldots, x_k^m)^\top \in \mathbb{R}^m$. We further assume that LS is linearly separable in feature space, i.e. there exists a hyperplane that correctly separates the samples of LS belonging to different classes, as shown in Figure 2.3.

The goal of a linear SVM is to find a model in the form:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{2.21}$$

where $\mathbf{w} = (w_1, \ldots, w_m)^\top \in \mathbb{R}^m$ and $b \in \mathbb{R}$. The class of a new sample characterized by an input $\mathbf{x}$ is then predicted according to the sign of $f(\mathbf{x})$.

A condition related to the hyperplane f is that it must classify the samples of LS without any error:

$$y_k f(\mathbf{x}_k) > 0, \forall k. \tag{2.22}$$

As multiple hyperplanes satisfy this condition, the idea is to select the one that will lead to the smallest generalization error. A good candidate is the

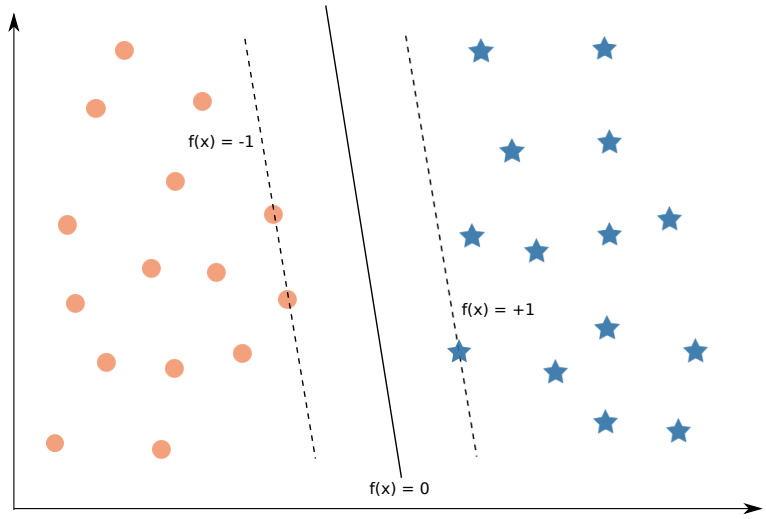


Figure 2.3: **Example of a linearly separable learning sample in two dimensions.** The plain line represents the hyperplane that separates the samples with the highest margin. Points located on the dashed lines are the support vectors.

hyperplane that maximizes the so-called *margin*, i.e. the distance from the hyperplane to the closest sample. Indeed it can be shown that the upper bound of the generalization error is on the order of $O(\frac{1}{\gamma})$, where γ is the margin (Shawe-Taylor and Cristianini, 2004). Furthermore, finding the hyperplane with the highest margin yields a convex optimization problem. In Figure 2.3, the maximum margin hyperplane is represented by the plain line.

The margin is given by:

$$\gamma = \min_k \frac{|f(\mathbf{x}_k)|}{\|\mathbf{w}\|_2} = \min_k \frac{y_k f(\mathbf{x}_k)}{\|\mathbf{w}\|_2}. \tag{2.23}$$

This leads to the following optimization problem:

$$\max_{\mathbf{w},b} \min_k \frac{y_k(\mathbf{w}^\top \mathbf{x}_k + b)}{\|\mathbf{w}\|_2}. \tag{2.24}$$

As rescaling $\mathbf{w}$ and b by some constant κ does not change the value of the margin, we can arbitrary set:

$$y_k(\mathbf{w}^\top \mathbf{x}_k + b) = 1, \tag{2.25}$$

for the samples $\mathbf{x}_k$ that are the closest to the hyperplane. These samples are called the *support vectors* and correspond to the points located on the dashed lines in Figure 2.3. Therefore, all the samples satisfy the condition:

$$y_k(\mathbf{w}^\top \mathbf{x}_k + b) \geq 1, \ \forall k. \tag{2.26}$$

The optimization problem then only amounts to the minimization of $\|\mathbf{w}\|_2$, or equivalently:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2, \tag{2.27}$$

subject to constraints (2.26). This constrained optimization problem can be solved by forming the Lagrangian function:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{k=1}^{N} \alpha_k (y_k(\mathbf{w}^\top \mathbf{x}_k + b) - 1), \tag{2.28}$$

where $\alpha_k \geq 0$ are the Lagrange multipliers. This function has to be minimized with respect to $\mathbf{w}$ and b, and maximized with respect to $\boldsymbol{\alpha}$. By setting to zero the derivatives of $L(\mathbf{w}, b, \boldsymbol{\alpha})$ with respect to $\mathbf{w}$ and b, we obtain the following conditions:

$$\begin{cases} \mathbf{w} = \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k, \\ \sum_{k=1}^{N} \alpha_k y_k = 0. \end{cases} \tag{2.29}$$

Substituting the equalities (2.29) in the Lagrangian function yields the *dual* maximization problem:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j, \tag{2.30}$$

subject to conditions:

$$\begin{cases} \alpha_k \geq 0, \\ \sum_{k=1}^{N} \alpha_k y_k = 0. \end{cases} \tag{2.31}$$

Formulating the optimization problem in this dual representation brings some advantages. The number of variables to optimize is now N, i.e. the number of Lagrange multipliers α_k, instead of $m + 1$. This is thus computationally efficient for the datasets in which the number of samples is much lower than the number of variables. Furthermore, the solution of the dual problem is usually sparse, as a significant number of α_k are equal to zero.

The predictive model can be expressed in terms of the parameters α_k by using the equalities (2.29) in Equation (2.21), which gives:

$$f(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k^\top \mathbf{x} + b. \tag{2.32}$$

Therefore, only the samples $\mathbf{x}_k$ of LS for which $\alpha_k \neq 0$ are used in the predictive model. It can be shown that these samples actually correspond to the support vectors.

Soft-margin SVM

So far, we searched for a hyperplane that perfectly separates the samples of different classes. However, the distribution of the classes may overlap in practice. In this case, a perfect separation of the samples of LS can lead to a poor generalization error, or simply does not exist.

A variant of the SVM was thus proposed, which allows some of the samples to be misclassified (Cortes and Vapnik, 1995). This variant introduces a *slack variable* ξ_k for each sample $\mathbf{x}_k$ and the classification constraints of Equation (2.26) are replaced by:

$$y_k(\mathbf{w}^\top \mathbf{x}_k + b) \geq 1 - \xi_k, \ \forall k, \tag{2.33}$$

where $\xi_k \geq 0$. Samples for which $\xi_k = 0$ are correctly classified and located at a distance from the hyperplane that is greater or equal to the margin. Samples for which $0 < \xi < 1$ are correctly classified but are located at a distance from the hyperplane that is lower than the margin. Samples for which $\xi = 1$ are on the hyperplane. Samples for which $\xi > 1$ are misclassified. The goal of the soft-margin SVM is still to maximize the margin, but also to softly penalize the samples that are on the wrong side of the hyperplane. The optimization problem thus amounts to:

$$\min_{\mathbf{w},\boldsymbol{\xi}} C \sum_{k=1}^{N} \xi_k + \frac{1}{2}\|\mathbf{w}\|_2^2, \tag{2.34}$$

subject to constraints (2.33), with $\xi_k \geq 0$. The constant $C > 0$ is the (inverse) regularization parameter that controls the trade-off between the minimization of the slack variables and the maximization of the margin (see Figure 2.4). When $C \to \infty$, the optimization problem is equivalent to the hard-margin problem (2.27).

Similarly to the hard-margin case, a dual representation of the problem can be formulated. The maximization problem is actually identical:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j, \tag{2.35}$$

but constraints on the parameters α_k are different:

$$\begin{cases} 0 \leq \alpha_k \leq C, \\ \sum_{k=1}^{N} \alpha_k y_k = 0. \end{cases} \tag{2.36}$$

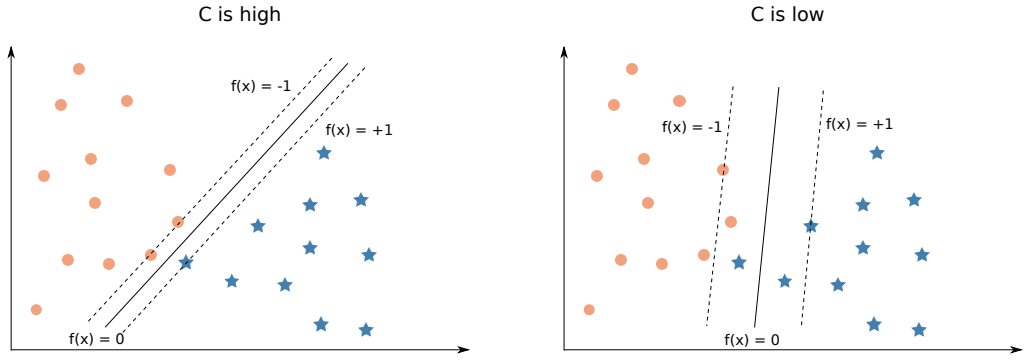


Figure 2.4: **Effect of the soft-margin SVM parameter C on the optimal hyperplane.** A high value of C highly penalizes the classification errors on the instances of the learning sample, while a low value of C allows some instances to be misclassified and increases the margin. In both plots, the plain line represents the optimal hyperplane and the dashed lines are on the margin. Points that are located either on the margin or inside the margin contribute to the predictive model.

We also have the same predictive model as in the hard-margin case:

$$f(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k^\top \mathbf{x} + b. \tag{2.37}$$

Only samples for which $\alpha_k \neq 0$ contribute to the predictive model. Samples for which $\alpha_k < C$ lie on the margin, while samples for which $\alpha_k = C$ lie inside the margin and can be either correctly classified ($\xi < 1$) or misclassified ($\xi > 1$).

2.3.2 Linear SVMs for regression

In the case of regression, the learning sample is given by:

$$LS = \{(\mathbf{x}_k, y_k)\}_{k=1}^{N}, \tag{2.38}$$

where $\mathbf{x}_k = (x_k^1, \ldots, x_k^m)^\top \in \mathbb{R}^m$ and $y_k \in \mathbb{R}$.

The goal of a linear SVM is to find a predictive model in the form:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{2.39}$$

that minimizes (Vapnik, 1995):

$$C \sum_{k=1}^{N} E_\epsilon(f(\mathbf{x}_k) - y_k) + \frac{1}{2}\|\mathbf{w}\|_2^2, \tag{2.40}$$

where E_ϵ is an *ϵ-insensitive error function*:

$$E_\epsilon(f(\mathbf{x}) - y) = \begin{cases} 0, & \text{if } |f(\mathbf{x}) - y| < \epsilon, \\ |f(\mathbf{x}) - y| - \epsilon, & \text{otherwise,} \end{cases} \tag{2.41}$$

with $\epsilon > 0$. Minimizing the first term in Equation (2.40) implies to find a model $f(\mathbf{x})$ such that all samples of LS lie inside an *ϵ-tube*:

$$f(\mathbf{x}_k) - \epsilon \leq y_k \leq f(\mathbf{x}_k) + \epsilon, \ \forall k. \tag{2.42}$$

Slack variables can be introduced, as for the soft-margin SVM for classification, except that now two slack variables $\xi_k \geq 0$ and $\widehat{\xi}_k \geq 0$ are used for each sample $\mathbf{x}_k$. Introducing the slack variables allows some of the samples to lie outside the ϵ-tube. Constraints (2.42) are replaced by:

$$y_k \leq f(\mathbf{x}_k) + \epsilon + \xi_k, \tag{2.43}$$

$$y_k \geq f(\mathbf{x}_k) - \epsilon - \widehat{\xi}_k, \tag{2.44}$$

and the optimization problem amounts to minimize:

$$C \sum_{k=1}^{N} (\xi_k + \widehat{\xi}_k) + \frac{1}{2} \|\mathbf{w}\|_2^2, \tag{2.45}$$

subject to these constraints.

By exploiting a Lagrangian function, this problem can be rewritten into the following dual problem:

$$\max_{\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}}} W(\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \widehat{\alpha}_i)(\alpha_j - \widehat{\alpha}_j) \mathbf{x}_i^\top \mathbf{x}_j - \epsilon \sum_{k=1}^{N} (\alpha_k + \widehat{\alpha}_k) + \sum_{k=1}^{N} (\alpha_k - \widehat{\alpha}_k) y_k, \tag{2.46}$$

where the Lagrange multipliers $\alpha_k \geq 0$ and $\widehat{\alpha}_k \geq 0$ (corresponding respectively to the slack variables ξ_k and $\widehat{\xi}_k$) must satisfy the conditions:

$$\begin{cases} 0 \leq \alpha_k \leq C, \\ 0 \leq \widehat{\alpha}_k \leq C, \\ \sum_{k=1}^{N} (\alpha_k - \widehat{\alpha}_k) = 0. \end{cases} \tag{2.47}$$

The weight vector $\mathbf{w}$ is then given by:

$$\mathbf{w} = \sum_{k=1}^{N} (\alpha_k - \widehat{\alpha}_k) \mathbf{x}_k. \tag{2.48}$$

Substituting equality (2.48) in Equation (2.39) leads to the following predictive model:

$$f(\mathbf{x}) = \sum_{k=1}^{N} (\alpha_k - \widehat{\alpha}_k) \mathbf{x}_k^\top \mathbf{x} + b. \tag{2.49}$$

The support vectors are the samples for which either $\alpha_k \neq 0$ or $\widehat{\alpha}_k \neq 0$. It can be shown that these samples correspond to those lying either on the boundaries or outside the ϵ-tube. All samples within the tube have both α_k and $\widehat{\alpha}_k$ equal to zero.

2.3.3 Parameters

The soft-margin SVM method comprises the parameter $C > 0$ that controls the trade-off between the regularization of the predictive model and the prediction accuracy on the instances of the learning sample. The effect of the value of C is shown in Figure 2.4 in the case of classification. When C is large, prediction errors are highly penalized. When C is decreased, the model is more regularized (i.e. $\|\mathbf{w}\|_2$ is decreased).

The SVM algorithm for regression has an additional parameter, which is the tube size ϵ used in the loss function. A too small value of ϵ would result in a high complexity of the model (the number of support vectors is large) and an overfitting of the training data, while a too large value of ϵ would result in underfitting. The optimal value of ϵ depends on the level of noise in the learning sample.

2.3.4 Variable importance measures

The simplest procedure for measuring the importance of an input variable X_i with linear SVMs, which is the one that we used in our work, is to calculate the absolute value of its corresponding weight w_i in the predictive model:

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i x_i + b. \tag{2.50}$$

As indicated in the previous sections, the elements of the weight vector $\mathbf{w}$ can be calculated by optimizing a Lagrangian function. For a classification problem, the weight of the ith input variable, $i = 1, \ldots, m$ is given by:

$$w_i = \sum_{k=1}^{N} \alpha_k y_k x_k^i, \tag{2.51}$$

where α_k is the Lagrange multiplier related to the kth instance of the learning sample, and y_k and x_k^i are respectively the values of the output and of the ith input variable in that instance. For a regression problem, we have:

$$w_i = \sum_{k=1}^{N} (\alpha_k - \widehat{\alpha}_k) x_k^i, \tag{2.52}$$

where α_k and $\widehat{\alpha}_k$ are the Lagrange multipliers related to the kth sample.

The SVM method is sensitive to the way the input variables are scaled (Ben-Hur *et al.*, 2008). Therefore, the weight of a variable in a SVM model depends in some manner on the range of values of this variable in the learning sample. In order to avoid this dependency, we normalize the values of the variables so that they are all comprised between -1 and +1 in the learning sample, before applying the SVM method (Chang and Lin, 2011).

Alternative methods for ranking the variables using SVMs were proposed. A well-known approach, SVM-RFE, is based on the Recursive Feature Elimination (RFE) procedure (Guyon *et al.*, 2002). SVM-RFE uses a backward elimination strategy that iteratively removes the least relevant variables (Kohavi and John, 1997). A first SVM model is learned using all m input variables and these variables are ranked according to the absolute values of the weights w_i. The least important variable is then removed and a new SVM model is learned from the remaining variables. This procedure is repeated until all the variables are removed. The final ranking that is returned by SVM-RFE consists in the inverse sequence of the eliminated variables, i.e. the variable that is removed at the kth iteration is ranked at position $m-k+1$. Note that other ranking methods using backward elimination like SVM-RFE were proposed by Rakotomamonjy (2003).

Compared to the baseline method, which ranks the features according to their individual relevance $|w_i|$, SVM-RFE allows to obtain a *feature subset ranking*. In such a ranking, the variables that are at the top of the list are not necessarily individually relevant, but become relevant when they are combined to each other. We however did not use the SVM-RFE procedure in our work. Abeel *et al.* (2010) showed that this method returns variable rankings that are clearly less stable than those returned by the baseline method. Furthermore, SVM-RFE has a high computational cost: to rank m variables, m SVM models have to be learned. Note that, in order to reduce the computational times, the algorithm can be modified to remove more than one variable at each iteration. However, this introduces an additional parameter, which is the number of variables to remove.

Table 2.1: **Contingency table.**

		Actual	
		1	0
Predicted	1	TP	FP
	0	FN	TN

TP: Number of true positives, FP: Number of false positives, TN: Number of true negatives, FN: Number of false negatives.

2.4 Performance metrics

The problems of feature selection and network inference can be seen as binary classification problems. The goal of a feature selection algorithm is to state whether a feature is relevant (1) or irrelevant (0), while a network inference task amounts to deciding whether a regulatory link is present (1) or absent (0) in the network. Feature selection and network inference algorithms can thus be evaluated using standard performance metrics from machine learning (assuming that the truth is known): *precision*, *recall* (also called *true positive rate* or TPR), and *false positive rate* (FPR). Given the contingency table in Table 2.1, these metrics are defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.53}$$

$$\text{recall (or TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.54}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{2.55}$$

The precision is thus the proportion of true positives among all predictions, the recall is the percentage of truly positives that are retrieved, and the false positive rate is the percentage of truly negatives that are considered positives.

Many feature selection (resp. network inference) algorithms work first by providing a ranking of the features (resp. regulatory links), from the most confident to the less confident according to an importance score. A threshold is then selected on this ranking to obtain a subset of features (resp. network). To evaluate such a ranking independently of the choice of a specific threshold, we can use both precision-recall (PR) and receiver operating characteristic (ROC) curves (see Figure 2.5). The PR curve plots for varying thresholds the precision versus the recall, whereas a ROC curve plots the true positive

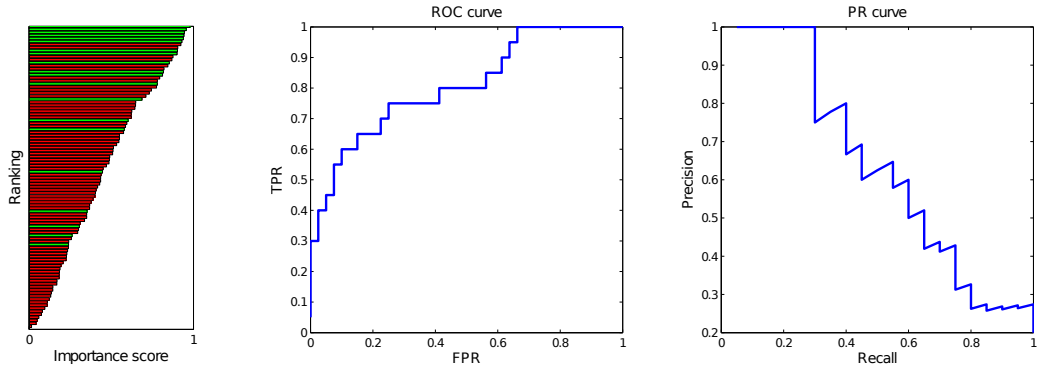


Figure 2.5: **PR and ROC curves.** A list of features (or regulatory links), ranked according to their importance score, is represented on the left. Green elements are the truly positives while red elements are the truly negatives. The ROC curve (middle) plots for varying thresholds on the importance scores the true positive rate versus the false positive rate. The PR curve (right) plots for varying thresholds the precision versus the recall. A perfect ranking yields a PR curve and a ROC curve that both have an area under the curve equal to one.

rate versus the false positive rate. A perfect ranking, i.e. a ranking where all the positives are located at the top of the list, yields a PR curve and a ROC curve that both have an area under the curve equal to one.

For problems where the number of negatives is much higher than the number of positives, which is typically the case of feature selection and network inference problems, it is usually advised to use the PR curve rather than the ROC curve to assess the performance of an algorithm (see e.g. Davis and Goadrich, 2006). Indeed, a large change in the number of false positives results in only a small change in the false positive rate used to compute the ROC curve. On the other side, the PR curve does not take the number of true negatives into account and thus enables a better visualization of the performance at the top of the ranking, i.e. the part of the ranking in which we are the most interested in.

Part II

Feature selection

3

Exploiting tree-based variable importances for feature selection

This chapter introduces a statistical procedure, called CER, which is based on permutation tests for extracting a subset of truly relevant variables from importance rankings derived from tree-based multivariate methods. This method is motivated with respect to a naive application of the permutation-based procedure to estimate the FDR (called pFDR). Several experiments are performed to illustrate the main features of both methods. A thorough evaluation of the CER and pFDR procedures, including their comparison with several other methods, is performed in Chapter 4. After the introduction of Section 3.1, Sections 3.2 and 3.3 respectively define the notion of feature relevance and the problem of selecting relevant features from a ranking. Section 3.4 studies the pFDR procedure on synthetics datasets, highlighting its main properties when applied to tree-based variable importances. Section 3.5 describes the proposed alternative approach (CER), its permutation-based estimation procedure, and the empirical results obtained with the same synthetic datasets. Section 3.6 shows some results on a real biological dataset. Finally, Section 3.7 concludes and gives a few directions for further research.

Contents

Publication related to this chapter:

Huynh-Thu, V. A., Wehenkel, L., and Geurts, P. (2008) Exploiting tree-based variable importances to selectively identify relevant variables. *JMLR Workshop and Conference Proceedings* **4**:60-73.

Peer-reviewed conference presentation:

Workshop on new challenges for feature selection in data mining and knowledge discovery, Antwerp (Belgium), 2008.

3.1 Introduction

Univariate hypothesis testing is widely used in the context of biomarker discovery in bioinformatics, where one seeks to identify variables (e.g. genes or genetic polymorphisms) that truly provide information about some biological condition of interest (e.g. disease status or treatment response). A classic procedure consists in applying a statistical test in order to compute a p-value for each variable of the considered problem and selecting variables that have a p-value lower than a chosen threshold. A disadvantage of univariate tests is that they can only identify variables that provide a significant amount of information about the output variable in isolation from the other inputs. Nowadays, when one seeks for multivariate interacting effects, one can resort to relevance scores provided by machine learning techniques such as tree-based methods. However, unlike the p-values returned by univariate tests, these relevance scores are usually not statistically interpretable. This lack of interpretability prevents the wide adoption of these methods by practitioners and also makes the identification of the truly relevant variables among the top-ranked ones, i.e. the determination of a relevance threshold, a very difficult task in practice.

One possible way of identifying relevant variables using tree-based variable importance measures would be to mimic the procedures that are used in the context of multiple hypothesis testing of univariate statistics (see Ge *et al.*, 2003 for a review). For example, one commonly used approach consists in ranking the features according to a relevance score derived from a (univariate) hypothesis test, and estimating, for each score threshold, the rate of

false positives (called the false discovery rate, or FDR) among the variables that have a score greater than the threshold. In this context, the FDR is usually estimated in a non-parametric way by assessing the average scores derived when randomly permuting the output values of the dataset.

In this chapter, we present results of the assessment of this FDR estimation approach when the relevance scores are derived from tree-based (multivariate) models. We show empirically that this simple procedure typically overestimates the real FDR in an unpredictable way and thus can lead to unreliable selections of relevant subsets. We explain this by the fact that, contrary to the univariate case, the tree-based importance scores of different variables are not independent of each other. We then propose a novel alternative procedure for assessing the presence of irrelevant variables among a subset of variables top-ranked by tree-based methods. For a given importance threshold, this procedure first assumes that all and only those variables that have received an importance higher than the threshold are truly relevant, and then estimates the probability that any of the other variables would receive an importance higher than the given threshold. Experiments suggest that the latter quantity (estimated by an appropriately adapted permutation scheme) indeed allows to rather well identify an importance threshold below which the risk of having at least one false positive rapidly increases. The procedure may thus serve to identify the maximal subset of truly relevant variables among those proposed by the importance scoring method, in a more robust way than the FDR-based approach.

3.2 About feature relevance

Among the various definitions of the relevance of a variable that have been proposed in the literature, we follow those provided by Kohavi and John (1997), who claim that two degrees of relevance are required: weak and strong.

Formally, we assume that we have at our disposal a learning sample LS of N instances of input-output pairs drawn from some unknown probability distribution. There are m input variables denoted $X_i, i = 1, \ldots, m$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)^\top$ be an instance of the vector of random variables $\mathbf{X} = (X_1, X_2, \ldots, X_m)^\top$, with probability $P(\mathbf{X} = \mathbf{x})$[1]. Let $\mathbf{V}$ be a subset of $\mathbf{X}$ and $\mathbf{X}^{-i}$ the vector of all variables except X_i. Y denotes the output variable. An input feature X_i is *strongly relevant* if it brings additional information about Y conditionally to all the other input variables, i.e. if there exist some

[1]Note that the following definitions are only applicable to discrete variables, but can be extended to continuous variables by changing probabilities $P(\mathbf{X} = \mathbf{x})$ to $P(\mathbf{X} \leq \mathbf{x})$.

values x_i, y, and $\mathbf{v}_i$ for which $P(X_i = x_i, \mathbf{X}^{-i} = \mathbf{v}_i) > 0$, such that:

$$P(Y = y | X_i = x_i, \mathbf{X}^{-i} = \mathbf{v}_i) \neq P(Y = y | \mathbf{X}^{-i} = \mathbf{v}_i). \tag{3.1}$$

A strongly relevant variable is thus always informative for characterizing the output variable. By contrast, a feature X_i is *weakly relevant* if it is not strongly relevant and there exists a subset of features $\mathbf{V}$ that do not include X_i and such that X_i brings additional information about Y conditionally to $\mathbf{V}$, i.e. if there exist some values x_i, y, and $\mathbf{v}_i$ for which $P(X_i = x_i, \mathbf{V}_i = v_i) > 0$, such that:

$$P(Y = y | X_i = x_i, \mathbf{V}_i = v_i) \neq P(Y = y | \mathbf{V}_i = v_i). \tag{3.2}$$

A weakly relevant variable thus brings at most the same information about Y as one or several other (strongly or weakly) relevant variable(s). Weakly relevant variables are also said to be *redundant*. In this thesis, we consider a feature *relevant* if it is strongly or weakly relevant. Hence, a feature is *irrelevant* if it is independent of the output conditionally to any subset of the other variables (including the empty subset).

3.3 Feature selection from a ranking

In this part of the thesis, we focus on the problem of selecting relevant features from a ranking. We assume that we have a machine learning algorithm $\mathcal{A}(LS)$ that computes from the learning sample LS a feature ranking, typically derived from an importance score s_i for each input variable X_i. These scores are not supposed to be independent and no further assumption is made about $\mathcal{A}$. The goal is then to determine a value k such that the subset composed of the k top-ranked variables contains the highest possible number of truly relevant features.

When the sole information available about the problem under consideration is limited to a training sample of input-ouput pairs, it is generally not possible to exactly identify the maximal subset of relevant inputs. Thus, any feature selection algorithm is at risk of either missing some sought features (false negatives) or of erroneously selecting some truly non desired ones (false positives). Among the different possible sensitivity/specificity compromises, we aim at high specificity, i.e. at identifying subsets of relevant variables that are top-ranked, while maintaining the rate of false positives as small as possible. This type of compromise is typically sought in the context of biomarker discovery where molecular variables are selected for further analysis and biological insight, while aiming at a very low false positive rate, because of high costs of subsequent experiments (see e.g. Saeys *et al.*, 2007).

3.4 False discovery rate

We considered for the feature ranking algorithm $\mathcal{A}$ two machine learning methods based on ensembles of trees, namely Random Forests and Extra-Trees, which compute variable importances as sums of entropy reductions in the form (2.11) for classification problems. For these two methods, we used the default parameter setting, i.e. no pruning of the trees and the number K of variables randomly selected at each node fixed at the square root of m. We grew ensembles of $T = 100$ trees, unless specified otherwise.

Without loss of generality, we further assume that the features are numbered according to their importance scores, i.e.

$$s_1 \geq s_2 \geq \ldots \geq s_m. \tag{3.3}$$

For a given importance threshold s_i, we consider that all variables whose importance is greater than s_i are relevant and our concern is to estimate the expected rate of truly irrelevant features among these variables, the so-called *false discovery rate* (FDR) (Storey and Tibshirani, 2003).

More formally, for a given importance threshold s_i, the FDR is defined as:

$$\text{FDR}(s_i) = E\left[\frac{V(s_i)}{R(s_i)}\right], \tag{3.4}$$

where $R(s_i)$ is the number of variables considered relevant at threshold s_i and $V(s_i)$ is the number of these variables that are truly irrelevant. The expectation is taken over different random learning samples drawn from the joint distribution of the variables.

To select a subset of variables, one can then check the FDR for increasing values of the threshold s_i and choose the minimum value of s_i such that $\text{FDR}(s_i) \leq \alpha$, where α is typically small and reflects the risk one is ready to accept in terms of false positives when selecting the variables.

3.4.1 Estimation by random permutations

To estimate this FDR, we adopted the same approximations as Listgarten and Heckerman (2007). When the number of features is large, one can approximate the expectation of the ratio by the ratio of the expectations:

$$\text{FDR}(s_i) = E\left[\frac{V(s_i)}{R(s_i)}\right] \approx \frac{E[V(s_i)]}{E[R(s_i)]}. \tag{3.5}$$

$E[R(s_i)]$ can be simply approximated by the observed $R(s_i)$, i.e. the number of variables with an importance greater or equal to s_i in the original

data. $E[V(s_i)]$ is approximated by the expectation $E[V(s_i)|H_0]$ over the null distribution H_0 stating that all variables are truly irrelevant. In other words, $E[V(s_i)]$ is taken as the expected number of variables that get an importance greater or equal to s_i when none of them are truly relevant. We simulate this null hypothesis by applying the same tree-based method that was used to produce the original importances on datasets obtained from the original data by randomly permuting the output values. This permutation decorrelates the output variable from the inputs, making them all irrelevant, but keeps the dependencies that exist between the features in the original data. We call pFDR (for "permutation-based FDR") the FDR estimated using this permutation scheme.

The algorithm of Table 3.1 describes the procedure that we use to compute the pFDRs for all observed importance value thresholds in a learning sample. Note that we enforce the monotonicity of the pFDR values in step 3 of the algorithm, so that a variable can be selected only if all the variables ranked above are also selected.

3.4.2 Experiments on artificial data

Artificial problems

To validate feature selection methods in a context where relevant variables are perfectly known, we generated two artificial problems by adapting the MATLAB®[2] code originally used to produce the *Madelon* dataset for the NIPS 2003 feature selection challenge[3]. Both problems are binary classification problems with continuous input variables. In each problem, each class is composed of a number of Gaussian clusters that are placed at random on the vertices of a hypercube in a p-dimensional space, where p is the number of relevant variables.

- **Dataset-3-20:** This dataset is composed of 200 objects and 20 variables. The first three are really relevant, while the others are pure Gaussian noise. The problem is such that the third variable is only relevant in combination with the first two.

- **Dataset-50-1000:** The second (larger) dataset is composed of 2000 objects and 1000 variables. 50 variables are relevant, among which 6 have been directly used to define the output and 44 are linear combinations of these 6 variables (these latter are thus redundant given the

[2] http://www.mathworks.com/

[3] http://www.clopinet.com/isabelle/Projects/NIPS2003/

Table 3.1: **Permutation algorithm for the computation of the pFDR.**

Inputs: A learning sample LS, a ranking algorithm $\mathcal{A}$, a significance level α.

Compute variable importances from the original data:

$$\{s_1, \ldots, s_m\} = \mathcal{A}(LS),$$

where s_i is the importance of variable X_i. Assume, without loss of generality, that the variables are numbered according to their importance s_i, i.e. $s_1 \geq s_2 \geq \ldots \geq s_m$.

1. for $p = 1$ to P (typically $P = 1000$):
   (a) Generate LS^p from LS by randomly permuting the output values.
   (b) Compute variable importance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$.
   (c) Compute $V_i^p = \#\{k : s_k^p \geq s_i\}$, for $i = 1, \ldots, m$.

2. Then at s_i, the FDR is estimated by

$$\text{pFDR}_i = \frac{1}{P} \frac{\sum_{p=1}^{P} V_i^p}{i}, \text{ for } i = 1, \ldots, m.$$

3. Enforce monotonicity by setting:

$$\text{pFDR}_1^* \leftarrow \text{pFDR}_1,$$

$$\text{pFDR}_i^* \leftarrow \max(\text{pFDR}_{i-1}^*, \text{pFDR}_i), \text{ for } i = 2, \ldots, m.$$

4. Select variables X_i such that $\text{pFDR}_i^* \leq \alpha$.

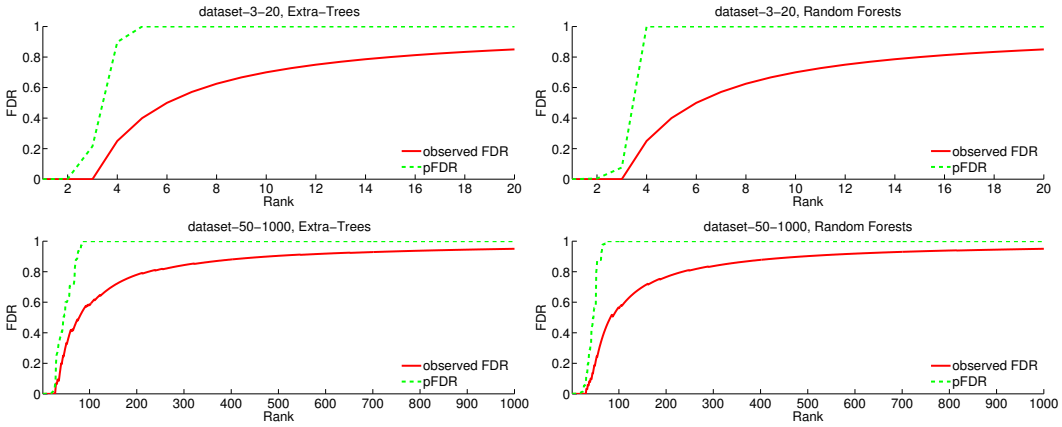


Figure 3.1: **pFDR and observed FDR for increasing rank.** Top on Dataset-3-20, bottom on Dataset-50-1000, left with Extra-Trees, right with Random Forests ($P = 1000$, $T = 100$, $K = \sqrt{m}$).

first 6 ones). The remaining 950 irrelevant variables are pure Gaussian noise.

Results

We applied the pFDR procedure on the two artificial problems described above. Since we perfectly knew the relevant variables for these two problems, we were also able to compute the *observed* FDR for a given subset of selected variables, i.e. the proportion of irrelevant variables among those selected. Figure 3.1 plots, for the two artificial datasets and the two tree-based methods, the pFDR calculated using the procedure of Table 3.1 and the observed FDR, both as a function of the rank of the variables. There is no important difference between Extra-Trees and Random Forests. In all cases, the pFDR overestimates the observed FDR. On the small dataset, both methods are able to find the three relevant variables (as illustrated by the fact that the observed FDR is equal to 0 until rank 3). However, the curve of the pFDR that already starts increasing for the third variable wrongly suggests that this variable is a false positive. On the larger dataset, using a typical threshold of 0.05 on the pFDR leads to the selection of 26 variables with Extra-Trees and 25 variables with Random Forests (all relevant in both cases), while the same threshold on the observed FDR would lead to 28 variables with Extra-Trees and 33 variables with Random Forests, with 5% of irrelevant among them.

3.4.3 Discussion

The overestimation of the FDR by the procedure of Table 3.1 can be explained, at least partially, by the fact that this procedure does not take into account the dependence between the importance values for different variables[4]. Indeed, as explained in Section 2.2.4, the sum of importances is roughly a constant for a given output distribution. In consequence, if a variable brings a lot of information about the output variable, there is much less left to be explained by the remaining variables, whether they are relevant or not. Thus, if a relevant variable receives a high importance, it potentially hides a less important but still relevant variable that may consequently receive an importance s_i that is small or even similar to that of an irrelevant variable in the permuted data. In this case, our estimation of $E[V(s_i)]$ from random permutations will be positively biased and thus our estimate of the FDR will be too conservative. This phenomenon is clearly apparent on the small artificial dataset, where the relevant variable X_3 gets an importance in the original ranking that is lower than the average importance obtained by the most important variable in the permuted data (see Figure 3.2).

Listgarten and Heckerman (2007) observed a similar effect when trying to estimate the false discovery rate among the edges predicted by a Bayesian network learning algorithm.

3.5 An alternative measure

In order to overcome this limitation of the FDR, we propose an alternative measure to be associated with each importance threshold s_i that takes into account the importances of the variables that are ranked above X_i. For a given importance threshold s_i, the procedure consists in computing the following conditional probability, which we call the *conditional error rate* (CER):

$$\text{CER}(s_i) = P(\max_{k=i,\dots,m} S_k^H \geq s_i | H_R^{1\to i-1}, H_I^{i\to m}), \tag{3.6}$$

where $H_R^{1\to i-1}$ denotes the hypothesis that all variables above X_i in the original ranking are relevant, $H_I^{i\to m}$ is the hypothesis that X_i and all the variables below X_i are irrelevant, and S_k^H $(k = 1, \dots, m)$ is the random variable denoting the importance of X_k under these two hypotheses. $\text{CER}(s_i)$ is thus the probability that at least one irrelevant variable among $m - i + 1$ gets

[4]Note that we are not talking here about the statistical dependence of the features that induces a dependence of their importances, however they are computed, but rather about the dependence between the importances that results from their joint computation by a multivariate approach.

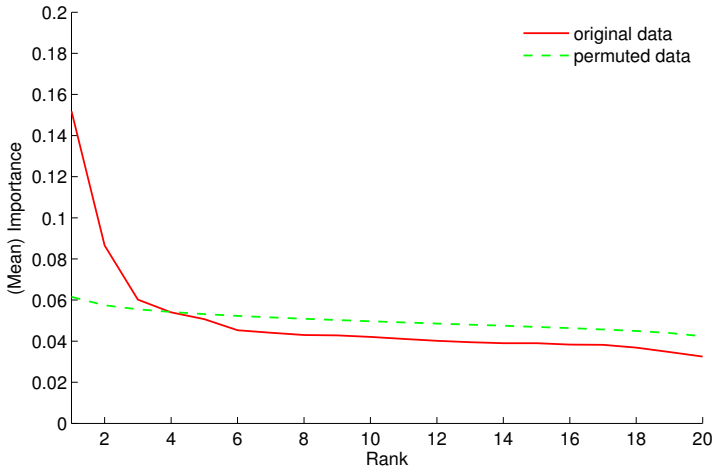


Figure 3.2: **Variable importance as a function of the rank.** Importance values were obtained with Extra-Trees on Dataset-3-20, from the original and the permuted data. In the latter case, importances were averaged over the P permutations ($P = 1000$, $T = 100$, $K = \sqrt{m}$).

an importance greater or equal to s_i, when these importances are computed under the assumption that variables $X_1, \ldots, X_{i-1}$ are all relevant.

This value can be interpreted as a measure of evidence against the hypothesis that all variables above the threshold s_i are relevant: a $\text{CER}(s_i)$ close to one means that it is very likely to observe an irrelevant variable with an importance above s_i while a $\text{CER}(s_i)$ close to zero means that it is very unlikely that an irrelevant variable could reach the threshold s_i. The limit between relevant and irrelevant variables in the ranking can then be determined by looking for the minimal threshold s_i such that $\text{CER}(s_i) \leq \alpha$, for some small value of α.

Since the CER tries to detect at least one false positive above the threshold, we expect it to evolve much more abruptly than the FDR and thus to indicate more clearly the risk of selecting some irrelevant variables in the ranking.

3.5.1 Estimation by random permutations

We propose to estimate the probabilities (3.6) by random permutations as well. $H_R^{1 \to i-1}$ is approximated by keeping the values of the output and of the first $i-1$ variables unchanged (which amounts to considering that variables X_1 to X_{i-1} are truly relevant), while hypothesis $H_I^{i \to m}$ is simulated

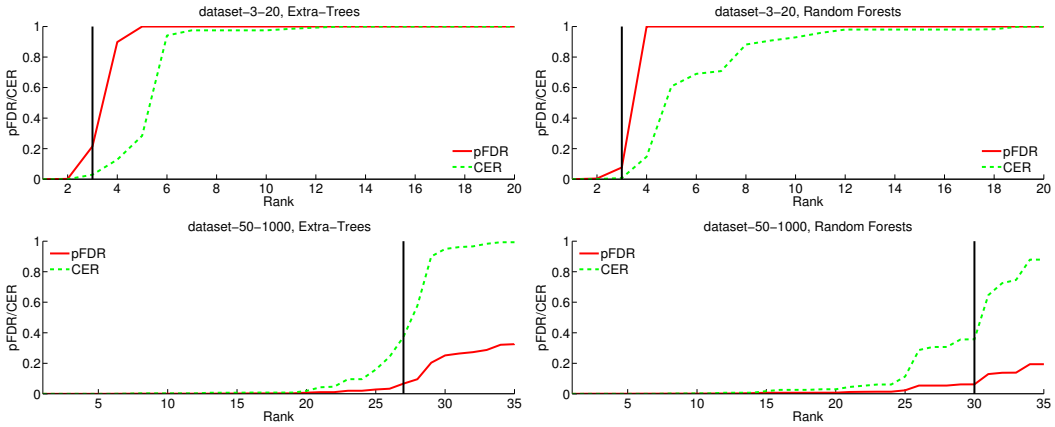


Figure 3.3: **CER and pFDR for increasing rank.** Top on Dataset-3-20, bottom on Dataset-50-1000, left with Extra-Trees, right with Random Forests. The vertical line indicates the rank beyond which the observed FDR is greater than zero and the position where the observed CER switches from 0 to 1 ($P = 1000$, $T = 100$, $K = \sqrt{m}$).

by randomly permuting the values of the variables X_i to X_m, that have an importance equal or smaller than s_i in the original ranking. To adhere as much as possible to the original joint distribution of the variables, they are furthermore permuted jointly, i.e. using the same permutation vector. The resulting procedure is described in Table 3.2.

Because the importances of the variables X_i to X_m in the random permutations are computed with the values of the variables X_1 to X_{i-1} being unchanged, these importances should not suffer from the same bias as in the estimation of the FDR. We thus expect that the algorithm of Table 3.2 will produce unbiased estimates of the CER and thus be more adapted to highlight the true frontier between relevant and irrelevant features than the procedure of Table 3.1 based on the pFDR.

3.5.2 Experiments on artificial data

Figure 3.3 compares the CER as estimated by the procedure of Table 3.2 with the pFDR as calculated by the procedure of Table 3.1, on the two datasets described in Section 3.4.2 and with the two ensemble methods.

On the small dataset, the CER correctly starts increasing at the fourth variable. It thus gives more chance than the pFDR to the third variable to be selected.

On the larger dataset, the transition region between low and high CER

Table 3.2: **Permutation algorithm for the computation of the CER.**

Inputs: A learning sample LS, a ranking algorithm $\mathcal{A}$, a significance level α.

Compute variable importances from the original data:

$$\{s_1, \ldots, s_m\} = \mathcal{A}(LS),$$

where s_i is the importance of variable X_i. Assume, without loss of generality, that the variables are numbered according to their importance s_i, i.e. $s_1 \geq s_2 \geq \ldots \geq s_m$.

1. for $i = 1$ to m :
   (a) for $p = 1$ to P (typically $P = 1000$):
      - Generate LS^p from LS by keeping the values of the output and of $X_1, \ldots, X_{i-1}$ fixed, and permuting the values of $X_i, \ldots, X_m$ with the same permutation vector.
      - Compute variable importance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$.

   (b) Then at s_i, the conditional probability (3.6) is estimated by:

$$\text{CER}_i = \frac{\#\{p : \max_{j=i,\cdots,m} s_j^p \geq s_i\}}{P}, \text{ for } i = 1, \ldots, m.$$

2. Enforce monotonicity by setting:

$$\text{CER}_1^* \leftarrow \text{CER}_1,$$

$$\text{CER}_i^* \leftarrow \max(\text{CER}_{i-1}^*, \text{CER}_i), \text{ for } i = 2, \ldots, m.$$

3. Select variables X_i such that $\text{CER}_i^* \leq \alpha$.

is quite well centered at the point where irrelevant variables start appearing in the ranking (indicated by the vertical line). Setting a small threshold $\alpha = 0.05$ on the CER and looking for the last variable X_i in the ranking such that $\text{CER}(s_i) \leq \alpha$ leads to the selection of 22 variables with Extra-Trees and 21 variables with Random Forests. In both cases, all the selected variables are truly relevant but the selection remains however quite conservative (with Extra-Trees, the first 27 variables are relevant and with Random Forests, the first 30). This is because the transition region between low and high CER is quite large (especially for Random Forests).

3.5.3 Link with FWER-based univariate procedures

The CER has a nice interpretation when the importance scores are actually derived from univariate statistics and a variable is, by definition, irrelevant when it satisfies the null hypothesis H_0 of the corresponding statistical test (e.g. s_i is the statistic t associated with variable X_i). Indeed, in this case, importances s_i are computed independently of each other and probability (3.6) can thus be rewritten as follows:

$$\begin{aligned} P(\max_{k=i,\dots,m} S_k^H \geq s_i | H_R^{1\to i-1}, H_I^{i\to m}) &= P(\max_{k=i,\dots,m} S_k^H \geq s_i | H_I^{i\to m}) \\ &= P(\max_{k=i,\dots,m} S_k^H \geq s_i | H_I^{1\to m}), \quad (3.7) \end{aligned}$$

where $H_I^{1\to m}$ is the hypothesis that all variables satisfy the null hypothesis H_0. Expression (3.7) corresponds precisely to the definition of Westfall and Young's *stepdown maxT adjusted p-values* (Ge *et al.*, 2003; Westfall and Young, 1993). The direct application of the procedure of Table 3.2 in this case thus produces estimates of these adjusted p-values by random permutations. Under some conditions about the distribution of the statistic, Westfall and Young (1993) showed that selecting all variables such that their adjusted p-values is lower than some threshold α guarantees that the family wise error rate, or FWER (i.e. the probability to include at least one false positive among the selected variables) is lower than α. In our context, however, given the strong dependency between the importances that are computed by tree-based methods, it is not clear whether this guarantee still applies.

3.6 Experiments on a real dataset

To highlight the behavior of both measures on a real problem, we ran experiments on a biological dataset. The goal of the study of Callow *et al.* (2000) was the identification of the genes with altered expression in the livers of

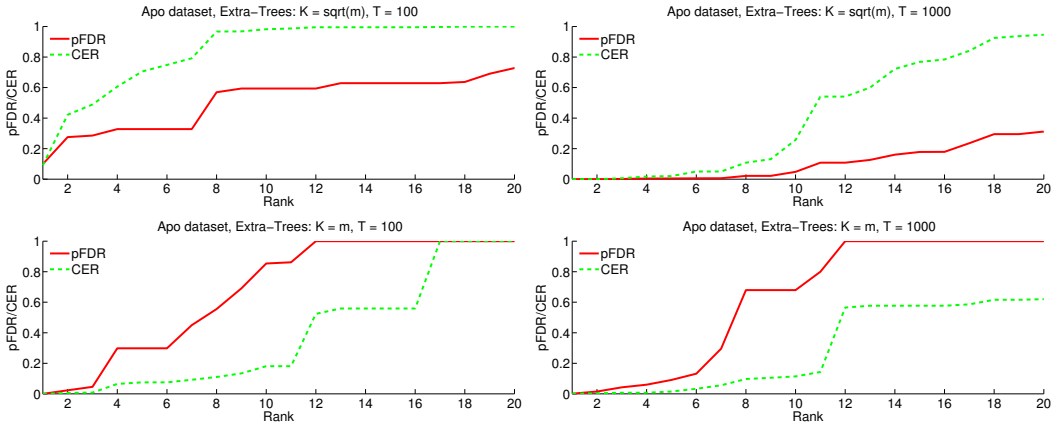


Figure 3.4: **CER and pFDR for increasing variable importance rank (Extra-Trees).** Top left with ($K = \sqrt{m}, T = 100$), top right with ($K = \sqrt{m}, T = 1000$). Bottom left with ($K = m, T = 100$), bottom right with ($K = m, T = 1000$). ($P = 1000$, in all cases.)

knock-out mice compared to control mice. The dataset[5] contains 5548 gene expression measurements for 16 mice divided into two classes: 8 wild-type mice and 8 mice whose Apo AI gene was knocked out. This dataset was also used by Ge *et al.* (2003) to compare several statistical procedures for controlling multiple testing issues for univariate statistical tests.

Although the truly relevant variables were unknown, Callow *et al.* (2000) identified eight variables that are differentially expressed using univariate statistical tests and their relevance was experimentally confirmed. We expect that multivariate approaches will at least highlight these eight variables and we will thus check their presence in the rankings below. Note however that this does not mean that only those eight variables are relevant. All additional variables found by our multivariate procedures would certainly deserve to be checked experimentally.

On this dataset, we applied the Extra-Trees algorithm with four different settings of its parameters, i.e. the number K of variables that are randomly selected at each node and the number T of trees in the ensemble: ($K = \sqrt{m}, T = 100$), ($K = \sqrt{m}, T = 1000$), ($K = m, T = 100$), and ($K = m, T = 1000$). The pFDR and CER as a function of the ranking are plotted in Figure 3.4 in all four cases.

Several conclusions can be drawn from these plots. First, using $K = \sqrt{m}$ and $T = 100$ does not bring interesting results on this dataset. The method is

[5] http://www.stat.berkeley.edu/users/terry/zarray/Html/apodata.html

unable to distinguish truly relevant variables from randomly permuted ones, which translates into a high value of both the pFDR and the CER. Simply increasing the number of ensemble terms already gives much better results in terms of the number of variables that appear relevant. A threshold of 0.05 on the CER selects a subset of 7 variables and a threshold of 0.05 on the pFDR gives 10 variables that actually contain the 8 variables identified by Callow *et al.* (2000). Only 3 of them were present in the top 10 variables with $T = 100$, confirming that the ranking is indeed improved by increasing T. It is interesting to note that, on the other hand, increasing T from 100 to 1000 does not affect the error rate on the prediction (which is equal to 25% in both cases, as estimated by leave-one-out), meaning that here prediction accuracy would not be a relevant criterion to assess the quality of the ranking. The high improvement of the pFDR and CER values when T is increased is here a consequence of the very high ratio between the number of variables and the number of examples that makes the random trees, and thus the corresponding rankings, highly unstable and thus requires to average a very large number of trees for stabilization.

When K is increased to its maximum value (i.e. randomization is reduced), the CER is also very much improved, even with $T = 100$. It shows an abrupt change between the 11th and the 12th variables, suggesting that about 11 variables are relevant. As a confirmation, the 8 variables identified by Callow *et al.* (2000) are again among these 11 ones. On the other hand, the pFDR seems to be highly overestimated in this case. We explain this by the fact that increasing K in Extra-Trees makes the model less random and thus increases the importance of the top-ranked variables relatively to the low-ranked ones, thus emphasizing the phenomenon highlighted in Section 3.4.3 and responsible for the overestimation of the FDR.

3.7 Discussion

In this chapter, we proposed and evaluated two statistical procedures to extract a subset of truly relevant variables from importance values obtained from tree-based multivariate methods.

The first method is a direct adaptation of FDR estimation schemes based on permutations used in the context of univariate statistics. Unfortunately, we found that this procedure, because it does not take into account the dependencies of the importance values derived from the tree-based methods, often strongly overestimates the actual FDR and can thus potentially lead to overly conservative selections of relevant subsets.

We therefore proposed a new statistic, called the *conditional error rate*

(CER), that explicitly takes into account the dependencies between the importance scores and thus leads to more robust feature selections. We also proposed a permutation procedure to empirically estimate the CER values with any given importance ranking scheme, and compared its performances with the FDR-based scheme on both artificial and real datasets. Our experiments suggest that the CER-based procedure leads to a more robust scheme for the selection of relevant variables among large numbers of irrelevant ones. They also suggest that for reliable identification of relevant features with tree-based ensemble methods, one should use very large ensembles, much larger than those needed for accurate prediction. This in turn highlights the fact that prediction performance may not be an appropriate measure for the identification of relevant features.

One drawback of the current CER estimation procedure is that it is very computationally demanding: $P \times m$ models are needed for the estimation of the CER for all possible importance thresholds, where m is the number of variables and P is the number of random permutations. The computing times can however be decreased, e.g. by stopping the procedure as soon as the estimated probability is greater than some threshold, as we are typically only interested in small values of the CER. In problems with a very high proportion of irrelevant variables, this truncation will lead to a very significant speed-up of the procedure.

It would be interesting to investigate better procedures to estimate the FDR. Several improved permutation procedures have been proposed to better estimate the FDR in the context of univariate tests. In Chapter 4, we consider the extension of the approach proposed by Ge *et al.* (2008) to importance measures derived from multivariate approaches.

Two related approaches to identify relevant features from a ranking were proposed by Stoppiglia *et al.* (2003) and Tuv *et al.* (2006). The common idea of both methods is to include random features in the learning sample and then to exploit their rank among the original features to determine a relevance threshold. Stoppiglia *et al.* (2003) suggested to introduce only one such random feature and they applied this idea in the context of linear models where the distribution of the rank of the random feature can be computed analytically. They also suggested in their conclusions that this distribution could be computed empirically for any other ranking method. Along a similar line, the approach of Tuv *et al.* (2006) introduces as many random features as there are input variables in the original problem and generates these features by permuting the values of the original variables. Relevant variables are then defined as those variables that receive an importance significantly greater than their permuted counterpart. In the procedure proposed by Tuv *et al.* (2006), this idea is actually wrapped into a gradient boosting type

algorithm that iteratively selects subsets of important variables but it could be also applied in a single step as an alternative to our approach. A detailed comparison of our method with these two approaches is performed in Chapter 4.

Another important direction of investigation is of course the application of our procedures in the context of other importance measures. In this chapter, we focused on classification problems. However, our analysis should carry over straightforwardly to regression measures based on variance reductions, in the form (2.12). It would be interesting also to consider the permutation-based importance measure proposed by Breiman (2001). In Chapter 4, we carry out some experiments with variable importances derived from the weights of linear SVM models.

4

Evaluation and comparison of feature selection methods

In this chapter, we present the results of the evaluation of several, existing and novel, procedures that extract relevant features from rankings derived from machine learning approaches, including the CER and pFDR methods presented in the previous chapter. Like these latter, the evaluated procedures work by replacing the variable relevance scores with measures that can be interpreted in a statistical way, such as p-values, false discovery rates, or family wise error rates, for which it is easy to determine a significance level. These procedures, which were assessed on several artificial and real datasets, differ greatly in terms of computing times and the tradeoff they achieve in terms of false positives and false negatives. Our experiments also clearly highlight that using model performance as a criterion for feature selection is not a good practice in general. After the short introduction of Section 4.1, Section 4.2 describes the feature selection methods that we evaluated. Section 4.3 describes the datasets that we used for our experiments, the performance metrics, and the compared ranking methods. Section 4.4 presents the results of the evaluation and finally, Section 4.5 concludes the chapter.

Contents

Publication related to this chapter:

Huynh-Thu, V. A., Saeys, Y., Wehenkel, L., and Geurts, P. (2011) Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. *Bioinformatics* **28**(13):1766-1774

Peer-reviewed conference presentation:

Benelux Bioinformatics Conference, Luxembourg (Luxembourg), 2011.

4.1 Introduction

In this chapter, we perform a large-scale evaluation of the CER and pFDR procedures, that are introduced in the previous chapter. We also compare them to several, existing and novel, other procedures that extract relevant features from a ranking returned by a multivariate algorithm. Like the CER and the pFDR, these procedures replace the original relevance score with a measure that can be interpreted in a statistical way and hence allow the user to determine a significance threshold in a more informed way. Most of these methods exploit a resampling procedure to estimate the false discovery rate (FDR) or the family wise error rate (FWER) among the k top-ranked features, for increasing values of k. Just like for standard univariate tests, the user can then choose a threshold on this new measure depending on the risk he/she is ready to take when deeming that all features above this threshold are relevant. Experiments on several artificial and real datasets show that most of these measures greatly help in the extraction of truly relevant features from a ranking derived from a multivariate approach. We also highlight that the common approach to this problem, i.e. selecting the top k features minimizing some cross-validated error, is not a good practice in general, as it typically leads to the selection of several irrelevant features.

4.2 Feature selection methods

We describe below several methods that have been developed for the selection of relevant variables from a ranking. We adopt the same setting as in Chap-

ter 3 and assume that we have an algorithm $\mathcal{A}(LS)$ that returns, from a learning sample LS, a relevance score s_i for each input variable $X_i, i = 1, \ldots, m$. We further assume, without loss of generality, that the features are numbered according to their relevance score, i.e. $s_1 \geq s_2 \geq \cdots \geq s_m$. Most of the presented methods then re-use $\mathcal{A}$ on a modified LS (obtained from a subsampling, a permutation, etc.) to replace each original relevance score s_i with a statistically interpretable measure. The intuition behind each method is given below and their detailed pseudo-code descriptions can be found in Appendix A.1.

4.2.1 Estimation of the generalization error of a model (err-$\mathcal{A}$ and err-TRT)

We include in our comparison, as a baseline method, the procedure based on the computation of the generalization error of a predictive model (see Geurts *et al.*, 2005 for an example). This method consists in estimating the error rate (resp. quadratic error) e_i of a classification (resp. regression) model that uses only the first i variables of the ranking, $\forall i = 1, \ldots, m$, and selecting the k top-ranked variables such that:

$$k = \arg \min_{i=1,\ldots,m} e_i. \tag{4.1}$$

The m predictive models can be learned using the algorithm $\mathcal{A}$ that was used to compute the ranking of variables and the generalization error of one model can be estimated using a cross-validation procedure (10-fold in all our experiments). We call this method *err-$\mathcal{A}$* to denote the fact that the same algorithm $\mathcal{A}$ is used both to rank the features and to estimate the error associated with each feature subset. A sharper threshold can be obtained by estimating the generalization error with an algorithm that is not robust to irrelevant variables, such as k-NN (Fukunaga and Hostetler, 1975) or totally randomized trees (TRT, i.e. Extra-Trees with parameter $K = 1$, Geurts *et al.*, 2006a). Compared to a robust algorithm, we expect the error of such a procedure to increase in a more abrupt way when irrelevant variables are introduced in the predictive model and therefore to yield a smaller number of selected variables. We used TRT in our experiments as this method is computationally less expensive than k-NN and we call the resulting feature selection method *err-TRT*.

One potential drawback of this approach is the fact that it is prone to *selection bias* (Ambroise and McLachlan, 2002), as the same instances of LS are used to rank the variables and to estimate the generalization error. This results in a too optimistic estimation of the errors e_i, and in particular of

the minimal error $\min_i e_i$, whose effect on the number of selected features is difficult to appraise. One could get better error estimates by ranking the features inside the cross-validation loop (Ambroise and McLachlan, 2002) but this would leave open the question of the selection of the final feature subset among the subsets generated within each fold.

4.2.2 Multiple testing with random permutations (pFDR, eFDR, and CER)

Besides the pFDR and CER procedures of Chapter 3, we evaluated another permutation-based approach, which was proposed by Ge *et al.* (2008) to estimate the FDR in the context of univariate rankings. This approach makes the assumption that the first $i-1$ variables are relevant and the FDR at threshold s_i is defined by:

$$\mathrm{FDR}(s_i) = E\left[\frac{V(s_i)}{V(s_i)+i-1}\middle| H_R^{1\to i-1}, H_I^{i\to m}\right], \tag{4.2}$$

where $V(s_i)$ is the number of false positives, $H_R^{1\to i-1}$ denotes the hypothesis that all variables ranked above X_i are relevant, and $H_I^{i\to m}$ is the hypothesis that X_i and all variables ranked below X_i are irrelevant. $V(s_i)$ is estimated in the following way. Let s_k^p be the relevance score of $X_k (\forall k = 1, \ldots, m)$, calculated from a random permutation of the data that simulates $H_R^{1\to i-1}$ and $H_I^{i\to m}$, and let $s_{(k)}^p$ be the k-th largest member of $\{s_i^p, \ldots, s_m^p\}$. $V(s_i)$ is then computed as:

$$V(s_i) = \max_{k=1,\ldots,m-i+1} \{k : s_{(1)}^p \geq s_i, s_{(2)}^p \geq s_{i+1}, \ldots, s_{(k)}^p \geq s_{i+k-1}\}. \tag{4.3}$$

The FDR estimated using Equations (4.2) and (4.3) is called eFDR. When applying this approach to rankings derived from a multivariate approach, we propose to use the same permutation scheme as in the CER approach, that consists in approximating $H_R^{1\to i-1}$ by keeping the values of the output variable and of the first $i-1$ variables unchanged, and $H_I^{i\to m}$ by randomly and jointly permuting the values of $X_i, \ldots, X_m$.

4.2.3 Empirical estimation of the null rank distribution (mr-test)

The *mr-test* (Zhang *et al.*, 2006) estimates an empirical distribution of the rank of an irrelevant feature, in order to derive a p-value p_i to be associated with each variable X_i, defined as the probability for an irrelevant variable to

be ranked above or at the same position as X_i. To estimate the distribution of the rank of an irrelevant variable, Zhang *et al.* (2006) proposed to proceed as follows. P feature rankings are obtained by applying the algorithm $\mathcal{A}$ on P resamplings of the original learning sample. The resampling procedure proposed by the authors consists in choosing randomly half of the instances of the original learning sample each time. Given a user-defined number k, the k variables that have on average the largest ranks among all the variables are considered putative irrelevant variables and the null rank distribution is estimated from their $k \times P$ ranks over the P rankings. The p-value p_i is then estimated as the proportion of these $k \times P$ ranks that are lower than the average rank of X_i over the P rankings.

As the p-values calculated using this procedure are *raw* p-values, the so-called multiple testing problem occurs, where the higher the number of variables in the considered problem, the higher the number of expected variables with a p-value lower than some threshold α, even if these variables are irrelevant. We therefore propose to apply a multiple testing correction procedure and to select the variables based on the corrected p-values. In our experiments, we used the Benjamini Hochberg correction (Benjamini and Hochberg, 1995), which was shown to control the FDR in the context of univariate statistical tests.

The main parameter of the mr-test procedure is the number k of putative irrelevant variables from which the empirical null rank distribution is estimated. A small value of k would result in overoptimistic selections of variables while a high value of k would be too conservative. In our experiments, k was fixed to $\frac{m}{2}$.

4.2.4 Introduction of random probes (1Probe and mProbes)

Stoppiglia *et al.* (2003) suggested to introduce one random feature in order to compute the probability p_i for this random feature to be ranked above or at the same position as X_i. They applied this idea in the context of linear models where each variable X_i is ranked according to the squared cosine of the angle between X_i and the output variable, and where therefore the distribution of the rank of the random feature can be computed analytically. However, to be able to apply this approach with any ranking procedure, the authors suggested in their conclusions to compute the null rank distribution empirically by artificially introducing random probes. We therefore propose the following procedure, that we call *1Probe*. In each of P iterations, we introduce in the original learning sample an additional variable X_{rand} whose

values are randomly sampled from $\mathcal{N}(0,1)$. We then estimate the p-value p_i by the rate of iterations where X_{rand} is ranked above X_i. As the p-values calculated using this procedure are prone to the multiple testing problem, we propose to correct them using the Benjamini Hochberg procedure, like in the mr-test procedure. Note that the *1Probe* method is parametric, since the choice of the distribution of the random probe can have an impact on its rank. The exact impact however depends on the ranking method used[1].

Along a similar line, the ACE method (Tuv *et al.*, 2009) introduces as many random features as there are input variables in the original problem. Each random feature is generated by permuting the values of one original variable. The method then assumes that an original variable is irrelevant if it has a relevance score not statistically higher than that of a random feature. In the original approach, a t-test is applied to determine the significance of each variable and the procedure is actually wrapped into a gradient boosting type algorithm that iteratively selects subsets of important variables. We propose to use a variant of ACE, that we call *mProbes*, where instead of applying a t-test, we simply compute the proportion of simulations where at least one random feature is ranked above X_i. We also drop the gradient boosting procedure and apply the approach in one single run. The value associated with X_i that is returned by mProbes thus estimates the FWER when selecting X_i and all the variables ranked above X_i.

4.2.5 Computational complexity

Although computing time is not a real issue in most applications, Table 4.1 shows the computational complexity of each method. Except err-$\mathcal{A}$ and err-TRT, all the methods have a common parameter P, which is the number of iterations or permutations. The higher the value of P, the better the (Monte Carlo) estimate of the FDR/FWER/p-value. P was fixed to 1000 in our experiments.

Among all methods, the mr-test has the lowest complexity as $\mathcal{A}$ is run on only half of the instances of the learning sample in each iteration. On the other hand, the eFDR and CER have the highest complexities if one wants to compute these measures for all m variables. However, as suggested in Section 3.7, the computing times of these procedures can be reduced by stopping them as soon as the eFDR/CER is greater than some significance level. It

[1]For example, the distribution of the random probe has no impact on standard trees, since the test selected at each node only depends on the ordering of the instances derived from the variables, and not the absolute values of the variables. It has however an impact on Extra-Trees, because of the randomization of the cut-point.

Table 4.1: **Computational complexity.**

Method	Complexity
CER	$M \times P \times C_{\mathcal{A}}(N, m)$
pFDR	$P \times C_{\mathcal{A}}(N, m)$
eFDR	$M \times P \times C_{\mathcal{A}}(N, m)$
mr-test	$P \times C_{\mathcal{A}}(\frac{N}{2}, m)$
1Probe	$P \times C_{\mathcal{A}}(N, m+1)$
mProbes	$P \times C_{\mathcal{A}}(N, 2m)$
err-$\mathcal{A}$	$C_{\mathcal{A}}(N, 1) + C_{\mathcal{A}}(N, 2) + \ldots + C_{\mathcal{A}}(N, m)$
err-TRT	$m \times N \times \log N$

P is the number of iterations, M is the number of variables for which one wants to compute the eFDR/CER, $C_{\mathcal{A}}(N, m)$ is the computational complexity of algorithm $\mathcal{A}$ when applied on a learning sample with N instances and m variables. For Random Forests, $C_{RF}(N, m) = O(\sqrt{m}.N.\log N)$. For SVMs, C_{SVM} lies between $O(m.N^2)$ and $O(m.N^3)$.

is also worth mentioning that all the methods can be easily parallelized on a computing grid.

4.3 Datasets and protocol

We describe in this section the artificial and real datasets that we used for our experiments, the performance metrics, and the compared ranking methods.

4.3.1 Artificial datasets

We generated two families of artificial problems to validate the feature selection methods in a context where relevant variables are perfectly known.

Linear. This is a linear two-class classification problem. All input variables are continuous and their values are sampled from $\mathcal{N}(0, 1)$. The output Y is given by:

$$Y = \text{sgn}\left(\sum_{i=1}^{p} w_i X_i + \epsilon\right), \tag{4.4}$$

where ϵ is a random noise with zero mean and the values of w_i are uniformly distributed random numbers between 0 and 1. Irrelevant variables, which

are pure Gaussian noise, are added to the p relevant variables.

Hypercube. As the artificial problems used in Chapter 3, this two-class classification problem was generated by adapting the MATLAB® code originally used to produce the *Madelon* dataset. All input variables are continuous and their values are sampled from $\mathcal{N}(0, 1)$. Each class is composed of a number of Gaussian clusters that are placed at random on the vertices of a hypercube in a p-dimensional space, where p is the number of relevant variables. Unlike the previous problem, the decision boundary is thus potentially non-linear. Irrelevant variables, which are pure Gaussian noise, are added to these p variables.

4.3.2 Microarray datasets

We performed experiments on three real gene expression datasets. For each dataset, the goal was to find a subset of genes that helps to discriminate between two groups of patients.

- **Prostate** (Dhanasekaran *et al.*, 2001). This dataset contains the expression levels of 4344 genes in 34 samples from patients with prostate cancer and 19 samples from men without documented prostate pathology.
- **Leukemia** (Golub *et al.*, 1999). This dataset contains the expression levels of 7129 genes in 47 samples from patients with acute lymphoblastic leukemia (ALL) and 25 samples from patients with acute myeloid leukemia (AML).
- **Breast** (Wang *et al.*, 2005). This dataset contains the expression levels of $\sim$ 22000 transcripts in 286 samples from patients with lymph-node-negative breast cancer, of whom 107 developed metastasis during the five years follow up while 179 were relapse-free.

4.3.3 Performance metrics

Each method returns a subset of features that it considers relevant. In the context of the artificial datasets where all relevant features are perfectly known, we used the precision and the recall to evaluate such subset, and we also compared these metrics to the following values:

- p_{max}: the precision of a method (called *rec-1*) that would select the first k variables of the ranking, where k is the smallest integer such that

$\{X_1, X_2, \ldots, X_k\}$ contains *all* the truly relevant variables (the recall is equal to one);

- r_{max}: the recall of a method (called *prec-1*) that would select the first k variables of the ranking, where k is the largest integer such that $\{X_1, X_2, \ldots, X_k\}$ contains *only* truly relevant variables (the precision is equal to one).

Finally, to evaluate a ranking of variables independently of the choice of a specific threshold, we used the area under the precision-recall curve (AUPR).

4.3.4 Compared ranking methods

We validated the feature selection methods in the context of three popular ranking algorithms; two representatives of multivariate techniques and one standard univariate method:

- Importance measures derived from the Random Forests procedure and based on sums of entropy reductions. The number K of randomly selected variables at each node of a tree was fixed to its default value $\sqrt{m}$ and ensembles of 1000 trees were grown.

- Importance measures derived from a linear support vector machine. The score s_i of feature X_i is simply taken as the absolute value of the coefficient w_i associated with the feature in the trained linear model. For our experiments, we used the LIBSVM library (Chang and Lin, 2011), with the regularization parameter C of the soft-margin SVM set to 1 (default value). We normalized the data so that the values of each variable were comprised between -1 and 1 in the learning sample before applying the linear SVM.

- The absolute value of the t statistics derived from a t-test.

4.4 Results

4.4.1 Artificial datasets

Comparison of the ranking methods

Figure 4.1 shows the AUPRs of the three ranking procedures (Random Forests, linear SVM, and t-test) on linear and hypercube datasets, as well as the AUPRs of a method that returns a random ranking for comparison. The

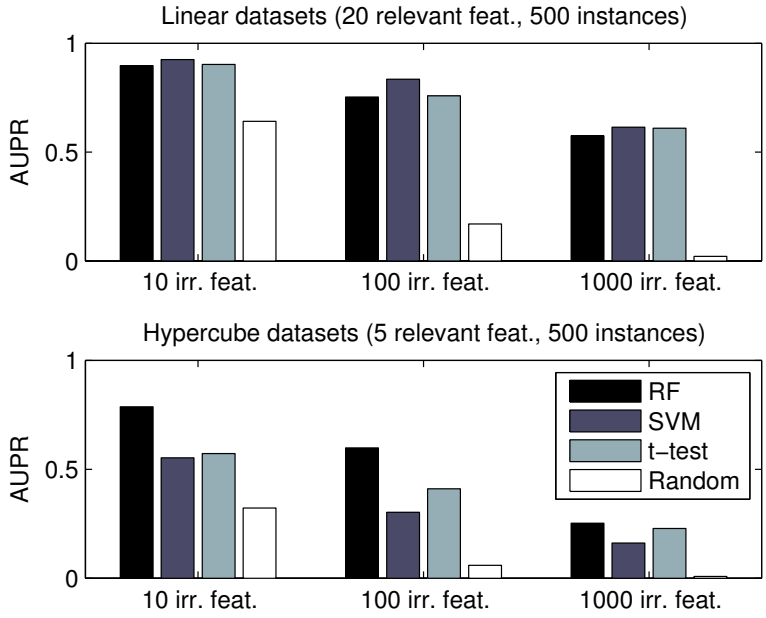


Figure 4.1: **AUPRs of each ranking method, for different numbers of irrelevant features.** *Random* is a method that randomly ranks the variables. Top on linear datasets, bottom on hypercube datasets. The AUPR values were averaged over 50 datasets in each case.

AUPR values were averaged over 50 randomly generated datasets in each case.

The three ranking methods perform better than the random procedure. The linear SVM yields the highest AUPRs on the linear datasets, although the t-test performs equally well for a high number of irrelevant features. On the (non-linear) hypercube datasets, the Random Forests procedure is the best performer.

Notice that the mr-test, 1Probe, and mProbes procedures each compute a statistical measure (p-value or FWER) associated with each variable X_i. These three methods thus potentially modify the original variable ranking by re-ordering the features according to the corresponding statistical measure. Nevertheless, the new rankings do not change much with respect to the original ranking. The corresponding AUPRs hardly vary, as shown in Figures A.1, A.2, and A.3 of Appendix A.2. On the other hand, the CER, pFDR, and eFDR procedures each estimate a statistic that corresponds to an *importance score* s_i rather than to a variable in itself. Therefore, the variables can not be re-ordered according to this statistic, and the monotonicity

of the estimated measures is enforced instead (see the pseudo-codes in Appendix A.1). The enforced monotonicity ensures that a variable X_i can be selected only if all the variables ranked above X_i are also selected.

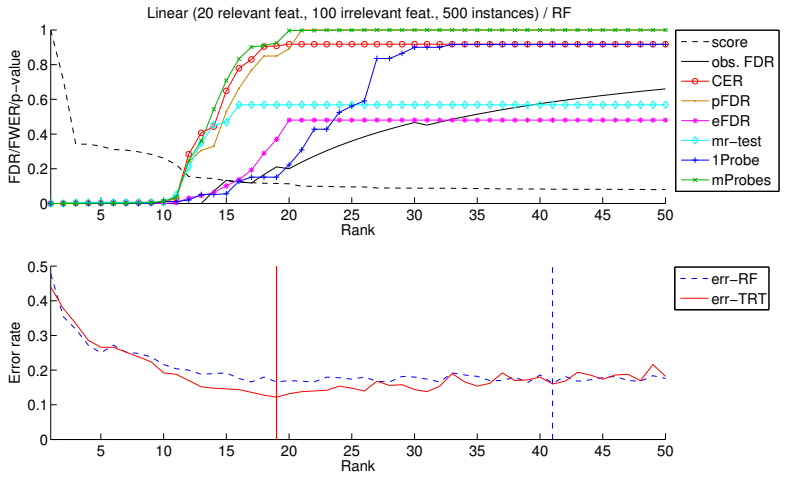
Interpretability of the curves

Figure 4.2(a) plots the curves of the different methods on a linear dataset with 20 relevant features. The Random Forests method was used as ranking procedure. At each rank i, we show the relevance score derived from the Random Forests, as well as the *observed* FDR, i.e. the proportion of truly irrelevant features among the i top-ranked variables. Nearly identical observed FDR curves are obtained when the variables are ranked using mr-test, 1Probe, and mProbes (see Figure A.4). Therefore, only the observed FDR related to the original ranking is plotted in Figure 4.2, for the sake of clarity.

We can see that selecting the variables based solely on the original relevance score is difficult as this score does not suggest any clear threshold (dashed curve in the top of Figure 4.2(a)). On the other hand, almost all studied methods successfully help to select variables. CER and mProbes provide a good estimation of the FWER as the transition between low and high CER/mProbes values is quite well centered at the point where irrelevant variables start appearing in the ranking (indicated by the observed FDR which becomes greater than zero). The pFDR overestimates the real FDR, as already observed in the results presented in Section 3.4 of Chapter 3, while the eFDR is closer to it. CER, pFDR, mr-test, and mProbes tend to be highly conservative, as their curves increase while the observed FDR is still equal to zero. By contrast, the values returned by eFDR and 1Probe become high only when larger subsets of top-ranked variables are considered. err-RF and err-TRT both select a high number of false positives. The minimal error rate of err-RF is obtained when the observed FDR is around 0.6, meaning that 60% of the selected variables are false positives, while, as expected, err-TRT selects fewer variables. Unlike the other methods, err-RF and err-TRT do not clearly highlight a threshold on the ranking. The error rate does not seem to be affected much by the introduction of irrelevant variables, resulting in rather flat curves.

The different methods generate similar curves when applied on a hypercube dataset (Figure A.5) and when the linear SVM is used as ranking procedure instead of the Random Forests (Figures A.6 and A.7).

(a) RF



(b) t-test

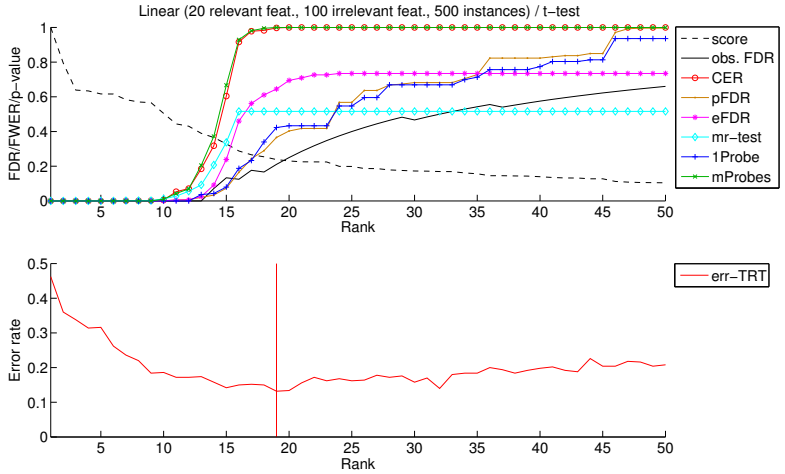


Figure 4.2: **Curves of the different methods on a linear dataset.** We used **(a)** the Random Forests and **(b)** the t-test as ranking methods. *score* is the relevance score derived from the Random Forests **(a)** or the absolute value of the statistic t derived from the t-test **(b)**. *obs. FDR* is the observed FDR. The dashed blue (resp. plain red) vertical line indicates the position of the lowest error rate for err-RF (resp. err-TRT).

Precision and recall of the methods

Figure 4.3 shows the precision and the recall of the methods on linear datasets, for different numbers of irrelevant variables. The Random Forests algorithm was used as ranking procedure and a significance level $\alpha = 0.05$ was chosen

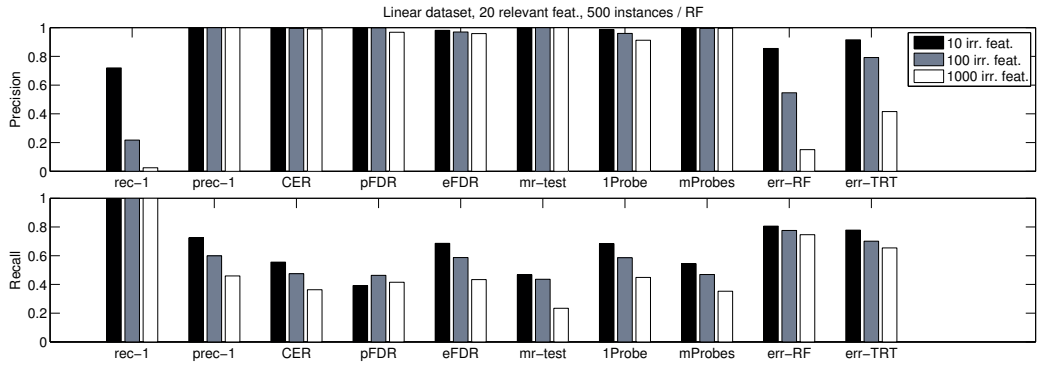


Figure 4.3: **Precision and recall on linear datasets, for different numbers of irrelevant features.** We used the Random Forests algorithm as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

(we used this significance level in all our experiments). The precision and recall values were averaged over 50 datasets in each case.

When the number of irrelevant variables increases, the recall of each method decreases. As already observed from the curves, CER, pFDR, mr-test, and mProbes are rather conservative. The precision of these methods remains always almost at its highest value and their recall never reaches the recall r_{max} of the prec-1 method. On the other side, eFDR and 1Probe trade some precision, which remains nevertheless high, for a recall that is higher and close to r_{max}. err-RF and err-TRT obtain the highest recall values but also the lowest precision levels. Moreover these precision levels clearly decrease when the number of irrelevant variables increases. err-TRT tends to select fewer variables than err-RF and has therefore a higher precision. Similar results are observed on hypercube datasets (Figure A.8) and when SVM is used as ranking procedure (Figures A.9 and A.10).

When we increase the number of instances in the learning samples, the recall of all the methods increases, as well as the precision of err-RF/SVM and err-TRT (see Figures 4.4, A.11, A.12, and A.13). We again observe three families of methods: those having a high precision and a recall lower than r_{max} (CER, pFDR, mr-test, and mProbes), those having a high precision and a recall close to r_{max} (eFDR and 1Probe), and those with a lower precision and a recall higher than r_{max} (err-$\mathcal{A}$ and err-TRT).

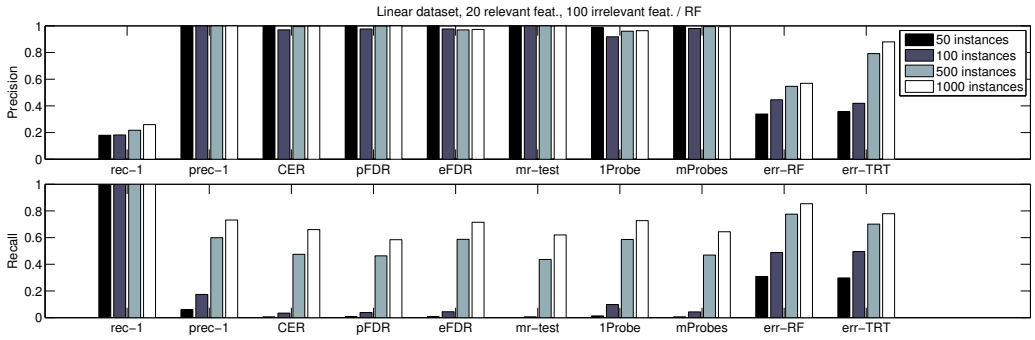


Figure 4.4: **Precision and recall on linear datasets, for different numbers of instances.** We used the Random Forests algorithm as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

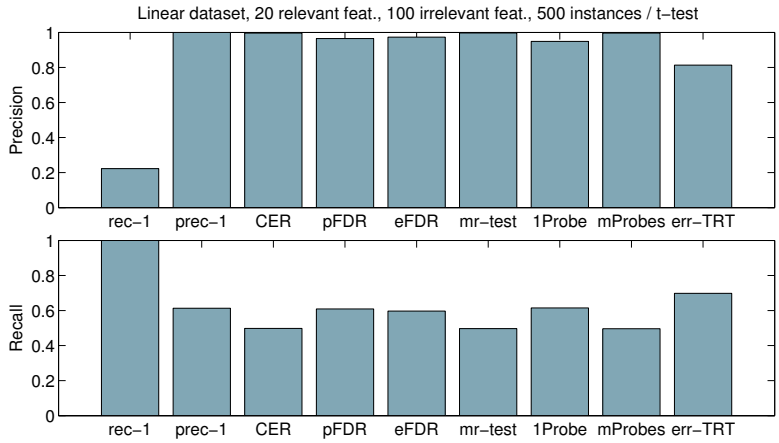


Figure 4.5: **Precision and recall on linear datasets.** We used the t-test as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets.

Univariate rankings

Figures 4.2(b) and 4.5 show respectively the score curves and precision/recall values of each method, when the relevance score of a variable is the absolute value of the statistic t computed by a t-test. All the results are similar to those obtained with a multivariate ranking method except for the pFDR which, when used with a t-test, provides a much better estimation of the real FDR (see Figure 4.2(b)) and has a recall equal to r_{max} while having a high precision (see Figure 4.5) .

4.4.2 Microarray datasets

In order to highlight the behavior of the different feature selection methods on real problems where the number of variables is much higher than the number of instances, we applied them on the three gene expression datasets described in Section 4.3.2.

On each problem, the number of genes selected by one method depends considerably on the chosen ranking procedure (see Table 4.2). For most methods, using the linear SVM leads to selections of empty subsets of genes while intermediate and large subsets are selected with the Random Forests and the t-test respectively. Compared to the artificial problems, the largest subsets are no longer obtained by err-$\mathcal{A}$ and err-TRT. These two methods select relatively small numbers of genes because the error rate e_i of a predictive model that uses only the i top-ranked genes is typically equal to zero for a low value of i (see Figure A.14 for the results with the linear SVM). This can be explained by the low number of instances compared to the number of genes and the fact that the error rate was estimated on instances that were also used to compute the gene ranking (the so-called selection bias, Ambroise and McLachlan, 2002). On the prostate and leukemia datasets, the largest subsets of genes are obtained by pFDR, eFDR, and 1Probe (used with a t-test). On the breast dataset, CER, pFDR, eFDR, mr-test, 1Probe, and mProbes select empty subsets when they are used with the Random Forests or the linear SVM, while with the t-test, they select subsets that are clearly smaller than those obtained on the prostate and leukemia datasets. These results can be explained by the fact that all these methods provide a correction for multiple testing and, as the breast dataset contains a very high number of genes, the correction is so strong that very few of them appear relevant.

Given a feature selection method and a ranking algorithm $\mathcal{A}$, the number of selected genes can also vary depending on the tuning of the parameters of $\mathcal{A}$. As an example, one parameter of Random Forests is the number T of trees that are grown in an ensemble. Increasing T from 1000 to 10000 results in larger subsets of selected genes for all the methods except err-RF and err-TRT (see Table 4.3). We already observed this phenomenon for the pFDR and the CER, on the Apo dataset in Section 3.6. Due to the very high ratio between the number of variables and the number of samples, the random trees, and thus the corresponding rankings, are highly unstable. Averaging a very large number of trees results in a stabilization and an improvement of the feature ranking, and thus the possibility to select more variables without including any false positive. However, in spite of this improvement, err-RF and err-TRT do not select more genes, meaning that the error rate would

Table 4.2: **Number of selected genes ($\alpha = 0.05$).**

	Prostate			Leukemia			Breast		
	RF	SVM	t-test	RF	SVM	t-test	RF	SVM	t-test
CER	58	0	391	47	0	152	0	0	3
pFDR	73	0	1499	83	0	1139	0	0	155
eFDR	391	0	1608	340	0	1047	0	0	127
mr-test	18	0	563	8	5	150	0	0	0
1Probe	54	0	1608	61	7	981	0	4	543
mProbes	91	0	350	63	10	145	0	0	3
err-$\mathcal{A}$	5	8	–	26	37	–	30	436	–
err-TRT	3	12	112	46	123	62	110	13	209

Table 4.3: **Number of selected genes with Random Forests on the prostate dataset ($\alpha = 0.05$).**

T	CER	pFDR	eFDR	mr-test	1Probe	mProbes	err-RF	err-TRT
1000	58	73	391	18	54	91	5	3
10000	136	88	668	193	444	215	4	3

T is the number of grown trees in an ensemble.

not be a relevant criterion to assess the quality of subsets of variables.

Finally, to highlight the behavior of the methods on a problem where none of the variables is truly relevant, we created a new dataset by randomly permuting the output values in the prostate dataset. Table 4.4 shows that the number of selected genes (averaged over 50 permuted datasets) is close to zero for CER, pFDR, eFDR, mr-test, 1Probe, and mProbes, while err-$\mathcal{A}$ and err-TRT both select a non negligible subset of false positives (they actually select more features than on the original dataset).

4.5 Conclusion

In this chapter, we evaluated several procedures that aim to identify, from a ranking, the maximal subset of variables that truly provide some information about an output variable. These procedures assume that a (multivariate)

Table 4.4: **Average number of selected genes on permuted prostate datasets ($\alpha = 0.05$).**

	CER	pFDR	eFDR	mr-test	1Probe	mProbes	err-$\mathcal{A}$	err-TRT
RF	0.04	0.08	0.08	0	1.36	0.04	29.28	20.80
SVM	0.08	0.1	0.08	0	0.08	0	77.72	24.92
t-test	0.16	3.46	2.70	0	6.34	0.14	–	44.74

ranking method $\mathcal{A}$ was first used to compute a relevance score for each variable of the considered problem and then extract relevant features from this ranking, by replacing the original relevance score with a measure that can be interpreted in a statistical way. Depending of the procedure, this measure is either the generalization error of a predictive model (err-$\mathcal{A}$ and err-TRT), the false discovery rate (pFDR, eFDR), the family wise error rate (CER, mProbes), or a p-value (mr-test, 1Probe).

Among the feature selection methods that we evaluated, err-$\mathcal{A}$ and err-TRT are the only ones that do not require to choose a significance level *a priori*. However on artificial problems, they always had the lowest precision among all methods and, on the permuted prostate datasets, they selected non negligible subsets of variables although none of them was truly relevant. Prediction performance thus does not appear to be an appropriate measure for the identification of relevant features.

The other methods all have a very high precision. One can distinguish highly conservative methods that avoid the inclusion of any irrelevant feature as much as possible (CER, pFDR, mr-test, and mProbes) and less stringent methods that trade some precision for a higher recall (eFDR and 1Probe). The choice between these two compromises clearly depends on the application. Among the more conservative methods, mr-test has the disadvantage of requiring the determination of an additional parameter k, which introduces some dependency with respect to the problem and ranking method used (although our default choice seems to be robust). The pFDR method is the simplest one but was shown to overestimate the real FDR in case of dependent scores. CER and mProbes reach a similar compromise. mProbes has a computational advantage over the CER method while this latter has a nice interpretation when the scores are derived from univariate scores. Among the less conservative methods, 1Probe has a clear advantage over the eFDR method in terms of computing times and is also conceptually much simpler. However, the variables selected by this method depend on the chosen dis-

tribution of the random probe, which makes it a parametric method. To conclude, our advice would be to use mProbes or CER when a very stringent method is needed, and eFDR otherwise.

5

Closure of Part II

This chapter summarizes our contributions to feature selection, as well as the main conclusions that can be drawn from our empirical studies. We also give some directions for future research.

Contents

5.1 Contributions

The work described in this part of the thesis was related to the problem of selecting relevant features from a variable ranking returned by a machine learning algorithm. This ranking is typically derived from a relevance score computed for each variable. However, because this relevance score is usually not statistically interpretable, choosing a threshold on this score, above which the variables are deemed relevant, is not an easy task.

To select features from such ranking, we proposed a procedure that replaces the variable relevance score with a statistical measure called the conditional error rate (CER). This procedure is based on a permutation scheme that specifically takes into account the fact that the relevance scores of the different variables are dependent on each other. While there is still a need to determine a threshold on the CER to select variables, the determination of this threshold is easier because this measure can be interpreted in a statistical way. Furthermore, the threshold is not dependent anymore on the considered problem and on the ranking method.

We performed a large-scale evaluation of the CER procedure and of other methods that also replace the relevance scores derived from a machine learning algorithm with statistically interpretable measures. These methods are either existing methods of the literature (pFDR, err-$\mathcal{A}$, mr-test) or novel methods inspired from existing ones (eFDR, 1Probe, mProbes). The pFDR and eFDR procedures rely on permutations of the data, err-$\mathcal{A}$ computes the generalization error of predictive models, the mr-test estimates the distribution of the rank of an irrelevant variable, and 1Probe and mProbes are based on the introduction of one and several random features respectively.

All the methods, except the ones based on the generalization error (err-$\mathcal{A}$ and err-TRT), have a very high precision when selecting variables from a ranking. One can distinguish highly conservative methods (CER, pFDR, mr-test, and mProbes) and less stringent methods that trade some precision for a higher recall (eFDR and 1Probe). The choice between these two compromises clearly depends on the application.

The methods that exploit the generalization error are the only ones that do not require to choose a significance level *a priori*. They actually allow to automatically select a threshold on a feature ranking. However our experiments on artificial data showed that these methods usually select subsets of features with a low precision. Prediction performance thus does not appear to be an appropriate measure for the identification of relevant features. This is also supported by experiments in the second part of the thesis (see Section 6.5).

5.2 Future research directions

Our experiments on the microarray datasets highlighted that the number of selected variables depends strongly on the precise values of the parameters of the ranking algorithm. For example, increasing the number of trees of the Random Forests method allows the different feature selection methods to select more genes (Table 4.3). It would thus be of great interest to find a way to automatically tune the parameters of the ranking algorithm. Recently, a great interest has raised for the procedures that evaluate feature selection methods by an analysis of their stability (see e.g. Abeel *et al.*, 2010). The rationale behind such stability analysis is that a good feature selection technique should select (nearly) the same subsets of features when small changes are made to the dataset. However, stability in itself is not a sufficient criterion for evaluating a subset of features, since a stable subset does not necessarily mean that all the features within this subset are relevant. Hence stability must be analyzed together with another quality criterion such as the predictive performance. An alternative to the stability analysis could be provided by the number of variables that are returned by the feature selection methods evaluated in this part of the thesis. Indeed, a higher number of variables with a low FDR or FWER indicates that it is more unlikely that an irrelevant variable reaches the top of the feature ranking, and hence that the ranking is more stable. In the future, we thus plan to explore further the use of the number of selected variables to tune the parameters of a ranking algorithm, independently of any consideration about predictive performance.

In this part of the thesis, we focused on the problem of finding all the relevant variables from a ranking. However, there exist some problems in which all the variables are relevant. For example, in the context of gene regulatory network inference, each gene of a network is actually regulated, either directly or indirectly, by all the other genes. The goal is then rather to identify its direct regulators. More generally, the problem consists in determining a minimal subset of relevant variables, such that no other variable conveys complementary information about the target conditionally to these variables (the so-called *Markov boundary*, Pearl, 1988). The adaptation of our procedures to solve this problem would be an interesting direction of future research.

Finally, although our goal was not to compare the different ranking methods, we observed from our experiments that a quite different subset of selected features could be obtained depending on the ranking method used. For example, on the microarray datasets, more significant variables are found with the t-test than with the Random Forests (Table 4.2). One could then wonder about the intrinsic differences that exist between these two ranking proce-

dures. Therefore, as future work, we would like to compare the rankings derived from the different methods, and check in which cases machine learning algorithms really have an advantage over simpler univariate procedures. Haury *et al.* (2011a) already proposed one such study, but they did not include the Random Forests in their evaluation.

Part III

Network inference

6

GENIE3: GEne Network Inference with Ensemble of trees

This chapter focuses on GENIE3, a new algorithm for the inference of gene regulatory networks from static steady-state expression data. GENIE3, which is based on feature selection with tree-based ensemble methods, was best performer in the DREAM4 and DREAM5 challenges (Marbach *et al.*, 2012, 2010, 2009; Prill *et al.*, 2010). The algorithm decomposes the prediction of a regulatory network of p genes into p regression problems. In each of the regression problems, the expression pattern of one of the genes (target gene) is predicted from the expression pattern of all the other genes (input genes), using tree-based methods. The importance of an input gene in the prediction of the target gene expression pattern is taken as an indication of a putative regulatory link. GENIE3 is simple and generic, making it adaptable to other types of genomic data and interactions. The chapter is organized in the following way. First, Section 6.1 consists in an overview of the currently existing network inference methods. Section 6.2 describes the GENIE3 algorithm. Section 6.3 presents the DREAM challenges. Section 6.4 shows the results obtained with GENIE3 on these challenges as well as on the M^{3D} *E. coli* dataset. Finally Section 6.5 concludes the chapter and discusses some ideas for further developments.

Contents

Publication related to this chapter:

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.

Invited conference presentations:

DREAM4 Conference, Boston (USA), 2009.
DREAM5 Conference, New York City (USA), 2010.

Peer-reviewed conference presentations:

Workshop on Machine learning in Systems Biololgy, Edinburgh (UK), 2010.
ISMB/ECCB, Vienna (Austria), 2011.

6.1 Background

Genetic regulatory networks (GRNs) (Bolouri, 2008) are central to all biological organisms, and their deciphering is crucial to understand the development, functioning, and pathology of these organisms. Once a remote theoretical possibility, this deciphering is now made possible by advances in genomics, most notably high-throughput profiling of gene expression patterns with DNA microarrays. These advances have prompted the development of a plethora of models of GRNs and algorithms to reverse-engineer them from expression data (Bansal *et al.*, 2007; Gardner and Faith, 2005; Lee and Tzou, 2009; Markowetz and Spang, 2007).

The simplest models of genetic regulatory networks are based on Boolean logic. Because of their simplicity, these Boolean network models have provided high-level insights into the design principles and emerging properties of GRNs (Kauffman, 1993). At the other end of the complexity spectrum are physical models mimicking the biological mechanisms at play, including promoter recognition, mRNA transcription, and protein translation. These models, typically based on systems of ordinary or stochastic differential equations, can generate realistic behavior (Gardner *et al.*, 2003). One of their main drawbacks is that they have high-dimensional parameter spaces, and

thus a large number of experimental data are needed for their identification. Nevertheless, hybrid methods involving ordinary differential equations have shown good performances on real-life genome-wide GRN inference (Bonneau *et al.*, 2006).

Models based on the statistical analysis of dependencies between expression patterns have an intermediate complexity, and have already been successfully applied to the inference of large GRNs. Early models used correlation coefficients between expression patterns of all pairs of genes to infer "coexpression networks" (Eisen *et al.*, 1998). However, correlation coefficients fail to capture more complex statistical dependencies (e.g. non-linear ones) between expression patterns, and thus more general measures of dependency based on mutual information (MI), have been proposed. The simplest model based on this measure, the "relevance network", computes MI between all pairs of genes and infers the presence of a regulatory interaction when MI is larger than a given threshold (Butte and Kohane, 2000). Various refinements have been proposed to try to discriminate between direct and indirect interactions in relevance networks. The CLR algorithm (Faith *et al.*, 2007a) modifies the MI score based on the empirical distribution of all MI scores. The ARACNE algorithm (Margolin *et al.*, 2006b) filters out indirect interactions from triplets of genes with the Data Processing Inequality (Cover and Thomas, 2006). Finally, MRNET (Meyer *et al.*, 2007) uses an iterative feature selection method based on a maximum relevance/minimum redundancy criterion.

Probabilistic graphical models have been widely used to model GRNs (Friedman, 2004). With respect to correlation or mutual information-based approaches, these methods are potentially able to model higher-order dependencies between the expression patterns of genes. Among these methods, Bayesian networks have been used since the advent of microarray technologies for GRN modeling and inference (Friedman *et al.*, 2000). A Bayesian network represents conditional independencies between random variables with a directed acyclic graph. Learning the structure of a Bayesian network is a non trivial problem (Auvray and Wehenkel, 2002; Chickering *et al.*, 2004), both from a theoretical and computational point of view, and several sophisticated heuristics have been proposed in the context of GRN inference (Auliac *et al.*, 2008; Yu *et al.*, 2004). One limitation of Bayesian networks for GRN inference is that these models do not allow the presence of cycles (feedback loops). While this limitation is partially circumvented by dynamic Bayesian networks (Perrin *et al.*, 2003; Yu *et al.*, 2004), these latter models can only be learned from time series expression data. Another family of probabilistic models that gained interest recently for GRN inference are Gaussian graphical models. These methods assume that gene expression values are jointly

Gaussian distributed and represent conditional independencies between genes by an undirected graph. The estimation of this graph for high-dimensional data is difficult but several robust solutions have been proposed in the literature (Ambroise *et al.*, 2009; Castelo and Roverato, 2009; Meinshausen and Bühlmann, 2006; Schäfer and Strimmer, 2005). Although often very effective, the main limitations of these methods are of course the Gaussianity assumption, which also implies linear dependencies between variables, and the undirected nature of the inferred regulatory links (although some heuristics have been proposed to direct them, Opgen-Rhein and Strimmer, 2007).

Within this context, we have developed a new GRN inference method based on variable selection with ensembles of regression trees. This method is called GENIE3 (for "GEne Network Inference with Ensemble of trees") and was best performer in the DREAM4 *In Silico Multifactorial* challenge in 2009 and in the DREAM5 *Network Inference* challenge in 2010. Its main features with respect to existing techniques are that it makes very few assumptions about the nature of the relationships between the variables (which can thus be non-linear) and can potentially capture high-order conditional dependencies between expression patterns. It also produces a *directed* graph of regulatory interactions and naturally allows the presence of feedback loops in the network. At the same time, it remains intuitive, computationally tractable, and easy to implement.

6.2 GENIE3

In this chapter, we focus on the unsupervised inference of gene regulatory networks from steady-state expression data. These data are obtained either from perturbation experiments or from observational experiments. A perturbation experiment consists in affecting the expression level of one or several gene(s) of the network. Examples of perturbation experiments comprise the knockout or overexpression of a gene, or the application of a drug. By contrast, observational experiments consist in observing the system without perturbing it. Observational data might correspond for example to expression profiles obtained from different patients or biological replicates. Such data are easier and less expensive to obtain than perturbation data and are thus more common in practice (Maathuis *et al.*, 2010). However, they are also less informative for the prediction of edge directionality (Maathuis *et al.*, 2010; Pournara and Wernisch, 2004; Werhli *et al.*, 2006) and therefore make the regulatory network inference task more challenging.

In Chapter 8, we describe how GENIE3 can be extended to other types of data, in particular expression data provided by time series experiments

and genetical genomics data.

In what follows, we define a (steady-state) learning sample from which to infer the network as a sample of N measurements:

$$LS = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \tag{6.1}$$

where $\mathbf{x}_k \in \mathbb{R}^p, k = 1, \ldots, N$ is the vector of expression values of all p genes in the kth experiment:

$$\mathbf{x}_k = (x_k^1, x_k^2, \ldots, x_k^p)^\top.$$

The goal of a network inference algorithm is to exploit the expression dataset LS to assign weights $w_{i,j} > 0, (i, j = 1, \ldots, p)$ to putative regulatory links from any gene i to any gene j, with the aim of yielding larger values for weights that correspond to actual regulatory interactions. We consider directed and unsigned edges, which means that $w_{i,j}$ can take a different value than $w_{j,i}$, and when gene i is connected to gene j, the former can be either an activator or a repressor of the latter. Note that in this thesis, we leave open the problem of choosing a threshold on the weights, in order to obtain a practical network, and focus on providing a ranking of the regulatory links.

To solve the network inference problem, the basic idea of our procedure is to decompose the problem of recovering a network involving p genes into p different subproblems, where each of these subproblems consists in identifying the regulators of one of the genes of the network. This idea has been exploited in other methods, such as MRNET (Meyer *et al.*, 2007) or the Graphical Lasso (Meinshausen and Bühlmann, 2006). Using expression data, the identification of the regulatory genes for a given target gene is defined as determining the subset of genes whose expression directly influences or is predictive of the expression of the target gene. Within the framework of supervised learning, this problem is equivalent to a feature selection problem. In this context, our solution exploits the embedded feature ranking mechanism of tree-based ensemble methods.

We first describe our procedure to solve the network inference problem using feature selection techniques and then specialize it to the case of tree-based ensemble methods.

6.2.1 Network inference as a feature selection problem

Our method makes the assumption that the expression of each gene in a given condition is a function of the expression of the other genes in the network in the same condition (plus some random noise). Denoting by $\mathbf{x}_k^{-j}$ the vector containing the expression values in the kth experiment of all genes except gene j:

$$\mathbf{x}_k^{-j} = (x_k^1, \ldots, x_k^{j-1}, x_k^{j+1}, \ldots, x_k^p)^\top,$$

we can write:

$$x_k^j = f_j(\mathbf{x}_k^{-j}) + \epsilon_k, \ \forall k, \tag{6.2}$$

where ϵ_k is a random noise with zero mean (conditionally to $\mathbf{x}_k^{-j}$). We further make the assumption that the function f_j only exploits the expression in $\mathbf{x}^{-j}$ of the genes that are direct regulators of gene j, i.e. genes that are directly connected to gene j in the targeted network. Recovering the regulatory links pointing to gene j thus amounts to finding those genes whose expression is predictive of the expression of the target gene. In machine learning terminology, this can be considered as a feature selection problem (in regression) for which many solutions exist (see Section 1.3). We assume here the use of a feature ranking technique that, instead of directly returning a feature subset, yields a ranking of the features from the most relevant to the less relevant for predicting the output.

The proposed network inference procedure is illustrated in Figure 6.1 and works as follows:

- For $j = 1$ to p:
  - Generate the learning sample of input-output pairs for gene j:

$$LS^j = \{(\mathbf{x}_k^{-j}, x_k^j), k = 1, \ldots, N\}. \tag{6.3}$$

  - Use a feature ranking technique on LS^j to compute confidence levels $w_{i,j}, \forall i \neq j$, for all genes except gene j itself.
- Combine the p individual gene rankings to get a global ranking of all regulatory links.

Note that depending on the interpretation of the weights $w_{i,j}$, their combination to get a global ranking of regulatory links is not trivial. We will see in the context of tree-based methods that it requires to normalize each expression vector appropriately.

6.2.2 Gene ranking with tree-based methods

The nature of the problem and the proposed solution put some constraints on candidate feature selection techniques. The nature of the functions f_j is unknown but they are expected to involve the expression of several genes (combinatorial regulation) and to be non-linear. The number of input features in each of these problems is typically much greater than the number of

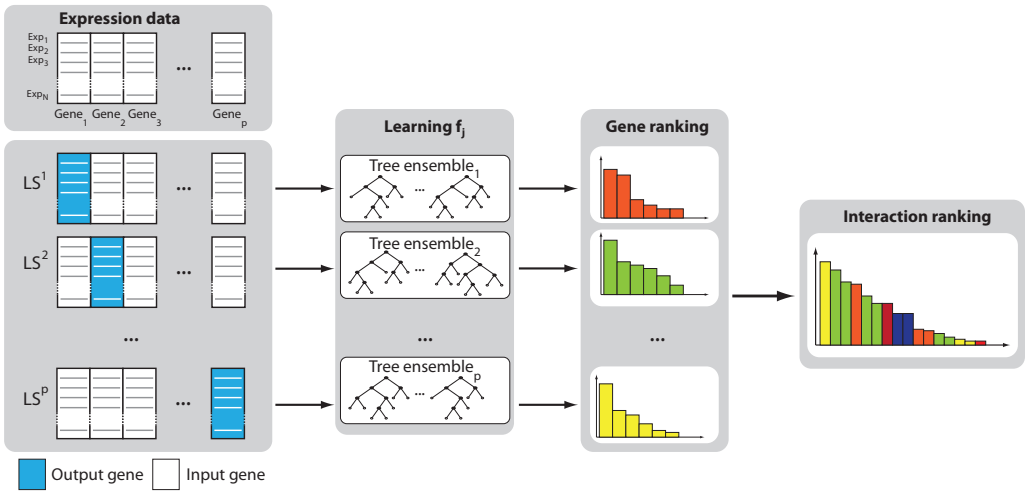


Figure 6.1: **GENIE3 procedure.** For each gene $j = 1, \ldots, p$, a learning sample LS^j is generated with expression levels of j as output values and expression levels of all other genes as input values. A function f_j is learned from LS^j and a local ranking of all genes except j is computed. The p local rankings are then combined to get a global ranking of all regulatory links.

observations. Computationally, since the identification of a network involving p genes requires to rerun the algorithm p times, it is also of interest for this algorithm to be fast and to require as few manual tuning as possible. Tree-based ensemble methods, such as Random Forests or Extra-Trees, are good candidates for that purpose. These methods do not make any assumption about the nature of the target function, can potentially deal with interacting features and non-linearity. They work well in the presence of a large number of features, are fast to compute, scalable, and essentially parameter-free.

6.2.3 Regulatory link ranking

In the GENIE3 procedure, a predictive model f_j in the form of an ensemble of trees is learned from LS^j, for each target gene j. Each tree-based model yields a separate ranking of the genes as potential regulators of a target gene j, derived from importance scores $w_{i,j}$ computed as sums of variance reductions in the form (2.12). As explained in Section 2.2.4, the sum of the importance scores of all input variables for a tree is usually very close to the

initial total variance of the output:

$$\sum_{i \neq j} w_{i,j} \approx N.\mathrm{Var}_j(LS'^j), \tag{6.4}$$

where LS'^j is the learning sample from which the tree was built (i.e. LS^j for the Extra-Trees method and a bootstrap sample for the Random Forests method) and where $\mathrm{Var}_j(LS'^j)$ is the variance of the target gene j estimated in the corresponding learning sample. As a consequence, if we trivially order the regulatory links according to the weights $w_{i,j}$, this is likely to introduce a positive bias for regulatory links towards the more highly variable genes. To avoid this bias, we first normalize the gene expressions so that they all have a unit variance in the training set, before applying the tree-based ensemble methods:

$$\mathbf{x}^j \leftarrow \frac{\mathbf{x}^j}{\sigma^j}, \ \forall j, \tag{6.5}$$

where $\mathbf{x}^j \in \mathbb{R}^N$ is the vector of expression levels of gene j in all N experiments and σ^j denotes its standard deviation. This normalization indeed implies that the different weights inferred from different models predicting the different gene expressions are comparable.

6.2.4 Computational complexity

The computational complexity of the Random Forests and Extra-Trees algorithms is on the order of $O(TKN \log N)$, where T is the number of trees, N is the learning sample size, and K is the number of randomly selected genes at each node of a tree. GENIE3's complexity is thus on the order of $O(pTKN \log N)$ since it requires to build an ensemble of trees for each of the p genes. The complexity of the whole procedure is thus log linear with respect to the number of measurements and, at worst, quadratic with respect to the number of genes (when $K = p - 1$).

To give an idea of the computing times, with our MATLAB® implementation of GENIE3, it takes 6.5 minutes to infer the five networks of the DREAM4 *Multifactorial* challenge and 24 hours to infer the *E. coli* network of the DREAM5 challenge (with known transcription factors), in both cases with Random Forests and $K = \sqrt{n_{TF}}$, where n_{TF} is the number of potential regulators (see Section 6.4 for the details of these experiments). These computing times were measured on a 16GB RAM, Intel L5420 2.50 GHz computer.

Note that, if needed, the algorithm can be easily parallelized as the p feature selection problems, as well as the different trees in an ensemble, are independent of each other.

6.2.5 Software availability

Our GENIE3 software is available from http://www.montefiore.ulg.ac.be/~huynh-thu/software.html.

6.3 The DREAM challenges

The Dialogue for Reverse Engineering Assessments and Methods (DREAM) initiative organizes an annual reverse engineering competition that comprises several challenges[1] (Marbach *et al.*, 2012; Prill *et al.*, 2010; Stolovitzky *et al.*, 2007, 2009). One of these challenges aims to evaluate the success of gene regulatory network inference algorithms on benchmarks of simulated and real data, in a double-blind way. Figure 6.2 illustrates the double-blind assessment procedure (using synthetic networks and data). For such a network inference challenge, the DREAM organizers typically generate or collect several gene expression datasets which are then provided to the participating teams. The goal of the challenge is, for each dataset, to provide a prediction of the underlying regulatory network, in the form of a list of all the potential (directed) regulatory links, ranked from the most to the less confident. Knowing the true networks (called *gold standards*), the organizers propose different statistics to evaluate a ranking of regulatory links corresponding to a network:

- AUPR: The area under the precision-recall (PR) curve;
- AUROC: The area under the receiver operating characteristic (ROC) curve;
- AUPR p-value: The probability that a given or larger AUPR is obtained by a random ranking of the potential network edges;
- AUROC p-value: The probability that a given or larger AUROC is obtained by a random ranking of the potential network edges.

Finally, to evaluate the performance of an algorithm on several networks, an overall score is used:

$$\text{overall score} = -0.5 \log_{10}(p_1 p_2), \tag{6.6}$$

where p_1 and p_2 are respectively the geometric means of AUPR p-values and AUROC p-values computed over the different networks. Thus, the higher the overall score, the better the performance of the algorithm.

[1] http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM_Project, http://www.the-dream-project.org/

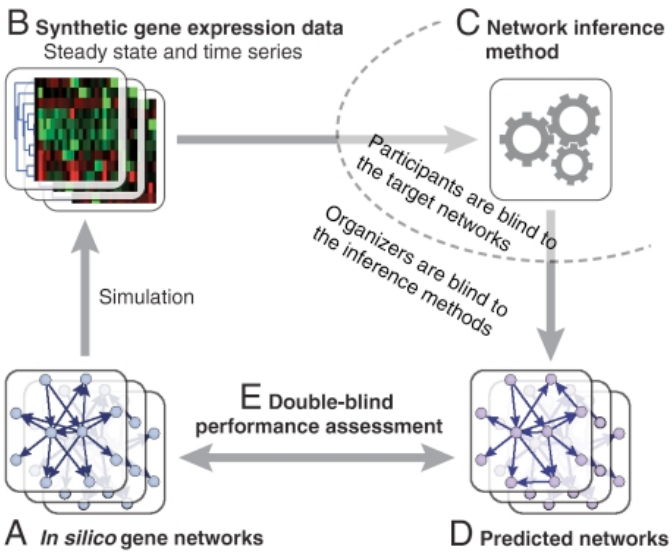


Figure 6.2: **Workflow of a network inference challenge of DREAM. A.** Several artificial gene regulatory networks are generated. **B.** Expression data are simulated from the artificial networks and provided to the challenge participants. **C.** Participants, being blind to the true networks, are asked to recover them from the provided data. **D.** and **E.** The organizers of the challenge evaluate the predictions of each participant, being blind to the inference algorithm that generated them. Figure taken from Marbach *et al.* (2010).

Note that the AUPR and AUROC p-values can be computed in different ways, depending on how a random ranking is defined. For the DREAM3 and DREAM4 editions, the organizers calculated the AUPR and AUROC scores for a large number of completely random orderings of the network edges and fitted the resulting histograms using stretched exponentials to obtain the null distributions (Stolovitzky *et al.*, 2009). For the DREAM5 challenge, the procedure for generating a random ranking was a bit different. In this procedure, the edge at each position k in one "random" ranking is randomly chosen among the edges that are ranked at the kth position in the rankings submitted by the challenge participants (Marbach *et al.*, 2012). Therefore, a p-value calculated using the first described procedure indicates the performance of a method compared to a random method, whereas a p-value calculated using the second procedure indicates its relative performance compared to the methods of the other challengers.

6.4 Results

We report below three series of experiments: first on the DREAM4 *In Silico Multifactorial* challenge, then on the M^{3D} *Escherichia coli* dataset, and finally on the DREAM5 *Network Inference* challenge.

6.4.1 The DREAM4 *Multifactorial* challenge

Challenge

The network inference challenge of the DREAM4 edition concerned *in silico* regulatory networks[2]. This challenge was divided into three sub-challenges, called *In Silico Size 10*, *In Silico Size 100*, and *In Silico Size 100 Multifactorial*. We only report here our results on this last sub-challenge. Results on the first two sub-challenges, which comprise time series experiments, are presented in Chapter 8.

The goal of the *In Silico Size 100 Multifactorial* sub-challenge was to infer five networks of $p = 100$ genes, each from multifactorial perturbation data. Multifactorial data are defined as static steady-state expression profiles resulting from slight perturbations of all genes simultaneously, and hence can be seen as observational data.

All networks and data were generated with GeneNetWeaver[3] (GNW, version 2.0) (Marbach *et al.*, 2009; Schaffter *et al.*, 2011). Network topologies were obtained by extracting subnetworks from the transcriptional regulatory networks of *E. coli* and *S. cerevisiae*. The subnetwork extraction method was adapted to preferentially include parts of the network with cycles but direct self-interactions were removed. The dynamics of the networks were simulated using a detailed kinetic model of gene regulation. Noise was added both in the dynamics of the networks and on the measurements of expression data. Multifactorial perturbations were simulated by slightly increasing or decreasing the basal activation of all genes of the network simultaneously by different random amounts. In total, the number of expression conditions N for each network was set to 100.

Predictions with GENIE3

We took part in the DREAM4 *In Silico Multifactorial* challenge. At the time of submission, the gold standard networks were unknown and it was thus impossible to choose the best one among several tree-based methods

[2] http://wiki.c2b2.columbia.edu/dream/index.php/D4c2
[3] http://gnw.sourceforge.net/genenetweaver.html

Table 6.1: **AUPR and AUROC scores for the DREAM4 *Multifactorial* challenge.**

	Method	NET1	NET2	NET3	NET4	NET5
AUPR	GENIE3-RF-sqrt	0.154	0.155	0.231	0.208	0.197
	2nd best	0.108	0.147	0.185	0.161	0.111
AUROC	GENIE3-RF-sqrt	0.745	0.733	0.775	0.791	0.798
	2nd best	0.739	0.694	0.748	0.736	0.745

GENIE3-RF-sqrt: GENIE3 using Random Forests with $K = \sqrt{p-1}$ and $T = 1000$. 2nd best: Second best performer in the DREAM4 *Multifactorial* challenge.

Table 6.2: **AUPR and AUROC p-values for the DREAM4 *Multifactorial* challenge.**

	Method	NET1	NET2	NET3	NET4	NET5	Overall p-value
AUPR p-value	GENIE3-RF-sqrt	3.3e-34	7.9e-54	1.8e-54	5.5e-47	4.6e-44	1.0e-46
	2nd best	5.6e-23	9.7e-50	6.6e-43	1.5e-35	4.4e-23	7.4e-35
AUROC p-value	GENIE3-RF-sqrt	3.3e-18	1.1e-28	9.7e-34	6.7e-33	1.9e-34	1.4e-29
	2nd best	1.7e-17	5.4e-21	4.9e-28	1.9e-23	1.1e-24	6.3e-23

GENIE3-RF-sqrt: GENIE3 using Random Forests with $K = \sqrt{p-1}$ and $T = 1000$. 2nd best: Second best performer in the DREAM4 *Multifactorial* challenge.

at our disposal. We thus submitted the rankings obtained by our GENIE3 procedure using the Random Forests algorithm with the default parameter $K = \sqrt{p-1}$ and growing $T = 1000$ trees[4].

Among twelve challengers, GENIE3 got the best performance with an overall score of 37.428. As a comparison, the score of the first runner-up was 28.165.

Table 6.1 shows the AUPR and AUROC values of our predictions and those of the first runner-up, and Table 6.2 shows their associated p-values, indicating that our predictions are significantly better than random guessing. On all networks, these scores are the highest among the twelve challengers. Individual PR and ROC curves for each network are collected in Figure B.1 in Appendix B.

[4]We used this value of T in all the experiments presented in this chapter.

Table 6.3: **Overall scores of GENIE3 for the DREAM4 networks.**

	RF-sqrt	RF-all	ET-sqrt	ET-all
Overall score	37.428	40.471	35.881	40.111

RF: Random Forests, ET: Extra-Trees, sqrt: $K = \sqrt{p-1}$, all: $K = p-1$.

Comparison of tree-based methods

We have subsequently applied GENIE3 on these same datasets, using the Extra-Trees algorithm, and also setting K to its maximum value ($K = p-1$). Table 6.3 shows the overall scores obtained with the four different combinations. The Random Forests and the Extra-Trees algorithms give comparable results, and the predictions are improved when the parameter K is increased, i.e. when the randomization is reduced. The overall best result is achieved when we use Random Forests with $K = p-1$, giving an overall score equal to 40.471. This result is slightly better than our initial submission to the challenge. Unless otherwise stated, all subsequent experiments on the DREAM4 datasets were carried out with this particular setting. Note that in this case, the algorithm simply corresponds to the Bagging method applied on standard regression trees.

Detailed analysis of the predictions

To have a more precise picture of the quality of the predictions obtained with GENIE3, Figure 6.3 depicts the ranking of the true regulators for all genes (genes are grouped according to their number of regulators), for the third network which is predicted with the highest AUPR score by our method. Similar plots for the other networks can be found in Figure B.2.

As observed in Figure 6.3, GENIE3 is able to retrieve the best regulator for about two thirds of the genes that have only one regulator. For genes with two regulators, the method retrieves one of the two regulators for about the same proportion of genes but is less good at retrieving the second regulator (only for one gene, the two regulators are at the top of the ranking). For genes with three or more regulators, even one regulator seems to be difficult to retrieve.

This suggests that the performance of GENIE3 at retrieving a regulator of one gene degrades as the number of regulators of this gene increases, as also observed by Marbach *et al.* (2010) from their analysis of the results of the DREAM3 challenge. To further check this hypothesis, we plotted in

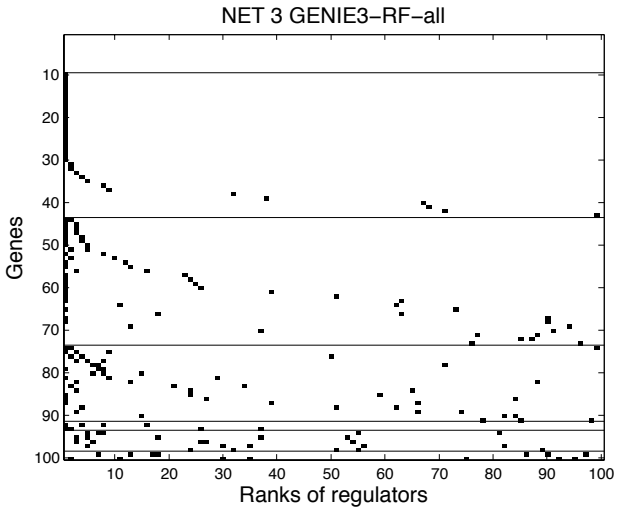


Figure 6.3: **Detailed results on DREAM4 NET3.** Ranking of the true regulators for all genes. Each row corresponds to a gene. Dots in each row represent the positions in the Random Forests ranking of the regulators of this gene. Genes are ordered on the y-axis according to their number of regulators in the gold standard network; those having the same number of regulators are grouped inside a horizontal block (from no regulator at the top to 6 regulators at the bottom). Inside each block, genes are ordered according to the median rank of their regulators. The ranking of interactions was obtained using Random Forests with $K = p - 1$ and $T = 1000$.

Figure 6.4 the median rank of the regulators of gene j, such that gene j is regulated by an increasing number of genes. The rank is presented here as a percentage, such that the first and last regulators of the ranking have a rank equal to 100% and 0% respectively. This plot clearly shows that the quality of the ranking monotonically decreases with the in-degree of the genes.

Undirected versus directed predictions

One interesting feature of GENIE3 is its potential ability to predict directed networks, while methods based on mutual information or correlation are only able to predict undirected networks.

To see to what extent the networks predicted by our method are asymmetric, we show in Table 6.4 for each network the proportion of predicted regulatory links for which the opposite link is not predicted. Notice that these predictions were obtained from the Random Forests ranking, by fixing a weight threshold such that the predicted network contains the same total

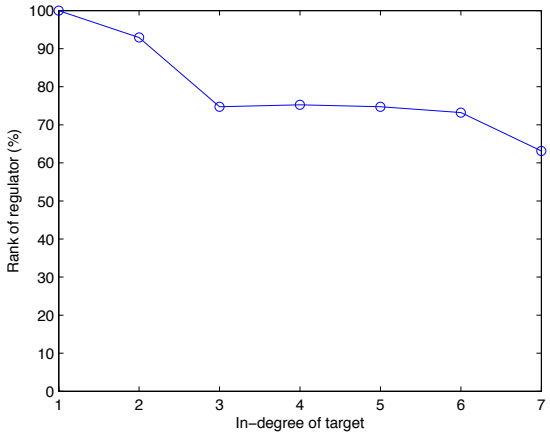


Figure 6.4: **Rank of regulators as a function of the in-degree of the target.** The in-degree of a target is its number of regulators. The dot corresponding to in-degree n is the median rank of regulators that regulate a gene with in-degree n, over the five networks. The rank is presented here as a percentage, such that the first and last regulators of each ranking have a rank equal to 100% and 0% respectively. The rankings of interactions were obtained using Random Forests with $K = p - 1$ and $T = 1000$.

Table 6.4: **Asymmetry of predicted and gold standard networks.**

	NET1	NET2	NET3	NET4	NET5
GENIE3-RF-all	50%	58%	48%	48%	58%
Gold standard	92%	94%	97%	96%	98%

The asymmetry of a network is measured by the proportion of regulatory links for which the opposite link is not present in the network.

number of edges as the gold standard. This percentage is compared with the same percentage computed for the gold standard. Our predicted networks are clearly more symmetric than the corresponding gold standards but they nevertheless contain a significant number of asymmetric predictions (52% of the links on the average, to be compared with 0% for a fully symmetric network).

Of course, the fact that GENIE3 predicts asymmetric networks does not ensure that the prediction of these asymmetric links is really informative; asymmetric predictions might precisely correspond to spurious predictions.

Table 6.5: **Error rates on edge directionality on the DREAM4 networks.**

Recall	5%	25%	50%	75%	100%
# Links	10	51	104	158	196
Error rate	20%	28%	27%	27%	26%

The error rate is the proportion of edges $i \to j$ in the gold standard network such that there is no edge $j \to i$ and for which our method wrongly predicts $w_{i,j} < w_{j,i}$. Each column corresponds to one value of the number of considered links of the gold standard, averaged over the five networks. These error rates were obtained using Random Forests with $K = p - 1$ and $T = 1000$, and averaged over the five networks.

To check this, we swapped the weights $w_{i,j}$ and $w_{j,i}$ for each pair of genes (i, j) and assessed the new resulting rankings. The overall score dropped from 40.471 to 14.674, suggesting that GENIE3 tends to correctly assign the highest weight to the true direction, given an undirected regulatory link.

To further assess the ability of our method to predict link directions, we computed the proportion of edges $i \to j$ in the gold standard network such that there is no edge $j \to i$ and for which our method wrongly predicts $w_{i,j} < w_{j,i}$. This can be considered as an error rate when our method is used for directing the edges of a known undirected network. Table 6.5 shows the average value of this error rate over the five networks, for increasing recall values, corresponding to different numbers of considered edges of the gold standard. Given that there are only two choices for a given link, a random ranking of the directed interactions would yield an error rate close to 50%. For all recall values, the error rate is significantly lower than 50%, suggesting that our method is a plausible approach for directing an undirected network. The error rate is smaller (20%) for the top-ranked interactions but it remains quite good (27%) even when considering less confident predictions.

Comparison to commonly used methods

We compared GENIE3 to several commonly used methods, which are three approaches based on the computation of mutual information (MI), namely CLR (Faith *et al.*, 2007a), ARACNE (Margolin *et al.*, 2006b), and MRNET (Meyer *et al.*, 2007), as well as one approach based on graphical Gaussian models (GGM) (Schäfer and Strimmer, 2005). All these four methods can only predict undirected networks. For these experiments, we used the original MATLAB® implementation of CLR (Faith *et al.*, 2007b) and the implemen-

Table 6.6: **Overall scores for the undirected networks of DREAM4.**

	GENIE3-RF-all	CLR	ARACNE	MRNET	GGM
Overall score	36.736	35.838	32.632	34.124	26.846

Links $i \to j$ and $j \to i$ were both assigned the same weights by CLR, ARACNE, MRNET, and GGM, while the predictions of GENIE3 were rendered symmetric by assigning to each pair (i, j) the maximum between $w_{i,j}$ and $w_{j,i}$.

Table 6.7: **Overall scores for the directed networks of DREAM4.**

	GENIE3-RF-all	CLR	ARACNE	MRNET	GGM
Overall score	40.471	31.57	28.488	30.435	23.705

Links $i \to j$ and $j \to$ were both assigned the same weights by CLR, ARACNE, MRNET, and GGM, while GENIE3 was used unmodified.

tations of ARACNE and MRNET in the *minet* R[5] package (Meyer *et al.*, 2008, 2009). To compute mutual information, we used a B-spline smoothing and discretization, as implemented in the CLR package, with the parameter setting used by Faith *et al.* (2007a) (10 bins and third order B-splines). For ARACNE, the tolerance parameter was optimized between 0 and 15%, as advised by Margolin *et al.* (2006b). For GGM, we used the *GeneNet* R package (Schäfer *et al.*, 2009; Schäfer *et al.*, 2006).

We carried two evaluations, the first one against the undirected gold standard (Table 6.6) and the second one against the directed gold standard (Table 6.7). In the first case, the predictions of GENIE3 were rendered symmetric by assigning to each pair (i, j) the maximum between $w_{i,j}$ and $w_{j,i}$. In the second case, links $i \to j$ and $j \to i$ were both assigned the same weights by the four undirected methods, while GENIE3 was used unmodified. In the undirected case, GGM gives the lowest score while all MI-based methods are equally good with only a slight advantage to our method. In the directed case, GENIE3 is significantly better than the four other methods that are constrained to predict undirected links.

[5] http://www.r-project.org/

GENIE3 and support vector machines

The framework of GENIE3 is general and other feature ranking methods can be used instead of tree-based methods to rank the putative regulators of each target gene. We thus carried out some experiments with feature rankings derived from linear models learned with support vector machines (SVMs), using the LIBSVM library (Chang and Lin, 2011).

For each gene $j = 1, \ldots, p$ of a network, a linear SVM model f_j is learned to predict the expression of this gene, from the expression levels of the other genes. The score of a regulatory link directed from one gene i to gene j is then given by the absolute value of the weight $w_{i,j}$ of gene i in the model f_j. To ensure that the weights are comparable across the p models, they are normalized in the following way:

$$|w_{i,j}| \leftarrow \frac{|w_{i,j}|}{\sum_{i \neq j} |w_{i,j}|}. \tag{6.7}$$

Figure 6.5 compares the performance of the SVMs to the Random Forests, on the directed and undirected networks. The parameter ϵ of the loss function in the SVM algorithm was fixed to its default value 0.1 and different values of the (inverse) regularization parameter C were tested. For the directed networks, the highest overall score (equal to 27.070) is obtained with $C = 10^{-4}$, and for the undirected networks, the highest overall score (equal to 27.797) is obtained with $C = 10^{-3}$. These scores remain however much lower than those obtained with the Random Forests. Linear SVMs are thus not competitive with tree-based methods on the DREAM4 *Multifactorial* challenge, but they would nevertheless have been ranked third among the official participating teams (with $C = 10^{-4}$).

6.4.2 The M^{3D} *E. coli* dataset

In addition to the DREAM4 *Multifactorial* challenge, we carried out experiments with our method on the inference of the regulatory network of *Escherichia coli*, which is well-studied and hence is used by several authors as a benchmark.

Dataset

The dataset of expression profiles was retrieved from the Many Microbe Microarrays (M^{3D}) database (Faith *et al.*, 2008) (version 4 build 6). It contains 907 *E. coli* microarray expression profiles of 4297 genes collected from the GEO (Barrett *et al.*, 2011), ArrayExpress (Parkinson *et al.*, 2009), and

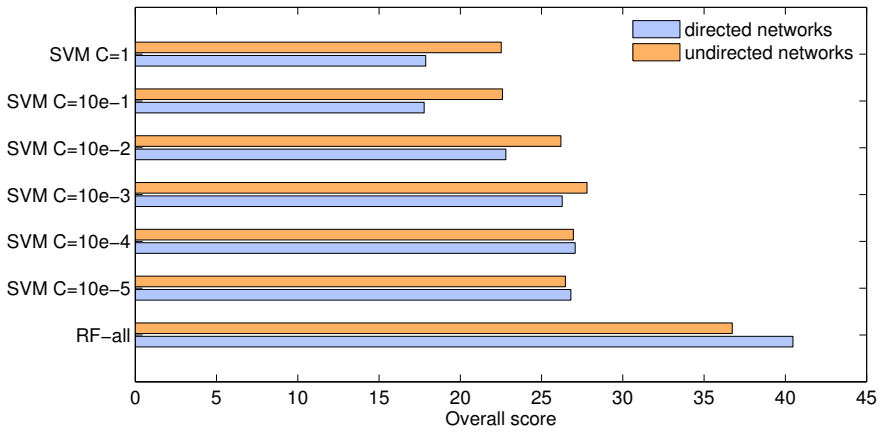


Figure 6.5: **Overall scores of GENIE3 for the DREAM4 networks.** SVM: weights of edges are absolute values of the weights of linear SVM models ($\epsilon = 0.1$). RF-all: weights of edges are derived from Random Forests models ($K = p - 1, T = 1000$).

ASAP (Glasner *et al.*, 2003) databases, as well as from individual investigators. The expression data were uniformly normalized using Robust Multichip Averaging (RMA) (Bolstad *et al.*, 2003). The resulting expression dataset is thus a compendium collecting different microarray experiments carried out in different laboratories. Note that some experiments are actually time series experiments. However, when applying a network inference procedure to the M^{3D} dataset, we considered each time point of a time series experiment as a separate (static) experiment, without taking into account any temporal aspect.

To validate the network predictions, we used 3433 experimentally confirmed regulatory interactions among 1471 genes that have been curated in RegulonDB (version 6.4) (Gama-Castro *et al.*, 2008). We considered the interactions annotated with at least one "weak" or "strong" evidence according to RegulonDB evidence classification.

Results

As a first experiment on the M^{3D} *E. coli* dataset, we adopted the same evaluation protocol as Faith *et al.* (2007a) that assumed to have prior knowledge about which genes of the gold standard (i.e. the experimentally confirmed interactions curated in RegulonDB) are transcription factors. In the context of our method, this makes each feature selection problem much easier as the regulators have to be identified among a much smaller set of genes. This

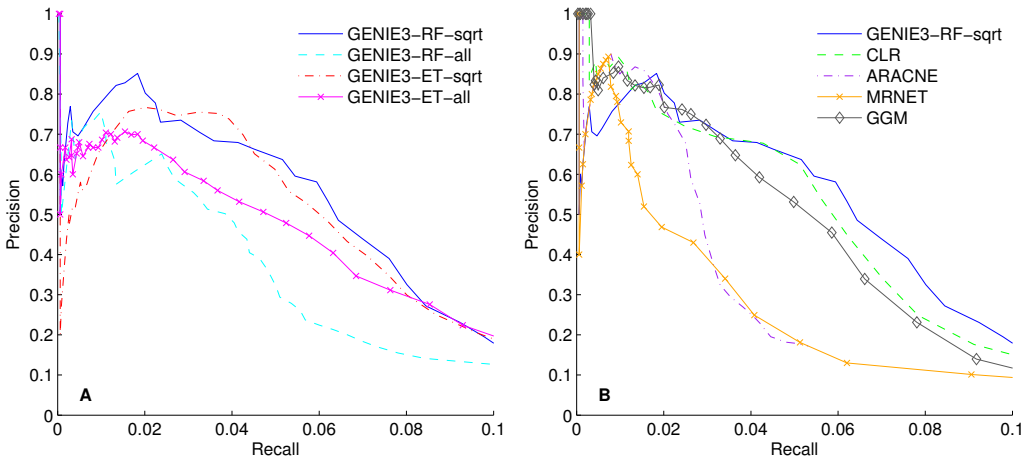


Figure 6.6: **PR curves of GENIE3 for the** M^{3D} ***E. coli*** **network.** Only known transcription factors were used as input genes. **A.** Comparison between the four different settings of the tree procedure. **B.** Comparison to other approaches.

also makes undirected and directed methods almost equally applicable since all links are automatically directed from transcription factors to genes. The only edges whose direction is not enforced are the links connecting two transcription factors. Figure 6.6(A) shows the precision-recall curves for the four different settings of the tree-based procedure. Contrary to the DREAM4 networks, setting $K = \sqrt{n_{TF}}$, where n_{TF} is the number of potential regulators, improves the performance compared to $K = n_{TF}$, RF-sqrt leading to the best precision-recall curve. Figure 6.6(B) compares this method with the four undirected methods, CLR, ARACNE, MRNET, and GGM, using exactly the same protocol. The predictions obtained using GENIE3 with Random Forests and $K = \sqrt{n_{TF}}$ outperform those obtained using ARACNE and MRNET, and give a precision-recall curve comparable to CLR and GGM (although less good for small recall values). Figure B.3 in Appendix B shows the ranking of the regulators of each gene, obtained with GENIE3-RF-sqrt.

As a second experiment, we simulated conditions similar to the DREAM4 challenge, where transcription factors were unknown, and tried to infer the network using as input features in each step of our procedure all 1471 genes except the target gene itself. For this experiment, precision never exceeded 6%, even for the smallest values of recall (see Figure 6.7). This indicates that the predictions are extremely poor and only slightly better than random

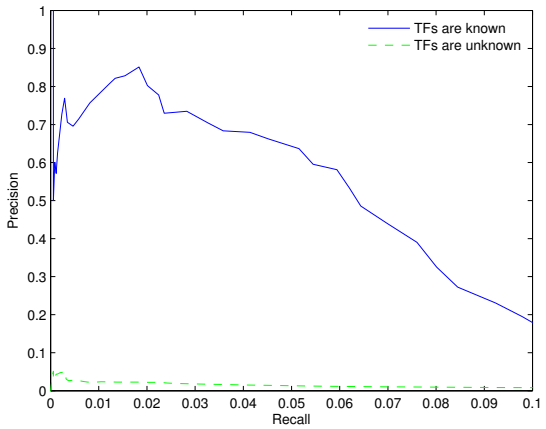


Figure 6.7: **PR curves for the** M^{3D} ***E. coli* network.** Blue plain curve: Only known transcription factors were used as input genes. Green dashed curve: All the genes were used as input genes. In both cases, the ranking of interactions was obtained using Random Forests with $K = \sqrt{n_{TF}}$ and $T = 1000$.

guessing.

With respect to the results obtained in the DREAM4 challenge, these results are disappointing. The larger number of genes in this case does not explain everything since it also comes with an increase of the number of observations. Actually in both cases, the number of observations is comparable to the number of genes. However, since the *E. coli* dataset is a collection of expression data compiled from experiments carried out in different laboratories, there may be some redundancy among these experiments or some bias in their selection. They are thus probably not as statistically useful as the really randomized and i.i.d. perturbation data generated for the DREAM4 *Multifactorial* challenge. Other potential reasons for these poor results are the fact that the gold standard network is not complete and also the discrepancy that exists between the simulation model used to generate the DREAM4 data and the real regulation mechanism of *E. coli*.

6.4.3 The DREAM5 *Network Inference* challenge

Challenge

The DREAM4 multifactorial datasets and networks were somewhat unrealistic: small number of genes in each network, expression datasets that were not high-dimensional (the number of genes was equal to the number of samples

in each dataset), i.i.d. perturbation data, etc. For the DREAM5 edition, the organizers thus aimed to generate a challenge closer to the reality. The DREAM5 *Network Inference* challenge[6] evaluated the algorithms on one artificial network as well as two real networks related to two micro-organisms, which are *Escherichia coli* and *Saccharomyces cerevisiae*. Note that the identities of the micro-organisms were unknown at the time of the challenge and that gene names were anonymized.

Each dataset contained gene expression levels collected from different types of experiments, including multifactorial perturbations, drug perturbations, gene deletion and overexpression experiments, and time series (Marbach *et al.*, 2012). The *E. coli* and *S. cerevisiae* datasets were compiled from the GEO database and were uniformly normalized using RMA[7]. The *in silico* network and dataset were generated using GNW (version 3.0), in a way such that the expression dataset contained the same set of experiments as the *E. coli* compendium (i.e. same number and types of experiments).

To validate the predictions of the *E. coli* network, the DREAM organizers used the experimentally confirmed interactions that have been curated in RegulonDB (version 6.8). They considered only the interactions annotated with at least one "strong evidence" according to RegulonDB evidence classification. For *S. cerevisiae*, they used a gold standard network derived from the analysis of ChIP binding data and evolutionary conserved transcription factor binding motifs (MacIsaac *et al.*, 2006). The network includes only interactions that have both strong evidence of binding and conservation[8].

Note that the challenge comprised a fourth dataset related to *Staphylococcus aureus*. As there is currently no gold standard network for this organism, algorithms were not evaluated on this dataset. Instead the pre-

[6] http://wiki.c2b2.columbia.edu/dream/index.php/D5c4

[7] A major part of the M^{3D} *E. coli* compendium that we exploited in Section 6.4.2 is actually common to the DREAM5 *E. coli* dataset.

[8] The challenge organizers actually tested the predictions of each team against different gold standard networks derived from MacIsaac *et al.* (2006) and obtained by varying the thresholds on the binding and the conservation (Marbach *et al.*, 2012). The gold standard based on the most stringent thresholds (strong evidence for both binding and conservation) is the one with which the inferred networks agree the most (i.e. high ratios between the obtained AUPRs and the AUPR of a random method are observed), and was hence retained as final gold standard. The organizers also tested two other gold standard networks derived from Hu *et al.* (2007) and from the YEASTRACT database (Abdulrehman *et al.*, 2011). The first gold standard leads to AUPRs that are not better than random predictions and the second gold standard leads to slightly higher AUPRs than the network derived from MacIsaac *et al.* (2006) and based on strong evidence. However, the YEASTRACT network was finally discarded because the evidence of its interactions is partially based on expression data. This gold standard network is therefore not completely independent of the predictions.

Table 6.8: **Datasets and gold standards of the DREAM5 *Network Inference* challenge.**

	Dataset			Gold standard		
	Chips	Genes	TFs	Edges	In-degree	Out-degree
In silico	805	1643	195	4012	3	23
S. aureus	160	2810	99	–	–	–
E. coli	805	4511	334	2066	2	15
S. cerevisiae	536	5950	333	3940	2	35

dictions of all the participating teams were used by the DREAM organizers to build a "community prediction" (see Chapter 7 for more details). This community network is the first comprehensive regulatory network available for this pathogen.

Table 6.8 shows the sizes of the different datasets, as well as the number of putative transcription factors (TFs) in each case. Indeed, as for the M^{3D} *E. coli* dataset, we had prior knowledge of which genes were candidate transcription factors for each network and only these transcription factors were allowed as regulatory genes in the predictions. The table also indicates, for each gold standard network, the number of edges, the average number of TFs that regulate each gene (in-degree), and the average number of genes regulated by each TF (out-degree).

Predictions with GENIE3

We took part in this challenge and submitted the rankings obtained by applying GENIE3 with the Random Forests and $K = \sqrt{n_{TF}}$, which was found to be the best performing combination on the M^{3D} *E. coli* dataset. Among 29 challengers, our method was the best performer with an overall score (as defined in Equation (6.6)) of 40.279. As a comparison, the score of the first runner-up was 34.023.

Table 6.9 shows the AUPR and AUROC values of our predictions and those of the first runner-up, and Table 6.10 shows their associated p-values[9]. GENIE3 yields much more accurate predictions for the *in silico* network

[9]We recall that these p-values do *not* indicate the performance of a method compared to a random method, but rather its relative performance compared to the other methods of the challenge.

Table 6.9: **AUPR and AUROC scores for the DREAM5 *Network Inference* challenge.**

	Method	*In silico*	*E. coli*	*S. cerevisiae*
AUPR	GENIE3-RF-sqrt	0.291	0.093	0.021
	2nd best	0.245	0.119	0.022
AUROC	GENIE3-RF-sqrt	0.815	0.617	0.518
	2nd best	0.780	0.671	0.519

GENIE3-RF-sqrt: GENIE3 using Random Forests with $K = \sqrt{n_{TF}}$ and $T = 1000$. 2nd best: Second best performer.

Table 6.10: **AUPR and AUROC p-values for the DREAM5 *Network Inference* challenge.**

	Method	*In silico*	*E. coli*	*S. cerevisiae*
AUPR p-value	GENIE3-RF-sqrt	1.6e-104	5.1e-20	1.6e-01
	2nd best	8.2e-53	1.1e-39	2.2e-02
AUROC p-value	GENIE3-RF-sqrt	3.1e-106	5.0e-11	1.1e-02
	2nd best	1.3e-56	1.6e-53	1.8e-03

GENIE3-RF-sqrt: GENIE3 using Random Forests with $K = \sqrt{n_{TF}}$ and $T = 1000$. 2nd best: Second best performer.

than for the real networks. Nevertheless, the p-values that we obtain for *E. coli* indicate that our method performs significantly better than most of the other methods used by the participants on this network. By contrast, our predictions of the *S. cerevisiae* network are only slightly better than random guessing (AUPR close to zero and AUROC close to 0.5). PR and ROC curves for each network are plotted in Figure B.4.

A detailed comparison of the methods used by the different participating teams for this challenge is provided in Chapter 7.

Comparison of tree-based methods

Table 6.11 shows for each network the AUPR scores obtained with the four settings of the tree-based procedure. Comparable results are obtained with Random Forests and Extra-Trees. However we observe a difference between the results obtained for the *in silico* network and those obtained for the real

Table 6.11: **AUPR scores of GENIE3 for the DREAM5 *Network Inference* challenge.**

	RF-sqrt	RF-all	ET-sqrt	ET-all
In silico	0.291	0.381	0.258	0.354
E. coli	0.093	0.067	0.094	0.079
S. cerevisiae	0.021	0.020	0.020	0.020

RF: Random Forests, ET: Extra-Trees, sqrt: $K = \sqrt{n_{TF}}$, all: $K = n_{TF}$.

networks. In the first case, the performance of GENIE3 is improved when K is increased to its maximum value ($K = n_{TF}$), while in the second case, the best performance is obtained when K is set to $\sqrt{n_{TF}}$.

TF-TF edge directionality

As potential TFs are known for each network, all regulatory links are automatically directed from these TFs to genes. However the directions of the edges connecting two TFs still have to be predicted. To assess the ability of GENIE3 to predict edge directionality, we computed the error rate on the direction of the TF-TF links, i.e. the proportion of edges $i \rightarrow j$ connecting two TFs in the gold standard network, such that there is no edge $j \rightarrow i$ and for which the method wrongly predicts $w_{i,j} < w_{j,i}$. Table 6.12 shows the error rates obtained on the different networks, for different numbers of considered TF-TF links. The error rates are quite acceptable on the *in silico* network (16% for the top-ranked links and 32% when considering all the links), but are higher on the real networks. When considering all the links, the error rate is equal to 39% on the *E. coli* network and 57% for *S. cerevisiae*. These latter results are much worse than what we obtained on the networks of the DREAM4 *Multifactorial* challenge.

Comparison to commonly used methods

Figure 6.8 compares GENIE3-RF-sqrt to CLR, ARACNE, MRNET, and GGM, and shows that GENIE3 compares very well to these methods on the *in silico* and *E. coli* networks.

An interesting observation can be made with respect to the results of GGM. Among all, this method returns the worse predictions on the *in silico* network but is the best performer on the real networks. On the *S. cerevisiae* network, GGM is even the only method that is able to recover some true

Table 6.12: **Error rates on TF-TF edge directionality on the DREAM5 networks.**

	In silico		*E. coli*		*S. cerevisiae*	
Recall	# Links	ER	# Links	ER	# Links	ER
10%	19	16%	2	50%	49	59%
25%	50	28%	22	45%	107	60%
50%	106	30%	55	33%	202	58%
75%	167	34%	97	37%	274	56%
100%	214	32%	121	39%	314	57%

The error rate (ER) is the proportion of edges $i \to j$ connecting two transcription factors in the gold standard network, such that there is no edge $j \to i$ and for which our method wrongly predicts $w_{i,j} < w_{j,i}$. Each line corresponds to one value of the number of considered TF-TF links of the gold standard. These error rates were obtained using Random Forests with $K = \sqrt{n_{TF}}$ and $T = 1000$.

regulatory links. On the other hand, MRNET is competitive with respect to the other methods on the *in silico* network, but gives poorer predictions for the *E. coli* network. These results, together with those obtained with GENIE3 when using different values of the parameter K of the trees, highlight the fact that there still exist some discrepancies between the real networks and datasets and their artificial counterparts, which deserve to be further studied (see Chapter 9 for further discussion on that matter).

We also tried to learn the different networks while ignoring the knowledge of the transcription factors and thus allowed any gene of a network to be a regulatory gene. Results are shown in Figure 6.9. As for the M^{3D} *E. coli* dataset, predictions are much poorer for each network. However most methods are still able to recover some regulatory links for the *in silico* network while they do not perform better than random guessing on the real networks.

GENIE3 and support vector machines

Finally we learned the different networks by using the GENIE3 framework with linear SVMs instead of tree-based methods. The results are shown in Figure 6.10, for different values of the parameter C. On the *in silico* network, the Random Forests method remains the best performer while higher scores are obtained with the SVMs on the real networks. For *E. coli*, the predictions obtained with SVMs and $C = 0.1$ even outperform (in terms of AUPR) the predictions obtained by the best official performer on this network. On

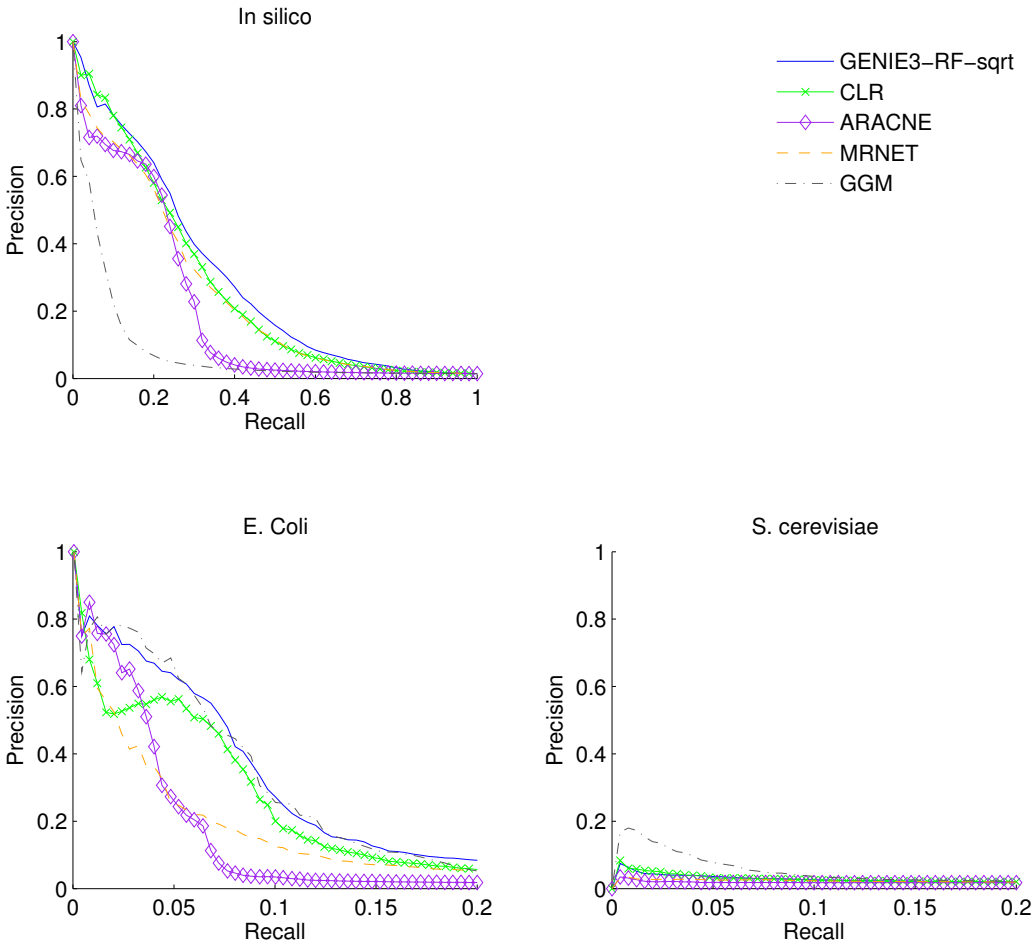


Figure 6.8: **PR curves for the DREAM5 networks.** Only known transcription factors were used as input genes. Note the change of scale on the recall-axis for *E. coli* and *S. cerevisiae*.

the *S. cerevisiae* network, the SVMs would have been ranked fifth with the same value of the C parameter, while GENIE3-RF-sqrt is ranked at the 9th position. Note that, like for the parameter K of the tree-based methods, the optimal value of C is problem-dependent.

6.4.4 Feature selection

In this work, we mainly focus on exploiting expression data in order to provide a ranking of the putative regulatory interactions. In some applications however, one would like to retrieve a practical predicted network rather than

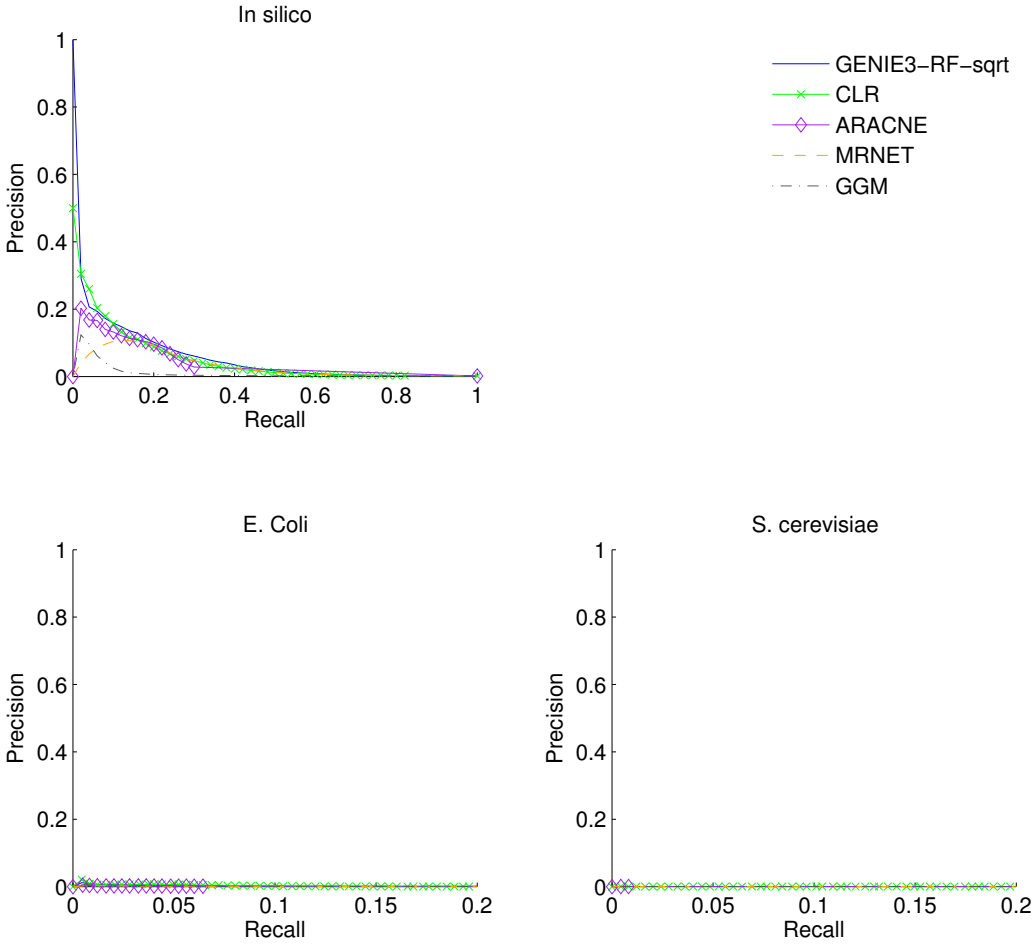


Figure 6.9: **PR curves for the DREAM5 networks.** All the genes were used as input genes. Note the change of scale on the recall-axis for *E. coli* and *S. cerevisiae*.

a ranking. For that purpose, one solution would be to apply a feature selection technique, such as one of those presented in Chapter 4, to each of the p rankings of putative regulatory genes, in order to select the top-ranked input genes as regulators of the corresponding target gene. We tried this procedure on the DREAM4 datasets, with each feature selection method evaluated in Chapter 4, using a significance level $\alpha = 0.05$.

Figure 6.11 shows the average precision and recall of the resulting predicted networks, when the weight $w_{i,j}$ of a regulatory link is respectively the importance of gene i in a Random Forests model (learned with $K = p - 1$) predicting the expression of gene j, the weight of gene i in a linear SVM model

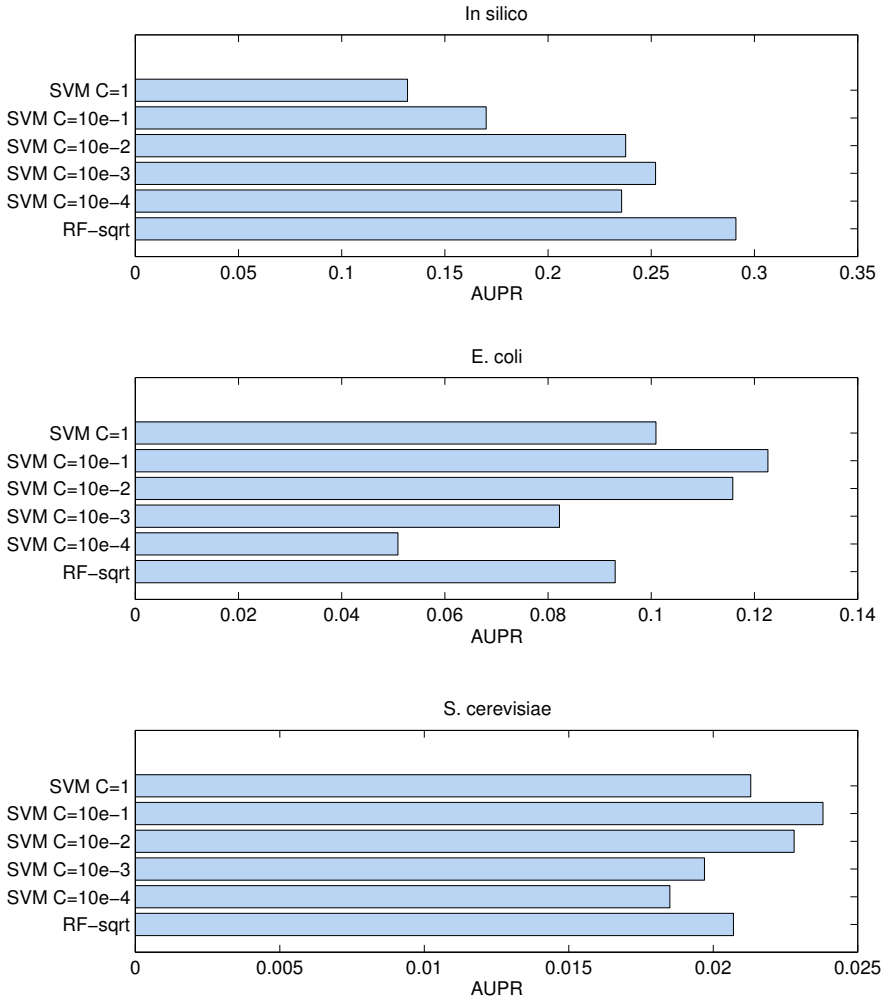


Figure 6.10: **AUPR scores of GENIE3 for the DREAM5 networks.** SVM: weights of edges are absolute values of the weights of linear SVM models ($\epsilon = 0.1$). RF-sqrt: weights of edges are derived from Random Forests models ($K = \sqrt{n_{TF}}, T = 1000$).

(learned with C and ϵ fixed to their default value, 1 and 0.1 respectively) predicting the expression of gene j, and the absolute value of the Spearman's rank correlation between the expression pattern of gene i and the expression pattern of gene j. We also show for comparison the precision and recall of two other methods, rec-1 and prec-1. Given a ranking of candidate regulatory genes corresponding to a target gene, rec-1 is a method that selects the subset of k top-ranked genes, where k is the smallest integer such that the subset contains all the true regulators of the target gene. On the other

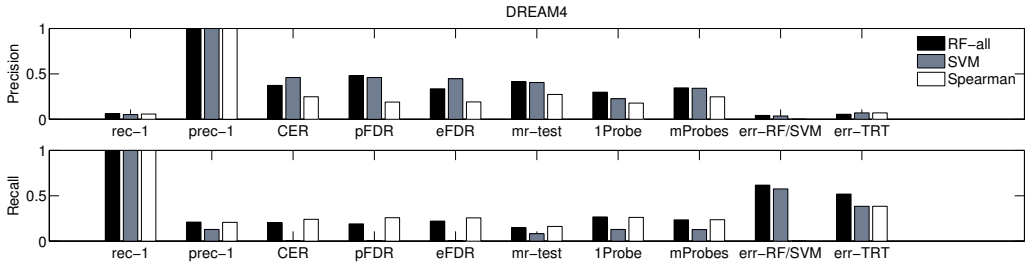


Figure 6.11: **DREAM4 networks.** Precision and recall of the resulting predicted networks, when different feature selection methods are applied to the rankings of putative regulatory genes (with $\alpha = 0.05$). The precision and recall values were averaged over the five networks.

side, prec-1 selects the subset of k top-ranked genes, where k is the largest integer such that the subset contains only true regulators. As in Chapter 4, we observe that the err-RF/SVM and err-TRT procedures have the highest recall values and the lowest precision levels. However the precision levels of all the methods are now much lower than those obtained on the datasets of Chapter 4. None of them is higher than 50%, meaning that more than half of the regulatory links predicted by each method are actually false positives. This result can be explained (at least partially) by the indirect effects that all the genes have on each other. Even if there is no direct regulatory link from gene i to gene j in the gold standard network, the expression of gene i can still be predictive of the expression of gene j, through one or several other genes (e.g. gene i regulates some gene k which in turn regulates gene j). In conclusion, each gene of the network is indirectly regulated by (almost) all the other genes, but most of these indirect regulatory links are considered false positives in our evaluation because they are not part of the gold standard.

To check if the different methods would select regulatory links $i \rightarrow j$ such that the expression of gene i is really irrelevant for the prediction of the expression of gene j, we generated five datasets that each contain 100 samples and 200 genes. In each dataset, the expression values of the first 100 genes are the original expression values contained in one of the DREAM4 datasets while the expression values of the 100 remaining genes are uniformly distributed random numbers between 0 and 1. Then a selected regulatory link from gene i to any target gene is considered a true positive if $i \leq 100$ and a false positive if $i > 100$. Results are shown in Figure 6.12, where indeed we can observe that all the methods except err-RF/SVM and err-TRT hardly select a noise gene as regulator (the precision is almost equal to one).

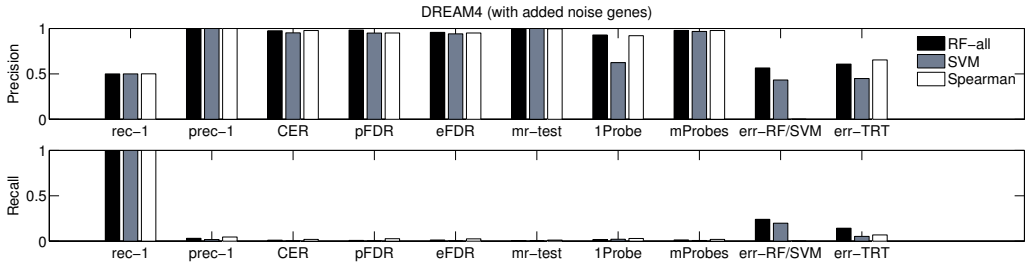


Figure 6.12: **DREAM4 networks with additional noise genes.** Precision and recall of the resulting predicted networks, when different feature selection methods are applied to the rankings of putative regulatory genes (with $\alpha = 0.05$). The precision and recall values were averaged over the five networks.

This experiment clearly highlights the fact that the different feature selection methods presented in Chapter 4 are designed to identify the *maximal* subset of relevant variables, i.e. all variables that get to their position in the ranking because they convey at least some information about the output and not merely by chance. In the context of network inference, we would need a method to determine a *minimal* subset of variables that convey all information about an output (and thus make all other variables conditionally irrelevant) to avoid the inclusion of indirect effects. An optimal treatment of this problem would probably require to adopt a more global approach of the problem, exploiting jointly all the individual rankings of all target genes.

6.5 Discussion

We developed GENIE3, a procedure that aims to recover a gene regulatory network from steady-state expression data. This procedure decomposes the problem of inferring a network of size p into p different feature selection problems, the goal of each being to identify the regulators of one of the genes of the network. Among different feature selection methods, we chose to use tree-based ensemble methods. These methods do not make any assumption about the nature of gene regulation and can potentially deal with combinatorial regulations and non-linearity. They work well in the presence of a large number of genes, are fast to compute and scalable.

GENIE3 got the best overall performances in the DREAM4 *In Silico Multifactorial* challenge and in the DREAM5 *Network Inference* challenge. The method is competitive with existing algorithms to decipher the genetic regulatory network of *Escherichia coli* assuming that transcription factors

are known (whether using the M^{3D} or the DREAM5 expression dataset). When no prior knowledge is available about transcription factors, our results on the *E. coli* network were however not better than random guessing.

Several discrepancies between artificial and real networks were observed. When applying GENIE3 on the DREAM4 *Multifactorial* datasets and on the DREAM5 *in silico* dataset, improved predictions were obtained by increasing the main parameter K of the tree-based methods, i.e. the number of randomly selected attributes at each node of one tree, to its maximum value ($K = n_{TF}$), while on the *E. coli* dataset, the best rankings of interactions were obtained by using $K = \sqrt{n_{TF}}$. Also, some methods that performed better than others on the *in silico* network of DREAM5 performed worse on the real networks, and vice versa. The reasons of these differences between artificial and real networks deserve to be further analysed.

In our research, we focus on providing a ranking of the regulatory interactions. In some applications however, one would like to retrieve a practical predicted network rather than a ranking. For that purpose, we tried to apply the feature selection techniques presented in Chapter 4 to each "local" ranking of putative regulatory genes obtained from the DREAM4 datasets, in order to select the top-ranked genes as regulators of the corresponding target gene. However, the resulting networks had an average precision that was lower than 50%. We attribute this low precision to the fact that each gene of a network is actually indirectly regulated by almost all the other genes of the network, but most of these indirect links are considered false positives because they are not part of the gold standard. We therefore need to adapt the different feature selection procedures in order to select, for each target gene, a minimal subset of non-redundant genes as regulators. Another direction of research would be to extend these techniques to help determining a threshold on the "global" ranking of the regulatory links, by assessing the significance of the top-ranked interactions.

The GENIE3 algorithm, which returns rankings of regulatory links, can be improved along several directions. As tree-based ensemble methods, we used the Random Forests and the Extra-Trees algorithms, that both gave comparable results. However, the performances of these methods depend to some extent on their main parameter K. It would thus be of interest to find a way to automatically tune this parameter. A first solution could be to select the value of the parameter that leads to the best performance for the prediction of the expression values, i.e. that minimizes the mean square error (MSE) estimated by cross-validation:

$$\text{MSE} = \sum_{j=1}^{p} \sum_{k=1}^{N} (x_k^j - f_j(\mathbf{x}_k^{-j}))^2. \tag{6.8}$$

Unfortunately, this solution did not work on the M^{3D} *E. coli* dataset, where using $K = n_{TF}$ led to a lower mean square error but a less good precision-recall curve.

There is also a potential room for improvement on the way variable importance scores are normalized. One apparent drawback of the measure we proposed is that it does not take into account the quality of the trees in generalization. Indeed since our trees are fully grown, importance weights satisfy Equation (6.4) which, given our normalization, attributes equal weights to all tree models irrespective of their quality when used to predict the expression values of the target gene. We tried to correct for this bias by normalizing the variable importance scores by the effective variance reduction brought by the model as estimated by cross-validation but it actually deteriorated the performances. The question of the optimal normalization remains thus open at this stage.

Notice that the two last results meet one of the conclusions of the Part II of this thesis, stating that using prediction accuracy as a criterion for assessing the quality of a ranking and selecting relevant features is often counter-productive.

Our experiments on the DREAM4 and DREAM5 datasets show that GENIE3 is able to predict to some extent the direction of the edges, even though it only exploits steady-state measurements. However, the reason why we are able to predict directionality is not clear. Predicting causality from static data alone is commonly admitted to be a difficult problem, although being possible in some situations (Bontempi and Meyer, 2010). As a matter of fact, we will see in Chapter 8 that either using time series data or combining steady-state expression data with genetic data allows to significantly improve the predictions made on the direction of the regulatory links.

Bayesian networks are methods that also potentially allow to predict edge directionality, and a comparison with this family of methods is performed in Chapter 7, using the DREAM5 challenge. Note that with respect to our approach, Bayesian networks do not allow for the presence of (directed) cycles in the predicted network, which could be a limiting factor for networks such as those in DREAM4 that contain cycles by construction.

Several procedures using regression trees have already been proposed to solve the regulatory network inference problem. Most of these procedures exploit other kinds of data in addition to expression data, e.g. counts of regulatory motifs that serve as binding sites for transcription factors (Phuong *et al.*, 2004; Ruan and Zhang, 2006), or ChIP-based binding data (Xiao and Segal, 2009). The closest work to ours is the procedure developed by Segal *et al.* (2005), that recovers module networks from expression data, so that the genes in each module share the same regulators in the network and the same

conditional probability distribution, represented by a (single) regression tree.

Finally, although we exploited tree-based ensemble methods, our framework is general, and other feature selection techniques could have been used as well. Actually, several existing methods for network inference can be interpreted as special instances of this framework. In particular, mutual information, as used in Relevance Networks (Butte and Kohane, 2000) or CLR (Faith *et al.*, 2007a), is a common dependency measure exploited in filter-kind approaches for feature selection (Saeys *et al.*, 2007). MRNET (Meyer *et al.*, 2007) also considers each gene in turn as the target output and exploits the maximum relevance/minimum redundancy feature selection method to rank its candidate regulators. Like Relevance Networks and CLR, this method reduces all the information contained in the expression data to mutual information between all pairs of genes, while our approach is by nature multivariate. Meinshausen and Bühlmann (2006) showed that finding the zero entries in the inverse covariance matrix of a multivariate Gaussian distribution can be solved by applying the LASSO embedded feature selection mechanism using each gene in turn as the target output, which links Gaussian graphical models with our approach. While the latter assumes that the functions f_j in Equation (6.2) are linear, our approach can be seen as a relaxation of this assumption by exploiting a non-parametric supervised learning method. Note however that preliminary experiments with feature rankings derived from linear SVM models showed that using non-linear models does not seem to be an advantage for the inference of the *E. coli* and *S. cerevisiae* networks. On the other hand, the tree-based methods yielded better performances on the artificial problems, which are non-linear by construction.

7

Results of the DREAM5 *Network Inference* challenge

This chapter summarizes the results presented in the publication of Marbach *et al.* (2012), in which the organizers of the DREAM challenges assessed the performances of the methods used by the teams that participated in the DREAM5 *Network Inference* challenge. They then presented a simple procedure to derive community-based networks, that combines the predictions of all the methods of the DREAM community, and that has a more robust performance than most of the individual methods across the different networks. They also performed a machine learning-based analysis that shows that the knockout and overexpression of a transcription factor are experiments that are highly informative for the prediction of the target genes of that transcription factor. Section 7.1 briefly recalls the framework of the DREAM5 *Network Inference* challenge, Section 7.2 presents its main results, and Section 7.3 concludes the chapter.

We are part of the authors of this publication, as members of the DREAM5 consortium. While we have critically read and approved the manuscript, all experiments have been performed by the DREAM organizers. Therefore, this chapter should not be considered as a contribution of this thesis, but we nevertheless introduce it here as it sheds new light on the GENIE3 method.

Contents

This chapter is a summary of the following publication:

Marbach, D., Costello, J. C., Küffner, R., Vega, N., Prill, R. J., Camacho, D. M., Allison, K. R., the DREAM5 Consortium (including Geurts, P., Huynh-Thu, V. A., Irrthum, A., Saeys, Y., and Wehenkel, L.), Kellis, M., Collins, J. J., and Stolovitzky, G. (2011) Wisdom of crowds for robust gene network inference. *Nature Methods* **9**:796-804.

7.1 The DREAM5 challenge

In this publication, the organizers of the DREAM challenges presented the results of the DREAM5 *Network Inference* challenge, whose goal was the inference of the transcriptional regulatory networks of a prokaryotic organism (*Escherichia coli*), an eukaryotic organism (*Saccharomyces cerevisiae*), and a human pathogen (*Staphylococcus aureus*), as well as an *in silico* network. The framework of the challenge is illustrated in Figure 7.1. To infer each network, participants had at their disposal a compendium containing gene expression levels collected in hundreds of different conditions, including environmental perturbations, drug perturbations, gene deletion and overexpression experiments, and time series. The *E. coli*, *S. cerevisiae*, and *S. aureus* compendia were compiled from the Gene Expression Omnibus (GEO) database (Barrett *et al.*, 2011), while the *in silico* network and dataset were generated using GeneNetWeaver (Marbach *et al.*, 2009; Schaffter *et al.*, 2011), in a way such that the expression dataset contained the same set of experiments as the *E. coli* compendium. In addition to expression data, a list of candidate transcription factors (TFs) was provided for each compendium, as well as a number of descriptive features for each microarray experiment. The names of all the genes and TFs were anonymized. The goal of the challenge was to provide predictions for each network, in the form of a list of directed regulatory links between TFs and target genes, ranked from the most confident to the less confident. The gold standards used by the organizers to evaluate the submitted predictions consisted in sets of experimentally verified interactions for *E. coli* and *S. cerevisiae*, and the known *in silico* network. Algorithms were not evaluated on the inference of the *S. aureus* network, as there is currently no gold standard for that organism. More details concerning the challenge are provided in Section 6.4.3 of Chapter 6.

The results of the challenge are presented in the next section. Besides the assessment of 35 network inference methods, the organizers also presented

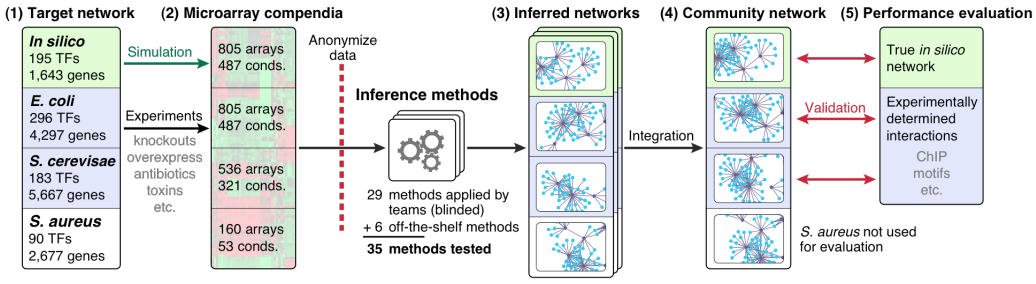


Figure 7.1: **The DREAM5 *Network Inference* challenge. 1.** The goal of the challenge was to infer one artificial gene regulatory network, as well as the networks of *E. coli*, *S. cerevisiae*, and *S. aureus*. **2.** To infer each network, the participants had at their disposal a compendium of gene expression profiles collected in various experimental conditions and in which the names of the genes and transcription factors were anonymized. **3.** 29 teams participated in the challenge. In addition, the challenge organizers applied 6 "off-the-shelf" inference methods. **4.** They also integrated the predictions of all the teams to build community networks. **5.** The predictions returned by each method were evaluated by comparing them to the true *in silico* network and to experimentally confirmed interactions for *E. coli* and *S. cerevisiae*. Methods were not evaluated on the inference of the *S. aureus* network, as few verified interactions currently exist for this organism. Figure taken from Marbach *et al.* (2012).

an integration scheme to derive community-based networks and showed that this community-based procedure produces more robust predictions than most of the individual algorithms. They also validated experimentally several *de novo* community predictions for *E. coli* and made available the community-based *S. aureus* network, which is the first comprehensive regulatory network for this organism.

7.2 Results of the challenge

7.2.1 Network inference methods

In total, 29 teams participated in the challenge. In addition, the organizers applied six commonly used "off-the-shelf" inference methods. The 35 methods are listed in Tables 7.1 and 7.2, and are divided into six categories, based on the descriptions of the methods provided by the participants: *(Linear) Regression*, *Mutual information (MI)*, *Correlation*, *Bayesian networks*, *Other* (containing methods that do not belong to any of the previous categories), and *Meta* (containing methods that combine several different approaches).

Table 7.1: **Categorization of the network inference methods (Part 1/2).**

ID	Synopsis	
	REGRESSION: TFs are selected by TG specific (1) sparse regression and combined by (2) stability selection.	
1	(1) Lasso; (2) the regularization parameter is chosen so that 5 TFs are selected per TG in each bootstrap sample.	Haury *et al.* (2011b)
2	(1) Steady state and time series data are combined by group lasso; (2) bootstrapping.	Yuan and Lin (2006)
3	Combination of lasso (TESLA toolbox) and Bayesian piecewise linear regression models learned from RJMCMC simulations.	Lèbre *et al.* (2010)
4	(1) Lasso; (2) bootstrapping.	Meinshausen and Bühlmann (2010)
5	(1) Lasso; (2) area under the stability selection curve.	Meinshausen and Bühlmann (2010)
6	Application of the Lasso toolbox GENLAB using standard parameters.	van Someren *et al.* (2006)
7	Lasso, regression models are combined by the maximum regularization parameter value selecting a given edge for the first time.	Meinshausen and Bühlmann (2010)
8	Linear regression determines the contribution of a TF to the expression of the TG.	D'Haeseleer *et al.* (1999)†
	MUTUAL INFORMATION: Interactions are ranked based on variants of mutual information (MI).	
1	CLR: for a given $TF_x - TG_y$ edge, an MI re-scoring scheme reflects TG_y and TF_x in the distribution of all TGs and all TFs respectively.	Faith *et al.* (2007a)†
2	MI is computed from discretized expression values.	Butte and Kohane (2000)†
3	ARACNE: Augments MI by a kernel estimator to avoid discretization and by the data processing inequality to distinguish interaction direction.	Margolin *et al.* (2006a)†
4	Topology is estimated by MI. The direction of edges is determined by Markov-blanket (HITON-PC)/Bayesian local causal discovery (BLCD).	Mani and Cooper (2004)
5	Topology is estimated by MI and Pearson's correlation. The direction of edges is determined by HITON-PC/BLCD.	Mani and Cooper (2004)
	CORRELATION: Interactions are ranked based on variants of correlation.	
1	Absolute value of Pearson's correlation coefficient.	Butte and Kohane (2000)
2	Signed value of Pearson's correlation coefficient.	Butte and Kohane (2000)†
3	Signed value of Spearman's correlation coefficient.	Butte and Kohane (2000)†
	BAYESIAN NETWORKS optimize posterior probabilities by different heuristic searches.	
1	Simulated annealing (catnet R package), aggregation of three runs.	Balov and Salzman (2011)
2	Simulated annealing (catnet R package).	Balov and Salzman (2011)
3	Max-Min Parent and Children algorithm (MMPC), bootstrapped datasets.	Tsamardinos *et al.* (2003)
4	Markov blanket algorithm (HITON-PC), bootstrapped datasets.	Aliferis *et al.* (2010)
5	Markov boundary induction algorithm (TIE*), bootstrapped datasets.	Statnikov and Aliferis (2010)
6	Models TF KO/OE data and time series by dynamic Bayesian networks (Infer.NET toolbox).	Minka *et al.* (2010)

Within each category, methods are ordered according to their overall performance. †Off-the-shelf algorithm applied by challenge organizers. TF: transcription factor, TG: target gene, TF KO/OE data: measurements where TFs have been deleted or overexpressed. Table modified from Marbach *et al.* (2012).

Table 7.2: **Categorization of the network inference methods (Part 2/2).**

ID	Synopsis	
	OTHER APPROACHES: Network inference by heterogeneous and novel methods.	
1	GENIE3: A Random Forest is trained to predict TG expression. Putative TFs are selected as tree nodes if they consistently reduce the variance of the TG.	Huynh-Thu *et al.* (2010)
2	TF-TG co-dependencies are detected by the non-linear correlation coefficient η^2 (two-way ANOVA). TF KO/OE data receive increased weights.	Küffner *et al.* (2011)
3	TFs are selected maximizing the conditional entropy for a given TG. TGs are represented as Boolean vectors with probabilities, avoiding discretization.	Karlebach and Shamir (2011)
4	Putative TFs are preselected from TF KO/OE data or by Pearson's correlation. TFs are then tested by iterative Bayesian Model Averaging (BMA).	Yeung *et al.* (2005)
5	A Gaussian noise model was used to estimate if the expression of a TG changes significantly in TF KO/OE measurements.	Yip *et al.* (2010)
6	After scaling, TGs are clustered by Pearson's correlation. A neural network was trained (genetic algorithm) and parametrized (back-propagation).	Sîrbu *et al.* (2011)
7	Data were discretized by Gaussian mixture models and clustering (Ckmeans); Detects interactions by generalized logical network model (χ^2 test).	Song *et al.* (2009)
8	The χ^2 test was applied to evaluate the probability of a simultaneous shift in TF and TG expression in TF KO/OE experiments.	Song *et al.* (2009)
	META PREDICTORS combine (1) several approaches by calculating (2) aggregate scores.	
1	(1) Calculates z-scores for TG in TF KO data, applies time-lagged CLR for time series, and linear ODE models constrained by Lasso; (2) resampling.	Greenfield *et al.* (2010)
2	(1) Pearson's correlation, mutual information, and CLR; (2) rank average.	Qiu *et al.* (2009)
3	(1) Calculates TG responses in TF KO data, applies full-order partial correlation and TF-TG co-deviation analysis; (2) weighted average with weights trained on simulated data	Pinna *et al.* (2011a)
4	(1) CLR filtered by negative Pearson's correlation, least angle regression (LARS) of time series, and TF KO/OE data; (2) combination by z-scores.	Watkinson *et al.* (2009)
5	(1) Pearson's correlation, differential expression (limma, Gauss tail), and time series analysis (maSigPro); (2) Naive Bayes.	Conesa *et al.* (2006)

Within each category, methods are ordered according to their overall performance. †Off-the-shelf algorithm applied by challenge organizers. TF: transcription factor, TG: target gene, TF KO/OE data: measurements where TFs have been deleted or overexpressed. Table modified from Marbach *et al.* (2012).

Within each category, methods are ranked according to their overall performance. Note that our GENIE3 method is the first method in the *Other* category.

7.2.2 Performance of the network inference methods

As described in Section 6.3 of Chapter 6, the predictions of a network returned by a method were assessed using precision-recall (PR) and receiver operating characteristic (ROC) curves, and the performance across several networks was summarized with an overall score defined in Equation (6.6). Figure 7.2(a) shows the overall score of each method, as well as their performance on each individual network (using the area under the PR curve, or AUPR).

Most of the methods have a performance that greatly varies across the different networks. For example the *Regression 2* method has the highest AUPR on the *in silico* network but is among the worst performers on the *E. coli* network. On the other hand, the *Meta 4* method yields better predictions than most methods for the *E. coli* network, but performs less good on the artificial one. Two methods have clearly the best overall performances. These methods are GENIE3 (*Other 1*) and an algorithm based on two-way ANOVA (*Other 2*) (Küffner *et al.*, 2011). Interestingly, the six off-the-shelf methods that are commonly used are outperformed by several other methods. A great variation can also be observed among the performances of the methods that are in a same category, leading to the conclusion that the good performance of a method is more due to the details of its implementation and parameter tuning rather than the general methodology. However, all the methods of the *Bayesian networks* and *Correlation* categories tend to have the lowest overall scores among all the methods.

The inference methods of all categories yield much better predictions for the *in silico* and *E. coli* networks than for the *S. cerevisiae* network, for which the different AUPRs that are obtained are comparable to the AUPR of random predictions. At least two reasons can explain the poor results obtained on this network. First, the *S. cerevisiae* gold standard was constructed based mostly on ChIP experiments. These experiments check the physical binding of a TF to the promoter region of a target gene, but can not state if this binding has an effect on the regulation. The *S. cerevisiae* gold standard thus contains many false positives. The second reason is that *S. cerevisiae* is an eukaryote organism, in which the genetic regulation is far more complex than in prokaryote organisms such as *E. coli*. For example, regulation may happen at the post-transcriptional level, which results in a decreased correlation between the mRNA expression levels of a TF and those of its target genes, and

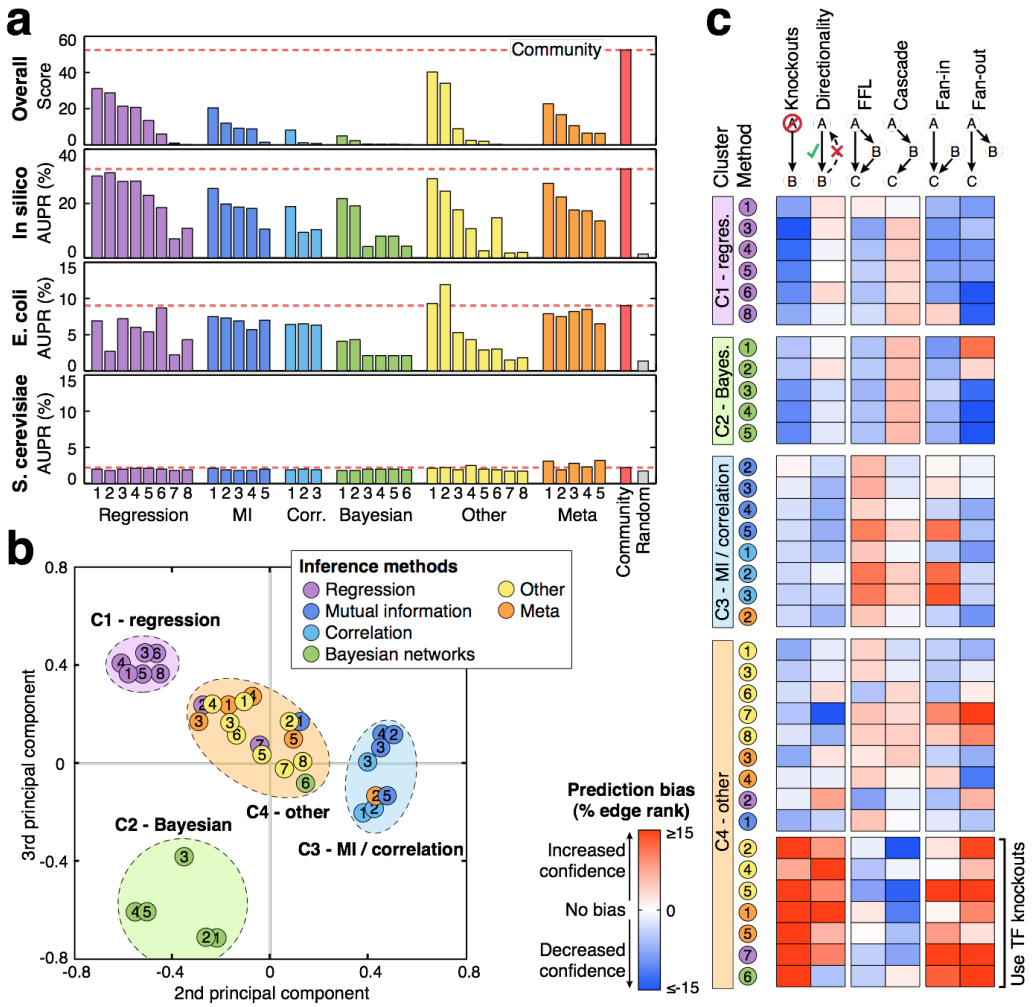


Figure 7.2: **Evaluation of the network inference methods. a.** This figure shows the overall performance of the 35 inference methods across networks (overall score) as well as their performance on each individual network (AUPR). The methods are categorized and numbered according to Tables 7.1 and 7.2. The rightmost bar shows the performance of random predictions. The red bar shows the performance of the integrated community network. GENIE3 is the *Other 1* method. **b.** Clustering of the methods using principal component analysis. **c.** Biases of the methods for predicting different types of interactions. A motif colored in red (resp. blue) indicates that the corresponding method tends to predict more (resp. less) reliably the interactions that are part of that motif. Figures taken from Marbach *et al.* (2012).

therefore a decreased performance of the methods that only exploit mRNA expression data to infer regulatory networks (Hu *et al.*, 2007; Michoel *et al.*, 2009).

7.2.3 Clustering and motif analysis

The predictions returned by the different inference methods were analyzed by principal component analysis (PCA). For this purpose, the authors constructed a *prediction matrix*, in which each row corresponds to an inference method and each column corresponds to an interaction[1], and the element (i, j) of the matrix indicates the rank of the jth interaction according the the ith method. The dimensionality of this matrix was then reduced by PCA. The second and third principal components[2] show four clusters of methods, corresponding largely to the categories of Tables 7.1 and 7.2 (see Figure 7.2(b)). Therefore, even if the methods of a same category show different performances, they tend to predict similar interactions.

The challenge organizers also performed a motif analysis, whose goal is to evaluate if the methods tend to systematically predict less or more reliably specific types of interactions. They considered six interaction types, each being part of a motif type and illustrated in Figure 7.2(c). For each inference method, they computed the average rank r_m of all the edges of the gold standard that are part of a given motif m, as well as the average rank $r_{\bar{m}}$ of the edges that are not part of the motif. The prediction bias is then given by $r_m - r_{\bar{m}}$. A positive (resp. negative) bias indicates that the method tends to rank higher (resp. lower) the interactions that are part of the motif m. We can see that GENIE3 shows the same biases as the MI- and correlation-based methods, although these biases are relatively weak compared to those of some other methods. Our method predicts less reliably the cascade motifs, i.e. it tends to rank high the (false) indirect links, but it yields more confident predictions for the edges that are part of a feed-forward loop. As on the networks of the DREAM4 *Multifactorial* challenge (see Section 6.4.1), GENIE3 has more difficulties to recover the regulators of the genes having a high in-degree, as well as the regulators having a high out-degree.

7.2.4 Information content of different experiment types

For all datasets, a description of each microarray experiment was provided to the challenge participants, together with the expression data. Experiments

[1] The prediction matrix comprises the interactions related to the four datasets.

[2] The first principal component actually correlates with the mean AUPR and is thus less characteristic of the intrinsic properties of the methods.

comprised time series, drug perturbations, gene knockout and overexpression experiments, as well as combinations of them. To see how informative each type of experiment is for the inference of a network, the DREAM organizers used the following supervised learning-based procedure, inspired from Mordelet and Vert (2008). For each TF, they generated the following dataset:

$$LS_{TF} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, p\}, \tag{7.1}$$

where p is the number of genes (not including the TF) in the original compendium, $\mathbf{x}_i \in \mathbb{R}^N$ is the vector containing the expression levels of the ith gene in all N experiments, and y_i is a binary variable stating if the ith gene is a target or not of the TF. The targets and non-targets of the TF were defined by the gold standard network. A predictive model, classifying the targets and the non-targets of the TF, was then learned from LS_{TF}, using either a decision tree or a linear SVM, and the importance, or weight, of each microarray experiment in the predictive model was computed. A high weight thus indicates that the corresponding experiment is highly informative for the prediction of the targets of the TF. Individual weights were then averaged across the experiments of a particular type and across the TFs. These average weights are shown in Figure 7.3, in which we clearly see that the knockout and overexpression of a TF are highly informative for the inference of the regulatory links involving that TF. Note that the *S. cerevisiae* compendium comprised only three knockout experiments, all for the same TF. Results are therefore not reliable for this organism.

Actually, several network inference methods used for the challenge explicitly exploit the information about the manipulation of a TF to identify its target genes. Compared to the methods that consider knockout/overexpression experiments at the same level as the other types of experiments, these methods obtain more confident predictions for the interactions involving the manipulated TFs, and predict more accurately the direction of the edges connecting two transcription factors (see Figure 7.2(c)). Note that they are also able to predict more reliably the edges such that the regulator has a high out-degree or such that the target gene has a high in-degree, but this is partly due to the fact that the knockout experiments were preferentially performed for the TFs that regulate a high number of genes.

7.2.5 Community networks

The authors also presented a procedure to construct *community networks*, that combines the predictions of all the participating teams. The procedure simply consists in ranking the regulatory links according to their average rank

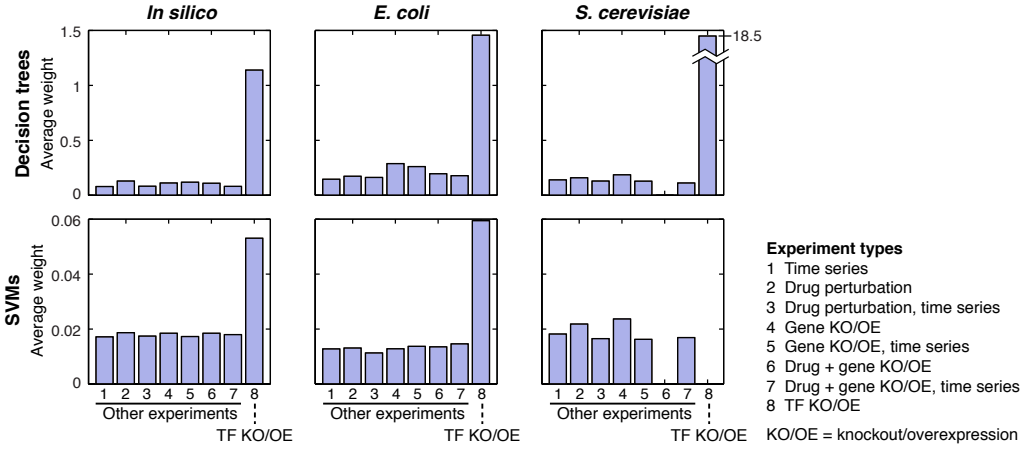


Figure 7.3: **Information content of the different experiment types.** A feature selection method was used to measure the weight of each type of experiment in the prediction of the interactions. The experiments were divided into 8 types: time series, drug perturbation, gene knockout or overexpression (KO/OE), and combinations thereof. The 8th bar in each plot corresponds to the importance of TF KO/OE experiments for predicting the targets of that TF. Figure taken from Marbach *et al.* (2012).

across all the methods. The rank r_I of an interaction I in the community prediction is thus given by:

$$r_I = \sum_{k=1}^{K} r_I^k, \tag{7.2}$$

where K is the number of inference methods ($K = 29$ [3]) and r_I^k is the rank of interaction I according to the kth method.

As shown in Figure 7.2(a), the community-based procedure has a robust performance across the three networks, having an overall score greater than all 35 individual methods. It obtains the highest AUPR score for the *in silico* network and ranks respectively at the third and 6th positions for the *E. coli* and *S. cerevisiae* networks. Therefore, the authors argued that, as none of the individual network inference methods is the best performer for each of the three networks, it is difficult to choose the most appropriate method to infer an unknown network and one should rather resort to community-based procedures. Note however that the two best performing individual procedures, i.e. GENIE3 (*Other 1*) and the ANOVA-based procedure (*Other 2*), have also a robust performance across the networks and remain competitive

[3]The 6 off-the-shelf methods were not considered to build the community predictions.

with the community-based procedure.

Figure 7.4(a) and (b) show respectively the *E. coli* and *S. aureus* community networks. Both networks were obtained by using a threshold on the corresponding rankings, in order to obtain networks with 1688 regulatory links. This number of edges was chosen because it corresponds to a precision of 50% for *E. coli*. Both networks have a modular structure. A module is a group of TFs and genes that are more densely connected among each other than to other genes that are not part of the group. The authors identified the different modules for each network and checked if they were significantly enriched for Gene Ontology (GO) terms (The Gene Ontology Consortium, 2000). They found that modules were strongly enriched for specific biological processes, with only few processes being enriched in more than one module. For example, a module of the *S. aureus* network is enriched for GO terms related to pathogenesis.

Novel regulatory links can be found in the *E. coli* community network, i.e. interactions that are not curated in the RegulonDB database. The authors selected 53 of these interactions, involving five different TFs (see Figure 7.4(c)). These interactions were experimentally tested and 23 of them were determined as true positives, corresponding to a precision of $\sim$ 40%. Note that this precision varies strongly across the different TFs. For example, all the selected novel target genes of rhaR were experimentally determined as false positives, suggesting that the precision of the interactions predicted by a network inference method can vary strongly across individual TFs.

7.3 Discussion

Marbach *et al.* (2012) provided a unique assessment of 35 network inference methods, that cover a wide range of different, existing and novel, approaches. The methods were evaluated in the context of the DREAM5 *Network Inference* challenge, whose goal was to recover an *in silico* network, as well as the *E. coli* and *S. cerevisiae* regulatory networks.

The evaluated methods typically yield much better predictions for the *in silico* and *E. coli* networks than for the *S. cerevisiae* network. The poor results on the *S. cerevisiae* network can be partly explained by the rather unreliable gold standard that was used to assess the performances of the methods, as well as by the higher complexity of the genetic regulation mechanisms of the eukaryotes. To infer the regulatory network of such an organism, one should therefore use other types of data besides expression data, such as DNA sequences of the promoters of the genes. In Chapter 8, we show that the performance of our GENIE3 procedure can be improved by exploiting

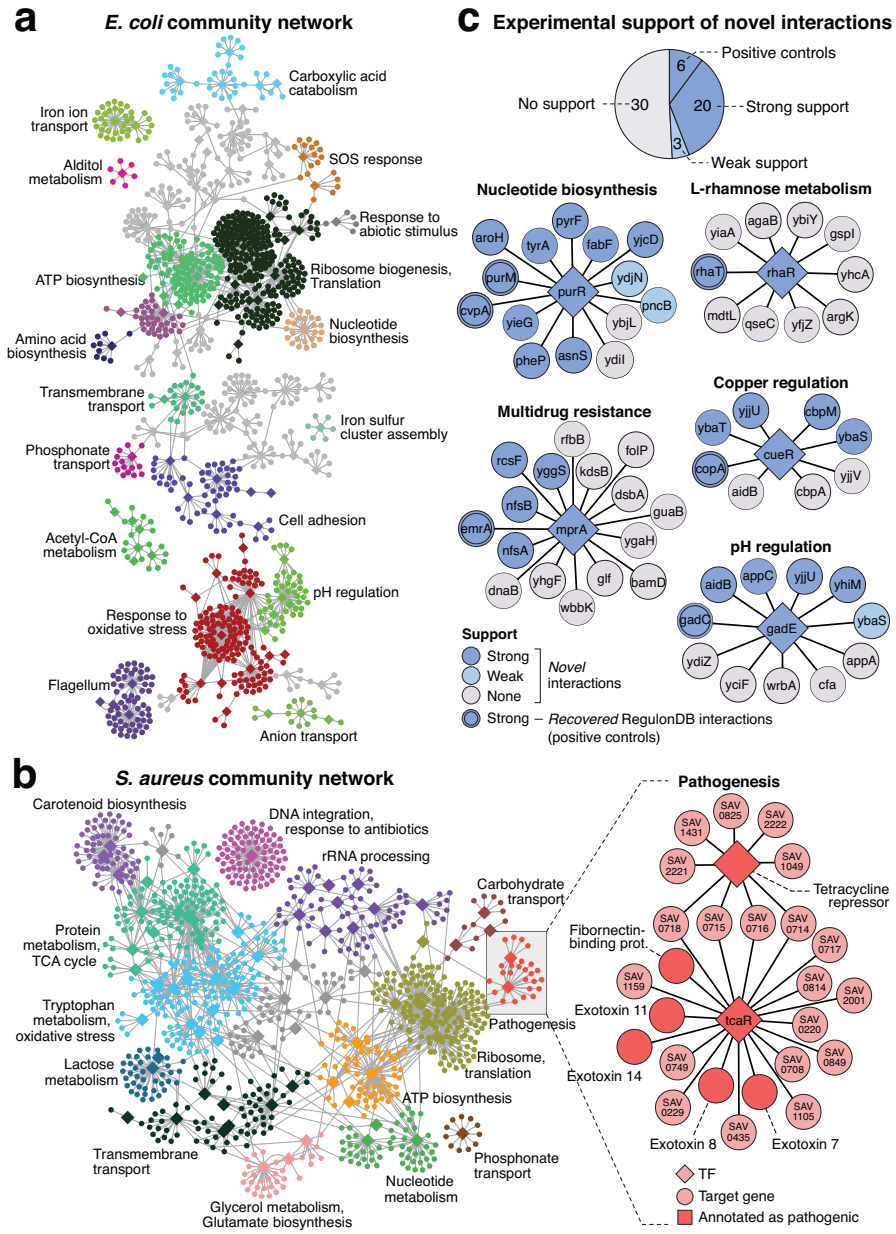


Figure 7.4: **Community networks.** **a.** *E. coli* community network. **b.** *S. aureus* community network. Both networks were obtained using a cutoff of 1688 edges on the rankings of regulatory links obtained with the community-based procedure. This number of edges was chosen because it corresponds to a precision of 50% for *E. coli*. Colored modules are enriched for particular GO terms. **c.** 53 novel *E. coli* interactions were experimentally tested, among which 23 were determined as true positives. Figure taken from Marbach *et al.* (2012).

information about genetic markers in addition to expression data.

Most of the evaluated inference methods do not have a robust performance across the different networks of the DREAM5 challenge. Some methods that perform better than others on the *in silico* network return less good predictions for *E. coli*, and vice versa. The authors therefore suggested to use a community-based procedure to infer unknown networks, rather than to choose an individual method that would have an uncertain performance. The presented procedure consists in ranking the regulatory links corresponding to a network, according to their average rank across the different methods of the DREAM community. They showed that the community-based predictions are quite robust across the networks of the DREAM5 challenge and yield an overall score that is higher than all individual methods. Several regulatory links of the *E. coli* community network were experimentally verified and the *S. aureus* community network is the first comprehensive regulatory network available for this pathogen.

An interesting analysis was also performed to evaluate the importance of the different types of experimental conditions for the inference of the networks. The analysis showed that the knockout and the overexpression of a transcription factor are particularly informative for the retrieval of the regulatory links involving this transcription factor. As future work, we therefore would like to extend our GENIE3 procedure such that it can explicitly take into account the information brought by the knockout or overexpression of a gene (some experiments in this direction are performed in Section 8.1.4 of Chapter 8). For example, this information could be used in a pre-processing step, in order to measure the effect of the manipulation of a TF on the expression of each gene. Then, when learning the tree-based model predicting the expression of a particular target gene, we would remove from the input variables the TFs whose manipulation does not affect the expression of that target gene.

8

Extensions of GENIE3

In Chapter 6, we introduced GENIE3, an algorithm that exploits static steady-state expression data to infer gene regulatory networks. In the present chapter, we describe extensions of the GENIE3 procedure that aim to recover regulatory networks from time series data and from genetical genomics data. We show that using another type of data in addition to steady-state expression data can improve significantly the network predictions. In particular, time series data and genotyping data allow to predict more accurately the direction of the edges of the targeted networks. Section 8.1 describes two procedures that integrate steady-state data and time series data. One of these two procedures is competitive with other algorithms on the DREAM3 and DREAM4 network inference challenges. In Section 8.2, we propose several methods to infer networks from genetical genomics data and show that one of these methods outperforms the official best performing algorithm of the DREAM5 *Systems Genetics* challenge. Finally, Section 8.3 concludes the chapter.

Contents

8.1 Time series of gene expressions

Expression data can be divided in two main categories: steady-state data and time series data. The first type of data is obtained by applying a perturbation to the system under study and measuring the gene expression levels once the system has reached a steady state. By contrast, the second type of data is obtained by measuring the gene expression levels at several time points following the perturbation. Because of their high cost, time series data are less common than static steady-state data, but they provide a more complete picture of the system under study (Bar-Joseph, 2004). In particular, they contain much more information about causal effects (De Smet and Marchal, 2010). Therefore many methods exploiting time series expression data have been developed in order to infer gene regulatory networks, e.g. methods based on time-lagged correlation analysis (Schmitt Jr *et al.*, 2004), dynamic Bayesian networks (Yu *et al.*, 2004), and ordinary differential equations (Bonneau *et al.*, 2006).

In this section, we present a modified version of the GENIE3 procedure that exploits time series data to infer regulatory networks. Compared to the original procedure, a tree-based regression model is still learned to predict the expression of each target gene j from the expression levels of the candidate regulatory genes, but the expression of the target gene in the corresponding learning sample LS^j is shifted by h time points, where h denotes a given time horizon, with respect to the expression values of the input genes. Given the expression levels of the input genes at some time point t, the learned tree-based model thus predicts the expression of the target gene at time point $t + h$. We also present two extensions of this procedure that allow to combine steady-state data and time series data.

8.1.1 Inference from time series data

In what follows, we suppose that we have at our disposal expression data provided by a time series experiment, i.e. an ensemble of gene expression levels measured at T time points following a perturbation of the system:

$$LS_T = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T\}, \tag{8.1}$$

where $\mathbf{z}_t \in \mathbb{R}^p, t = 1, \ldots, T$ is a vector containing the expression values of p genes at the tth time point:

$$\mathbf{z}_t = (z_t^1, z_t^2, \ldots, z_t^p)^\top. \tag{8.2}$$

We further assume that the time intervals between two measurements are equal.

To infer a regulatory network from such data, we propose an extension of our GENIE3 procedure, called **GENIE3-time**. In this procedure, we make the assumption that the expression of each gene of the network at a time point $t + h$, where $h > 0$ denotes a given time horizon, is a function of the expression of the other genes of the network at time point t. Denoting by $\mathbf{z}_t^{-j}$ the vector containing the expression values at time point t of all the genes except gene j, we thus assume that we can write:

$$z_{t+h}^j = g_j^h(\mathbf{z}_t^{-j}) + \epsilon_t, \forall t, \tag{8.3}$$

where ϵ_t is a random noise. As in the original GENIE3 procedure, we further assume that g_j^h only exploits the expression in $\mathbf{z}^{-j}$ of the genes that directly regulate gene j in the underlying network. To learn the function g_j^h, we apply a tree-based ensemble method to the following learning sample:

$$LS_{T,h}^j = \{(\mathbf{z}_t^{-j}, z_{t+h}^j), t = 1, \ldots, T - h\}. \tag{8.4}$$

In this learning sample, the vector of expression values of the target gene j is thus shifted by h time points with respect to the vectors of the input genes. Tree-based importance scores $w_{i,j}^h (\forall i \neq j)$, in the form of sums of variance reductions (2.12), are then computed from the model g_j^h for all input genes.

If M time course experiments are available, i.e. we have at our disposal M datasets LS_T obtained from M different perturbations of the network, we follow the same procedure by learning g_j^h from the concatenation of the learning samples:

$$LS_{T,h}^j = \{LS_{T,h,1}^j, LS_{T,h,2}^j, \ldots, LS_{T,h,M}^j\}, \tag{8.5}$$

where $LS_{T,h,k}^j$ is the learning sample in the form (8.4), generated from the kth time course experiment. Note that we assume that the time intervals separating two expression measurements are the same in the M experiments.

As in the original GENIE3 procedure, the output vector $\mathbf{z}_{T,h}^j$ of each learning sample $LS_{T,h}^j$ is normalized so that the expression of the target gene j has a unit variance in the learning sample, in order to avoid a positive bias for the regulatory links towards the more highly variable genes:

$$\mathbf{z}_{T,h}^j \leftarrow \frac{\mathbf{z}_{T,h}^j}{\sigma_{T,h}^j}, \tag{8.6}$$

where $\sigma_{T,h}^j$ is the standard deviation of $\mathbf{z}_{T,h}^j$.

As it will be shown in Section 8.1.4, quite different predictions $w_{i,j}^h$ can be obtained depending on the value of the time horizon h. Choosing an

appropriate value for h can be a difficult task and to avoid to make this choice, we propose to learn each weight $w_{i,j}^h$ with all possible values of h (i.e. $h = 1, \ldots, T-1$), and use the average of the different weights as final prediction. The weight of the edge directed from gene i to gene j in the predicted network is thus given by:

$$w_{i,j} = \frac{1}{T-1} \sum_{h=1}^{T-1} w_{i,j}^h. \tag{8.7}$$

8.1.2 Inference from time series and steady-state data

Now we assume that, besides a time series expression dataset LS_T, we also have at our disposal the expression levels of the genes in N steady-state conditions:

$$LS_S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \tag{8.8}$$

where $\mathbf{x}_k \in \mathbb{R}^p, k = 1, \ldots, N$ is a vector containing the expression values at steady-state of the p genes in the kth experiment:

$$\mathbf{x}_k = (x_k^1, x_k^2, \ldots, x_k^p)^\top. \tag{8.9}$$

We propose two procedures to infer a gene regulatory network that exploit the two types of data. In the first procedure, called **GENIE3-mean**, two separate models are respectively learned from steady-state data and from time series data, while in the second procedure, called **GENIE3-comb**, a unique model is learned from both datasets.

In the GENIE3-mean procedure, we assume that the observed steady-state and time series expression levels of each gene $j = 1, \ldots, p$ are explained by two respective models f_j and g_j^h. As in the original GENIE3 method, we assume that the steady-state expression of gene j in a condition depends in some manner on the steady-state expression levels of the other genes in the same condition:

$$x_k^j = f_j(\mathbf{x}_k^{-j}) + \epsilon_k, \forall k, \tag{8.10}$$

while the expression of this gene at time point $t+h$ following a perturbation of the network depends in another manner on the expression levels of the other genes at time point t:

$$z_{t+h}^j = g_j^h(\mathbf{z}_t^{-j}) + \epsilon_t, \forall t. \tag{8.11}$$

Two different rankings of regulatory links are thus respectively learned from steady-state and time series data. The first ranking is obtained by applying to LS_S the original GENIE3 approach while the second ranking is obtained

by applying GENIE3-time to LS_T. The two rankings are then aggregated by a weighted averaging of the different pairs of scores corresponding to the same regulatory link. The edge directed from gene i to gene j will thus have the following final score:

$$w_{i,j} = \frac{\mu_S.w^S_{i,j} + \mu_T.w^T_{i,j}}{\mu_S + \mu_T}, \tag{8.12}$$

where $w^S_{i,j}$ is the importance of gene i in f_j, $w^T_{i,j}$ is its importance in g^h_j (averaged over all possible values of h), and μ_S and μ_T are the weights given to steady-state data and time series data respectively.

The second proposed procedure, GENIE3-comb, assumes that both steady-state data and time series data can be explained by a unique model f^h_j:

$$\begin{cases} x^j_k = f^h_j(\mathbf{x}^{-j}_k) + \epsilon_k, \forall k, \\ z^j_{t+h} = f^h_j(\mathbf{z}^{-j}_t) + \epsilon_t, \forall t. \end{cases} \tag{8.13}$$

This amounts to considering steady-state data as time series expression measurements that do not change over time. The function f^h_j is learned by applying a tree-based ensemble method to the learning sample obtained by concatenating the two types of data:

$$LS^j_{C,h} = \left\{\{(\mathbf{x}^{-j}_k, x^j_k), k = 1, \ldots, N\}, \{(\mathbf{z}^{-j}_t, z^j_{t+h}), t = 1, \ldots, T-h\}\right\}, \tag{8.14}$$

in which we respectively normalize the output vectors $\mathbf{x}^j$ and $\mathbf{z}^j_{T,h}$ of the steady-state and time series datasets in the following way:

$$\mathbf{x}^j \leftarrow \frac{\mathbf{x}^j - \bar{\mathbf{x}}^j}{\sigma^j}, \quad \mathbf{z}^j_{T,h} \leftarrow \frac{\mathbf{z}^j_{T,h} - \bar{\mathbf{z}}^j_{T,h}}{\sigma^j_{T,h}}, \tag{8.15}$$

where $\bar{\mathbf{x}}^j$ and $\bar{\mathbf{z}}^j_{T,h}$ are respectively the means of $\mathbf{x}^j$ and $\mathbf{z}^j_{T,h}$, and σ^j and $\sigma^j_{T,h}$ denote their respective standard deviations. This normalization ensures that the expression values of gene j coming from the two datasets are comparable and that the output vector of $LS^j_{C,h}$ has a unit variance.

8.1.3 The DREAM challenges

We evaluated the proposed procedures on the synthetic datasets of the DREAM3 and DREAM4 network inference challenges, which are both described below.

DREAM3 *In Silico Network* challenge

The goal of the DREAM3 *In Silico Network* challenge[1] was to recover artificial networks from simulated steady-state and time series data. There were three sub-challenges called *In Silico Size 10*, *In Silico Size 50*, and *In Silico Size 100*. The goal of each of the three sub-challenges was to recover five networks of 10, 50, and 100 genes respectively. In each case, the provided steady-state data contained the expression levels of the genes for the wild-type (unperturbed network) and the systematic knockout and knockdown of each gene. The knockout of a gene was simulated by setting its transcription rate to zero, while its knockdown was simulated by reducing its transcription rate by half. To generate the time series data, 4, 23, and 46 perturbations experiments were simulated for the networks of size 10, 50, and 100 respectively. Each experiment consisted in applying a multifactorial perturbation to the network, simulated by changing the initial expression levels of all the genes. Gene expression levels were provided at 21 time points (with equal time intervals) following each perturbation. As an illustration, Figure 8.1 (top) shows the expression profiles of the genes of a network of size 10 in a time course experiment. All data were generated using GeneNetWeaver (version 1.0) (Marbach *et al.*, 2009; Schaffter *et al.*, 2011).

DREAM4 *In Silico Network* challenge

The DREAM4 *In Silico Network* challenge contained two sub-challenges involving time series data, called *In Silico Size 10* and *In Silico Size 100*[2]. The goal of each of the two sub-challenges was to recover five artificial networks of 10 and 100 genes respectively. In each case, the provided steady-state data contained the expression levels of the genes for the wild-type and the systematic knockout and knockdown of each gene (simulated in the same way as for the DREAM3 challenge). For the *In Silico Size 10* sub-challenge, additional steady-state data were available in the form of expression levels obtained after the application of multifactorial perturbations. To generate the time series data, 5 and 10 perturbation experiments were simulated for the networks of size 10 and 100 respectively. Contrary to the experiments of DREAM3, each time series experiment of DREAM4 consisted in strongly increasing or decreasing the initial expression of about one third of the genes, thereby simulating a physical or chemical perturbation rather than a multifactorial perturbation. Gene expression data were provided for 21 time points (with equal time intervals) in each case. Figure 8.1 (bottom) shows

[1] http://wiki.c2b2.columbia.edu/dream/index.php/D3c4
[2] http://wiki.c2b2.columbia.edu/dream/index.php/D4c2

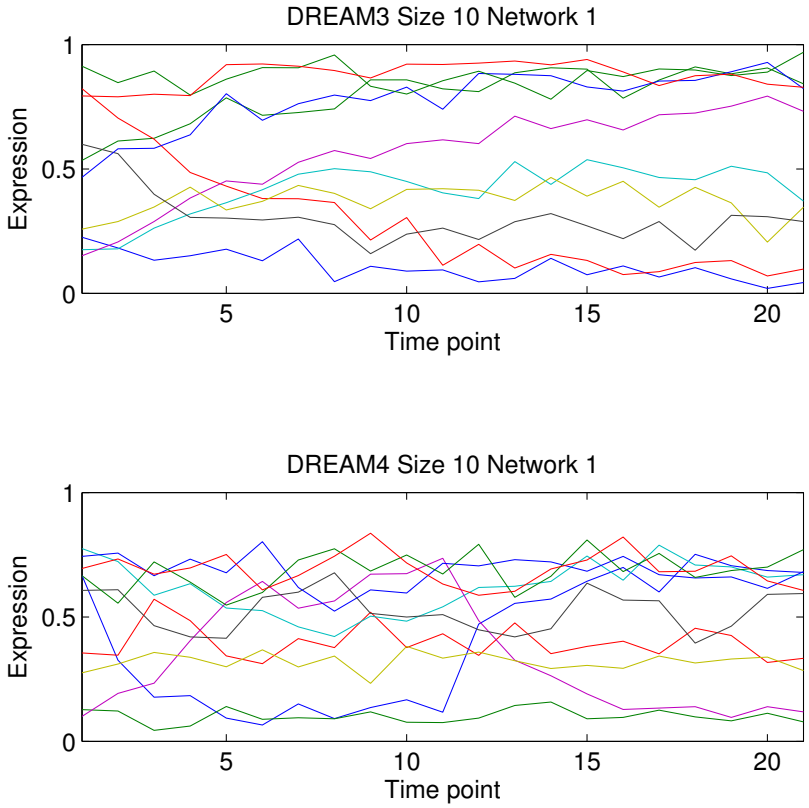


Figure 8.1: **Gene expression profiles.** Each panel shows the expression levels of the genes of a network, at each time point of a time course experiment. Each curve corresponds to a gene. Top: First network of size 10 of the DREAM3 challenge. Bottom: First network of size 10 of the DREAM4 challenge.

the expression profiles of the genes of a network of size 10 in a time course experiment. All data were generated using GeneNetWeaver (version 2.0).

Note that each simulated time course experiment consisted in applying a perturbation to the network at time $t = 0$ and removing this perturbation after 10 time points, making the system return to its original state. In our different experiments with GENIE3, we considered the application and the removal of the perturbation as two separate sub-experiments. We therefore divided the data to obtain 10 (resp. 20) time course experiments for the networks of size 10 (resp. 100), each one containing 11 time points (the 11th time point of each original experiment being the last point of the first sub-experiment and also the first point of the second sub-experiment).

8.1.4 Results

Performance of the methods

For each sub-challenge of DREAM3 and DREAM4, we applied the following procedures:

- GENIE3, applied to the dataset obtained by merely concatenating steady-state and time series data. We thus consider each time point of a time course experiment a separate static condition;
- GENIE3, applied to the steady-state data;
- GENIE3-time, applied to the time series data;
- GENIE3-mean, exploiting both types of data;
- GENIE3-comb, exploiting both types of data.

All the predictions were obtained using the Random Forests method with its main parameter K fixed to $p-1$, where p is the number of genes, and growing $T = 1000$ trees. For the GENIE3-mean method, we used as default values the following values for the weights μ_S and μ_T:

$$\mu_S = \frac{N_S}{N_S + N_T}, \quad \mu_T = \frac{N_T}{N_S + N_T}, \tag{8.16}$$

where N_S and N_T are the sizes of the steady-state and time series datasets respectively. These values give more weight to the larger dataset.

Figure 8.2 compares, in terms of AUPR scores, the performances of the different procedures. The corresponding PR curves can be found in Figures C.1 and C.2. In each sub-challenge, the highest AUPR scores are obtained with GENIE3-mean and GENIE3-comb. As discussed by Yip *et al.* (2010), steady-state data and time series data probably contain different and complementary information about the underlying networks, giving an advantage to the procedures that integrate both types of data. However, applying the original GENIE3 procedure to the concatenation of both datasets, such that each time point of a time course experiment is simply considered a separate static condition, results in relatively poor predictions. In some cases, these predictions are even worse than those obtained when only steady-state data are used, stressing the importance of taking into account a time horizon $h > 0$ when exploiting time series data. GENIE3-comb obtains a higher score than GENIE3-mean in each sub-challenge. Learning a single model from steady-state and time series data, rather than two separate models,

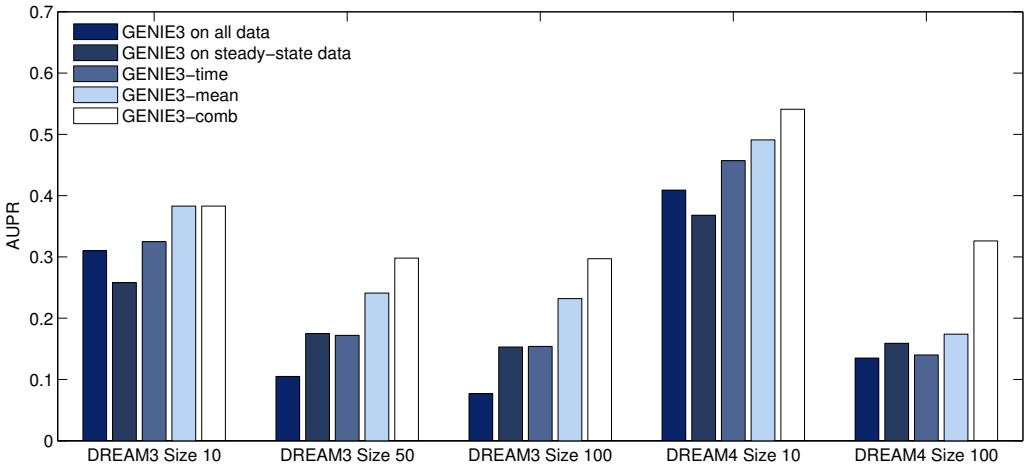


Figure 8.2: **AUPR scores for the DREAM3 and DREAM4 networks.** For each sub-challenge, the AUPR values are those obtained when applying respectively GENIE3 to the concatenation of steady-state and time series data, GENIE3 to the steady-state data, GENIE3-time, GENIE3-mean (with μ_S and μ_T fixed to their default values), and GENIE3-comb. The AUPR values of each method were averaged over the five networks of each sub-challenge.

thus seems to capture in a more efficient way the complementary information that they contain, and therefore results in a significant improvement of the network predictions.

Influence of the weights of GENIE3-mean

Figure 8.3 shows the AUPR scores obtained by GENIE3-mean, for different values of the weights μ_S and μ_T, which correspond respectively to the steady-state data and the time series data. The performance of the method typically does not change much when varying the values of the weights. We can nevertheless observe slightly better performances for larger values of μ_T. This result can be explained by the fact that in all the sub-challenges except the DREAM4 *In Silico Size 100* sub-challenge, the time series datasets contain much more samples than the steady-state datasets. Our default values of μ_S and μ_T seem to be a good choice, as they correspond to the (nearly) optimal values. However, regardless of the values that are chosen for the weights, GENIE3-comb typically yields better performances.

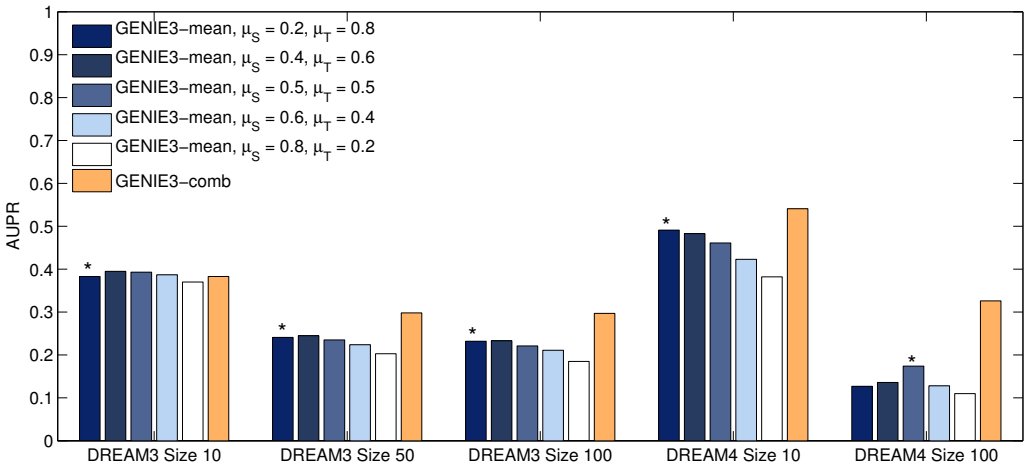


Figure 8.3: **AUPR scores of GENIE3-mean and GENIE3-comb.** The AUPRs of GENIE3-mean are shown for different values of the weights μ_S and μ_T, which correspond respectively to the steady-state and the time series data. For each sub-challenge, an asterisk indicates the bar corresponding to the default values. The AUPR values of each method were averaged over the five networks of each sub-challenge.

Influence of the time horizon

In our different procedures, we propose to compute the weights $w_{i,j}^h$ of the regulatory links with all possible values of the time horizon h, and use their average values as final predictions. This allows to avoid to choose a value for this parameter, which can be a difficult task. Figure 8.4 shows, for five networks, the AUPR values obtained with GENIE3-comb, for different values of h, as well as the AUPR resulting from the average weights (horizontal dashed line). Similar plots for the other networks are shown in Figures C.3 and C.4. We can see that quite different results can be obtained depending on the value of the time horizon and that averaging the weights over all possible values of h results in performances that are comparable and sometimes even better than those obtained by the best performing values of h. Note that using a higher value for h tends to result in a lower AUPR because the number of samples in the resulting learning sample is lower and most of the information contained in the time series dataset is hence not used.

Direction of the edges

It is commonly admitted that, compared to steady-state data, time series data are more informative for the prediction of the direction of the edges

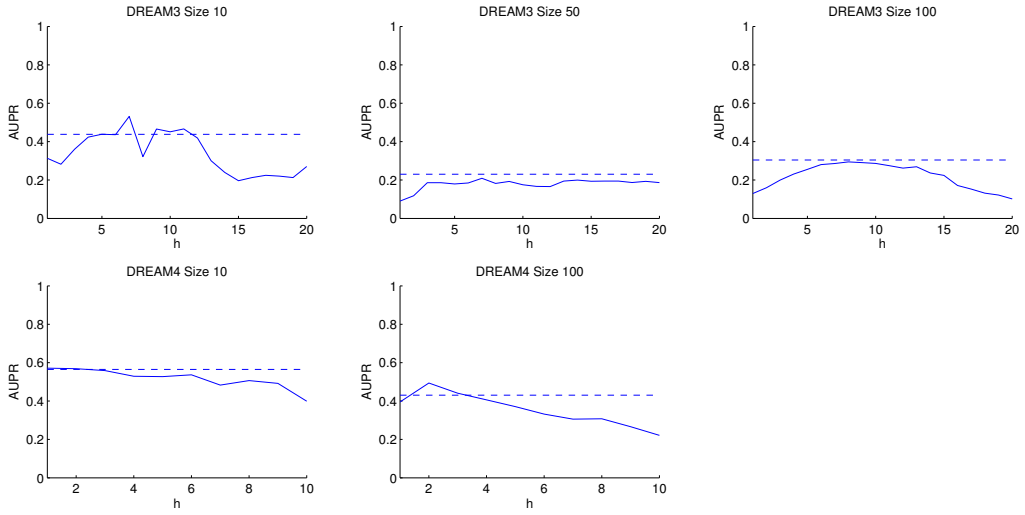


Figure 8.4: **AUPR scores of GENIE3-comb.** Each plot is related to one of the five networks of one sub-challenge and shows the AUPRs obtained with GENIE3-comb, for different values of the time horizon h, as well as the AUPR obtained when the weights of the regulatory links are averaged over all possible values of h (horizontal dashed line).

of the targeted network. We indeed observe this result when inferring the different networks of the DREAM3 and DREAM4 challenges. Figure 8.5 indicates the average error rate on the edge directionality of each GENIE3 method, computed as the proportion of edges $i \rightarrow j$ in the gold standard network such that there is no edge $j \rightarrow i$ and for which the method wrongly predicts $w_{i,j} < w_{j,i}$. We can see that the highest error rate is typically obtained when time series data are not exploited, i.e. when GENIE3 is applied to steady-state data only, and that using time series data allows to predict more accurately the direction of the edges. The lowest error rates on edge directionality are typically obtained with GENIE3-comb, which yields an average error rate equal to 29%.

Comparison with the best performers

Table 8.1 compares, for each sub-challenge, the mean AUPR score of GENIE3-comb with the one of the best performer in this sub-challenge. The table also shows the rank of GENIE3-comb among the official participating teams (according to the overall score). GENIE3-comb is ranked among the top-performing algorithms in all the sub-challenges. However, the AUPR scores indicate that our method remains significantly outperformed by the differ-

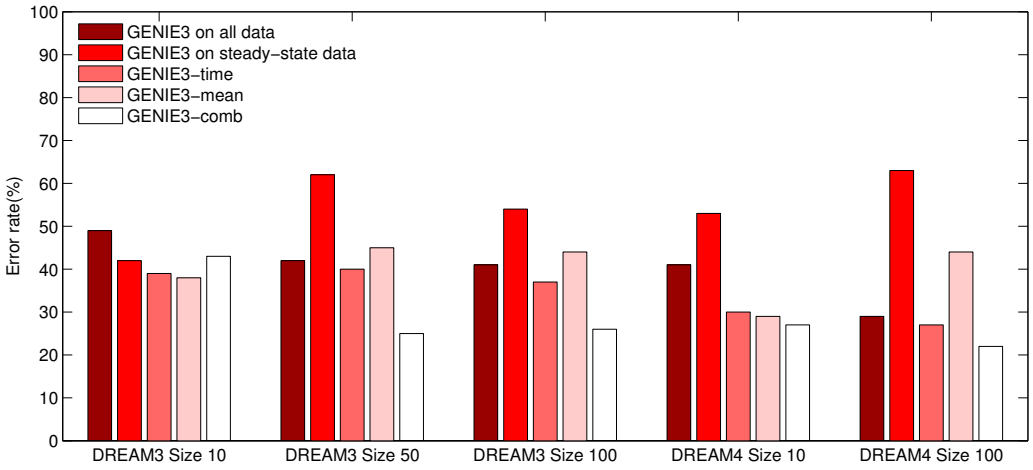


Figure 8.5: **Error rates on edge directionality on the DREAM3 and DREAM4 networks.** For each sub-challenge, the error rates are those obtained when applying respectively GENIE3 to the concatenation of steady-state and time series data, GENIE3 to the steady-state data, GENIE3-time, GENIE3-mean (with μ_S and μ_T fixed to their default values), and GENIE3-comb. The error rate is the proportion of edges $i \to j$ in the gold standard network such that there is no edge $j \to i$ and for which the method wrongly predicts $w_{i,j} < w_{j,i}$. The error rates of each method were averaged over the five networks of each sub-challenge.

ent best performing algorithms. These algorithms include a procedure that learns noise models and differential equation models from knockout data and time series data respectively (Yip *et al.*, 2010, best performing team of the three DREAM3 sub-challenges), a procedure based on Petri Nets with Fuzzy Logic (Küffner *et al.*, 2010, best performing team of the DREAM4 *In Silico Size 10* sub-challenge), and two procedures based on z-scores measuring the change in gene expression levels when a particular gene is deleted (Greenfield *et al.*, 2010; Pinna *et al.*, 2010, best performing teams of the DREAM4 *In Silico Size 100* sub-challenge). Three of these four methods make an intensive use of the steady-state expression data resulting from the systematic knockout of each gene of the network, highlighting, as some results presented in Chapter 7, the importance of this type of data for the inference of regulatory networks.

To check if our GENIE3-comb procedure could be improved by an appropriate use of the knockout data, we combined it with the inference method of Greenfield *et al.* (2010). In this latter procedure, the weight of the edge directed from gene i to gene j is given by the following median corrected

Table 8.1: **Comparison with the best performers.**

	Rank	GENIE3-comb	Best
DREAM3 Size 10	3rd	0.383	0.698
DREAM3 Size 50	3rd	0.298	0.589
DREAM3 Size 100	2nd	0.297	0.579
DREAM4 Size 10	4th	0.541	0.809
DREAM4 Size 100	3rd	0.326	0.373

Rank: rank of GENIE3-comb among the official challengers, according to the overall score. GENIE3-comb: Mean AUPR obtained with GENIE3-comb. Best: Mean AUPR obtained by the best performer in the sub-challenge. The AUPR scores of each method were averaged over the five networks of each sub-challenge.

z-score (MCZ):

$$w_{i,j} = \frac{x^j_{i,ko} - x^j_{wt}}{\sigma_j}. \tag{8.17}$$

$x^j_{i,ko}$ is the expression of gene j when the gene i is deleted. x^j_{wt} is the expected wild-type expression of gene j, computed as its median expression over the following conditions: the wild-type, the knockout experiments, and the initial time points of all time series experiments. σ_j is the standard deviation of gene j over these same conditions. To combine MCZ with GENIE3-comb, we simply take the products of the scores of the two methods. The weight of the edge $i \to j$ is given by:

$$w_{i,j} = w^{\text{MCZ}}_{i,j} \times w^{\text{GENIE3}}_{i,j}, \tag{8.18}$$

where the weights obtained by the two methods are rescaled in order to be comprised between 0 and 1. The final weight $w_{i,j}$ will thus have a high value if the edge $i \to j$ is ranked in the top of the list by both methods.

As shown in Figure 8.6, the predictions of the networks are indeed improved when the two methods are combined. Actually, this procedure based on the combination would have been ranked first in all the sub-challenges, except the DREAM4 *In Silico Size 10*, where it would have been ranked third.

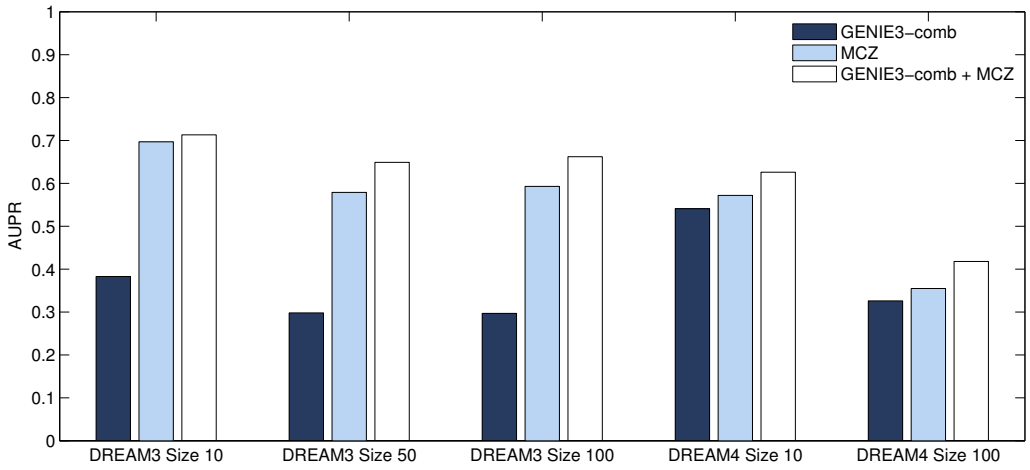


Figure 8.6: **AUPR scores for the DREAM3 and DREAM4 networks.** For each sub-challenge, the AUPR values are those obtained when applying respectively GENIE3-comb, MCZ, and the combination of these two methods. The AUPR values of each method were averaged over the five networks of each sub-challenge.

8.2 Genetical genomics data

The goal of genetical genomics is to exploit the natural variations that exist between the DNA sequences of related individuals and that can represent the randomized and multifactorial perturbations necessary to recover gene regulatory networks (Jansen, 2003; Jansen and Nap, 2001). In such a study, two strains that are widely separated in terms of genetic background are crossed (Figure 8.7(a)) and their children are self-crossed during several generations in order to produce a *recombinant inbred line* (RIL) segregating population (Figure 8.7(b)). The genomes of the individuals of this population comprise random segments of the genomes of the two original parents and genetic differences can therefore be detected between them, representing multifactorial genetic perturbations. Each individual is then analyzed by microarray expression profiling (Figure 8.7(c)), as well as by genetic marker analysis (Figure 8.7(d)). The goal of this second analysis is to identify which variations occur at specific locations, defined by the genetic markers, in the genome of an individual. Note that the individuals of a RIL population are such that each genetic marker can have two possible states only[3].

Multiple methods have been developed to infer gene regulatory networks from genetical genomics data. Several methods infer causal regulatory rela-

[3]This is due to the fact that each individual of a RIL population is *homozygous*.

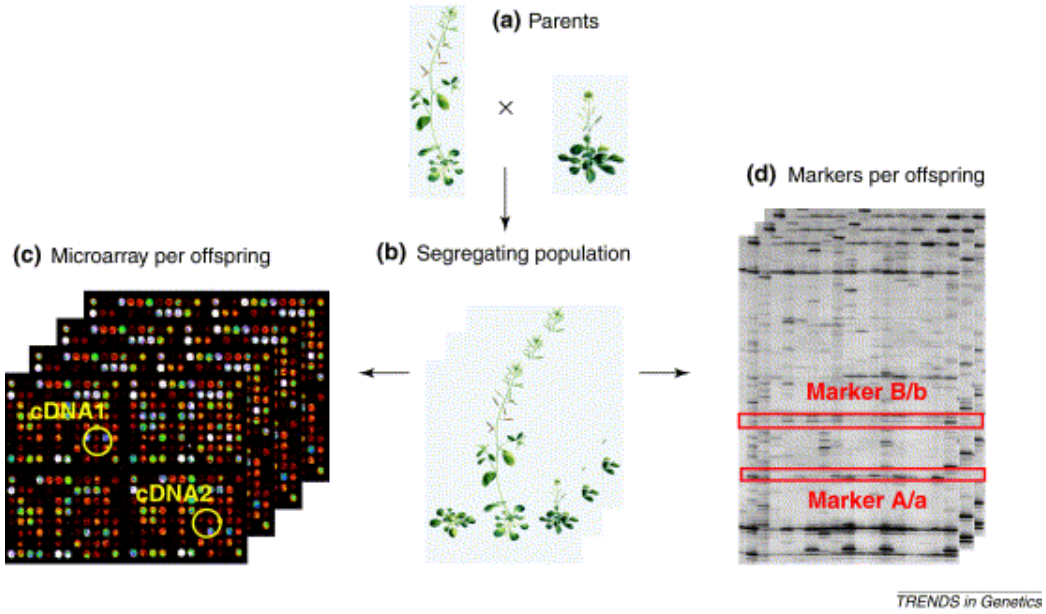


Figure 8.7: **Genetical genomics.** Two genetically diverse strains are crossed and their children are self-crossed during multiple generations, thereby producing a segregating population, i.e. a population of individuals in which genetic differences can be detected. Each individual in this population is then analyzed by microarray expression profiling and genetic marker analysis. Figure taken from Jansen and Nap (2001).

tionships among pairs of genes, including procedures that rely on statistical tests to identify causal links (Chen *et al.*, 2007) and approaches based on the fitting of causal models (Kulp and Jagalur, 2006; Schadt *et al.*, 2005). Other methods are based on the analysis of the correlation between expression profiles of genes located in a particular genomic region and expression profiles of genes that are potentially affected by the markers located in this region (Bing and Hoeschele, 2005). Methods that study the regulatory relationships at a systems-level include approaches based on Bayesian networks (Li *et al.*, 2005; Vandel *et al.*, 2010; Zhu *et al.*, 2004), structural equation models (Li *et al.*, 2006; Liu *et al.*, 2008), and the orientation of the edges of an undirected network using genetic markers as causal anchors (Aten *et al.*, 2008; Chaibub Neto *et al.*, 2008).

In what follows, we assume that each gene whose (steady-state) expression is profiled is analyzed for one single (functional) genetic marker, located either in the promoter region of the gene or in its coding region. In these conditions, genetic markers can have mainly two effects on gene expression, which are both illustrated in Figure 8.8. When a marker is located in the

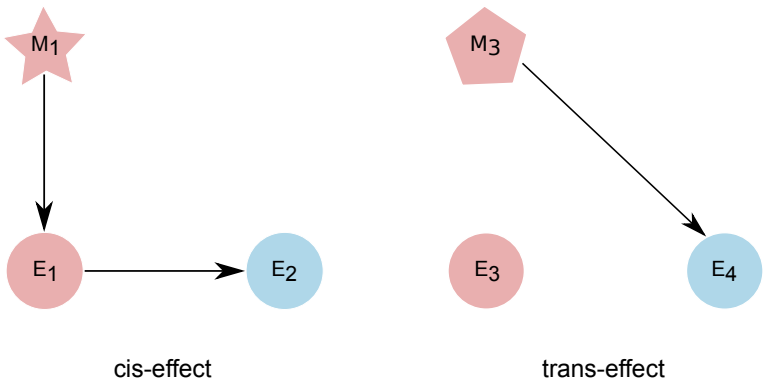


Figure 8.8: **Effects of genetic markers on gene expression.** In this example, gene 1 regulates gene 2 and gene 3 regulates gene 4. M_i represents the marker of gene i and E_i represents the expression of gene i, $i \in \{1, 2, 3, 4\}$. M_1 is in the promoter region of gene 1 and affects the expression of gene 1, which in turn affects the expression of gene 2. M_3 is in the coding region of gene 3 and does not affect the expression of gene 3, but affects the expression of the genes targeted by gene 3.

promoter region of a gene, it has a *cis*-effect on its expression, by affecting the rate of transcription of the gene. In such a case, there is a correlation between the state of the marker (M_1) and the expression of the gene (E_1). Furthermore if this gene regulates another gene, the expression of its target gene (E_2) is also affected by the state of the marker. On the other side, when a marker is located in the coding region of a gene, it does not affect the expression of this gene, but rather the properties (e.g. the structure) of the protein that is produced by the gene, and hence the expression of each gene regulated by this protein. In this case, we say that the marker (M_3) has a *trans*-effect on the expression of the target genes (E_4). We can thus assume that there is always a correlation between the state of the marker of a regulatory gene and the expression of the genes that it regulates, either by *cis*-effect or by *trans*-effect. It is therefore of great interest to exploit information about genetic markers for the inference of regulatory networks. In the following section, we describe how the GENIE3 procedure can be modified to incorporate genetic markers.

8.2.1 Inference from genetical genomics data

We assume that we have at our disposal a dataset containing the steady-state expression levels of p genes measured in N individuals, as well as the state

of one genetic marker for each of these genes in the same N individuals:

$$LS = \{(\mathbf{e}_1, \mathbf{m}_1), (\mathbf{e}_2, \mathbf{m}_2), \ldots, (\mathbf{e}_N, \mathbf{m}_N)\}, \tag{8.19}$$

where $\mathbf{e}_k \in \mathbb{R}^p$ and $\mathbf{m}_k \in \{0,1\}^p, k = 1, \ldots, N$ are respectively the vectors of expression levels and marker states of the p genes in the kth individual:

$$\begin{cases} \mathbf{e}_k = (e_k^1, e_k^2, \ldots, e_k^p)^\top, \\ \mathbf{m}_k = (m_k^1, m_k^2, \ldots, m_k^p)^\top. \end{cases} \tag{8.20}$$

We propose two extensions of GENIE3 integrating genetic data and expression data for the inference of gene regulatory networks. Both procedures are based on the computation of tree-based regression models predicting the expression of each gene $j = 1, \ldots, p$ of the network. The first procedure, called **GENIE3-gen-1**, learns a single model from both types of data, while the second procedure, called **GENIE3-gen-2**, learns two separate models, one based on the genetic markers and the other one based on the expression data.

The GENIE3-gen-1 procedure assumes that a unique model f_j explains the expression of a gene j in a given individual, knowing the expression levels and the states of the genetic markers of the different genes of the network:

$$e_k^j = f_j(\mathbf{e}_k^{-j}, \mathbf{m}_k) + \epsilon_k, \forall k, \tag{8.21}$$

where $\mathbf{e}_k^{-j}$ is the vector containing the expression levels of all the genes except gene j in the kth individual and ϵ_k is a random noise. Notice that $\mathbf{m}_k$ contains the state of the marker of gene j. Indeed, it often happens that a genetic marker contributes to the expression of the gene in which it is located (*cis*-acting polymorphism). Including the marker of gene j in the input variables thus avoids to wrongly attribute to another regulator the part of the expression of gene j that is actually explained by the marker. To learn f_j, we apply a tree-based ensemble method to the following learning sample, in which the output feature is the expression of gene j and marker states are added to expression levels as input features.

$$LS_{e,m}^j = \{((\mathbf{e}_k^{-j}, \mathbf{m}_k), e_k^j), k = 1, \ldots, N\}. \tag{8.22}$$

Tree-based importance scores $w_{i,j}^e (i \neq j)$ and $w_{i,j}^m, i = 1, \ldots, p$, corresponding respectively to the expression and the genetic marker of gene i can then be computed from the model f_j, in the form of sums of variance reductions (2.12).

In the second proposed procedure, GENIE3-gen-2, we assume that two different models f_j^e and f_j^m can both explain the expression of a gene j in a

given individual, either from the expression levels of the other genes, or from the states of the genetic markers:

$$\begin{cases} e_k^j = f_j^e(\mathbf{e}_k^{-j}) + \epsilon_k, \forall k, \\ e_k^j = f_j^m(\mathbf{m}_k) + \epsilon'_k, \forall k. \end{cases} \tag{8.23}$$

The functions f_j^e and f_j^m are respectively learned from two learning samples. In both learning samples, the output variable is the expression of gene j. In the first learning sample, input features are gene expression levels:

$$LS_e^j = \{(\mathbf{e}_k^{-j}, e_k^j), k = 1, \ldots, N\}, \tag{8.24}$$

while in the second learning sample, input features are the states of the genetic markers:

$$LS_m^j = \{(\mathbf{m}_k, e_k^j), k = 1, \ldots, N\}. \tag{8.25}$$

The tree-based importance score $w_{i,j}^e$ of the expression of an input gene i is then computed from f_j^e, while the tree-based importance score $w_{i,j}^m$ of its genetic marker is computed from f_j^m.

In both procedures, we thus obtain, for each input gene i, two separate importance scores $w_{i,j}^e$ and $w_{i,j}^m$, corresponding respectively to the expression and the marker of gene i. To aggregate these two scores, we again propose two procedures. In the first procedure, the final weight of the edge directed from gene i to gene j is the sum of the importance scores:

$$w_{i,j} = w_{i,j}^e + w_{i,j}^m. \tag{8.26}$$

The edge will thus have a high weight if *either* the marker or the expression of gene i is predictive of the expression of gene j. In the second aggregation procedure, we consider the product of the importance scores:

$$w_{i,j} = w_{i,j}^e \times w_{i,j}^m. \tag{8.27}$$

The edge directed from gene i to gene j will thus have a high weight if the marker and the expression of gene i are *both* predictive of the expression of gene j.

8.2.2 The DREAM5 *Systems Genetics* challenge

We applied the proposed procedures to the synthetic datasets of the DREAM5 *Systems Genetics* challenge which is described below.

Challenge

Besides the *Network Inference* challenge, the DREAM5 edition comprised another challenge, called the *Systems Genetics* challenge. This challenge concerned the inference of *in silico* regulatory networks from genetical genomics data[4]. It was divided into three sub-challenges. The goal of each sub-challenge was to infer five networks from populations of 100, 300, and 999 individuals respectively. Each of the 15 networks contained 1000 genes and for each individual, expression levels of these genes were provided, as well as the state of one genetic marker for each gene. All data were generated using SysGenSIM[5] (Pinna *et al.*, 2011b).

8.2.3 Results

Performance of the methods

Figure 8.9 shows the AUPR scores obtained by the different GENIE3 procedures, all applied using the Random Forests method with its main parameter K fixed to the number of input variables and growing $T = 1000$ trees. The related PR curves are plotted in Figure C.5. As expected, the performance of each method improves when the number of individuals for which data are available increases. The scores also indicate that genetic markers are much more informative than expression data for the inference of the networks. For all the datasets, only exploiting genetic data ("GENIE3 on markers") already results in significantly more accurate predictions than learning from expression data alone. The performance can nevertheless be highly improved when both types of data are integrated, indicating that expression data can still bring some complementary information besides genetic data.

Given an aggregation procedure (either sum or product of the importance scores), better performances are obtained when two separate models are respectively learned from the two types of data (GENIE3-gen-2), instead of one single model (GENIE3-gen-1). The less good performance of GENIE3-gen-1 can be explained by the fact that when the inputs comprise continuous and discrete variables (with a low number of categories), the Random Forests method has a positive bias for the continuous variables when selecting a variable at a test node (Strobl *et al.*, 2007). Indeed, since a continuous variable provides more possible cut-points than a variable with a low number of categories, it has more chance to provide the highest variance reduction on the local node, and hence to be selected for the test, even if it is actually less

[4] http://wiki.c2b2.columbia.edu/dream/index.php/D5c3

[5] http://sysgensim.sourceforge.net/

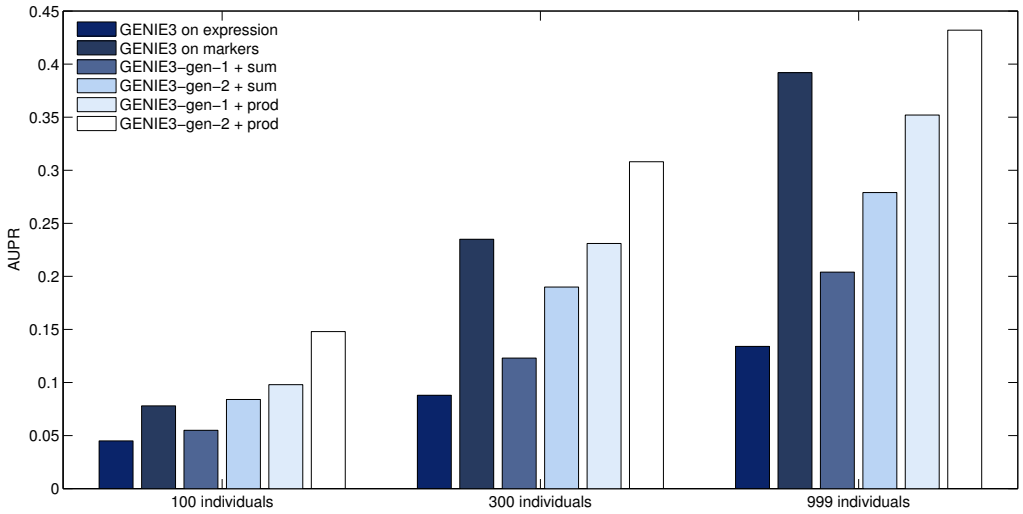


Figure 8.9: **AUPR scores for the DREAM5 *Systems Genetics* challenge.** GENIE3 on expression: original GENIE3 procedure applied to expression data. GENIE3 on markers: weight of edge $i \to j$ is the weight $w^m_{i,j}$ computed from f^m_j as defined in Equation (8.23). Sum: weight of edge $i \to j$ is the sum of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. Prod: weight of edge $i \to j$ is the product of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. The AUPR values of each method were averaged over the five networks of each sub-challenge.

or equally informative globally. Therefore, in GENIE3-gen-1, which learns a joint model from the gene expression values (continuous variables) and from the states of the genetic markers (discrete variables), the importance $w^e_{i,j}$ of the expression of each gene i tends to be higher than the importance $w^m_{i,j}$ of its marker, as shown in Figure 8.10. This bias is not encountered in GENIE3-gen-2, which learns two models, each one from a set of variables that are of the same type.

For both procedures GENIE3-gen-1 and GENIE3-gen-2, higher scores are obtained when the importance scores $w^e_{i,j}$ and $w^m_{i,j}$ are aggregated by taking their product rather than their sum, i.e. when we consider that both the genetic marker and the expression of a regulating gene are important for the prediction of the expression of a target gene. This conservative aggregation procedure allows to give a lower weight to a lot of false edges, in particular by predicting more accurately the direction of the regulatory links. For example, if a gene G_1 regulates a gene G_2, with the inverse being not true, it may happen that the expression of G_2 is still predictive of the expression of G_1,

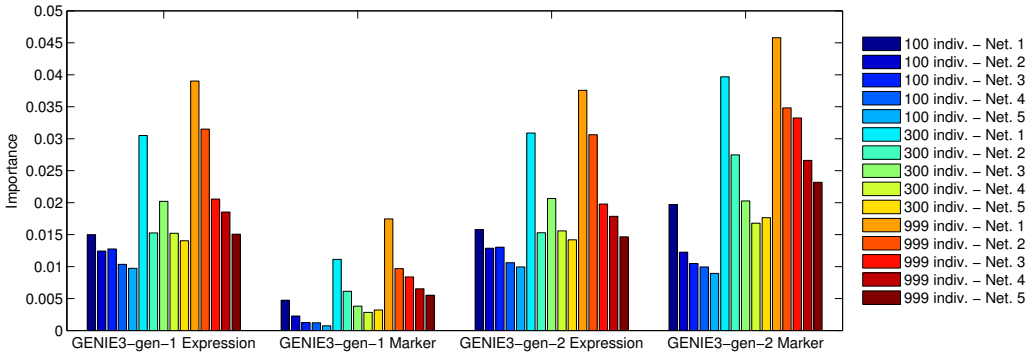


Figure 8.10: **Importance of expressions and markers.** This figure shows, for each network, the average weight $w^e_{i,j}$ obtained from the expression profiles, as well as the average weight $w^m_{i,j}$ obtained from the markers, both computed over the edges $i \to j$ that are part of the gold standard network. The weights $w^e_{i,j}$ and $w^m_{i,j}$ are those obtained with GENIE3-gen-1 (left part of the figure) and GENIE3-gen-2 (right part of the figure).

but the genetic marker of G_2 will not be informative. Therefore, while the sum of the weights related to the false edge $G_2 \to G_1$ can be relatively high with respect to the sum of the weights related to the true edge $G_1 \to G_2$, their product tends to be close to zero (Figure 8.11).

Direction of the edges

Figure 8.12 shows the error rates on the direction of the edges. Compared to exploiting expression data alone, using information about genetic markers greatly helps for the prediction of the direction of the edges. As mentioned in the previous paragraph, lower error rates are indeed obtained when the importance scores $w^e_{i,j}$ and $w^m_{i,j}$ are aggregated by their product instead of their sum.

Comparison with the best performers

Figure 8.13 compares, in terms of AUPR scores, GENIE3-gen-2 to the four methods that were used by the official best performing team of the DREAM5 *Systems Genetics* challenge (Team SaAB: M. Vignes, J. Vandel, N. Ramadan, D. Allouche, C. Cierco, S. de Givry, B. Mangin, T. Schiex, from INRA Toulouse, France)[6]. These methods are respectively based on Dantzig regression (Candès and Tao, 2007), LASSO regression (Tibshirani, 1996), static

[6]These results were provided by Jimmy Vandel, during his visit in our department in February, 2011.

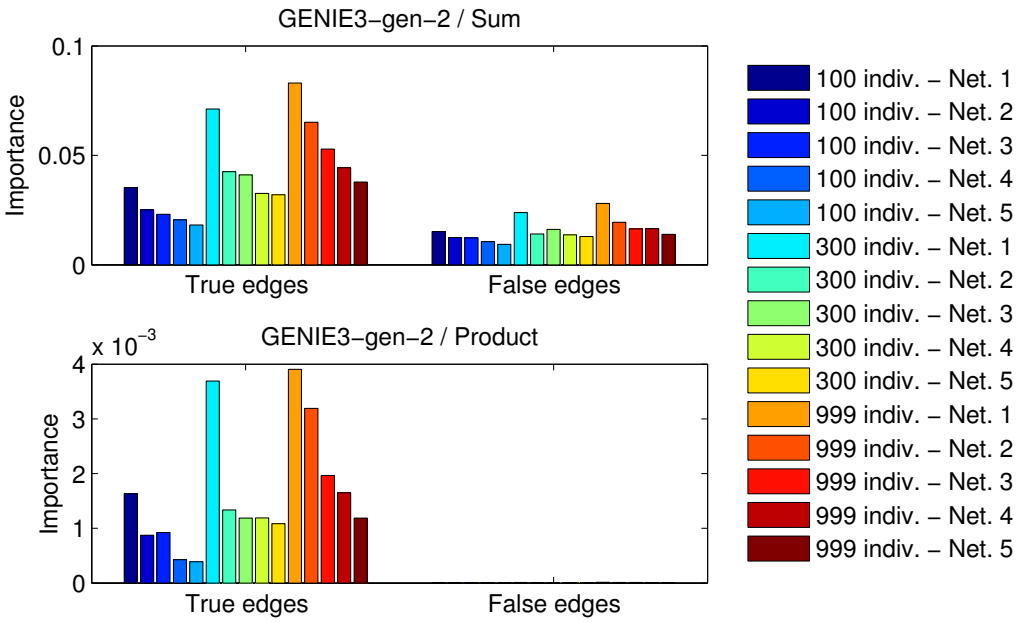


Figure 8.11: **Weights of the regulatory links returned by GENIE3-gen-2.** This figure shows, for each network, the average weight $w_{i,j}$ returned by GENIE3-gen-2, computed over the edges $i \to j$ that are part of the gold standard and such that there is no edge $j \to i$ (true edges), and over the inverse edges $j \to i$ (false edges). Top: weight of edge $i \to j$ is the sum of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. Bottom: weight of edge $i \to j$ is the product of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i.

Bayesian network learning (Friedman *et al.*, 2000), and a meta-analysis using Fisher's inverse χ^2 test (Fisher, 1925) to combine the predictions obtained by the first three methods. The AUPR scores indicate that our procedure significantly outperforms these four methods in each of the three sub-challenges.

Influence of the network density

Besides the study of the effect of the dataset size (number of individuals) on the predictions returned by inference methods, the DREAM5 *Systems Genetics* challenge was also designed to study the effect of the connectivity of a network on the predictions. Each sub-challenge thus included networks with various numbers of edges. Figure 8.14 shows the effect of the network density on the predictions returned by GENIE3-gen-2 and the methods of the best performer. Clearly, in each sub-challenge, the ability of all the methods

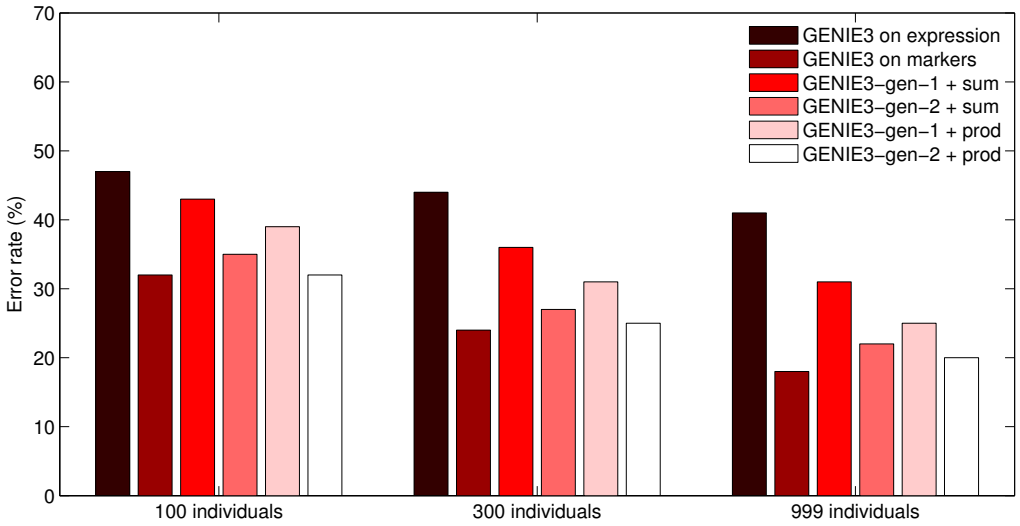


Figure 8.12: **Error rates on edge directionality on the networks of the DREAM5 *Systems Genetics* challenge.** GENIE3 on expression: original GENIE3 procedure applied to expression data. GENIE3 on markers: weight of edge $i \to j$ is the weight $w^m_{i,j}$ computed from f^m_j as defined in Equation (8.23). Sum: weight of edge $i \to j$ is the sum of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. Prod: weight of edge $i \to j$ is the product of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. The error rate is the proportion of edges $i \to j$ in the gold standard network such that there is no edge $j \to i$ and for which the method wrongly predicts $w_{i,j} < w_{j,i}$. The error rates of each method were averaged over the five networks of each sub-challenge.

at recovering a network tends to decrease as the number of edges in the network increases and regulatory interactions become more complex.

We can also observe an effect of the network density on the results presented in Figure 8.10 and 8.11. In these figures, the networks within each sub-challenge are numbered according to the density (network 1 has the lowest number of edges and network 5 has the highest number of edges). When the number of edges increases and the regulation of a gene becomes "more combinatorial", the importance values of the regulators of a target gene decrease, since the information related to the expression of the target gene is shared between more regulators.

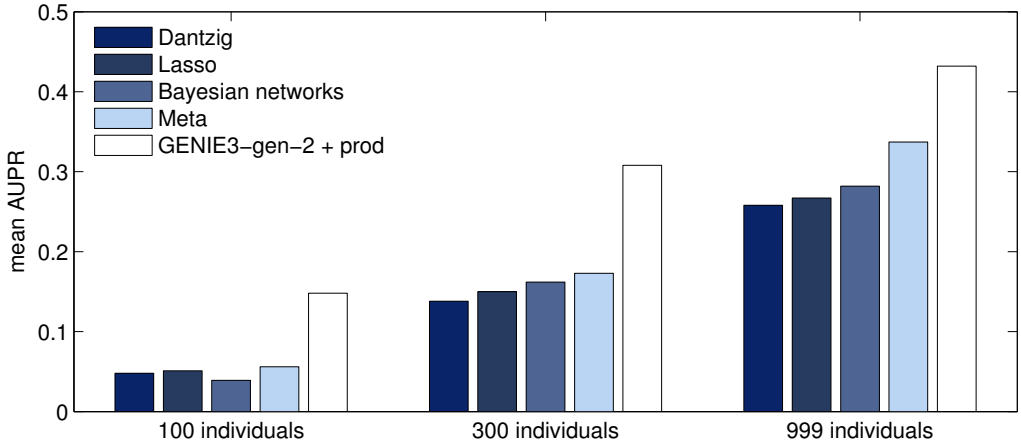


Figure 8.13: **AUPR scores for the DREAM5 *Systems Genetics* challenge.** The AUPR values are those obtained by GENIE3-gen-2 (with the product of the different pairs of importance scores related to the same regulatory link), as well as by the methods used by the official best performing team of the challenge, comprising LASSO regression, Dantzig regression, Bayesian networks, and a combination of these three methods (Meta). The AUPR values of each method were averaged over the five networks of each sub-challenge.

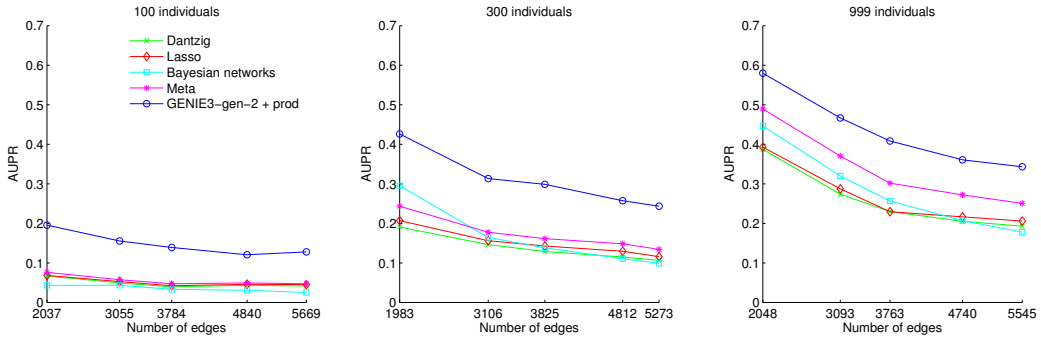


Figure 8.14: **Influence of the network density.** The AUPR values are those obtained by GENIE3-gen-2 (with the product of the different pairs of importance scores related to the same regulatory link), as well as by the methods used by the official best performing team of the challenge, comprising LASSO regression, Dantzig regression, Bayesian networks, and a combination of these three methods (Meta).

8.3 Discussion

In this chapter, we proposed different extensions of the GENIE3 method in order to infer gene regulatory networks from time series data and from genetical genomics data. All these extensions are based on the same core procedure, which consists in decomposing the problem of inferring a regulatory network of p genes into p different feature selection problems, the goal of each being to retrieve the regulators of one of the genes of the network. Each feature selection problem is then solved by applying a tree-based ensemble method.

We proposed two procedures that integrate steady-state data and time series data for the inference of gene regulatory networks. The best performing of these two procedures considers steady-state data as time series measurements that do not fluctuate over time, and learns for each gene of the network a tree-based model predicting the expression of this gene at a time $t + h$, where h denotes a given time horizon, knowing the expression levels of the other genes at time t. The importance of the expression of a candidate input gene at time t for the prediction of the expression of the target gene at time $t + h$ is taken as weight for the edge directed from the input gene to the target gene in the predicted network. Compared to the original GENIE3 procedure, this extension allows to predict much more accurately the directionality of the edges of the network. On the different DREAM3 and DREAM4 sub-challenges, the procedure is competitive with the top performing methods but still remains significantly outperformed by the best performing algorithm of each sub-challenge. We could improve our method by combining it with the procedure of one of the best performers, that exploits more appropriately the steady-state expression data resulting from gene knockouts. However, this latter procedure requires a complete dataset comprising the systematic knockout of each gene of the targeted network, which may be unrealistic.

Among the different procedures that we proposed to infer regulatory networks from genetical genomics data, the best performing one consists in learning, for each gene of the network, two tree-based models predicting the expression of that gene: one from the expression levels of the other genes, and the other one from the states of the genetic markers of the genes. Then, the weight that is given to the edge directed from one input gene to the target gene in the predicted network is the product of the importances of the expression of the input gene and of its genetic marker for the prediction of the expression of the target gene. Results obtained on the artificial networks of the DREAM5 *Systems Genetics* challenge showed that genetic markers bring much more information about the regulatory networks than expression

data and are thus highly helpful for their recovering. Also, our inference procedure actually outperforms the official best performing algorithm of the challenge.

The DREAM challenges allowed us to make a first evaluation of the performances of our different procedures on time series and genetic data. However, these challenges are solely based on networks and data that are artificial. As future works, we thus would like to apply our methods on real datasets. In a first time, we could restrict ourselves to organisms for which there exists a proper gold standard regulatory network, such as *E. coli* and *S. cerevisiae*, in order to be able to evaluate the predictions returned by our methods. For time series data, we could for example use the *E. coli* and *S. cerevisiae* compendia of the DREAM5 *Network Inference* challenge, that both contain expression data obtained from different time course experiments. In these compendia, the time intervals between two expression measurements are usually different from one time course experiment to another, and we thus have to find a way to deal with these different intervals in our procedures. We tried to interpolate the gene expression values of the DREAM5 *E. coli* dataset in order to get equal time intervals, but the resulting predictions (obtained using GENIE3-comb) were not better than our initial submission to the challenge (i.e. the predictions obtained using the original GENIE3 procedure). Concerning genetical genomics data, datasets related to various organisms are publicly available, such as the *S. cerevisiae* dataset of Brem and Kruglyak (2005). However, in our different procedures, we assume that each gene whose expression is measured in N individuals is also analyzed for one single genetic marker in each of these N individuals. Unfortunately, this situation is usually not encountered in real datasets. We will thus have to modify our methods in order to deal with missing data, and also to establish a procedure to aggregate the importance scores of different genetic markers related to the same gene.

9 Closure of Part III

This chapter provides a discussion on our contributions to gene network inference and on results obtained from our empirical experiments. We finally give some directions of future research on extensions of the GENIE3 method.

Contents

9.1 Discussion

While the second part of this thesis focused on the identification of relevant variables from a feature ranking, this third part was dedicated to the application of feature selection/ranking techniques to the problem of gene regulatory network inference. The problem of inferring a network of p genes can indeed be viewed as p feature selection problems, the goal of each being to retrieve the regulators of one of the genes of the network.

We exploited this framework in GENIE3, a procedure that aims to recover a gene regulatory network from steady-state expression data. GENIE3 got the best overall performances in the DREAM4 *In Silico Multifactorial* challenge and in the DREAM5 *Network Inference* challenge, and is competitive with existing algorithms to decipher the genetic regulatory network of *Escherichia coli*. We also extended the GENIE3 method to the inference of networks from time series data and from genetical genomics data respectively, and showed that these extensions are competitive with the official best performing algorithms in different DREAM challenges.

The framework of GENIE3 and its extensions are all based on the ranking of the putative regulatory genes for each gene of the network. Among different feature ranking methods, we chose to use tree-based ensemble methods. Several reasons can explain the success of these methods in the inference of gene regulatory networks.

First, a gene is expected to be jointly regulated by several regulators. Therefore, tree-based methods, that are potentially able to detect multivariate interacting effects between variables, have an advantage over methods that consider pairwise interactions only, such as the methods based on the computation of correlation or mutual information.

A second reason for the success of tree-based methods is that these methods can deal with high-dimensional datasets. By contrast, several complex methods, such as Bayesian networks or methods based on systems of ordinary or stochastic differential equations, can yield very good performances on small networks, but fail at recovering larger networks because of computational issues or because they need much more experimental data.

Tree-based methods are also non-parametric and hence do not make any strong assumption about the nature of the regulation, which can thus be non-linear. However, it is not clear yet that using non-linear models constitutes a real advantage, at least for the inference of the *E. coli* and *S. cerevisiae* networks. We indeed obtained better performances on these networks by using linear SVMs instead of tree-based methods (Figure 6.10). Note also that the third overall best performing algorithm of the DREAM5 *Network Inference* challenge is based on the same framework as GENIE3, but uses a LASSO

regression approach to generate each gene ranking (Marbach *et al.*, 2012). Unlike the SVMs, the LASSO performed less good than the Random Forests on the *E. coli* network, indicating that predictions can be very different depending on the linear method used. Nevertheless, Random Forests perform better than linear methods on artificial problems, which are non-linear by construction.

Our experiments on the networks of the DREAM4 *In Silico Multifactorial* challenge showed that GENIE3 is able to predict to some extent the direction of the edges of these networks, even though it only exploits steady-state expression measurements (Table 6.5). This is commonly admitted to be a difficult problem and we are not yet in a position to explain why our method is able to predict directionality to some extent. In addition, higher error rates on edge directionality are obtained on the other networks of the DREAM4 challenge, as well as on those of the DREAM3 challenge, when only steady-state data are exploited (see Figure 8.5), and on the networks of the DREAM5 *Systems Genetics* challenge when only expression data are exploited (see Figure 8.12). One could therefore question the ability of GENIE3 to predict edge directionality from steady-state data alone. On the other hand, exploiting additional data such as time series data or information about genetic markers clearly allows to predict more accurately the direction of the edges.

We could observe from our different experiments some discrepancies between the results obtained on artificial data and those obtained on real data. For example, in the DREAM5 *Network Inference* challenge, improved predictions of the artificial network were obtained with GENIE3 by increasing the parameter K of the Random Forests to its maximum value, while for the *E. coli* network, the best ranking of interactions was obtained with $K = \sqrt{n_{TF}}$, where n_{TF} is the number of candidate regulators. The challenge organizers generated the artificial expression dataset so that it contains the same set of experiments as the *E. coli* compendium. The differences in the predictions are thus not due to differences in the sizes of the datasets or the types of experiments that they each contain.

A potential difference could reside in the fact that the gold standard network of *E. coli* is not complete and hence contains false negative edges. To check this hypothesis, we removed some of the interactions of the artificial gold standard network of DREAM5 (Figure 9.1(A)). For both values of the parameter K ($\sqrt{n_{TF}}$ and n_{TF}), the performance of GENIE3 degrades as we remove more interactions. However, setting $K = n_{TF}$ results in a better performance in all cases, suggesting that the presence of false negatives in the *E. coli* gold standard network is not a cause of the differences observed on this network.

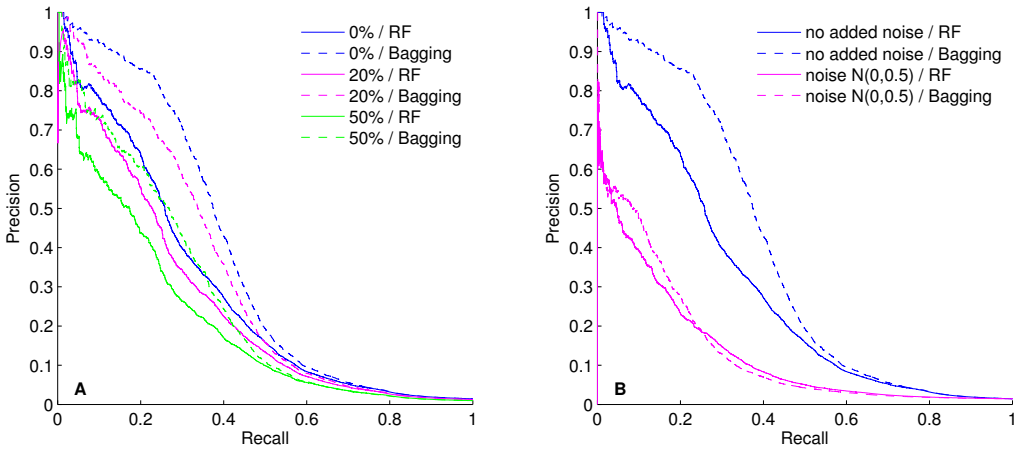


Figure 9.1: **Effects of the false negatives in the gold standard and of the noise in the dataset.** These PR curves evaluate the performance of GENIE3 at recovering the artificial network of the DREAM5 *Network Inference* challenge. **A.** Interactions of the gold standard were randomly removed. The percentage indicated in the legend is the proportion of edges of the gold standard that were removed. **B.** Noise was added to the expression dataset. In each case, the predictions were obtained using GENIE3 with Random Forests, either with $K = \sqrt{n_{TF}}$ (RF) or with $K = n_{TF}$ (Bagging), where n_{TF} is the number of putative regulators.

Another reason could be the discrepancy that exists between the simulation model used to generate the artificial data and the real regulation mechanism of *E. coli*. To simulate expression data, GeneNetWeaver (Marbach *et al.*, 2009; Schaffter *et al.*, 2011) models the rates of change of mRNA and protein concentrations, and adds noise both in the dynamics of the networks and on the measurement of expression data. It does not take into account additional layers of regulation, such as the one formed by non-coding regulatory RNA molecules. As the real regulation mechanism of *E. coli* is probably more complex than the one simulated by GeneNetWeaver, the *E. coli* expression data appear more "noisy" than the artificial data. We thus checked how the performance of GENIE3 varies when adding further noise to the artificial data, in the form of a Gaussian noise $\sim \mathcal{N}(0, 0.5)$ (Figure 9.1(B)). As expected, the predictions are worse when noise is added, for both values of the parameter K of the tree-based method. We can however observe that the performances obtained with $K = \sqrt{n_{TF}}$ and with $K = n_{TF}$ become equivalent, a result that is close to what we obtain on the real networks. This could be explained by the fact that using $K = \sqrt{n_{TF}}$ results in pre-

dictive models that overfit less the data than those obtained with $K = n_{TF}$ and that are therefore more robust to the noise. Note that this might also explain why linear methods perform better than tree-based methods on the real datasets. Their linearity is compensated by a greater robustness to noise and overfitting.

9.2 Extensions of GENIE3

According to Figure 1.4, which proposes a categorization of the network inference methods, the original GENIE3 procedure is a method which is direct (it considers individual interactions only), unsupervised (it does not assume prior knowledge about the targeted network), and non-integrative (it exploits expression data only).

The different extensions of GENIE3 to genetical genomics data that we proposed are also direct and unsupervised, but they are integrative as they combine expression data and information about genetic markers to infer a network. GENIE3 could be extended to many other types of data that would bring complementary information about the targeted regulatory networks, such as information about transcription factor binding sites, or physical interactions between proteins. In particular, we would like to exploit data related to microRNAs. These small non-coding RNA molecules are key players in the regulation of gene expression and exploiting them for the inference of regulatory networks would certainly be of great interest. Nowadays, there is an increasing number of publicly available datasets containing microRNA and mRNA expression levels measured in the same conditions. Exploiting these datasets would potentially help to predict more accurately the regulatory links between the genes, as an additional layer of regulation would be taken into account (see e.g. Bonnet *et al.*, 2010).

It would also be interesting to extend GENIE3 to other categories presented in Figure 1.4. Instead of a direct algorithm, we could consider a module-based procedure that infers a regulatory network while searching for modules, i.e. groups of genes that are regulated by the same transcription factors. To identify a module, we could for example exploit multiple-output trees (Blockeel *et al.*, 1998; Geurts *et al.*, 2006b), which would learn models predicting the expression values of several target genes simultaneously, i.e. those that belong to a given module. Note that we carried out some preliminary experiments in this direction during the course of our thesis, but we were not able to outperform the original GENIE3 method.

We could also modify GENIE3 in order to have a method that infers networks in a supervised way, i.e. by exploiting prior partial knowledge

about the targeted networks. We could for example follow the same idea as SIRENE (Mordelet and Vert, 2008). This method consists in learning, for each transcription factor, a SVM model predicting if a gene is regulated or not by this transcription factor, using the expression profile of the gene. Each model is built from a learning sample containing genes that are known to be regulated (or not) by the corresponding transcription factor, and in which each input variable is the gene expression in a given condition. The rationale behind this framework is that if two genes are regulated by the same transcription factor, they should have similar expression profiles. However, by considering only one transcription factor at a time, SIRENE does not take into account the fact that the regulation of a gene can be combinatorial. For example, two genes can be regulated by a same transcription factor, but have different expression profiles because one of the two genes is also regulated by another transcription factor. Therefore, instead of learning a model for each transcription factor, we can learn a single multiple-output model (e.g. multiple-output trees) in which each output corresponds to a transcription factor. This model would predict the links between a gene and all the transcription factors simultaneously.

Another interesting direction of future research would be to extend GENIE3 to the *differential networking* problem, where the goal is to identify the subparts of the regulatory network that change between healthy and disease-affected tissues (de la Fuente, 2010). We could simply start by inferring two networks using GENIE3, one from the healthy samples and one from the disease samples, and identifying the regulatory links whose weights vary the most between the two networks.

Finally, an obvious goal would be to be able to infer the human regulatory network (Belcastro *et al.*, 2011). However, the direct application of GENIE3 to the human genome would raise some computational issues, due to the very high number of genes. A solution could be to apply a procedure that filters the candidate input genes, before learning each tree-based model predicting the expression of a target gene. We could for example use the expression data resulting from gene knockout experiments, and remove from the input variables the transcription factors whose deletion does not affect the expression of the target gene.

Part IV

Appendices

A

Evaluation and comparison of feature selection methods - Supplementary information

A.1 Pseudo-codes

We assume that we have a learning sample LS of N instances of input-output pairs drawn from some unknown probability distribution. LS contains m input variables denoted $X_i, i = 1, \ldots, m$. We also assume that we have an algorithm $\mathcal{A}(LS)$ that outputs from the learning sample a feature ranking, derived from a relevance score s_i for each input variable X_i. We assume, without loss of generality, that the features are numbered according to their relevance score, i.e.

$$s_1 \geq s_2 \geq \cdots \geq s_m.$$

A.1.1 err-$\mathcal{A}$

Inputs: LS, $\mathcal{A}$.

1. For $i = 1, \ldots, m$:
   (a) Generate LS_i from LS by including only variables $X_j, j = 1, \ldots, i$.
   (b) Learn a predictive model $M_i = \mathcal{A}(LS_i)$.
   (c) Estimate the generalization error e_i of model M_i using ten-fold cross-validation.
2. Determine the minimum value of k such that:

$$e_k = \min_{i=1,\ldots,m} e_i.$$

3. Select variables $X_i, i = 1, \ldots, k$.

A.1.2 pFDR

Inputs: $LS, \mathcal{A}, \alpha$.

1. For $p = 1, \ldots, P$:
   (a) Generate LS^p from LS by randomly permuting the output values.
   (b) Compute variable relevance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$.
   (c) Compute $V_i^p = \#\{k : s_k^p \geq s_i\}$, for $i = 1, \ldots, m$.
2. Then at s_i, the FDR is estimated by:
$$\text{pFDR}_i = \frac{1}{P} \frac{\sum_{p=1}^{P} V_i^p}{i}, \text{ for } i = 1, \ldots, m.$$
3. Enforce monotonicity by setting:
$$\text{pFDR}_1^* \leftarrow \text{pFDR}_1,$$
$$\text{pFDR}_i^* \leftarrow \max(\text{pFDR}_{i-1}^*, \text{pFDR}_i), \text{ for } i = 2, \ldots, m.$$
4. Select variables X_i such that $\text{pFDR}_i^* \leq \alpha$.

A.1.3 CER

Inputs: $LS, \mathcal{A}, \alpha$.

1. For $i = 1, \ldots, m$:
   (a) For $p = 1, \ldots, P$:
   - Generate LS^p from LS by keeping the output values and the values of $X_1, \ldots, X_{i-1}$ fixed, and by randomly and jointly permuting the values of $X_i, \ldots, X_m$.
   - Compute variable importance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$.

   (b) Then at s_i, the FWER is estimated by :
$$\text{CER}_i = \frac{1}{P} \cdot \#\{p : \max_{k=i,\ldots,m} s_k^p \geq s_i\}.$$
2. Enforce monotonicity by setting:
$$\text{CER}_1^* \leftarrow \text{CER}_1,$$
$$\text{CER}_i^* \leftarrow \max(\text{CER}_{i-1}^*, \text{CER}_i), \text{ for } i = 2, \ldots, m.$$
3. Select variables X_i such that $\text{CER}_i^* \leq \alpha$.

A.1.4 eFDR

Inputs: $LS, \mathcal{A}, \alpha$.

1. For $i = 1, \ldots, m$:

   (a) For $p = 1, \ldots, P$:

   - Generate LS^p from LS by keeping the output values and the values of $X_1, \ldots, X_{i-1}$ fixed, and by randomly and jointly permuting the values of $X_i, \ldots, X_m$.
   - Compute variable relevance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$. Let $s_{(j)}^p$ be the j-th largest member of $\{s_i^p, \ldots, s_m^p\}$.
   - Define V_i^p as the unique integer k such that:

$$s_{(1)}^p \geq s_i, \ldots, s_{(k)}^p \geq s_{i+k-1}, \text{ and } s_{(k+1)}^p < s_{i+k}.$$

   (b) Then at s_i, the FDR is estimated by:

$$\text{eFDR}_i = \frac{1}{P} \sum_{p=1}^{P} \frac{V_i^p}{V_i^p + i - 1}.$$

2. Enforce monotonicity by setting:

$$\text{eFDR}_1^* \leftarrow \text{eFDR}_1,$$

$$\text{eFDR}_i^* \leftarrow \max(\text{eFDR}_{i-1}^*, \text{eFDR}_i), \text{ for } i = 2, \ldots, m.$$

3. Select variables X_i such that $\text{eFDR}_i^* \leq \alpha$.

A.1.5 mr-test

Inputs: $LS, \mathcal{A}, k, \alpha$.

1. For $p = 1, \ldots, P$:

   (a) Generate LS^p by randomly sampling (without replacement) half of the instances of LS.

   (b) Compute variable relevance scores $\{s_1^p, \ldots, s_m^p\} = \mathcal{A}(LS^p)$.

   (c) Rank the variables according to their score s_i^p. Let r_i^p be the rank of variable X_i in the pth ranking.

2. Compute the mean rank of each variable over the P rankings:

$$\bar{r}_i = \frac{1}{P} \sum_{p=1}^{P} r_i^p, \text{ for } i = 1, \ldots, m.$$

3. Identify the k variables with the highest mean ranks $\bar{r}_i$.

4. Let $r_{(j)}, j = 1, \ldots, k \cdot P$, be the ranks of these k variables in the P rankings. Then at s_i, compute the following p-value:

$$p_i = \frac{1}{k \cdot P} \cdot \#\{j : r_{(j)} \leq \bar{r}_i\}, \text{ for } i = 1, \ldots, m.$$

5. Correct the p-values with the Benjamini Hochberg procedure.

6. Select variables X_i such that $p_i \leq \alpha$.

A.1.6 1Probe

Inputs: $LS, \mathcal{A}, \alpha$.

1. For $p = 1, \ldots, P$:

   (a) Create a random feature X_{rand} by randomly sampling its values from a normal distribution $\mathcal{N}(0, 1)$.

   (b) Add this random feature to the original learning sample: $LS^p = LS \cup X_{rand}$.

   (c) Compute variable relevance scores $\{s_1^p, \ldots, s_m^p, s_{rand}^p\} = \mathcal{A}(LS^p)$.

2. Then at s_i, compute the following p-value:

$$p_i = \frac{1}{P} \cdot \#\{p : s_{rand}^p \geq s_i^p\}.$$

3. Correct the p-values with the Benjamini Hochberg procedure.

4. Select variables X_i such that $p_i \leq \alpha$.

A.1.7 mProbes

Inputs: $LS, \mathcal{A}, \alpha$.

1. For $p = 1, \ldots, P$:
   (a) Create variable X_i^p by randomly permuting the values of X_i in LS, for $i = 1, \ldots, m$.
   (b) Create a new learning sample $LS^p = LS \cup \{X_1^p, \ldots, X_m^p\}$.
   (c) Compute variable relevance scores $\{v_1, \ldots, v_m, v_1^p, \ldots, v_m^p\} = \mathcal{A}(LS^p)$.
2. Then at s_i, the FWER is estimated by:

$$\text{FWER}_i = \frac{1}{P} \cdot \#\{p : \max_{k=1,\ldots,m} v_k^p \geq v_i\}.$$

3. Select variables X_i such that $\text{FWER}_i \leq \alpha$.

A.2 Supplementary figures

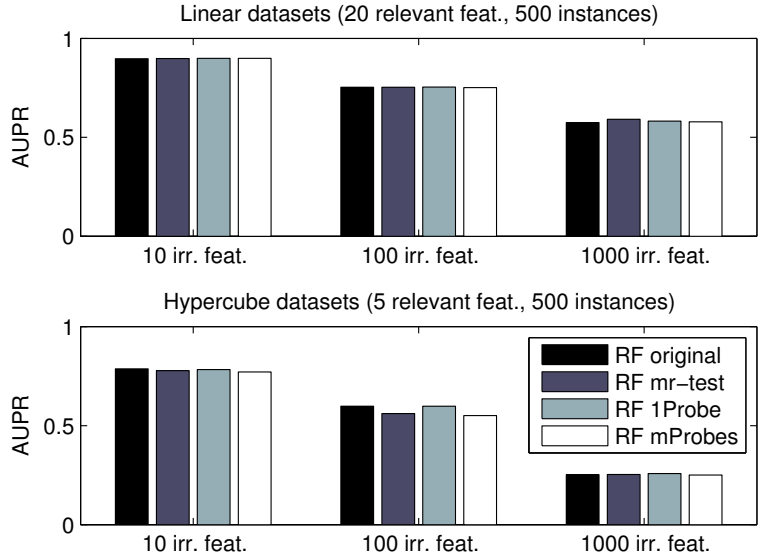


Figure A.1: **AUPR scores.** These AUPRs are those obtained when the variables are ranked according respectively to the importance score derived from Random Forests, the p-value derived from mr-test, the p-value derived from 1Probe, and the FWER derived from mProbes (the three last methods being used with Random Forests). Top on linear datasets, bottom on hypercube datasets, for different numbers of irrelevant features. The AUPR values were averaged over 50 datasets in each case.

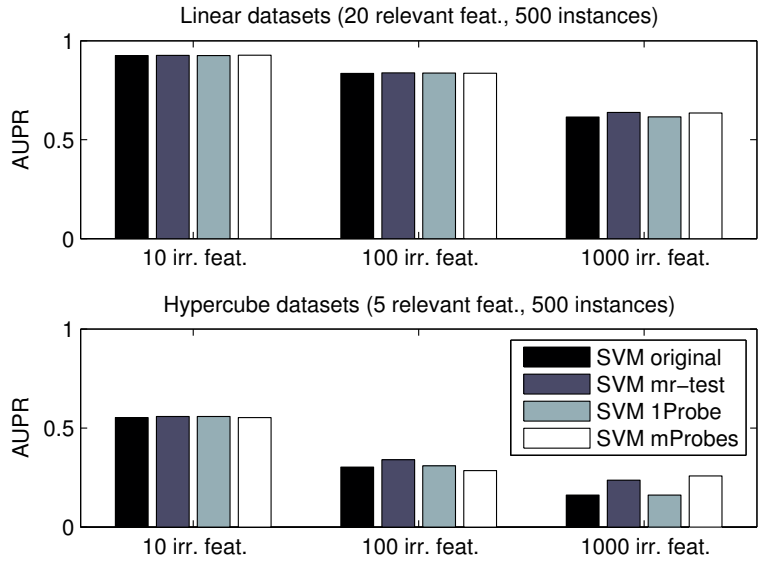


Figure A.2: **AUPR scores.** These AUPRs are those obtained when the variables are ranked according respectively to the importance score derived from a linear SVM, the p-value derived from mr-test, the p-value derived from 1Probe, and the FWER derived from mProbes (the three last methods being used with a linear SVM). Top on linear datasets, bottom on hypercube datasets, for different numbers of irrelevant features. The AUPR values were averaged over 50 datasets in each case.

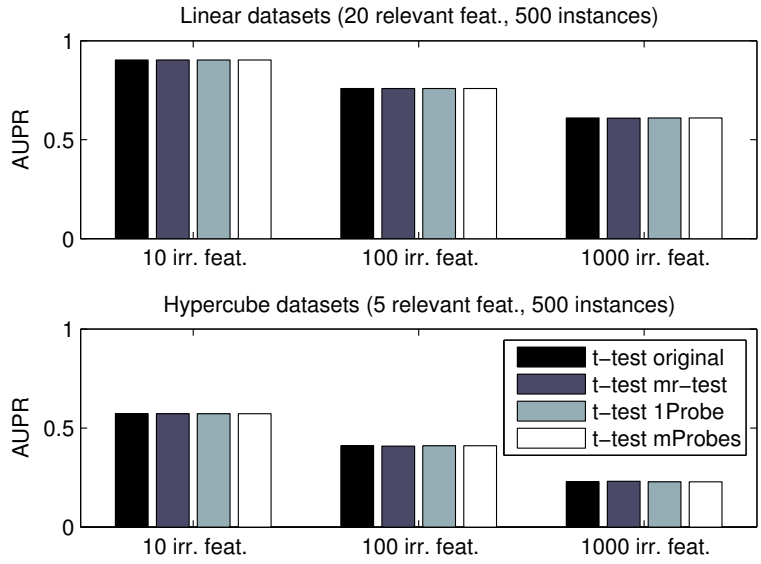


Figure A.3: **AUPR scores.** These AUPRs are those obtained when the variables are ranked according respectively to the statistic t computed by a t-test, the p-value derived from mr-test, the p-value derived from 1Probe, and the FWER derived from mProbes (the three last methods being used with a t-test). Top on linear datasets, bottom on hypercube datasets, for different numbers of irrelevant features. The AUPR values were averaged over 50 datasets in each case.

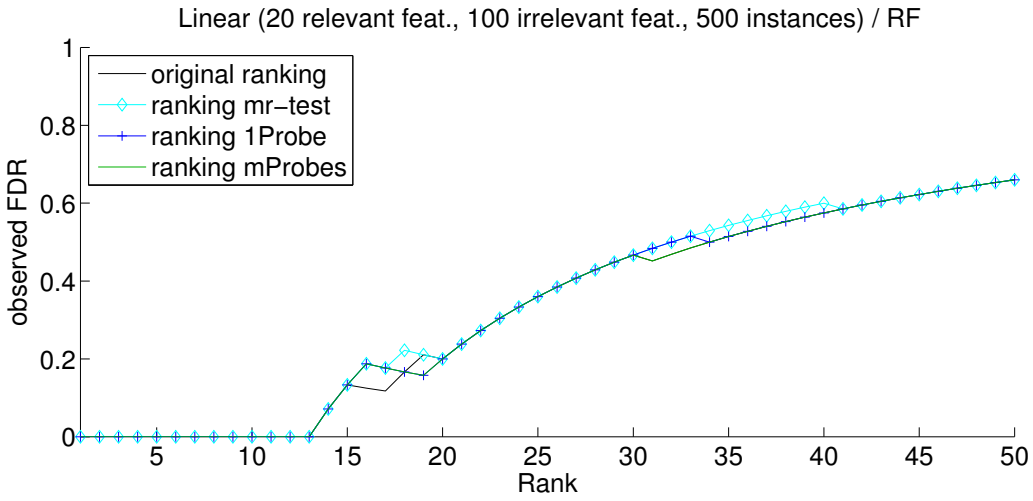


Figure A.4: **Curves of the observed FDR on a linear dataset.** These curves correspond to the original ranking obtained using Random Forests and the rankings obtained using mr-test, 1Probe, and mProbes respectively (all three methods being used with Random Forests).

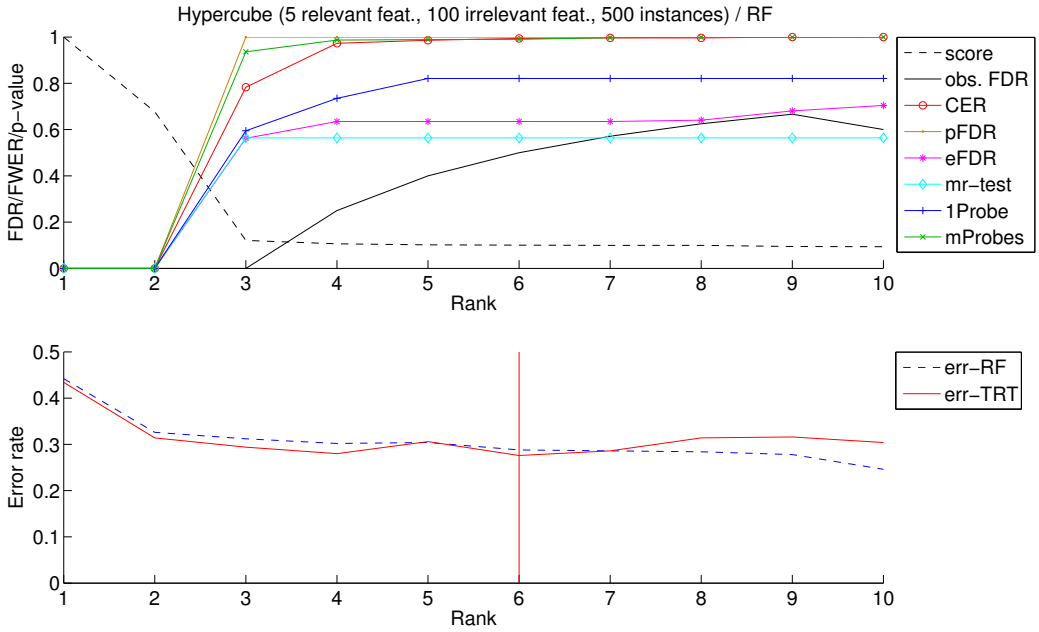


Figure A.5: **Curves of the different methods on a hypercube dataset.** We used the Random Forests as ranking method. *score* is the relevance score derived from the Random Forests. *obs. FDR* is the observed FDR. The red vertical line indicates the position of the lowest error rate for err-TRT. (The lowest error rate for err-RF is obtained at rank 13.)

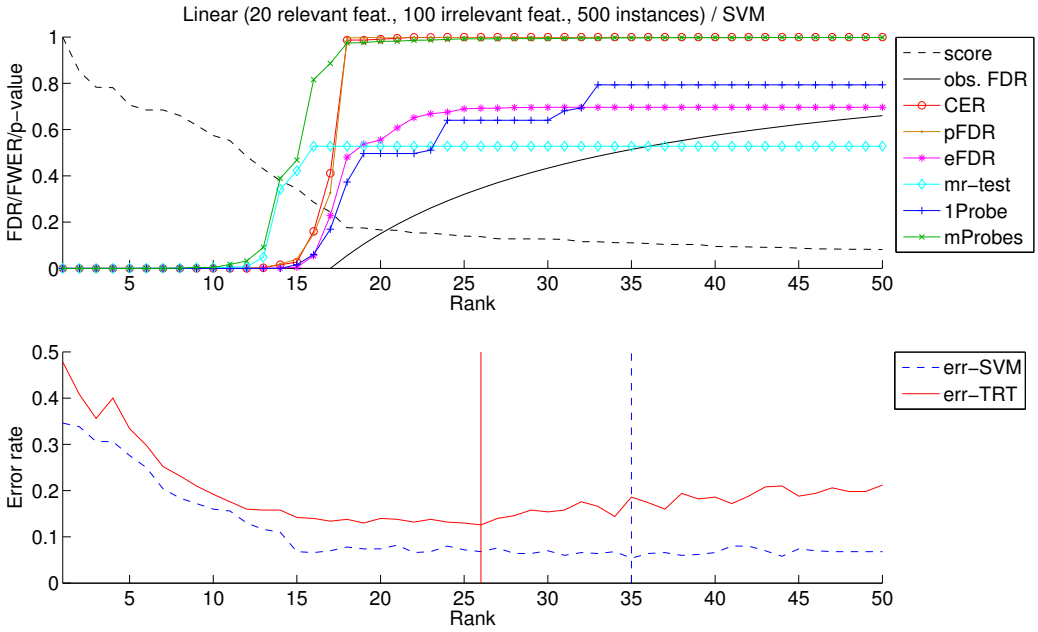


Figure A.6: **Curves of the different methods on a linear dataset.** We used the linear SVM as ranking method. *score* is the relevance score derived from the SVM. *obs. FDR* is the observed FDR. The dashed blue (resp. plain red) vertical line indicates the position of the lowest error rate for err-SVM (resp. err-TRT).

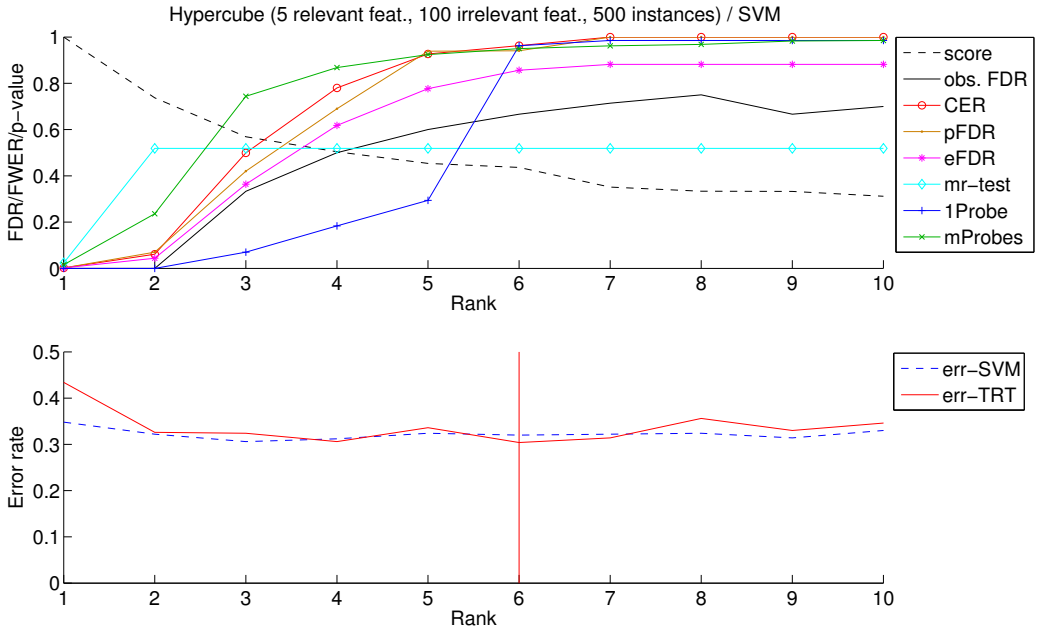


Figure A.7: **Curves of the different methods on a hypercube dataset.** We used the linear SVM as ranking method. *score* is the relevance score derived from the SVM. *obs. FDR* is the observed FDR. The red vertical line indicates the position of the lowest error rate for err-TRT. (The lowest error rate for err-SVM is obtained at rank 37.)

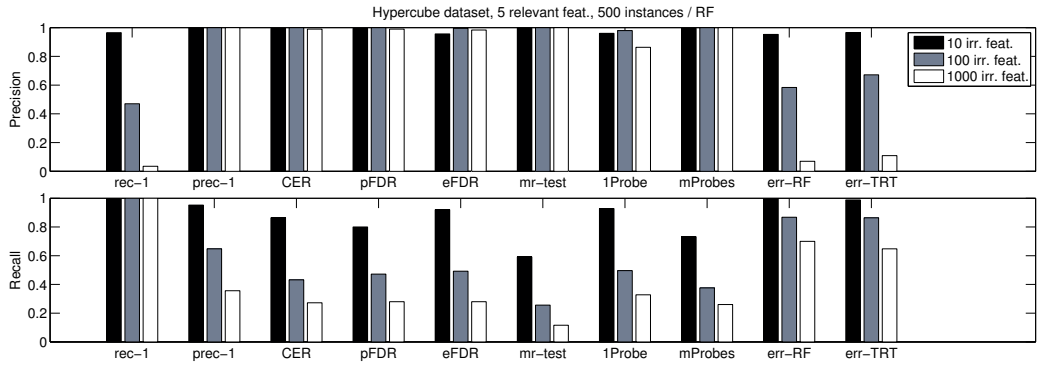


Figure A.8: **Precision and recall on hypercube datasets, for different numbers of irrelevant features.** We used the Random Forests algorithm as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

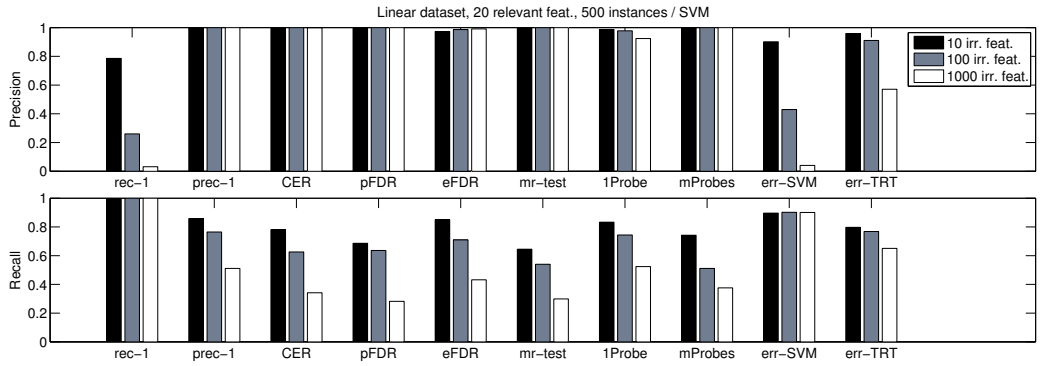


Figure A.9: **Precision and recall on linear datasets, for different numbers of irrelevant features.** We used the linear SVM as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

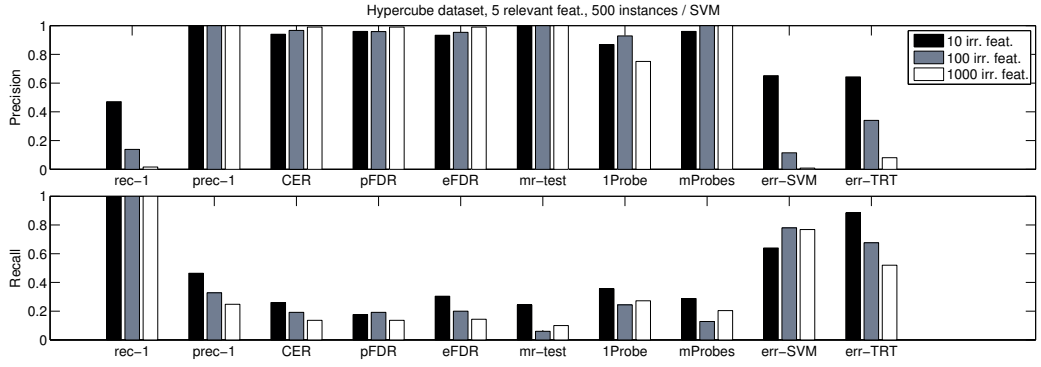


Figure A.10: **Precision and recall on hypercube datasets, for different numbers of irrelevant features.** We used the linear SVM as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

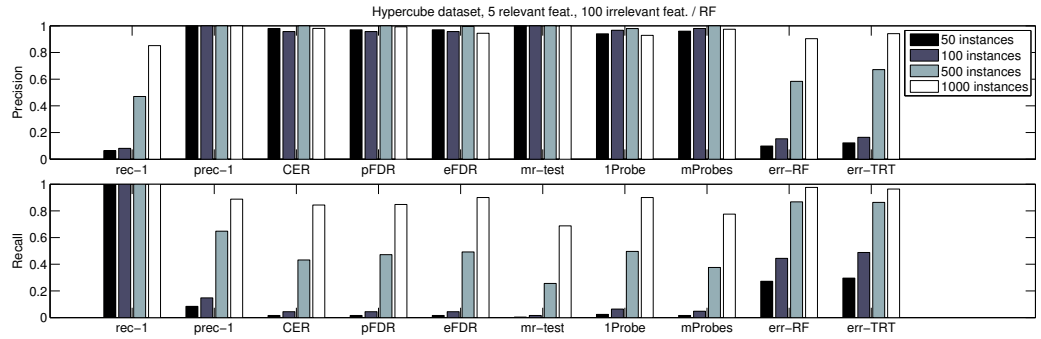


Figure A.11: **Precision and recall on hypercube datasets, for different numbers of instances.** We used the Random Forests algorithm as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

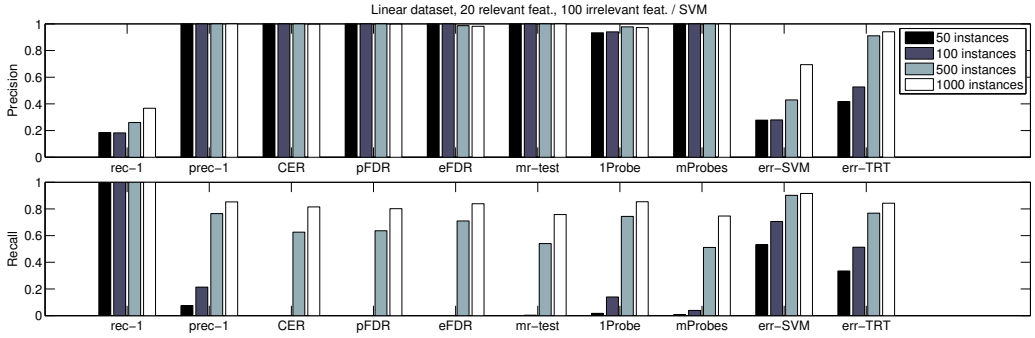


Figure A.12: **Precision and recall on linear datasets, for different numbers of instances.** We used the linear SVM as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

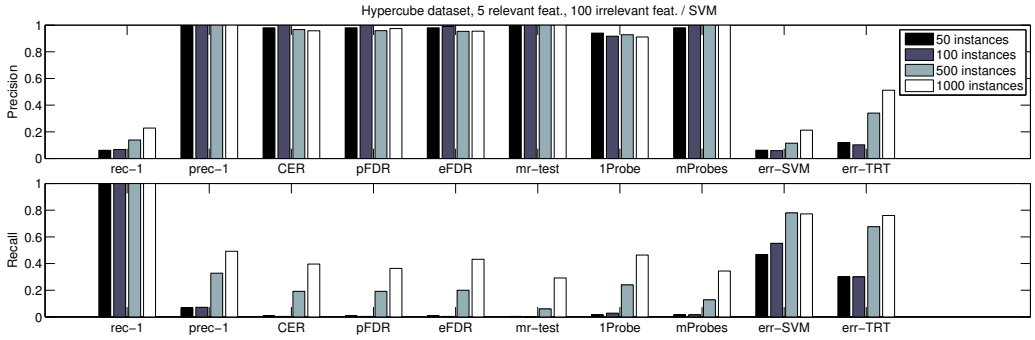


Figure A.13: **Precision and recall on hypercube datasets, for different numbers of instances.** We used the linear SVM as ranking method and $\alpha = 0.05$. The precision and recall values were averaged over 50 datasets in each case.

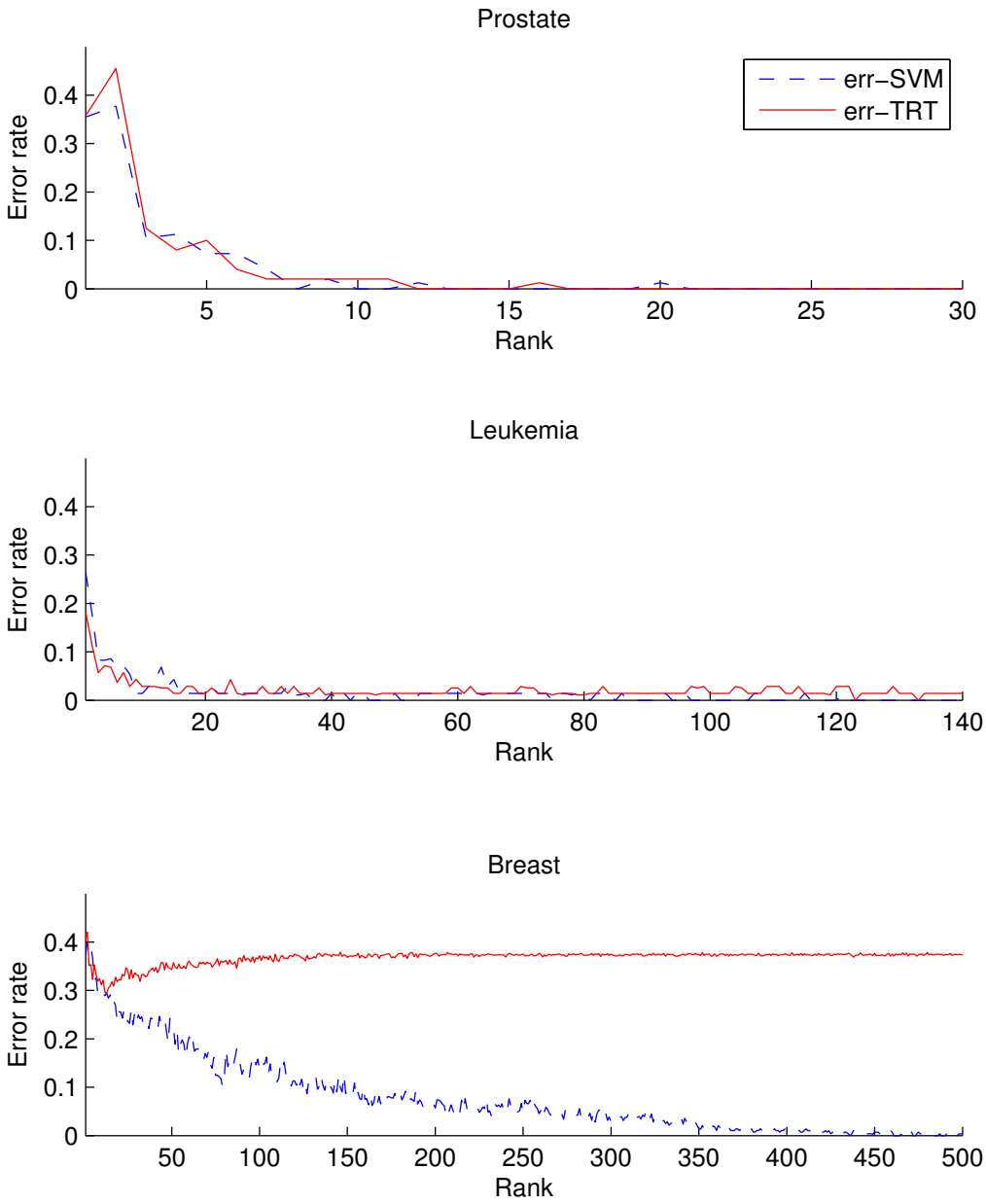


Figure A.14: **Curves of err-SVM and err-TRT, on the microarray datasets.** We used the linear SVM as ranking method.

B

GENIE3: GEne Network Inference with Ensemble of trees - Supplementary figures

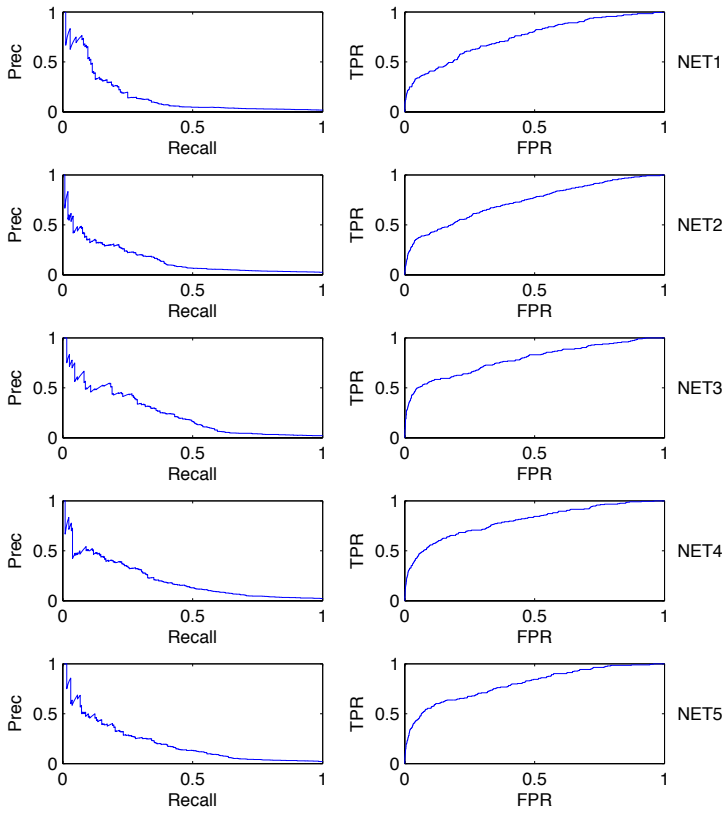


Figure B.1: **PR and ROC curves of GENIE3 for each network of the DREAM4 *Multifactorial* challenge.** Left: PR curves. Right: ROC curves. Prec: Precision. FPR: False Positive Rate. TPR: True Positive Rate. The rankings of interactions were obtained using Random Forests with $K = \sqrt{p-1}$, where p is the number of genes, and growing $T = 1000$ trees.

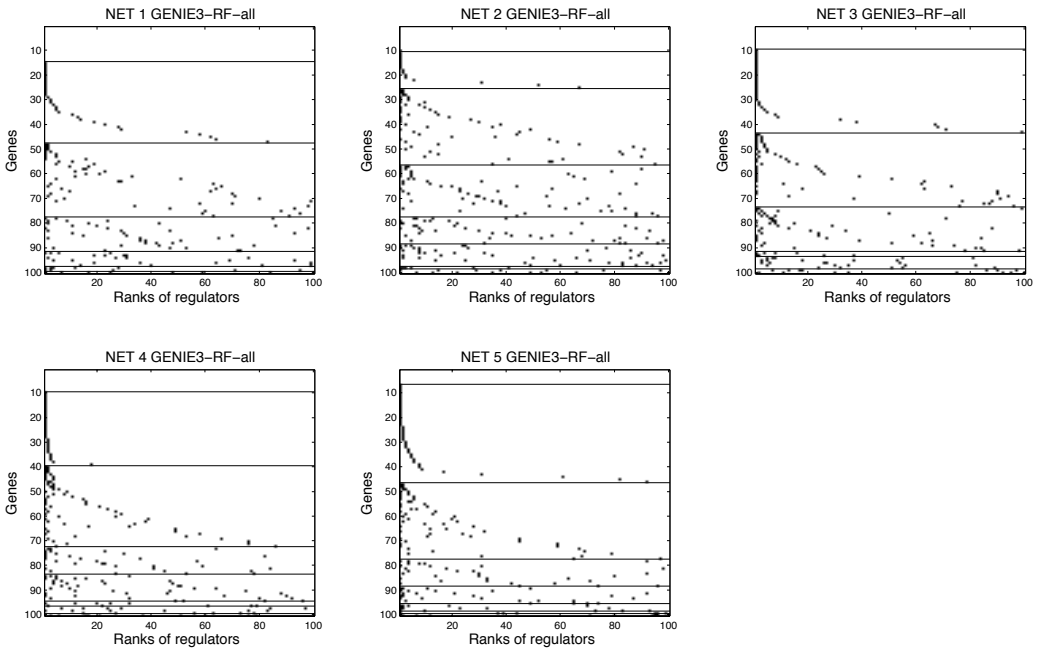


Figure B.2: **Ranking of the true regulators of each gene on DREAM4 networks.** Each row in a figure corresponds to a gene. Dots in each row represent the positions in the Random Forests ranking of the regulators of this gene. Genes are ordered on the y-axis according to their number of regulators in the gold standard network; those having the same number of regulators are grouped inside a horizontal block. Inside each block, genes are ordered according to the median rank of their regulators. The rankings of interactions were obtained using Random Forests with $K = p - 1$, where p is the number of genes, and growing $T = 1000$ trees.

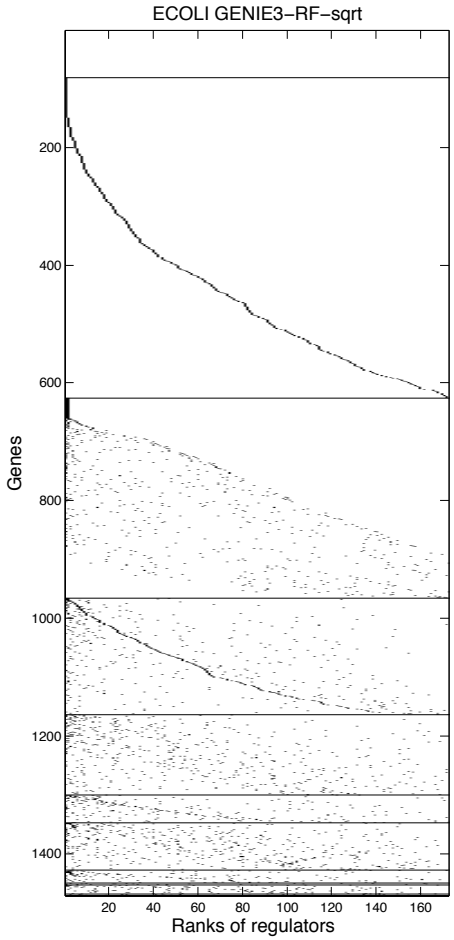


Figure B.3: **Ranking of the true regulators of each gene on the *E. coli* network.** Each row in a figure corresponds to a gene. Dots in each row represent the positions in the Random Forests ranking of the regulators of this gene. Genes are ordered on the y-axis according to their number of regulators in the gold standard network; those having the same number of regulators are grouped inside a horizontal block. Inside each block, genes are ordered according to the median rank of their regulators. The ranking of interactions was obtained using Random Forests with $K = \sqrt{n_{TF}}$, where n_{TF} is the number of known transcription factors, and growing $T = 1000$ trees.

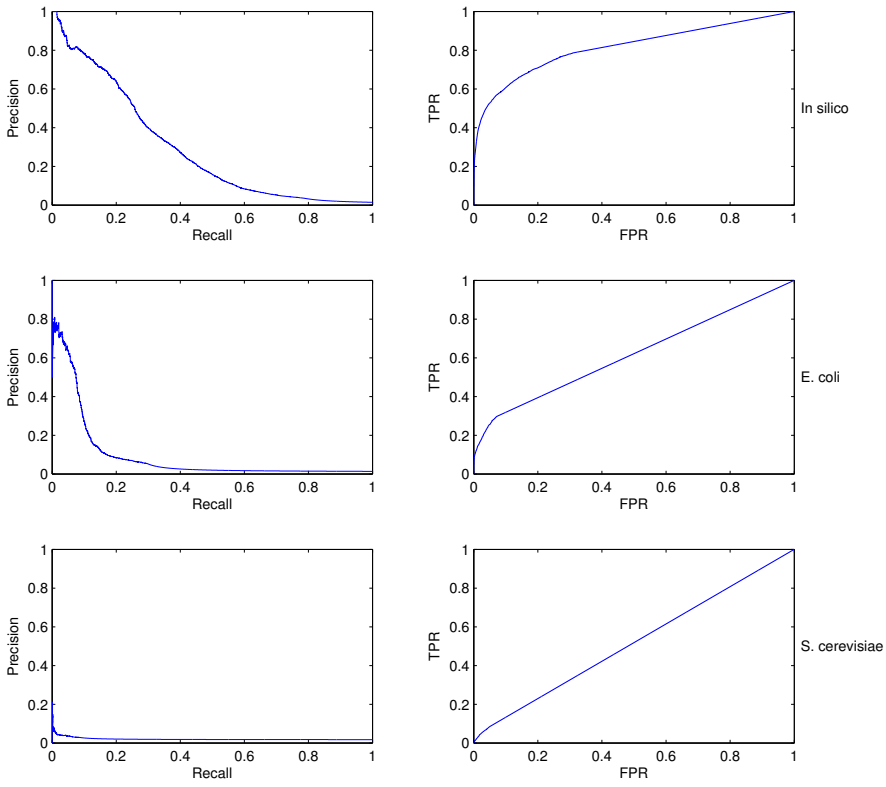


Figure B.4: **PR and ROC curves of GENIE3 for each network of the DREAM5 *Network Inference* challenge.** Left: PR curves. Right: ROC curves. FPR: False Positive Rate. TPR: True Positive Rate. The rankings of interactions were obtained using Random Forests with $K = \sqrt{n_{TF}}$, where n_{TF} is the number of potential transcription factors, and growing $T = 1000$ trees.

C

Extensions of GENIE3 - Supplementary figures

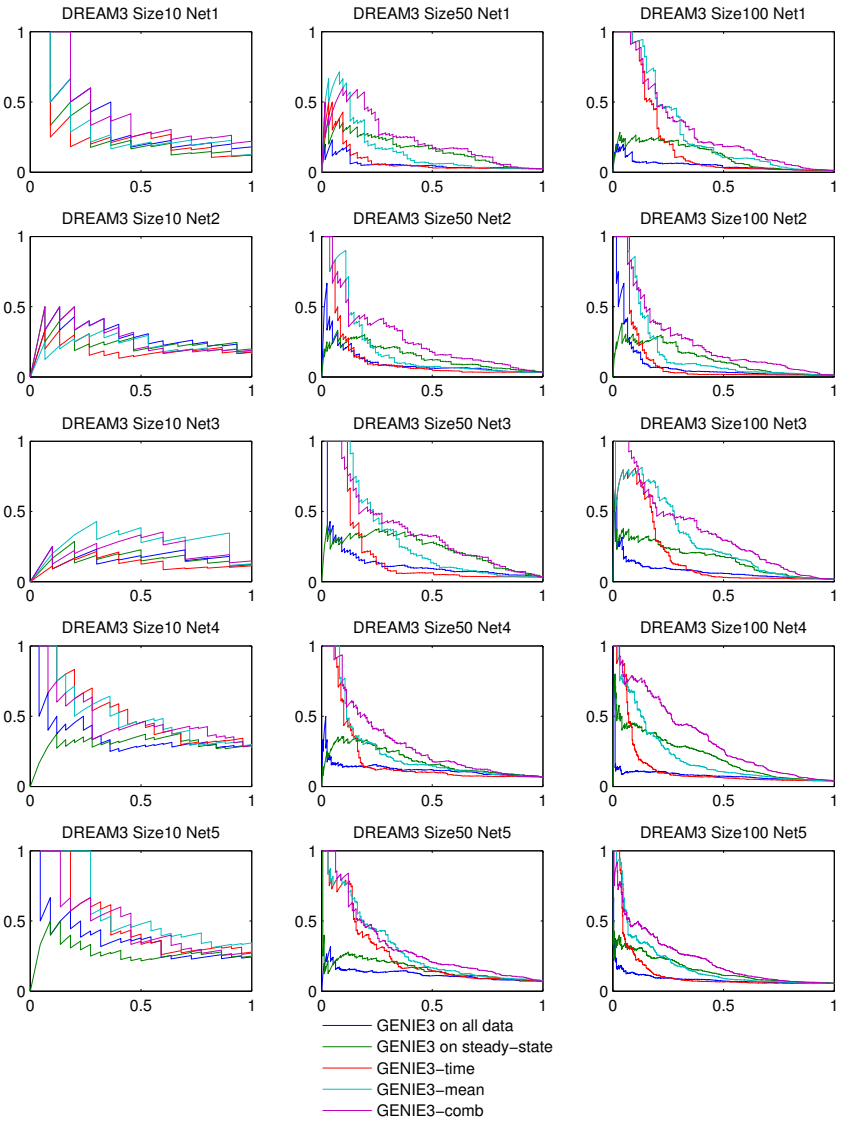


Figure C.1: **PR curves for the networks of the DREAM3 *In Silico Network* challenge.** On each plot, the x-axis is the recall and the y-axis is the precision. The PR curves are those obtained when applying respectively GENIE3 to the concatenation of steady-state and time series data, GENIE3 to the steady-state data, GENIE3-time, GENIE3-mean (with μ_S and μ_T fixed to their default values), and GENIE3-comb. The rankings of interactions were obtained using Random Forests with $K = p - 1$, where p is the number of genes, and growing $T = 1000$ trees.

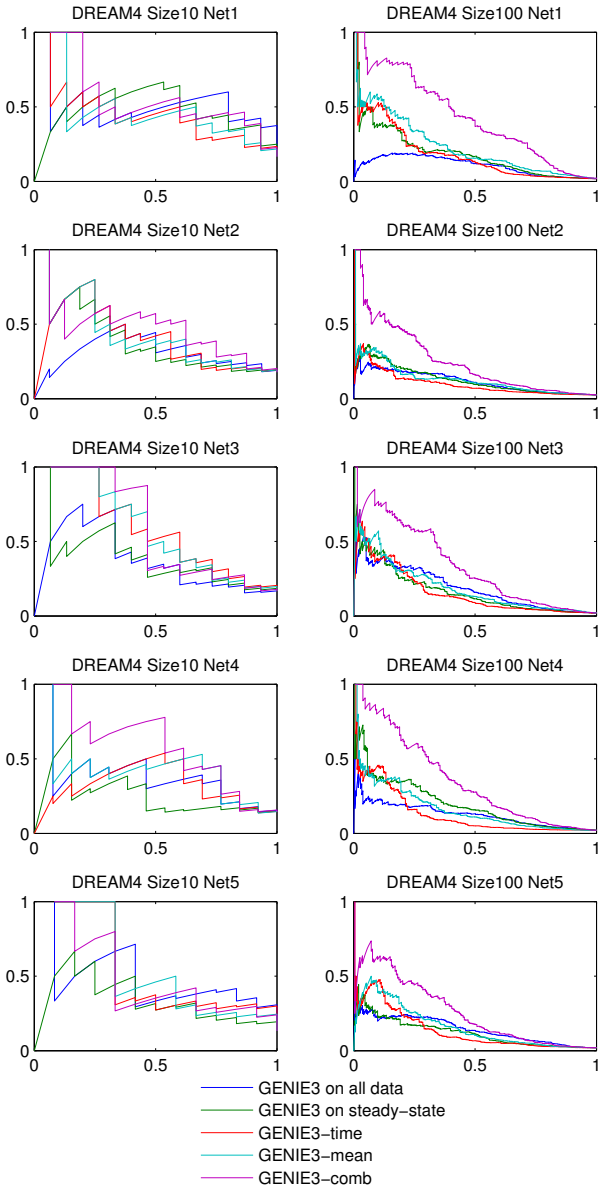


Figure C.2: **PR curves for the networks of the DREAM4 *In Silico Network* challenge.** On each plot, the x-axis is the recall and the y-axis is the precision. The PR curves are those obtained when applying respectively GENIE3 to the concatenation of steady-state and time series data, GENIE3 to the steady-state data, GENIE3-time, GENIE3-mean (with μ_S and μ_T fixed to their default values), and GENIE3-comb. The rankings of interactions were obtained using Random Forests with $K = p - 1$, where p is the number of genes, and growing $T = 1000$ trees.

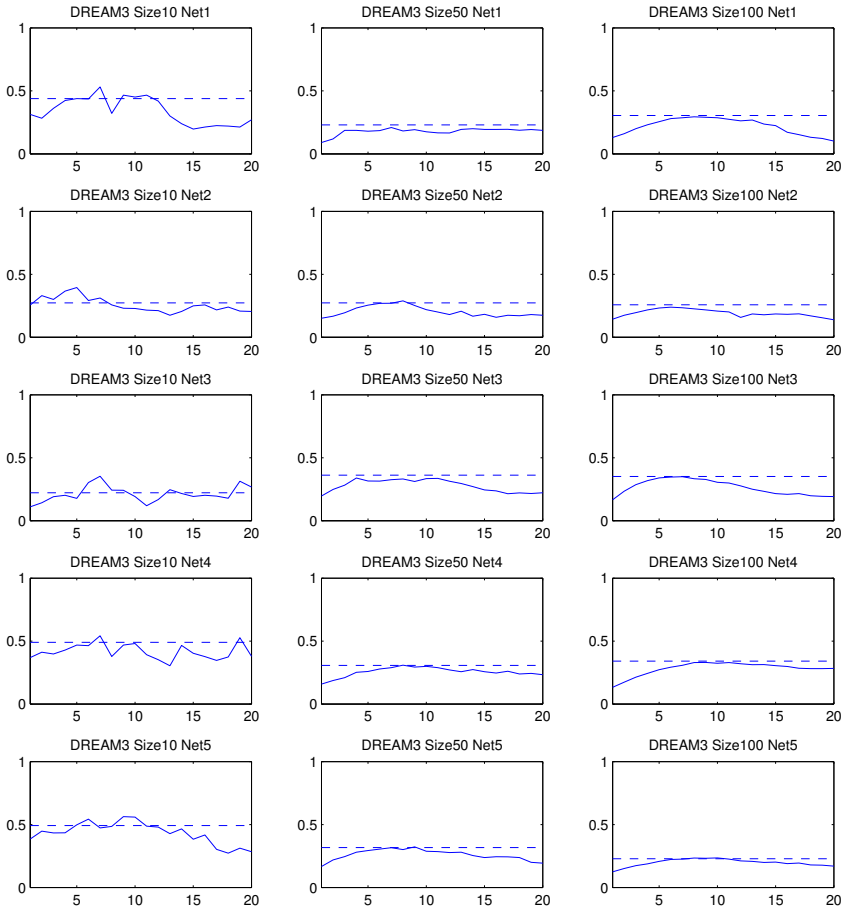


Figure C.3: **AUPRs of GENIE3-comb for the DREAM3 networks.** Each plot is related to one network and shows the AUPRs obtained with GENIE3-comb, for different values of the time horizon h, as well as the AUPR obtained when the weights of the regulatory links are averaged over all possible values of h (horizontal dashed line). On each plot, the x-axis is the time horizon h and the y-axis is the AUPR. The rankings of interactions were obtained using Random Forests with $K = p - 1$, where p is the number of genes, and growing $T = 1000$ trees.

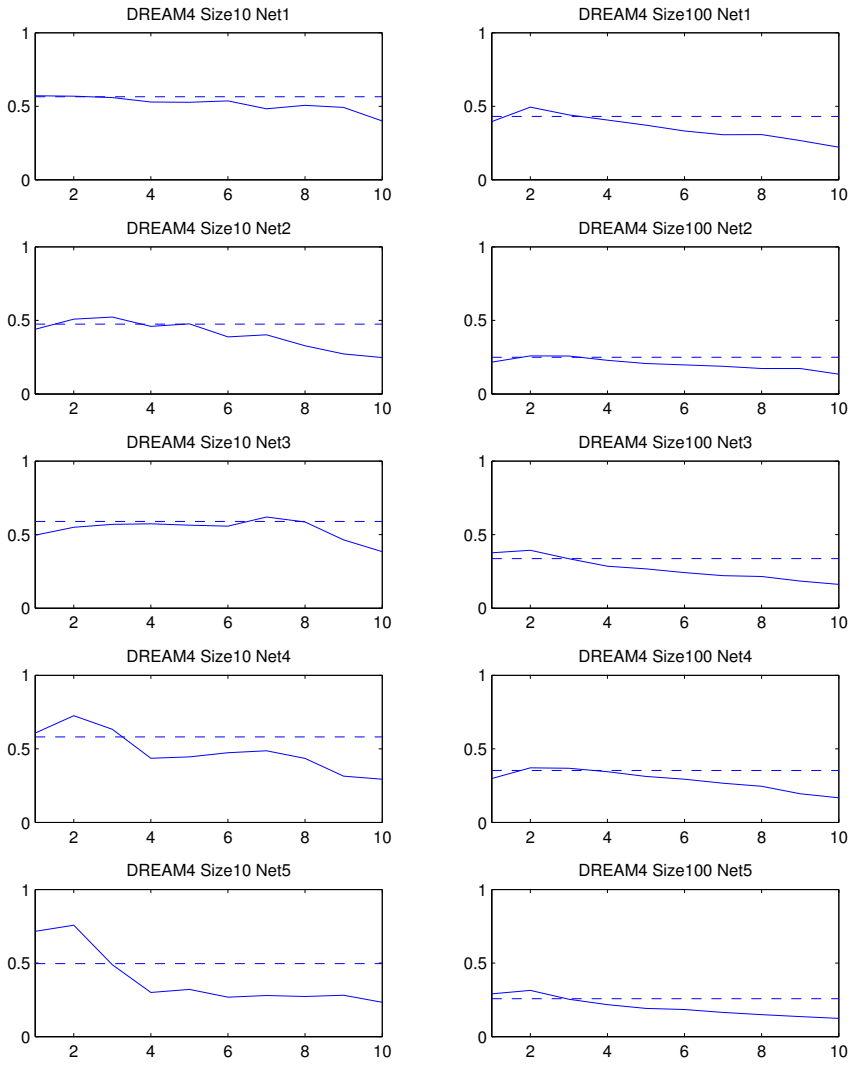


Figure C.4: **AUPRs of GENIE3-comb for the DREAM4 networks.** Each plot is related to one network and shows the AUPRs obtained with GENIE3-comb, for different values of the time horizon h, as well as the AUPR obtained when the weights of the regulatory links are averaged over all possible values of h (horizontal dashed line). On each plot, the x-axis is the time horizon h and the y-axis is the AUPR. The rankings of interactions were obtained using Random Forests with $K = p - 1$, where p is the number of genes, and growing $T = 1000$ trees.

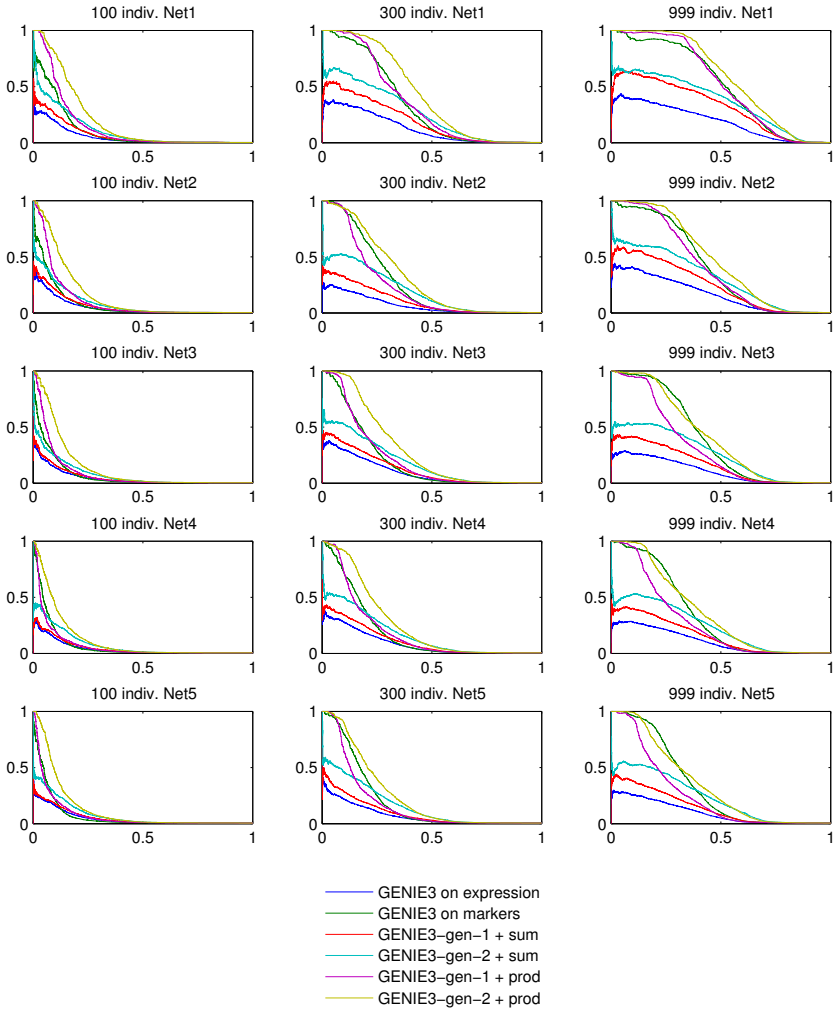


Figure C.5: **PR curves for the networks of the DREAM5 *Systems Genetics* challenge.** On each plot, the x-axis is the recall and the y-axis is the precision. GENIE3 on expression: original GENIE3 procedure applied to expression data. GENIE3 on markers: weight of edge $i \rightarrow j$ is the weight $w^m_{i,j}$ computed from f^m_j as defined in Equation (8.23). Sum: weight of edge $i \rightarrow j$ is the sum of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. Prod: weight of edge $i \rightarrow j$ is the product of the importance $w^e_{i,j}$ of the expression of gene i and the importance $w^m_{i,j}$ of the marker of gene i. The rankings of interactions were obtained using Random Forests with K fixed to the number of input variables, and growing $T = 1000$ trees.

Bibliography

Abdulrehman D, Monteiro PT, Teixeira MC, Mira NP, Lourenço AB, dos Santos SC, *et al.* (2011). YEASTRACT: providing a programmatic access to curated transcriptional regulatory associations in *Saccharomyces cerevisiae* through a web services interface. *Nucleic Acids Research* **39**: D136–D140.

Abeel T, Helleputte T, Van de Peer Y, Dupont P, Saeys Y (2010). Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics* **26**: 392–398.

Aliferis CF, Statnikov A, Tsamardinos I, Mani S, Koutsoukos XD (2010). Local causal and markov blanket induction for causal discovery and feature selection for classication Part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research* **11**: 171–234.

Ambroise C, Chiquet J, Matias C (2009). Inferring sparse gaussian graphical models with latent structure. *Electronic Journal of Statistics* **3**: 205–238.

Ambroise C, McLachlan GJ (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Science* **99**: 6562–6566.

Aten JE, Fuller TF, Lusis AJ, Horvath S (2008). Using genetic markers to orient the edges in quantitative trait networks: The NEO software. *BMC Systems Biology* **2**: 34.

Auliac C, Frouin V, Gidrol X, D'Alché-Buc F (2008). Evolutionary approaches for the reverse-engineering of gene regulatory networks: A study on a biologically realistic dataset. *BMC Bioinformatics* **9**: 91.

Auvray V, Wehenkel L (2002). On the construction of the inclusion boundary neighbourhood for markov equivalence classes of bayesian network structures. In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, pp. 26–35.

Balov N, Salzman P (2011). How to use catnet package. http://rwiki.sciviews.org/doku.php?id=packages:cran:catnet.

Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007). How to infer gene networks from expression profiles. *Mol Syst Biol* **3**: 78.

Bar-Joseph Z (2004). Analyzing time series gene expression data. *Bioinformatics* **20**: 2493–2503.

Barrett T, Troup DB, Wilhite SE, Ledoux P, Evangelista C, Kim IF, *et al.* (2011). NCBI GEO: archive for functional genomics data sets – 10 years on. *Nucleic Acids Research* **39**: D1005–D1010.

Belcastro V, Siciliano V, Gregoretti F, Mithbaokar P, Dharmalingam G, Berlingieri S, *et al.* (2011). Transcriptional gene network inference from a massive dataset elucidates transcriptome organization and gene function. *Nucleic Acids Research* .

Ben-Hur A, Ong CS, Sonnenburg S, Schölkopf B, Rätsch G, Lewitter F (2008). Support vector machines and kernels for computational biology. *PLoS Computational Biology* **4**: e1000173.

Benjamini Y, Hochberg Y (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)* **57**: 289–300.

Bing N, Hoeschele I (2005). Genetical genomics analysis of a yeast segregant population for transcription network inference. *Genetics* **170**: 533–542.

Bishop CM (2006). *Pattern Recognition and Machine Learning*. Springer.

Blockeel H, De Raedt L, Ramon J (1998). Top-down induction of clustering trees. In: Shavlik J (ed.), *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, pp. 55–63.

Bolouri H (2008). *Computational Modeling of Gene Regulatory Networks - a Primer*. Imperial College Press.

Bolstad BM, Irizarry RA, Åstrand M, Speed TP (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* **19**: 185–193.

Bonneau R, Reiss DJ, Shannon P, Facciotti M, Hood L, Baliga NS, *et al.* (2006). The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol* **7**: R36.

Bonnet E, Tatari M, Joshi A, Michoel T, Marchal K, Berx G, *et al.* (2010). Module network inference from a cancer gene expression data set identifies microRNA regulated modules. *PLoS ONE* **5**: e10162.

Bontempi G, Meyer PE (2010). Causal filter selection in microarray data. In: *Proceedings of the 27th International Conference on Machine Learning*. Omnipress, pp. 95–102.

Boser BE, Guyon I, Vapnik VN (1992). A training algorithm for optimal margin classifiers. In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, pp. 144–152.

Breiman L (1996). Bagging predictors. *Machine Learning* **24**: 123–140.

Breiman L (2001). Random forests. *Machine Learning* **45**: 5–32.

Breiman L, Friedman JH, Olsen RA, Stone CJ (1984). *Classification and Regression Trees*. Wadsworth International (California).

Brem RB, Kruglyak L (2005). The landscape of genetic complexity across 5,700 gene expression traits in yeast. *Proceedings of the National Academy of Sciences* **102**: 1572–1577.

Butte AJ, Kohane IS (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac Symp Biocomput* pp. 418–429.

Callow MJ, Dudoit S, Gong EL, Speed TP, Rubin EM (2000). Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research* **10**: 2022–2029.

Candès E, Tao T (2007). The dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics* **35**: 2313–2351.

Castelo R, Roverato A (2009). Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *Journal of Computational Biology* **16**: 213–227.

Chaibub Neto E, Ferrara CT, Attie AD, Yandell BS (2008). Inferring causal phenotype networks from segregating populations. *Genetics* **179**: 1089–1100.

Chang CC, Lin CJ (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* **2**: 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen LS, Emmert-Streib F, Storey JD (2007). Harnessing naturally randomized transcription to infer regulatory relationships among genes. *Genome Biology* **8**: R219.

Chickering DM, Heckerman D, Meek C (2004). Large-sample learning of bayesian networks is NP-hard. *Journal of Machine Learning Research* **5**: 1287–1330.

Conesa A, Nueda MJ, Ferrer A, Talón M (2006). maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* **22**: 1096–1102.

Cortes C, Vapnik V (1995). Support vector networks. *Machine Learning* **20**: 273–297.

Cover TM, Thomas JA (2006). *Elements of Information Theory 2nd Edition*. Wiley-Interscience.

Davis J, Goadrich M (2006). The relationship between Precision-Recall and ROC curves. In: *Proceedings of the 23rd international conference on Machine learning*. ACM, Pittsburgh, Pennsylvania, ICML '06, pp. 233–240.

de la Fuente A (2010). From 'differential expression' to 'differential networking' – identification of dysfunctional regulatory networks in diseases. *Trends in Genetics* **26**: 326–333.

De Smet R, Marchal K (2010). Advantages and limitations of current network inference methods. *Nat Rev Micro* **8**: 717–29.

D'Haeseleer P, Wen X, Fuhrman S, Somogyi R (1999). Linear modeling of mRNA expression levels during CNS development and injury. *Pacific Symposium on Biocomputing* **4**: 41–52.

Dhanasekaran S, Barrette T, Ghosh D, Shah R, Varambally S, Kurachi K, *et al.* (2001). Delineation of prognostic biomarkers in prostate cancer. *Nature* **412**: 822–826.

Eisen MB, Spellman PT, Brown PO, Botstein D (1998). Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A* **95**: 14863–14868.

Faith JJ, Driscoll ME, Fusaro VA, Cosgrove EJ, Hayete B, Juhn FS, *et al.* (2008). Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. *Nucleic Acids Research* **36 (Database issue)**: D866–70.

Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, *et al.* (2007a). Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biology* **5**: e8.

Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, *et al.* (2007b). Supplemental website for: Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. http://gardnerlab.bu.edu/data/PLoS_2007/.

Fisher RA (1925). *Statistical methods for research worker*. Oliver and Boyd, Edinburgh and London.

Friedman JH (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* **29**: 1189–1232.

Friedman N (2004). Inferring cellular networks using probabilistic graphical models. *Science* **303**: 799–805.

Friedman N, Linial M, Nachman I, Pe'er D (2000). Using bayesian networks to analyze expression data. *Journal of computational biology* **7**: 601–620.

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

Gama-Castro S, Jimnez-Jacinto V, Peralta-Gil M, Santos-Zavaleta A, Pealoza-Spinola MI, Contreras-Moreira B, *et al.* (2008). RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Research* **36 (Database issue)**: D120–4.

Gardner TS, di Bernardo D, Lorenz D, Collins JJ (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science* **301**: 102–105.

Gardner TS, Faith JJ (2005). Reverse-engineering transcription control networks. *Physics of Life Reviews* **2**: 65–88.

Ge Y, Dudoit S, Speed TP (2003). Resampling-based multiple testing for microarray data analysis. Tech. Rep. 633, Department of Statistics, University of California, Berkeley.

Ge Y, Sealfon SC, Speed TP (2008). Some step-down procedures controlling the false discovery rate under dependence. *Statistica Sinica* **18**: 881–904.

Geurts P, Ernst D, Wehenkel L (2006a). Extremely randomized trees. *Machine Learning* **36**: 3–42.

Geurts P, Fillet M, de Seny D, Meuwis MA, Malaise M, Merville MP, *et al.* (2005). Proteomic mass spectra classification using decision tree based ensemble methods. *Bioinformatics* **21**: 3138–3145.

Geurts P, Irrthum A, Wehenkel L (2009). Supervised learning with decision tree-based methods in computational and systems biology. *Mol Biosyst* **5**: 1593–605.

Geurts P, Wehenkel L, d'Alché Buc F (2006b). Kernelizing the output of tree-based methods. In: Cohen W, Moore A (eds.), *Proceedings of the 23rd International Conference on Machine Learning*. ACM, pp. 345–352.

Glasner JD, Liss P, Plunkett III G, Darling A, Prasad T, Rusch M, *et al.* (2003). ASAP, a systematic annotation package for community analysis of genomes. *Nucleic Acids Research* **31**: 147–151.

Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, *et al.* (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* **286**: 531–537.

Greenfield A, Madar A, Ostrer H, Bonneau R (2010). DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PLoS ONE* **5**: e13397.

Guyon I, Elisseeff A (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research* **3**: 1157–1182.

Guyon I, Weston J, Barnhill S, Vapnik V (2002). Gene selection for cancer classification using support vector machines. *Machine learning* **46**: 389–422.

Hastie T, Tibshirani R, Friedman J (2001). *The elements of statistical learning: data mining, inference and prediction*. Springer.

Haury AC, Gestraud P, Vert JP (2011a). The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. Technical report HAL-00559580.

Haury AC, Vera-Licona P, Mordelet F, Vert JP (2011b). Inferring gene regulatory networks with the stabilized lasso. In preparation.

Hecker M, Lambeck S, Toepfer S, van Someren E, Guthke R (2009). Gene regulatory network inference: Data integration in dynamic models – a review. *Biosystems* **96**: 86–103.

Hu Z, Killion PJ, Iyer VR (2007). Genetic reconstruction of a functional transcriptional regulatory network. *Nature Genetics* **39**: 683–687.

Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE* **5**: e12776.

Huynh-Thu VA, Saeys Y, Wehenkel L, Geurts P (2012). Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. *Bioinformatics* **28**: 1766–1774.

Huynh-Thu VA, Wehenkel L, Geurts P (2008). Exploiting tree-based variable importances to selectively identify relevant variables. *JMLR: Workshop and Conference proceedings* **4**: 60–73.

Hyafil L, Rivest RL (1976). Constructing optimal binary decision trees is NP-complete. *Information Processing Letters* **5**: 15–17.

Jansen RC (2003). Studying complex biological systems using multifactorial perturbation. *Nature Reviews Genetics* **4**: 145–151.

Jansen RC, Nap JP (2001). Genetical genomics: the added value from segregation. *TRENDS in Genetics* **17**: 388–391.

Karlebach G, Shamir R (2011). Constructing logical models of gene regulatory networks by integrating transcription factor-DNA interactions with expression data: an entropy based approach. Submitted.

Kauffman SA (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.

Kloosterman W, Plasterk R (2006). The diverse functions of microRNAs in animal development and disease. *Dev Cell* **11**: 441–450.

Kohavi R, John GH (1997). Wrappers for feature subset selection. *Artificial intelligence* **97**: 273–324.

Küffner R, Petri T, Tavakkolkhah P, Windhager L, Zimmer R (2011). Inferring gene regulatory networks by ANOVA. In preparation.

Küffner R, Petri T, Windhager L, Zimmer R (2010). Petri nets with fuzzy logic (PNFL): Reverse engineering and parametrization. *PLoS ONE* **5**: e12807.

Kulp DC, Jagalur M (2006). Causal inference of regulator-target pairs by gene mapping of expression phenotypes. *BMC Genomics* **7**: 125.

Lèbre S, Becq J, Devaux F, Stumpf MPH, Lelandais G (2010). Statistical inference of the time-varying structure of gene-regulation networks. *BMC Systems Biology* **4**: 130.

Lee WP, Tzou WS (2009). Computational methods for discovering gene networks from expression data. *Brief Bioinform* **10**: 408–423.

Li H, Lu L, Manly KF, Chesler EJ, Bao L, Wang J, *et al.* (2005). Inferring gene transcriptional modulatory relations: a genetical genomics approach. *Human Molecular Genetics* **14**: 1119–1125.

Li R, Tsaih SW, Shockley K, Stylianou IM, Wergedal J, Paigen B, *et al.* (2006). Structural model analysis of multiple quantitative traits. *PLoS Genetics* **2**: e114.

Listgarten J, Heckerman D (2007). Determining the number of non-spurious arcs in a learned DAG model: Investigation of a bayesian and a frequentist approach. In: *Proceedings of Twenty-Third Conference on Uncertainty in Artificial Intelligence*. UAI Press, pp. 251–258.

Liu B, de la Fuente A, Hoeschele I (2008). Gene network inference via structural equation modeling in genetical genomics experiments. *Genetics* **178**: 1763–1776.

Maathuis MH, Colombo D, Kalisch M, Bühlmann P (2010). Predicting causal effects in large-scale systems from observational data. *Nature Methods* **7**: 247–248.

MacIsaac KD, Wang T, Gordon DB, Gifford DK, Stormo GD, Fraenkel E (2006). An improved map of conserved regulatory sites for *Saccharomyces cerevisiae*. *BMC Bioinformatics* **7**: 113.

Mani S, Cooper GF (2004). A bayesian local causal discovery algorithm. In: *Proceedings of the World Congress on Medical Informatics*. pp. 731–735.

Marbach D, Costello JC, Küffner R, Vega N, Prill RJ, Camacho DM, *et al.* (2012). Wisdom of crowds for robust gene network inference. *Nature Methods* **9**: 796–804.

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010). Revealing strengths and weaknesses of methods for gene network

inference. *Proceedings of the National Academy of Sciences* **107**: 6286–6291.

Marbach D, Schaffter T, Mattiussi C, Floreano D (2009). Generating realistic *in silico* gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology* **16**: 229–239.

Margolin AA, Nemenman I, Basso K, Wiggins C, Stolovitzky G, Dalla Favera R, *et al.* (2006a). ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics* **7**: S7.

Margolin AA, Wang K, Lim WK, Kustagi M, Nemenman I, Califano A (2006b). Reverse engineering cellular networks. *Nature Protocols* **1**: 663–672.

Markowetz F, Spang R (2007). Inferring cellular networks–a review. *BMC Bioinformatics* **8 Suppl 6**: S5.

Maston GA, Evans SK, Green MR (2006). Transcriptional regulatory elements in the human genome. *Annu Rev Genomics Hum Genet* **7**: 29–59.

Meinshausen N, Bühlmann P (2006). High-dimensional graphs and variable selection with the lasso. *Ann Statist* **34**: 1436–1462.

Meinshausen N, Bühlmann P (2010). Stability selection. *J Royal Statist Soc B* **72**: 417–473.

Meyer PE, Kontos K, Lafitte F, Bontempi G (2007). Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J Bioinform Syst Biol* **2007**: 79879.

Meyer PE, Lafitte F, Bontempi G (2008). minet: A R/Bioconductor package for inferring large transcriptional networks using mutual information. *BMC Bioinformatics* **9**: 461.

Meyer PE, Lafitte F, Bontempi G (2009). *minet: Mutual Information Network Inference.* R package version 2.0.0. **URL:** *http://CRAN.R-project.org/package=minet*

Michoel T, De Smet R, Joshi A, Van de Peer Y, Marchal K (2009). Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. *BMC Systems Biology* **3**: 49.

Minka T, Winn J, Guiver J, Knowles D (2010). Infer.NET 2.4. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

Mordelet F, Vert JP (2008). SIRENE: supervised inference of regulatory networks. *Bioinformatics* **24**: i76–i82.

Opgen-Rhein R, Strimmer K (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology* **1**: 37.

Parkinson H, Kapushesky M, Kolesnikov N, Rustici G, Shojatalab M, Abeygunawardena N, *et al.* (2009). ArrayExpress update – from an archive of functional genomics experiments to the atlas of gene expression. *Nucleic Acids Research* **37**: D868–D872.

Pearl J (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Perrin BE, Ralaivola L, Mazurie A, Bottani S, Mallet J, d'Alche Buc F (2003). Gene networks inference using dynamic bayesian networks. *Bioinformatics* **19**: ii138–48.

Phuong TM, Lee D, Lee KH (2004). Regression trees for regulatory element identification. *Bioinformatics* **20**: 750–757.

Pinna A, Soranzo N, de la Fuente A (2010). From knockouts to networks: Establishing direct cause-effect relationships through graph analysis. *PLoS ONE* **5**: e12912.

Pinna A, Soranzo N, De Leo V, de la Fuente A (2011a). Elucidating transcriptional regulatory networks from heterogeneous gene expression compendia. In preparation.

Pinna A, Soranzo N, Hoeschele I, de la Fuente A (2011b). Simulating system genetics data with SysGenSIM. *Bioinformatics* **27**: 2459–2462.

Pournara I, Wernisch L (2004). Reconstruction of gene networks using bayesian learning and manipulation experiments. *Bioinformatics* **20**: 2934–2942.

Prill RJ, Marbach D, Saez-Rodriguez J, Sorger PK, Alexopoulos LG, Xue X, *et al.* (2010). Towards a rigorous assessment of systems biology models: The DREAM3 challenges. *PLoS ONE* **5**: e9202.

Qiu P, Gentles AJ, Plevritis SK (2009). Fast calculation of pairwise mutual information for gene regulatory network reconstruction. *Computer Methods and Programs in Biomedicine* **94**: 177–180.

Rakotomamonjy A (2003). Variable selection using SVM-based criteria. *Journal of Machine Learning Research* **3**: 1357–1370.

Ruan J, Zhang W (2006). A bi-dimensional regression tree approach to the modeling of gene expression regulation. *Bioinformatics* **22**: 332–340.

Saeys Y, Inza I, Larranaga P (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics* **23**: 2507–2517.

Schadt E, Lamb J, Yang X, Zhu J, Edwards S, GuhaThakurta D, *et al.* (2005). An integrative genomics approach to infer causal associations between gene expression and disease. *Nature Genetics* **37**: 710–717.

Schäfer J, Opgen-Rhein R, , Strimmer K (2009). *GeneNet: Modeling and Inferring Gene Networks*. R package version 1.2.4. **URL:** *http://CRAN.R-project.org/package=GeneNet*

Schäfer J, Opgen-Rhein R, Strimmer K (2006). Reverse engineering genetic networks using the GeneNet package. *R News* **6/5**: 50–53.

Schäfer J, Strimmer K (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology* **4**: 1175.

Schaffter T, Marbach D, Floreano D (2011). GeneNetWeaver: *In silico* benchmark generation and performance profiling of network inference methods. *Bioinformatics* **27**: 2263–2270.

Schmitt Jr WA, Raab RM, Stephanopoulos G (2004). Elucidation of gene interaction networks through time-lagged correlation analysis of transcriptional data. *Genome Research* **14**: 1654–1663.

Segal E, Pe'er D, Regev A, Koller D, Friedman N (2005). Learning module networks. *Journal of Machine Learning Research* **6**: 557–588.

Shawe-Taylor J, Cristianini N (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Sîrbu A, Ruskin HJ, Crane M (2011). Stages of gene regulatory network inference: the evolutionary algorithm role. *Evolutionary Algorithms* .

Song MJ, Lewis CK, Lance ER, Chester EJ, Yordanova RK, Langston MA, *et al.* (2009). Reconstructing generalized logical networks of transcriptional regulation in mouse brain from temporal gene expression data. *EURASIP Journal on Bioinformatics and Systems Biology* **2009**: 545176.

Statnikov A, Aliferis CF (2010). Analysis and computational dissection of molecular signature multiplicity. *PLoS Computational Biology* **6**: e1000790.

Stolovitzky G, Monroe D, Califano A (2007). Dialogue on reverse-engineering assessment and methods: The DREAM of high-throughput pathway inference. *Annals of the New York Academy of Sciences* **1115**: 11–22.

Stolovitzky G, Prill RJ, Califano A (2009). Lessons from the DREAM2 challenges. *Annals of the New York Academy of Sciences* **1158**: 159–95.

Stoppiglia H, Dreyfus G, Dubois R, Oussar Y (2003). Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research* **3**: 1399–1414.

Storey JD, Tibshirani R (2003). Statistical significance for genomewide studies. *Proc Natl Acad Sci U S A* **100**: 9440–9445.

Strobl C, Boulesteix AL, Zeileis A, Horthorn T (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics* **8**: 25.

The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nature Genetics* **25**: 25–9.

Tibshirani R (1996). Regression shrinkage and selection via the Lasso. *Journal of the royal statistical society series b-methodological* **58**: 267–288.

Tsamardinos I, Aliferis CF, Statnikov A (2003). Time and sample efficient discovery of markov blankets and direct causal relations. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 673–678.

Tuv E, Borisov A, Runger G, Torkkola K (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research* **10**: 1341–1366.

Tuv E, Borisov A, Torkkola K (2006). Feature selection using ensemble based ranking against artificial contrasts. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*. pp. 2181–2186.

van Someren EP, Vaes BLT, Steegenga WT, Sijbers AM, Dechering KJ, Reinders MJT (2006). Least absolute regression network analysis of the murine osteoblast differentiation network. *Bioinformatics* **22**: 477–484.

Vandel J, Mangin B, Vignes M, de Givry S (2010). Extended bayesian scores for reconstructing gene regulatory networks. In: *ECCS-10 workshop on graphical models for reasoning on biological systems: computational challenges*.

Vapnik VN (1995). *The nature of statistical learning theory*. Springer.

Wang Y, Klijn JG, Zhang Y, Sieuwerts AM, Look MP, Yang F, *et al.* (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet* **365**: 671 – 679.

Watkinson J, Liang KC, Wang X, Zheng T, Anastassiou D (2009). Inference of regulatory gene interactions from expression data using three-way mutual information. *Ann N Y Acad Sci* **1158**: 302–313.

Wehenkel L (1998). *Automatic learning techniques in power systems*. Kluwer Academic, Boston.

Werhli AV, Grzegorczyk M, Husmeier D (2006). Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. *Bioinformatics* **22**: 2523–2531.

Westfall PH, Young SS (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. John Wiley & Sons.

Xiao Y, Segal MR (2009). Identification of yeast transcriptional regulation networks using multivariate random forests. *PLoS Computational Biology* **5**: e1000414.

Yeung KY, Bumgarner RE, Raftery AE (2005). Bayesian model averaging: development of an improved multi-class, gene selection and classification tool for microarray data. *Bioinformatics* **21**: 2394–2402.

Yip KY, Alexander RP, Yan KK, Gerstein M (2010). Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PLoS ONE* **5**: e8121.

Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004). Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics* **20**: 3594–603.

Yuan M, Lin Y (2006). Model selection and estimation in regression with grouped varaibles. *J Royal Statist Soc B* **68**: 49–67.

Zhang C, Lu X, Zhang X (2006). Significance of gene ranking for classification of microarray samples. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **3**: 1–9.

Zhu J, Lum PY, Lamb J, GuhaThakurta D, Edwards SW, Thieringer R, *et al.* (2004). An integrative genomics approach to the reconstruction of gene networks in segregating populations. *Cytogenetic and Genome Research* **105**: 363–374.