



Kernel methods for gene regulatory network inference

Arnaud Fouchet

► To cite this version:

Arnaud Fouchet. Kernel methods for gene regulatory network inference. Bioinformatics [q-bio.QM]. Université d'Evry-Val-d'Essonne, 2014. English. NNT : 2014EVRY0058 . tel-01804286

HAL Id: tel-01804286

<https://hal.science/tel-01804286>

Submitted on 31 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ D'ÉVRY VAL D'ESSONNE
ÉCOLE DOCTORALE DES SCIENCES DE L'INGÉNIEUR

THÈSE

présentée par

Arnaud FOUCHET

pour obtenir le grade de

Docteur en sciences de l'Université d'Évry Val d'Essonne

Spécialité : Informatique

Kernel Methods for Gene Regulatory Network Inference

| | | |
|----------------------|----------------------|--------------------------|
| <i>Rapporteurs :</i> | Pierre GEURTS | - Université de Liège |
| | Alain RAKOTOMAMONJY | - Université de Rouen |
| <i>Examineurs :</i> | George MICHAILIDIS | - University of Michigan |
| | Céline ROUVEIROL | - Université Paris-Nord |
| | Marie SZAFRANSKI | - ENSIIE |
| <i>Directeurs :</i> | Jean-Marc DELOSME | - Université d'Évry |
| | Florence d'ALCHÉ-BUC | - Université d'Évry |

Contents

| | | |
|----------|--|----------|
| 1 | Context | 9 |
| 1.1 | Introduction | 9 |
| 1.2 | Gene Regulatory Networks - an introduction | 10 |
| 1.2.1 | How are genes expressed ? | 10 |
| 1.2.2 | How are genes differentially expressed ? | 11 |
| 1.2.3 | How is gene expression measured ? | 11 |
| 1.2.4 | GRN Inference | 12 |
| 1.2.5 | Available data | 13 |
| 1.3 | Machine Learning | 13 |
| 1.3.1 | Generalities on supervised Learning | 15 |
| | Bias-variance tradeoff | 17 |
| | Curse of dimensionality | 18 |
| | Overfitting | 18 |
| 1.3.2 | Linear model | 20 |
| | Ridge-regression | 21 |
| | LASSO | 21 |
| 1.3.3 | Kernel Methods | 22 |
| | Kernel ridge-regression | 25 |
| 1.4 | GRN Inference from gene expression data | 25 |
| 1.4.1 | Scoring methods | 26 |
| | Correlation | 26 |
| | Mutual information | 27 |
| | Covariance | 28 |
| | Z-score | 28 |
| 1.4.2 | Modeling | 29 |
| | Linear Methods | 29 |
| | Boolean Model | 30 |

| | | |
|----------|---|-----------|
| | Bayesian methods | 31 |
| | Other approaches | 33 |
| 1.5 | Assessment of GRNI methods | 34 |
| 1.6 | Problem formulation | 35 |
| I | Estimation of Partial Derivatives for Network Inference | 39 |
| 2 | Estimation of Partial Derivatives | 43 |
| 2.1 | Introduction | 43 |
| 2.2 | Consistent estimation of mean partial derivatives | 44 |
| 2.3 | Kernel methods for partial derivative estimation | 46 |
| 2.4 | Approximation of integrals | 48 |
| 2.5 | Numerical simulations | 49 |
| 2.6 | Conclusion | 50 |
| 3 | Network inference using partial derivatives of kernel-based models | 53 |
| 3.1 | Introduction | 53 |
| 3.2 | Partial derivatives of kernel-based models | 54 |
| 3.2.1 | Jacobian matrix estimation for GRN inference | 54 |
| 3.2.2 | Ensemble of randomized kernel-based models | 55 |
| | Learning from heterogeneous data types | 56 |
| | Dataset building | 57 |
| | Subspaces and subsamples building | 57 |
| 3.2.3 | p -value | 59 |
| 3.2.4 | Voting procedure | 60 |
| 3.3 | Other measures for feature importance in a model | 60 |
| 3.3.1 | Feature relevance | 61 |
| 3.3.2 | Sensitivity analysis | 62 |
| 3.4 | Hyperparameter selection | 63 |
| 3.5 | Experiments on small-scale networks | 63 |
| 3.5.1 | Data: DREAM3 Challenge | 63 |
| 3.5.2 | Performance evaluation | 64 |
| 3.5.3 | Performances without ensemble methods | 64 |
| 3.5.4 | Analysis of p -value | 65 |
| 3.5.5 | Performances on DREAM3 50-gene and 100-gene Net- works | 66 |
| 3.6 | Experiments on real and real-sized networks | 69 |

| | | |
|-------|---|----|
| 3.6.1 | Datasets | 69 |
| 3.6.2 | Using other GRN Inference methods: A consensus approach | 69 |
| 3.6.3 | Performance | 70 |
| 3.7 | Conclusion | 70 |

II Kernel Feature Weighting for Network Inference 77

4 Local Kernels and feature selection 81

| | | |
|-------|---|-----|
| 4.1 | Introduction | 81 |
| 4.2 | Feature weighting methods | 82 |
| 4.2.1 | <i>Local Kernel</i> Approach | 83 |
| 4.2.2 | Filtering methods | 86 |
| | Kernel target alignment (KA) | 86 |
| | RReliefF | 87 |
| 4.2.3 | Embedded methods | 91 |
| | Recursive Feature Elimination | 91 |
| | KerNel Iterative Feature Extraction - KNIFE | 93 |
| 4.3 | Hyper-parameter selection method | 94 |
| 4.3.1 | Cross-Validation (CV) | 94 |
| 4.3.2 | Stability | 95 |
| 4.3.3 | Block-Stability | 95 |
| 4.3.4 | Bayesian and Akaike Information Criterion | 96 |
| 4.4 | Experiments | 97 |
| 4.4.1 | Hyperparameter selection | 97 |
| 4.4.2 | Performances on size 10 networks | 98 |
| 4.4.3 | Performances on size 50 networks | 100 |
| 4.5 | Conclusion | 102 |

5 LocKNI: Local Kernel for Network Inference 105

| | | |
|-------|--|-----|
| 5.1 | Introduction | 105 |
| 5.2 | $\ell_2 - MKL$ and Prior Knowledge Incorporation | 106 |
| 5.2.1 | $\ell_2 - MKL$ | 106 |
| 5.2.2 | Prior knowledge incorporation | 107 |
| 5.3 | Ensemble method | 108 |
| 5.4 | Experimental results | 109 |
| 5.4.1 | Hyperparameter choice | 109 |
| 5.4.2 | Datasets | 109 |

| | | |
|-------|---|-----|
| 5.4.3 | Network inference by consensus of methods | 111 |
| 5.4.4 | Error analysis on Network N1 | 112 |
| 5.4.5 | Incorporation of prior knowledge | 113 |
| 5.5 | Conclusion | 116 |

| | |
|-------------------|------------|
| Appendices | 123 |
|-------------------|------------|

Introduction

“Mathematics is biology’s next microscope, only better; biology is mathematics’ next physics, only better” wrote Joel E. Cohen in 2004. The microscope has allowed many breakthroughs in biology. It has revealed a formerly invisible world, the existence of microorganisms such as microbes, and gave birth to cellular biology. Similarly, mathematics can shine a new light on biological problems. Computational methods can test many candidates to find drugs to block a virus such as HIV. Furthermore, mathematics offer tools to visualize and understand data. Conceiving such tools is an important challenge in today’s biology, as the quantity of data is booming. Firstly, scientists’ collaboration and internet databases have increased the amount of available data. Secondly, the price to acquire new data has dropped in many fields. For example, in 1990 started the Human Genome Project, which successfully sequenced a human genome after 13 years of work, mobilizing a hundred laboratories and costing three billion dollars. Today, companies consider sequencing someone’s genome for less than a thousand dollars, and within thirty hours. Molecular biology is switching from an era of data scarcity to an era of data abundance. One hopes that mathematics will reveal knowledge from large data, as the microscope revealed images of micro-organisms.

For mathematicians, biology raises many challenges. The dynamics of living systems are caused by interactions of many actors, on various scales. In most biological cases, mathematicians have very few data compared to the problem complexity, not to mention the noise surrounding those data. However, challenges in the empirical sciences have fostered discoveries in mathematics, in particular physics. As Joseph Fourier said, “L’étude approfondie de la nature est la source la plus féconde des découvertes mathématiques”¹. His study of heat propagation lead to modeling by partial derivative equations, Fourier analysis, Fourier series,

¹“Nature’s In-depth study is the most prolific source of mathematic discoveries”

etc. Similarly, the theory of distributions was inspired by challenges in physics.

The synergy between mathematics, computer science and biology has been highly visible in the last decades. Biological problems have found useful available tools in the two other fields. Yet, there is no doubt that collaboration can go much further, and can increase our understanding of biological mechanisms. In particular, much bioinformatic research is devoted to identify gene regulatory network (GRN). A GRN is a collection of DNA segments that interact indirectly with each other, interact with other substances in the cell and respond to the external environment. For example, in the presence of sugar, a yeast cell will turn on genes to process the sugar to alcohol. Yeast's GRN commanded this process, which is necessary to yeast's living as it made the yeast cell gain energy to multiply - incidentally, this process is necessary to men for wine-making. With the multiplication of data -in particular gene expression data-, scientists hope to discover the GRN's mechanisms. This knowledge would increase our understanding of the living system, and may help creating cure against specific pathology. Considering a living system's malfunction, one could see which genes are involved in that malfunction, then target some genes in the functional pathway to make the living system healthy.

Bioinformaticians have tackled the GRN inference (GRNI) problem, with the relevant mathematical tools. A common approach to GRNI in bioinformatic is reverse-engineering. Given gene expression data, bioinformaticians build a model that mimic the observed genes' behavior. Then, the bioinformaticians know what regulatory interactions happen in his model, and assume that the same interactions happen in the living system. Mathematical tools exist to model an observed system. In particular, many tools are provided by machine learning, a field of computer science and mathematics that sprang up with computational capacities and its applicability to many fields (image recognition, text categorization, spam detection). Nevertheless, gene modeling is a very demanding task: observed data contain much noise, gene have a nonlinear behavior, some actors of the GRN are not observed, the scale of the data demands computationally fast methods, data types are heterogeneous and, compared to the problem complexity, few data are available.

Among the relevant tools, kernel functions provide methods that are robust-to-noise, able to model any nonlinear behavior and computationally fast. Besides, kernel-based models are the result of minimizing a loss function. By adding

constraints or modifying the loss function, one can incorporate prior knowledge or external sources of information to reduce the problem's complexity. Nevertheless, kernel methods have scarcely been used for GRN Inference. The reason was mainly their lack of interpretability. In this work, I present two contributions to network inference using interpretable kernel methods. The two approaches originate from the same idea: a model is interpretable if the importance of input features for the output prediction can be weighted. In the first approach, I assume that partial derivatives of a perfect gene model should reflect the GRN. Indeed, if the concentration of a regulator gene changed, it should affect the concentration of the regulated gene. I demonstrate that partial derivatives of kernel-based models can consistently estimate the mean of partial derivatives of the ideal model. Thus I will interpret feature importance in a kernel-based model by its partial derivatives. In the second approach, I turn to multiple kernel-based models with multiple *local kernels*, each kernel being devoted to one feature. The idea is to find the optimal linear combination of these *local kernels* for the modeling of a target gene. The linear combination will weight the importance of each *local kernel*, thus of each feature. To get more stability and to tackle high dimension data in both approaches, ensemble of those models are built using a double scheme of randomization which provides a drastic improvement in terms of performance. Besides, this randomization scheme allow learning from heterogeneous data types. These two ways to interpret kernel models are then used to infer GRNs. On real and realistically simulated datasets, these methods show state-of-the-art performances: on some well-referenced datasets, they perform better than current state-of-the-art methods. They do under-perform on some other datasets. However, combining a kernel GRN inference method with other state-of-the-art GRN inference methods lead to substantial improvement over state-of-the-art.

This thesis is organized as follows: in Chapter 1, I introduce informally the GRN Inference problem, as well as the relevant ideas of machine learning and kernel methods, and describe current GRN inference methods. In the Chapter 2, I demonstrate that kernel methods consistently estimate any continuous linear form of the partial derivatives. In Chapter 3, I give several methods to interpret a kernel model, and observe that, on many realistically simulated data, using partial derivatives is the most efficient method for network inference. Then, I use this method on real and real-sized networks, showing complementary results to other GRNI methods. In Chapter 4, I describe prevalent kernel feature selection methods, and develop two methods: one based on multiple kernel learning,

another based on kernel alignment. On realistically simulated data, these two methods yield better results than other kernel feature selection methods. In Chapter 5, one of these two feature selection methods is used on real data, and show state-of-the-art performances. I suggest a modification of this method, to take into account prior information and specificities of biological data. Incorporation of reasonable prior knowledge greatly enhances the performances of this method.

Chapter 1

Context

1.1 Introduction

In the twentieth century, many breakthroughs have profoundly transformed the field of biology. Firstly, as early as 1930, new knowledge and technologies have allowed scientists to analyze living systems and phenomena on a molecular scale; most notably, the discovery of DNA. Secondly, starting in the nineties, new tools have been developed to measure gene or protein expression levels, giving scientists another type of data to analyze and understand cells and living systems. The cost of these tools has greatly decreased over the years. Also, scientific collaboration through data publication—in journals or in databases—greatly increased. As a result, the amount of available data augmented, creating new possibilities for scientific research.

Along with experimental design, *i.e.* choosing which experiments to perform, the study of large data is a considerable step toward new knowledge. Mathematics and computer science provide powerful tools for the exploration of large datasets. The inter-disciplinary field of bioinformatics has developed in this context. This is the study of mathematical and computational tools for the analysis of biological systems. Bioinformatics includes storing and visualizing biological data, and assessing relationships between phenomena. In particular, many bioinformatics methods have been developed for the problem of gene regulatory network inference (GRN inference, or GRNI). This thesis is devoted to developing new tools for GRNI.

This chapter will explain the matter and current approaches. In the first section, I introduce informally the gene regulatory network inference problem. Secondly,

I describe and discuss available data and the technology to produce these data. Thirdly, I introduce a central tool in bioinformatics: machine learning, and, in particular, kernel methods. Fourthly, I present current methodologies and discuss their most efficient version. In the fifth section, I mention key papers comparing GRNI methods, and give their conclusions.

1.2 Gene Regulatory Networks - an introduction

All of our cells contain the same genetic information, contained in our DNA. Nevertheless, skin cells are different from liver or kidney cells. These differences come about because different genes are expressed at high levels in different tissues. So, how are genes “expressed” ? The “central dogma of molecular biology” asserts that “DNA make RNA make protein” [1], as illustrated in Figure 1.1.

1.2.1 How are genes expressed ?

DNA consists of sequences of nucleobases A, T, G and C¹. These sequences hold all necessary information for the development and functioning of a living system. In particular, these sequences hold the system’s genes. Nowadays, many definitions exist for a “gene”. The following definition, from [2], is sufficient for this thesis: “*a gene is a locatable region of genomic sequence, corresponding to a unit of inheritance, which is associated with transcribed regions, regulatory regions, and or other functional sequence regions*”, with the following explanations:

- *a transcribed region* is a sequence of A, T, G and C that will be “copied” (transcribed) into messenger RNA. Most messenger RNAs are then “read” by ribosomes, which translates such a sequence into a sequence of amino acids, which forms the protein carrying out the function coded in the gene.
- *a regulatory region* is a segment of DNA capable of increasing or decreasing gene expression. A particular regulatory region, shared by almost all genes, is known as the promoter, which provides a position that is recognized by the transcription machinery when a gene is about to be transcribed and expressed. A gene can have more than one promoter, resulting in different RNAs.

¹Adenine, Thymine, Guanine and Cytosine

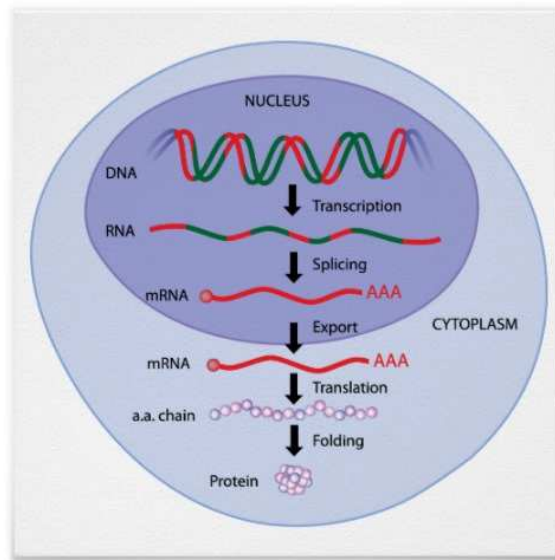


Figure 1.1: Molecular biology's central dogma

Only 1.5% of the human genome consists of protein-coding nucleobases. Some of the noncoding DNA is transcribed in microRNAs, which regulate gene expression, but the purpose of most of the noncoding DNA is yet unknown.

1.2.2 How are genes differentially expressed ?

Some of the proteins hold a gene regulatory function. They are called “*Transcription Factors*” (sometimes called sequence-specific DNA-binding factors). These proteins bind to specific DNA sequences, controlling gene expression, by promoting (activator) or blocking (repressor) the gene's transcription into mRNA. Thus, expression of specialized genes can regulate other genes, leading to a specialized cell.

1.2.3 How is gene expression measured ?

New technologies allow us to learn many aspects of the genome. In particular, gene expression can be evaluated through RNA sequencing, which gives a snapshot of RNA's presence and quantity in a given tissue at a given time. With mRNA

concentration, one has a measure of gene expression - even though other post transcriptional gene regulation events exist, such as RNA interference.

The main tool for RNA concentration measure consists in DNA microarrays. To measure an mRNA concentration, one uses a probe, which is a complementary sequence of a part of this mRNA. Each spot of a DNA microarray contains millions of copies of a probe. After extraction from a cell, mRNAs are spread over the array, where they bind to their specific complementary sequence. The array is then washed to remove unbound sequences. Then, the array is scanned with a laser. Each probe produces a fluorescent signal, whose intensity is linked to the number of bound mRNAs.

This method faces several noise sources, and noise reducing methods have been developed. For example, probes can "cross-hybridize", *i.e.* bind with the wrong target. To control cross-hybridization, some arrays pair probes that should work (Perfect Match, PM) with probes that should not (Mismatch MM). PM is perfectly complementary to the sequence of interest, and MM is the same as PM for all but one base. Further discussion of DNA microarrays gene expression measure problematics can be found in [3].

The reader must be aware of three main sources of noise in gene expression data. First, there is a noise inherent to observing a living system, especially on these scales. Second, probes have different binding affinities to their target mRNA. Thus, it is difficult to tell whether "gene A beats gene B in experiment 1", as opposed to "there is more gene A in experiment 1 than in experiment 2". Microarrays only produce relative measurements of gene expression. Thirdly, the number of genes is well above the number of available experiments. This is known as the "*large p small N*" framework, where p is the problem dimension (here, the number of genes) and N is the number of data samples (here, the number of available experiments). This is a very challenging framework, with uncertainties on the results.

1.2.4 GRN Inference

The gene regulatory network represents gene interactions at the transcription level, *i.e.* which gene regulates which gene. Knowledge of a living system's GRN has many potential applications. It would enhance our understanding of the system. As a possible consequence, it could help the development of cures. Considering a pathology resulting from a system's malfunction, one could see which genes are involved in that malfunction, then target some genes in the functional pathway to enhance or inhibit this function, and hopefully cure the pathology. This explains

why GRN inference has become a major challenge in biology.

GRN can be described by a graph whose nodes are genes and in which a directed edge from node i to node j means that gene i regulates gene j . This is a simplified view, which does not take into account several key players such as microRNAs. In fact, a regulatory interaction involves DNA, mRNAs and proteins. All these elements are merged into one element, the gene representative, see Figure 1.2. Let us call A the adjacency matrix of this graph : $a_{ij} = 1$ if j regulates i , 0 otherwise. Network inference usually refers to the estimation of this matrix A .

1.2.5 Available data

A p -gene system is observed through mRNA expression levels in steady-state and time-series data. Steady-state data consist in the concentration of each of the p genes in a particular cell. Noting \mathbf{x}_k the vector containing the p concentrations in experiment k , x_k^i is the concentration of the i^{th} gene in the k^{th} experiment. The different types of measures that may be available are: steady-state measures on unperturbed individuals, called wild-type data; mRNA concentration of perturbed individuals, on which a gene's mRNA concentration has been increased or diminished; knock-out experiments, in which a gene is knocked out, hence its expression is null. These data are called "perturbed" or "perturbational data".

Time-series, measures of genes' mRNA concentrations through time, are also available. The vector $\mathbf{x}(t, u)$ groups concentrations of all genes at time t in experiment u . Usually, the individual has received an exogenous perturbation or signal at time $t = 0$, such as a heat step, presence of a molecule like glucose, or exposure to radiations. A finite number of observations is available; the system is observed at times t_1, \dots, t_k .

1.3 Machine Learning

Machine learning aims at extracting information or knowledge from data. GRN inference takes special interest in supervised learning, a branch of machine learning identifying the link between input variables \mathbf{X} and a response or output variable \mathbf{Y} . Supervised learning is useful in many domains (computer vision, medical imaging, bioinformatics, etc.), and has fostered the development of many performant algorithms. In particular, Gaussian kernel methods have valuable properties in theory (consistency) and in practice (efficiency, robustness-to-noise), thus are good candidates for GRN Inference.

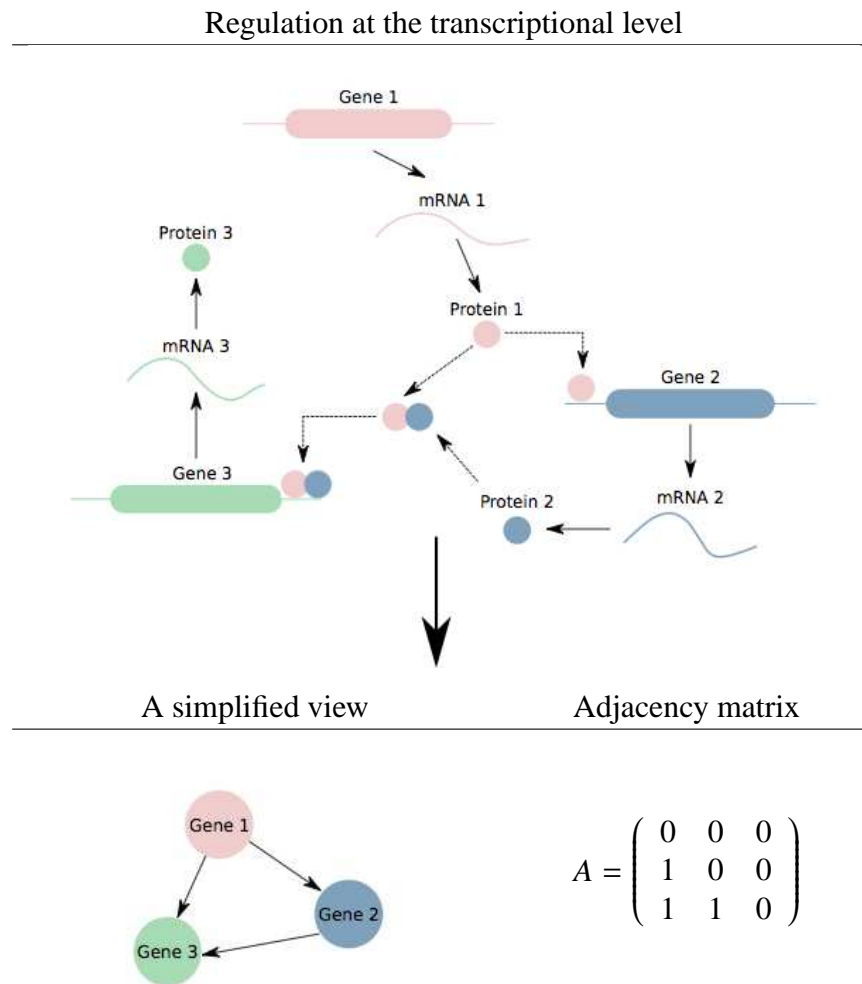


Figure 1.2: Regulatory interactions are simplified into a network representing only genes. Image modified from [4]

In this section, I first present generalities of supervised learning: notations, goals and difficulties. Next, I describe linear methods, and the associated solutions to supervised learning problems. Finally, I present kernel methods, which can be seen as an extension of linear methods.

1.3.1 Generalities on supervised Learning

The objective of supervised learning is to identify the link between some input variables $\mathbf{x} = (x^1, \dots, x^p)'$ and an output variable, or response y . The random variables $(\mathbf{X}, \mathbf{Y}) \in (\mathcal{X}, \mathcal{Y})$ are assumed to be distributed according to a distribution \mathcal{P} . In all the problems of this thesis, \mathcal{X} is a finite p -dimensional space, \mathcal{Y} is either a subset of \mathbb{R} (regression problem), the discrete set $\{-1, 1\}$ (binary classification) or a discrete set $\{1, \dots, m\}$ (multi-class classification). A bold upper case letter, e.g \mathbf{X} , denotes a random variable, a lower case letter, such as y or x^j , a scalar, and a bold lower case denotes a column vector, e.g. $\mathbf{x} = (x^1, \dots, x^p)^T$. Variables (\mathbf{X}, \mathbf{Y}) are linked through a function f

$$\mathbf{Y} \sim f(\mathbf{X}) + \epsilon \quad (1.1)$$

with ϵ a zero-mean noise. N realizations of (\mathbf{X}, \mathbf{Y}) are observed and usually assumed independent and identically distributed (*i.i.d.*). They form the learning set $\mathcal{S} = (\mathbf{x}_i, y_i)_{i=1 \dots N}$. Given a prediction function $g \in \mathcal{F}(\mathcal{X}, \mathcal{Y})$, let $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$ be a loss function, quantifying the cost for predicting $g(\mathbf{x}_i)$ instead of y_i (see examples of classic loss functions in Figure 1.3). The *risk* of a function g , $R(g)$, is defined as the expected loss:

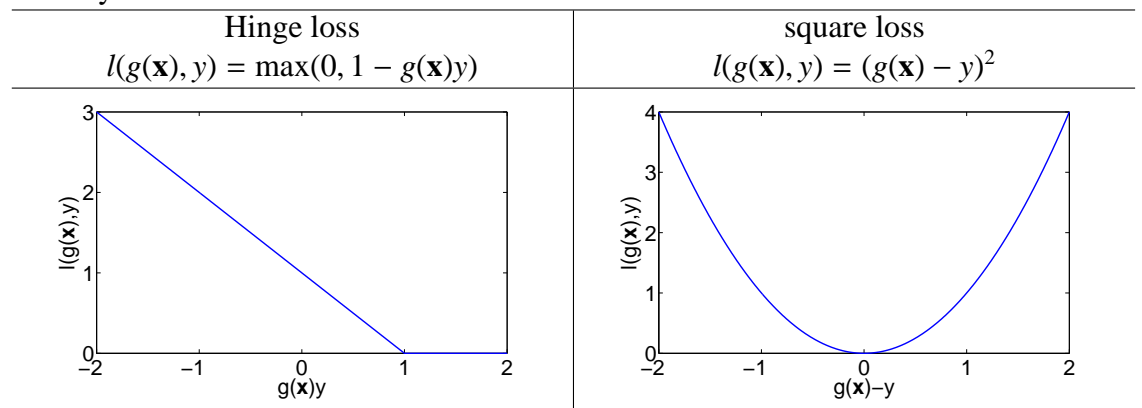
$$R(g) = \mathbb{E}_{\mathcal{P}}(l(g(\mathbf{X}), \mathbf{Y})) \quad (1.2)$$

The goal is to find function f^* , from \mathcal{H} the set of admissible functions, that minimizes the risk:

$$f^* = \arg \min_{g \in \mathcal{H}} R(g) \quad (1.3)$$

Unfortunately, the function $R(\cdot)$ is not known. However, the training set \mathcal{S} is available so that the *empirical risk*, *i.e.* the mean of the loss function of the

Binary classification



Regression

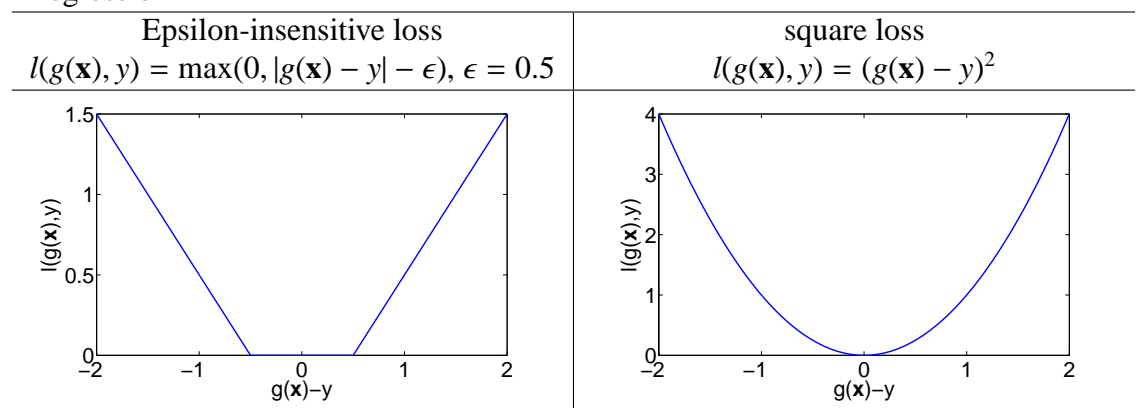


Figure 1.3: Examples of loss functions

training data, is minimized instead:

$$R_{emp}(g) = \frac{1}{N} \sum_{i=1}^N l(g(\mathbf{x}_i), y_i) \quad (1.4)$$

$$\hat{f} = \arg \min_{g \in \mathcal{H}} R_{emp}(g) \quad (1.5)$$

Bias-variance tradeoff

The choice of the functional space \mathcal{H} is critical for the performance of the empirical risk minimization approach. If one chooses too “*small*” a functional space, there could be no function in \mathcal{H} that approximate correctly the true function f and the true risk will be high. The model has *bias*.

On the other hand, the empirical risk minimizer \hat{f} depends of the learning set \mathcal{S} . If the functional space is too “*big*”, data may be insufficient, and the learned function \hat{f} may be very different of the true risk minimizer f^* . Those two sources of error can be decomposed this way:

$$\text{Model error} = R(\hat{f}) - R(f) , \quad (1.6)$$

$$= \underbrace{R(\hat{f}) - R(f^*)}_{(1)} + \underbrace{R(f^*) - R(f)}_{(2)} . \quad (1.7)$$

(1) is the error from learning from \mathcal{S} and not infinite data. (2) is the error from choosing the functional space \mathcal{H} .

To find the best model, one has to find the right balance between the bias and the variance of the model. This is quantified by the bias-variance tradeoff. Considering that the loss function is the square loss, the expected true risk of \hat{f} according to the distribution of the learning set \mathcal{S} :

$$\mathbb{E}_{\mathcal{S}}(R(\hat{f})) = \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(y - \hat{f}(\mathbf{x}))^2 \right] \right] \quad (1.8)$$

T.Hastie and R.Tibshirani [5] prove that this can be decomposed in the following manner—proof in appendix, page 125:

$$\mathbb{E}_{\mathcal{S}}(R(\hat{f})) = \mathbb{E}_{\mathcal{P}, \mathcal{S}}[\epsilon^2] + \mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}))^2 \right] + \mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f^*(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \quad (1.9)$$

$$= \sigma^2 + (\text{bias})^2 + (\text{variance}) \quad (1.10)$$

with σ^2 be the variance of the noise, $\sigma^2 = \mathbb{E}_{\mathcal{P}}[\epsilon^2]$. This exposes all the sources of error in supervised learning. σ^2 is the best error one can have, when trying to predict y from \mathbf{x} . The bias, $(\text{bias})^2 = \mathbb{E}_{\mathcal{P}}[(f(\mathbf{x}) - f^*(\mathbf{x}))^2]$, expresses the error from choosing functional space \mathcal{H} . This term can be high if \mathcal{H} is poorly chosen. Finally, if \mathcal{H} is too big, the learned function \hat{f} will greatly vary with the learning set \mathcal{S} , resulting in a third source of error.

Curse of dimensionality

The expression “*curse of dimensionality*”, coined by Richard Bellman (1961), illustrates the problem of the dramatic increase of the volume of data with the increase of dimension. I give an illustrative example in Figure 1.4, where I have drawn 64 points with coordinates in the interval $[0, 1]$. In one dimension, the whole space is well occupied. In two dimensions, the data become more sparse. In three dimensions, the space $[0, 1]^3$ is clearly under-sampled. To have, in the 10-dimensional space $[0, 1]^{10}$, the same space coverage as for 100 points in a 1-dimensional space, we would need 10^{20} points [6].

In this context, it becomes clear that one cannot identify the best prediction function in a “*big*” functional space \mathcal{H} by minimizing only the empirical risk.

Overfitting

Related to the “*curse of dimensionality*” and the bias-variance tradeoff is the danger of overfitting. Figure 1.5 shows a toy classification problem. Noting $\mathcal{U}(\mathcal{X})$ the uniform distribution in \mathcal{X} , the test and training data have been drawn according to the following distributions:

$$\mathbf{X} \sim \mathcal{U}([0, 1]^2) \quad (1.11)$$

$$\mathbf{Y} \sim \text{sign}(-3 + x_2 + 3x_1 + 2 \times \mathcal{U}([0, 1])) . \quad (1.12)$$

The linear model makes mistakes on the training set, but captures the true classification solution. The nonlinear classification function has perfect prediction on the training set, but would give worse results on the test set. To avoid overfitting, an approach is to regularize the prediction function. One minimizes the empirical risk under a constraint on the complexity of the function:

$$\min_{g \in \mathcal{H}} R_{\text{emp}}(g) \quad (1.13)$$

$$\text{s.t. } \Omega(g) \leq T \quad (1.14)$$

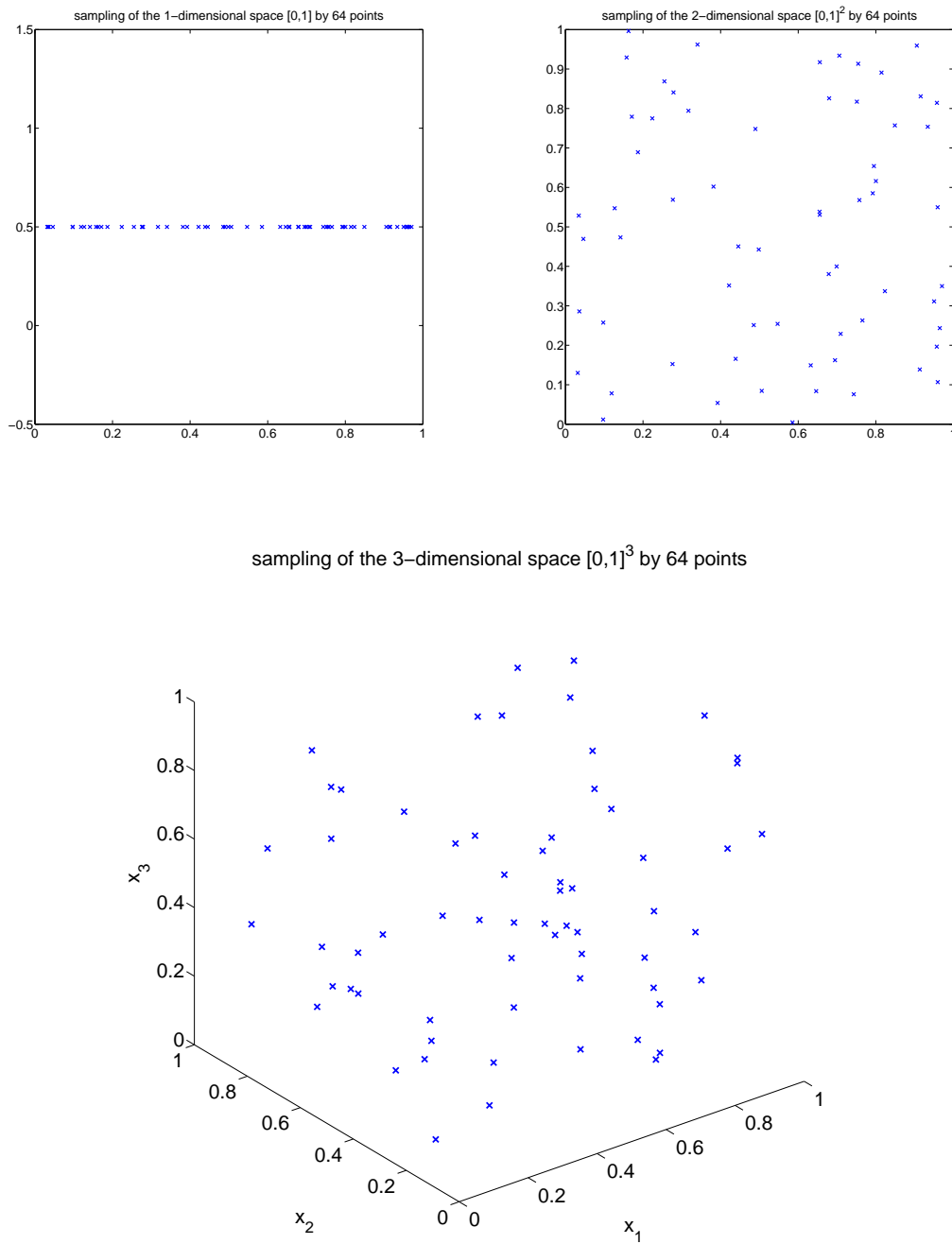


Figure 1.4: Illustration of the curse of dimensionality. As dimension increases, data become more sparse.

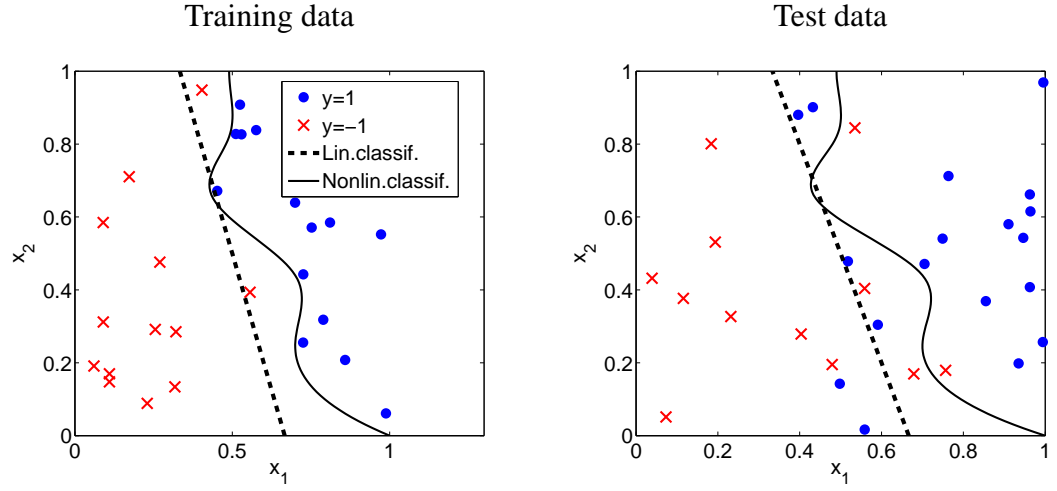


Figure 1.5: Example of overfitting by a nonlinear predictor.

with $\Omega : \mathcal{H} \rightarrow \mathbb{R}_+$ a convex penalty on the complexity of the function. This can be equivalently formulated as

$$\min_{g \in \mathcal{H}} R_{emp}(g) + \lambda \Omega(g) . \quad (1.15)$$

Even with linear functions, one faces the curse of dimensionality as well as the possibility of overfitting if the vector \mathbf{x} is high-dimensional, which is typically the case in bioinformatics. I will now show efficient regularized methods.

1.3.2 Linear model

The first functional space to be studied is the space of linear functions:

$$f_{lin}(\mathbf{x}) = \sum_{m=1}^p a_m x_m \quad (1.16)$$

$$= \langle \mathbf{a}, \mathbf{x} \rangle \quad (1.17)$$

The choice of the loss function $l(.,.)$ and of the complexity penalty $\Omega(.)$ will define various algorithms. For regression problems, the most common methods are ridge-regression (or Tikhonov regression) and LASSO. Each method insures smoothness in a different way.

Ridge-regression

In the case of regression, a popular loss function is the quadratic loss

$$l_{quad}(y, f_{lin}(\mathbf{x})) = (y - f_{lin}(\mathbf{x}))^2 . \quad (1.18)$$

When adding the ℓ_2 -norm as regularization term $\Omega(\cdot)$, the problem is known as ridge regression:

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_i (y_i - \langle \mathbf{a}, \mathbf{x}_i \rangle)^2 + \lambda \|\mathbf{a}\|_2^2 . \quad (1.19)$$

It admits a closed-form solution:

$$\hat{\mathbf{a}} = (X^T X + \lambda I_p)^{-1} (X^T) \mathbf{y} , \quad (1.20)$$

where I_p is the identity matrix of order p . The matrix $(X^T X + \lambda I_p)$ is always invertible; the problem is well-posed. As can be seen in equation (1.21),

$$|f_{lin}(\mathbf{x}_1) - f_{lin}(\mathbf{x}_2)| = |\langle \mathbf{a}, \mathbf{x}_1 - \mathbf{x}_2 \rangle| \leq \|\mathbf{a}\|_2 \|\mathbf{x}_1 - \mathbf{x}_2\|_2 , \quad (1.21)$$

the norm of \mathbf{a} bounds the ratio between the distance of two input points $(\mathbf{x}_1, \mathbf{x}_2)$ and their image through the function f_{lin} , $(f_{lin}(\mathbf{x}_1), f_{lin}(\mathbf{x}_2))$. The function f_{lin} is smooth in a Lipschitzian way.

LASSO

By choosing the ℓ_1 -norm as a penalty term, the problem is known as LASSO ("Least Absolute Shrinkage and Selection Operator"):

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_i (y_i - \langle \mathbf{a}, \mathbf{x}_i \rangle)^2 + \lambda \|\mathbf{a}\|_1 . \quad (1.22)$$

With this regularization term, the solution \mathbf{a} is sparse, *i.e.* many of its entries will be zeroes. Figure 1.6 gives a geometric intuition why. Level sets, or contour lines, are the curves such that the empirical risk has the same value for all values of \mathbf{a}

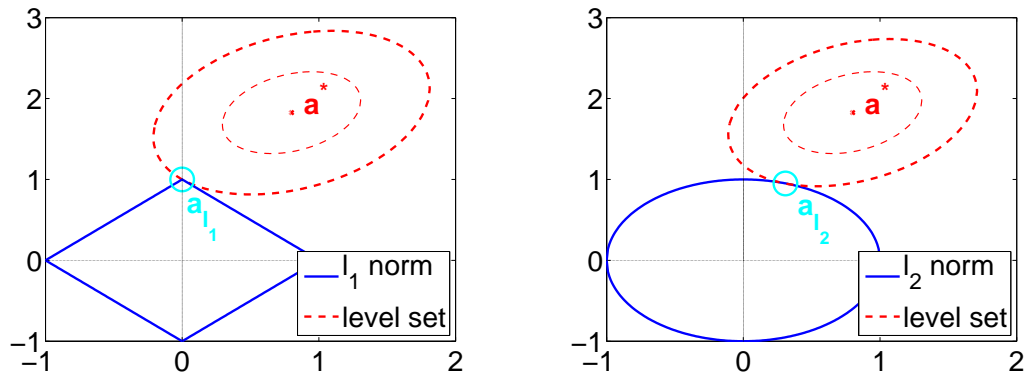


Figure 1.6: Illustration that ℓ_1 -norm induces sparsity. \mathbf{a}^* is the minimizer of the empirical risk. \mathbf{a}_ℓ is the minimizer of ℓ -regularized empirical risk. \mathbf{a}_ℓ will be the first intersection between a level set of the empirical risk and the level set of the regularization norm. Due to its form, the ℓ_1 -norm level set intersects the empirical risk level set at a vertex, where many features are null.

along them. At their center is \mathbf{a}^* the minimizer of the quadratic loss function on the learning set. The regularization can be seen as a second term to minimize, but also as a constraint. In the figure, it is interpreted as a constraint. One optimizes \mathbf{a} to minimize the empirical risk given that \mathbf{a} 's ℓ -norm is below a threshold T . The optimum, noted \mathbf{a}_ℓ , will be found at the first intersection between a contour line and the set of admissible values. With the ℓ_1 -norm, the intersection will be found at a vertex of the set of admissible points. The vertices of admissible points with ℓ_1 -norm are points with some zero components.

The property of giving sparse solutions makes the LASSO model interpretable. The few non-zero entries are the only relevant ones. This interpretability allowed LASSO to be used as a feature selection method and also to become a popular regression technique.

Unfortunately, many regression problems are not solved with sufficient accuracy by linear methods. I present in the following section an extension of those methods to nonlinear models.

1.3.3 Kernel Methods

To extend the linear model methods to non-linear separation, one approach is to map the data in a high dimensional space. In the example in Figure 1.7, taken

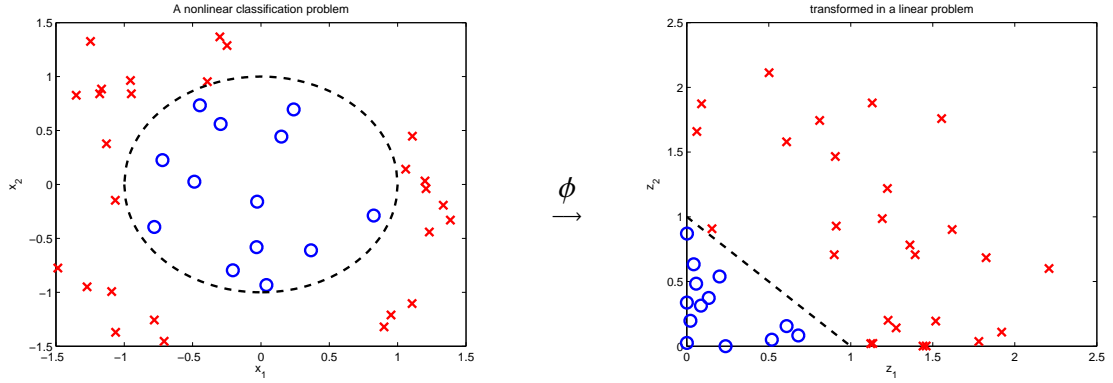


Figure 1.7: Illustration of a nonlinear problem transformed in a linear problem by the appropriate map ϕ

from [7], a linear model cannot correctly classify the data. But if we map the data according to $\phi : (x_1, x_2) \mapsto (z_1, z_2) = (x_1^2, x_2^2)$, a perfect linear classifier is available.

Clearly better classifiers can be built by mapping data in the proper space, but selecting the map ϕ for each problem would be a very hard task. This difficulty has been overcome by using kernel functions and the so-called kernel trick, explained below, without calculating the map or the higher dimensional space.

A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a kernel if it has the following properties:

- *symmetric*: for all $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
- *positive semi-definite*: for all $N \in \mathbb{N}$, for all $(\mathbf{x}_i)_{i=1 \dots N} \in \mathcal{X}^N$, for all $(\alpha_i)_{i=1 \dots N} \in \mathbb{R}^N$, $\sum_{i,j=1}^N \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

Aronszjan showed, in [8], that $k(\mathbf{x}, \mathbf{x}')$ is a scalar product in a particular functional space $\mathcal{H} \subset \mathcal{F}(\mathcal{X}, \mathbb{R})$. In particular, there exists a mapping ϕ such as $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$. The functional space \mathcal{H} has the property that, for any $f \in \mathcal{H}$, for any $\mathbf{x} \in \mathcal{X}$, $\langle f, k(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = f(\mathbf{x})$; \mathcal{H} is called a Reproducing Kernel Hilbert Space (RKHS). The best function in functional space \mathcal{H} is searched:

$$\hat{f} = \arg \min_{g \in \mathcal{H}} R_{emp}(g) + \lambda \Omega(g) \quad (1.23)$$

$$\hat{f}(\mathbf{x}) = \langle \hat{\mathbf{a}}, \phi(\mathbf{x}) \rangle_{\mathcal{H}} \quad (1.24)$$

Using as regularization term an increasing function of $\|g\|_{\mathcal{H}}$, the form of \hat{f} can be deduced from the representer theorem.

Theorem 1. Representer theorem. *Let X be a nonempty set and k a positive-definite real-valued kernel on $X \times X$ with corresponding reproducing kernel Hilbert space \mathcal{H} . Given a training sample $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in (X \times \mathbb{R})^N$, a strictly monotonically increasing real-valued function $\Omega : [0, \infty) \rightarrow \mathbb{R}$, and an arbitrary empirical risk function $L : (\mathbb{R} \times \mathbb{R})^N \rightarrow \mathbb{R} \cup \{\infty\}$, then for any $f^* \in \mathcal{H}$ satisfying*

$$\hat{f} = \arg \min_{f \in \mathcal{H}} L((y_1, f(\mathbf{x}_1)), \dots, (y_N, f(\mathbf{x}_N))) + \Omega(\|f\|) \quad (1.25)$$

\hat{f} admits a representation of the form:

$$\hat{f}(\cdot) = \sum_{i=1}^N \alpha_i k(\cdot, \mathbf{x}_i) \quad (1.26)$$

with $\alpha_i \in \mathbb{R}$ for all $1 \leq i \leq N$

This is the general formulation of the representer theorem, as given in [9]. Proof is given in appendix, page 126.

The kernel trick [10] consists in calculating all scalar products through the kernel k , and never computing the feature map ϕ

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad (1.27)$$

$$\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} \quad (1.28)$$

$$= \sum_{i,j=1}^N \alpha_i \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}} \quad (1.29)$$

$$= \sum_{i,j=1}^N \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (1.30)$$

With a convex loss l and a strictly increasing convex function Ω , solving the problem

$$\min_{f \in \mathcal{H}} \sum_i l(y_i, f(\mathbf{x}_i)) + \lambda \Omega(\|f\|_{\mathcal{H}}) \quad (1.31)$$

reduces to a convex problem in α , which is quickly solved by gradient descent. In the following work, I will mainly use the kernel-ridge regression.

Kernel ridge-regression

We now have tools to learn non-linear functions with regularization enforcing smoothness. I will use them with the quadratic loss and a regularization term

$$\min_{f \in \mathcal{H}} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 . \quad (1.32)$$

Noting K the Gram matrix ($K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$) and using the representer theorem, this amounts to solving:

$$\min_{\alpha \in \mathbb{R}^N} (\mathbf{y} - K\alpha)^T (\mathbf{y} - K\alpha) + \lambda \alpha^T K \alpha , \quad (1.33)$$

α admits the closed form solution

$$\hat{\alpha} = (K + \lambda I_N)^{-1} \mathbf{y} . \quad (1.34)$$

Note that minimizing the norm of function f give smoother functions in a Lipschitzian way, similarly to the ridge-regression case

$$|f(\mathbf{x}) - f(\mathbf{x}')| = | \langle f, \phi(\mathbf{x}) - \phi(\mathbf{x}') \rangle_{\mathcal{H}} | \quad (\mathcal{H} \text{ RKHS}) \quad (1.35)$$

$$\leq \|f\|_{\mathcal{H}} \times \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|_{\mathcal{H}} \quad (\text{Cauchy-Schwarz}) . \quad (1.36)$$

1.4 GRN Inference from gene expression data

Mathematical tools and, in particular, machine learning tools have been used to infer a GRN from gene expression data, without knowledge of existing interactions. Although they may use supervised learning tools, they are unsupervised methods as they try to predict the existence or absence of edges, but they were not trained with examples of existing or non-existing edges. They can be decomposed into two groups: *scoring* methods, where one evaluates the dependency of one gene to another through a pre-defined metric, and *modeling* methods, or reverse-engineering methods, where one creates a model \hat{f} with parameters $\hat{\theta}$ such as this model mimics observed expression data. The model is built so \hat{f} or $\hat{\theta}$ can be interpreted to infer a gene regulatory network. I give below a list of existing methods with short technical details.

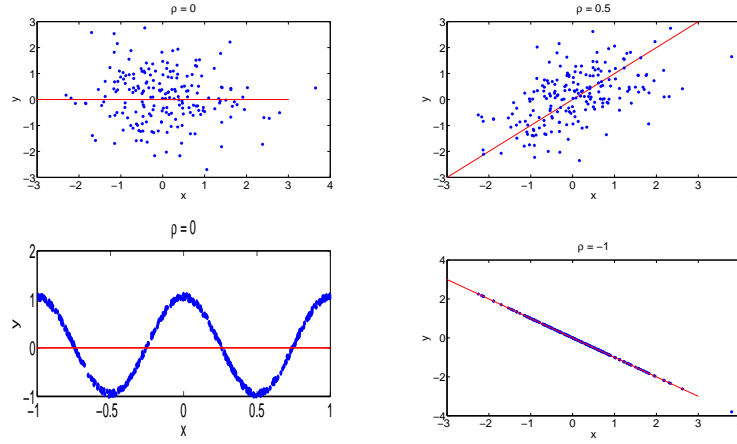


Figure 1.8: Examples of data (x, y) and their Pearson correlation. Each observed couple (x, y) is represented by a blue dot. The regression $y = ax$ is drawn in red. Pearson's correlation can miss nonlinear dependencies

1.4.1 Scoring methods

Statistic dependency measures are used in these methods to evaluate the likeliness of a link between gene i and gene j . Adaptations of these measures have also been suggested to be more relevant in the biological context, or to remove spurious interactions.

Correlation

Correlation studies the intensity of a link between two variables. Many forms of correlation measures exist (Spearman's rank correlation [11], Kendall's τ [12], Goodman and Kruskal's γ [13]); the following methods use Pearson's correlation ρ . With two variables x and y , ρ 's value is contained in $[-1, 1]$, and evaluate the linear dependency of those two variables. With \bar{x} (resp. \bar{y}) the mean of x (resp. y), (x_i, y_i) observed examples of the couple (x, y) , ρ is the cosine between the two centered vectors \mathbf{x} and \mathbf{y} and is defined as:

$$\rho(x, y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(\sum_i (x_i - \bar{x})^2)(\sum_i (y_i - \bar{y})^2)}} \quad (1.37)$$

Figure 1.8 shows some examples, including $\rho = 0$ for uncorrelated variables and $\rho = -1$ for $y = ax + b$ if $a < 0$. If gene j excites (resp. inhibits) gene i , i will

have low (resp. high) concentration when j has low concentration, and they will have a highly positive (resp. negative) correlation. If the correlation is superior to a threshold, a link between these two genes is inferred [14].

For more plausible biological networks, some researchers have proposed to post-process the obtained adjacency matrix to produce *scale-free* networks², see the *Symmetric-N* [21] or the *Asymmetric-N* algorithm [22].

Kuffner et al. used a non-linear correlation coefficient, called η^2 [23], for GRN inference. They improved this measure in [24] by giving additional weights to perturbed data.

Mutual information

Another metric is the mutual information I between a pair of random variables

$$I(x^i, x^j) = H(x^i) - H(x^i|x^j) = \sum_{x^i, x^j} p(x^i, x^j) \log \left(\frac{p(x^i, x^j)}{p(x^i)p(x^j)} \right)$$

Where $H(x^i)$ is the entropy of random variable x^i , measuring its unpredictability, and $H(x^i|x^j)$ is the entropy of x^i once x^j is observed, hence is always lower than $H(x^i)$. If gene j regulates gene i , its concentration will be responsible of much of gene i 's concentration, thus x^i should be much more predictable once x^j is observed, and the mutual information will be high. See the work on **Relevance Network** [25] for a first use of mutual information. Many network inference algorithm use mutual information, such as **ARACNE** [26].

Faith et al. suggest the **CLR** algorithm [27]. The mutual information between all gene pair (i, j) is computed. Then, authors can compute the p_i distribution of mutual information with gene i . For a randomly chosen gene z , the mutual information between gene i and z follows the distribution p_i , $MI(i, z) \sim p_i$. For a gene j , Faith et al. compute Z_{ij} (resp. Z_{ji}), the unlikeliness of observing $MI(i, j)$ given p_i (resp. p_j). They score the likelihood of an interaction with the following equation:

$$s(i, j) = \sqrt{Z_{ij}^2 + Z_{ji}^2} . \quad (1.38)$$

An edge between genes i and j is inferred if $s(i, j)$ is superior to a threshold.

²A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. This property is observed on many large-scale networks, such as the scientific collaboration network [15] or biological networks [16, 17, 18]. Note that the scale-free assumption for biological networks is contested in [19, 20]

Covariance

For the same reason as for correlation, genes interacting together should have relatively high absolute value for their covariance. Opgen-Rhein and Strimmer [28] suggest an estimation of covariance through an efficient multi-dimensional estimator: the James-Stein estimator [29]. **Gaussian graphical models** (GGM), also called concentration graph, covariance selection or Markov random field model [30, 31, 32], estimate the concentration matrix $\Omega = \Sigma^{-1}$, *i.e.* the inverse of the covariance matrix Σ , assuming all gene expression levels are distributed according to a multivariate normal distribution. The partial correlation between gene i and j knowing all other genes is related to the concentration matrix according to:

$$\rho(i, j | \{1, \dots, p\} \setminus \{i, j\}) = \frac{-\Omega_{ij}}{\sqrt{\Omega_{ii}\Omega_{jj}}} \quad (1.39)$$

In particular, Ω_{ij} should be equal to 0 if two genes do not interact, even if a third gene z interacts with both of them. Various methods exist to evaluate Ω or to test if the observed value of Ω_{ij} is sufficiently high to infer an edge between gene i and j . See [33, 34, 35, 36].

Z-score

Static, dynamic and perturbed data are available. In particular, an experiment with gene j perturbed should highlight a change of behavior in every gene that gene j regulates. Pinna et al. [37] measure the Z-scores: for each gene i , they calculate from wild type experiments its mean μ_i and standard deviation σ_i . Then the Z-score is computed as:

$$Z_{ij} = \frac{G_i^j - \mu_i}{\sigma_i} \quad (1.40)$$

with G_i^j the concentration of gene i in the experiments where gene j has been knocked out. Z_{ij} measures how “unlikely” the state G_i^j would be reached by chance if j did not regulate i . The authors also used a method to remove indirect edges. This last approach performed very well on synthetic data, where perturbed data is available for all genes.

1.4.2 Modeling

Another approach to gene regulatory network inference is through modeling of the genes. The parameter θ is sought such that:

$$x^i = f_{stat}^i(\mathbf{x}^{-i}, \theta) + \epsilon \quad \text{for static data ,} \quad (1.41)$$

$$\mathbf{x}(t + \tau) = f_{dyn}(\mathbf{x}(t), \theta) + \epsilon_t \quad \text{for time-series ,} \quad (1.42)$$

Where the function f is in a particular functional space. The network is learned from the parameter vector θ . I describe here the tried functional space and the procedure to identify θ .

Linear Methods

A linear dynamical model has the form:

$$x^i(t + \tau) = \sum_{j=1}^p \theta_{ij} x^j(t) \quad (1.43)$$

Where θ_{ij} gives the importance of gene j for the prediction of gene i . One would usually add constraints to obtain a sparse θ , and consequently have an interpretable model. Thus, the LASSO [38] was used in [39]. In biological systems, there may be a delay between the presence of a gene i and its influence on another gene j . Shojaie and Michailidis [40] build linear models depending on the system's state at several previous times ($\mathbf{x}(t), \mathbf{x}(t-1), \dots$).

Besides, linear methods offer a framework to learn from heterogeneous data. Given dynamic data, illustrating a dynamics $\mathbf{x}(t + \tau) = f_{dyn}(\mathbf{x}(t))$, and steady-state data, illustrating $x^i = f_{stat}^i(\mathbf{x}^{-i})$, one can learn a linear regression for each model θ_{dyn} and θ_{stat} that result from the same network. One wants θ_{dyn} and θ_{stat} to be null on genes not interacting with gene i , thus $(\theta_{dyn})_j = 0$ if $(\theta_{stat})_j = 0$, and reciprocally. Marbach et al. [41] used group-LASSO [42] and bootstrap samples; J.Chiquet et al. [43] also wanted $(\theta_{dyn})_j$ and $(\theta_{stat})_j$ to share the same sign, which they obtained through cooperative-LASSO. Another approach [44] consists in learning θ separately on each dataset, then find $\hat{\theta}$, the linear regression that would be closer to all the found regressions.

The state-of-the-art method TIGRESS [45] uses linear models combined with *stability selection* [46]. Given a target gene i and a potential regulator j , TIGRESS

wants to evaluate the probability that θ_{ij} is non-null. To this purpose, it uses LARS regressors [47], that iteratively select the L most important regulators for the linear model. TIGRESS runs R LARS regressor on modified data: the expression levels of candidate transcription factors are multiplied by a random number in the interval $[r, 1]$, and the model is trained on a random subsample of the data. TIGRESS now has, for all candidate transcription factor j and all $\ell \in [1, L]$, the frequency $F(i, j, \ell)$ with which the TF j was selected by LARS in the top ℓ features to predict the expression of gene i . By selecting a too small value for ℓ , many TF would have 0 score; by selecting too large a value for ℓ , several TF may have the same probability 1. Thus, TIGRESS will infer an edge from j to i with an averaged version of $F(i, j, \ell)$, with the score $s(i, j)$:

$$s(i, j) = \frac{1}{L} \sum_{\ell=1}^L F(i, j, \ell) . \quad (1.44)$$

$s(i, j)$ corresponds to the area under the curve of the probability to select transcription factor j for target gene i when selecting ℓ regulators, with ℓ varying from 1 to L . An edge from gene j to gene i is inferred if $s(i, j)$ is superior to a threshold.

Boolean Model

In the Boolean framework, a gene can only be active (1) or inactive (0). Let \tilde{x}^i be the discrete state of gene i , $\tilde{x}^i = x^i \geq \theta^i$, $\tilde{\mathbf{x}} = (\tilde{x}^1, \dots, \tilde{x}^p)'$. In the dynamical case, given a system state at time t , $\tilde{\mathbf{x}}(t)$, there exists a Boolean function giving the state at time $t + 1$. This function can be represented by a truth table, or a wiring diagram, as shown in Figure 1.9. For a p -gene system, there exists 2^{2^p} possible Boolean functions for each output. Finding the optimal Boolean function is a combinatorial problem, and its resolution by brute-force algorithms is computationally too expensive. Heuristics are used to simplify the problem. Akutsu et al. [48] limit the number of regulators that one gene can have to a constant K , thus drastically reducing the search space. In the REVEAL algorithm [49], S.Liang et al. start by identifying gene j which has highest mutual information with output gene i . They iteratively add to the list of regulators of gene i the gene which most improve the Boolean function.

Boolean networks focus on generic network behavior rather than quantitative biochemical details. Despite a simplification of the input variables, Boolean networks succeeded in retrieving meaningful biological information [50, 51, 52] and are able to model the behavior of biological systems [53]. Particularly, they allow to study attractor states for the system.

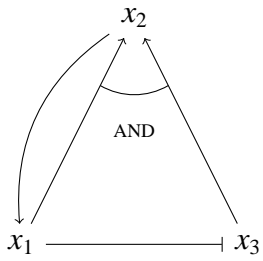
| Graph | Truth table | | | | | |
|---|-------------|-------|-------|--------|-------|-------|
| | Input | | | Output | | |
| | x_1 | x_2 | x_3 | x_1 | x_2 | x_3 |
|  | 0 | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 1 |
| | 0 | 1 | 0 | 1 | 0 | 1 |
| | 0 | 1 | 1 | 1 | 0 | 1 |
| | 1 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 1 | 1 | 0 |

Figure 1.9: Different representations of a Boolean function. Left: wiring diagram. \rightarrow : positive regulation (excitate); \vdash : negative regulation (inhibit). Right: truth table

Bayesian methods

Bayesian networks and Bayesian methods offer a probabilistic framework able to use noisy data and prior knowledge to infer dependencies. A causal relationship between variables is described by a directed acyclic graph G (for an example see Figure 1.10). Let Π_i be the parents of gene i in graph G , *i.e.* all genes j such as the edge $j \rightarrow i$ exists in G . The state of gene i is assumed to depend only on the state of its parents. Given a graph G , conditional probabilities $p(X_i = x_i | X_{\Pi_i} = \mathbf{x}_{\Pi_i})$ are estimated. Bayesian methods search for the graph G and probabilities $p(\cdot)$ that maximizes the probability of the observed data, with a penalty term representing

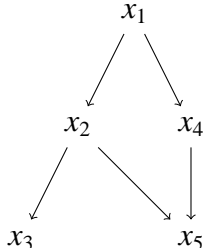
| Directed acyclic graph | Adjacency matrix |
|---|---|
|  | $G = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ |

Figure 1.10: Example of a Bayesian network, represented as a graph or an adjacency matrix

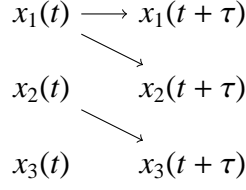


Figure 1.11: graph of a dynamical Bayesian network

a prior knowledge:

$$\mathcal{L}((\mathbf{x}_i)_{i=1\dots N}, G, \eta) = \text{prior}(G, \eta) \times \underbrace{\prod_{i=1}^N \left(\underbrace{\prod_{j=1}^p p_{\eta}(X_j = x_{ij} | X_{\Pi_j} = \mathbf{x}_{i\Pi_j})}_{\text{likelihood of observing } \mathbf{x}_i} \right)}_{\text{likelihood of observing all data}} \quad (1.45)$$

The prior is usually a penalty on the density of the graph, to obtain a sparse graph. Bayesian networks usually learn simplified dynamics, with discrete variables, or linear dynamics with continuous variables. Optimization of $\mathcal{L}((\mathbf{x}_i)_{i=1\dots N}, G, \eta)$ is an NP-hard problem, and is computationally prohibitive for brute-force algorithms [54, 55], thus most algorithms use heuristics or approximated solutions. Friedman et al. [56] use the *Sparse candidate algorithm* [57], in which parents of a node can only be found in a small subset of genes that are highly correlated with target gene. Another approach [58] is the estimation of a score of several networks through variational approximation methods [59, 60].

The assumption of an acyclic graph is false for biological networks, which contain feedback loops. This limitation is overcome by dynamical Bayesian network, see Figure 1.11. These networks can only learn from time-series data. The state of a gene at time t depends only on gene states at previous times [61, 62]. In the dynamical case, algorithms have been developed for learning nonlinear continuous dynamics [63, 64, 65].

Gene expression level are usually measured by mRNA concentration, but there are other components influence a gene's dynamic, such as protein concentration or microRNA. Bayesian methods can furthermore consider those unseen variables through latent variables. Bayesian methods alternatively identify the most likely

latent variables given the model and optimize the model given the data and current estimated latent variables [66, 67].

Other approaches

Other, more specific, models have been developed. **Kalman filters** model noisy and partially observed systems. They can model nonlinear dynamics, via variants such as “*Extended Kalman filters*” or “*Unscented Kalman filters*”, used for GRN inference in [68, 69, 70]. **S-Systems** offers very generic modeling, but are computationally intensive and no method guarantees to find the best solution for the model. They were used in [71, 72]. **Neural network** models mimic the way the brain functions, with neurons sending signals to others to calculate an output signal. They suffer from the same weaknesses as S-systems, and have been used in [73, 74]. **Gaussian processes** can model a nonlinear stochastic process such as gene dynamics, as shown by Aijo et al. [75]. **Logistic regression** offers a nonlinear interpretable model, useful for GRN inference [76]. Lim et al. [77] used **operator-valued kernels** to model genes. Using partial derivatives of their models, Lim et al. were able to infer GRNs.

Finally, best performances in several GRNI challenges were reached by the GENIE3 method [78]. GENIE3 learns a model $f^i(\mathbf{x}^{-i})$ with **Regression trees**. For a new input variable \mathbf{x} , f^i will make a binary test, for example $x^j \geq 0.5$. Following the answer to this binary test, input variable will either go in the left branch or the right branch of the tree. It will land on another node, with a new binary test classifying \mathbf{x}^{-i} , until it lands on a leaf, giving the value of $f^i(\mathbf{x}^{-i})$. After each binary test, $f(\mathbf{x}^{-i})$, the prediction for gene i , is more determined. One can interpret the importance of feature j in f^i as the amount of variance it reduces. For example, given a node \mathcal{N} , with its left branch \mathcal{N}_ℓ and its right branch \mathcal{N}_r . Let S (resp. S_ℓ , S_r) be all data from training set \mathcal{S} that reaches node \mathcal{N} (resp. branch \mathcal{N}_ℓ , \mathcal{N}_r). Noting $|S|$ the number of elements in S , $\text{Var}(S)$ the variance of x^i for all $\mathbf{x} \in S$, the variance reduction \mathcal{I} of node \mathcal{N} is computed following:

$$\mathcal{I}(\mathcal{N}) = |S|\text{Var}(S) - |S_\ell|\text{Var}(S_\ell) - |S_r|\text{Var}(S_r) \quad (1.46)$$

Importance of feature j is the sum of the variance reduction \mathcal{I} for all nodes \mathcal{N} where the binary test was done on x^j . A randomization scheme is added. Instead of uses a single regression tree, Huynh et al. uses **random forests**, *i.e.* many trees learned on random subsamples of the data.

1.5 Assessment of GRNI methods

Each author of a GRNI method tested his method in one or several of the following manners: inferring network from real gene expression data and comparison of the inferred network to the known GRN; inferring from real data and inspection by an expert of the inferred network; inferring from simulated data and comparison to the true regulatory network. Each method has its advantages and its drawbacks. When using real data and the known GRN, the results are questionable as large-scale GRN are, to a large extent, incomplete. The known GRN is called a *bronze* standard. To overcome errors from our limited knowledge of GRN, some author also justified their inferred network by observing that the topology of the network they inferred was in agreement with other studies. Using simulated data, one knows the true GRN producing the gene expression data. The inferred network is compared with a *gold* standard, but simulated data may badly reflect the real behavior of genes. In addition to questions on a dataset's quality to evaluate a GRNI method, most methods have some hyper-parameters to select a-priori. The arbitrariness of these parameters may put the study in question.

GRNI methods have also been evaluated by comparative studies. GRNI is an active field of research, and reviews quickly become outdated. I will mention two reviews: Emmert-Streib [79] gives an overview of correlation or mutual-information-based methods and summarizes papers comparing these methods; Narendra [80] compares 32 state-of-the-art methods on 15 real or realistically simulated data. No method clearly stands out. Authors essentially conclude that some methods are always underperforming, and should not be used.

Yet, comparative studies are not a perfectly fair manner to compare methods. Indeed, as the study's authors both use and evaluate methods, they might fine-tune the GRNI method or the dataset to produce better results, such as results stressing their method's advantages. This can be overcome by blind-challenges. In these competitions, organizers released gene expression data on several networks. Each contestant sent the networks inferred by his methods with no knowledge of the target network. In particular, Stolovitzky, Monroe and Califano created the Dialogue for Reverse-Engineering Assessments and Methods (DREAM) conferences and challenges [81] in 2006. They released data for many bioinformatic challenges, producing a fair evaluation of researchers' methods and many benchmark sets. Currently, there has been eight editions of DREAM challenges. In particular, the issue of GRNI was tackled in challenge four of the third edition and challenge four of the fifth edition. Organizers surveyed strengths and weaknesses of each method. They also reached several conclusions [82, 41]. Firstly, on simu-

lated data, when knock-out data are available for each gene, Z-score methods give excellent results, with a simple and fast algorithm. Secondly, results on a particular network depend mainly on the chosen model; two mutual-information-based methods would essentially find the same network. Thirdly, some methods give relatively good results: GENIE3, TIGRESS, ANOVA and CLR. Fourthly, and most importantly, the consensus of several methods usually performs better than methods taken individually. In the DREAM5 Challenge, Marbach et al. compared the results of each contestant to the results of averaging all contestants' responses. The consensus was always ranked in the three best methods, and often was the best. Besides, for consensus to perform best, methods using different modeling should be used: the consensus of a Bayesian, a mutual-information-based and a regression-based methods is expected to perform better than the consensus of three regression-based methods.

1.6 Problem formulation

In this manuscript, I suggest model-driven, nonparametric, GRN inference methods, using kernel functions and gene expression data. I set this work in the common simplified view, described in Section 1.2.4 and shown in Figure 1.2. Let p be the number of genes and A be the $p \times p$ adjacency matrix: $a_{ij} = 1$ if gene j regulates gene i , $a_{ij} = 0$ otherwise. To estimate this adjacency matrix A , I decompose this task into p independent tasks: for each gene i , I will estimate the row vector \mathbf{a}_i , assuming that I observe data with an additive noise whose covariance is diagonal.

As seen in Section 1.2.5, time-series data are available, allowing us to learn a model:

$$x^i(t + \tau) = f_{dyn,\tau}^i(\mathbf{x}^{-i}) + \epsilon_{dyn,t+\tau}^i, \quad (1.47)$$

with $\mathbf{x}^{-i} = (x^1, \dots, x^{i-1}, x^{i+1}, \dots, x^p)$ and $\epsilon_{dyn,t+\tau}^i$ a zero-mean noise. There also exists data with no time dependence, called steady-state data. They allow us to learn a model:

$$x^i = f_{stat}(\mathbf{x}^{-i}) + \epsilon^i, \quad (1.48)$$

with ϵ^i a zero-mean noise. Those models are learned using dynamic data, $\mathcal{S}_{dyn,\tau} = \{(\mathbf{x}^{-i}(t_1), x^i(t_1 + \tau)), \dots, (\mathbf{x}^{-i}(t_{N_\tau} - \tau), x^i(N_\tau))\}$, or static data, $\mathcal{S}_{stat} =$

$\{(\mathbf{x}_1^{-i}, x_1^i), \dots, (\mathbf{x}_N^{-i}, x_N^i)\}$. Both modeling use the same supervised learning tools, that will be described using the general notation $\mathcal{S} = \{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_N, y_N)\}$.

These data are observations of a biological system, hence they are noisy and result from nonlinear functions. Gene can be modeled through parametric approaches, where one assumes that gene dynamics is ruled by particular ordinary differential equations (ODE), and one has to find optimal parameters. These approaches can tackle nonlinear problems, but, in practice, they face prohibitive computational time on large dataset. Not to mention the relevance of the ODE for gene modeling.

Kernel functions, nonparametric models seen in Section 1.3.3, are good candidates for gene modeling. Very few works have applied kernel methods to GRN inference, mainly because they are not easily interpretable, so, even if they perfectly modeled genes, extracting the GRN from the kernel model would not be straightforward. This manuscript describes two GRN inference methods using kernels. In the first part, I propose a method to interpret a kernel model, called \widehat{Jac} . Given \hat{f}^i a model for gene i learned on steady-state or time-series data with kernel functions, \widehat{Jac} estimates \mathbf{a}_i through the partial derivatives of \hat{f}^i

$$\hat{a}_{ij} = \int \frac{\partial \hat{f}^i}{\partial x^j} d\mathbf{x} \quad (1.49)$$

In Chapter 2, I demonstrate that, under some assumptions on the distribution of the genes \mathbf{x} and the kernel function k , kernel methods consistently estimate the mean of partial derivatives. In Chapter 3, I describe other methods to interpret variable importance in a model \hat{f}^i . I suggest a method to learn from both steady-state and time-series data. I compare \widehat{Jac} to those other model interpretation methods and to state-of-the-art methods for GRN inference. In the second part, I develop an interpretable kernel-based model, called LocKNI:

$$x^i \approx \hat{f}^i(\mathbf{x}^{-i}, \mathbf{w}^i) + \epsilon, \quad (1.50)$$

with \mathbf{w}^i a feature weighting parameter; w_j^i measures the importance of gene j for the modeling of gene i . \mathbf{w}^i is learned from data. Then I estimate the adjacency matrix:

$$\mathbf{a}_i = \mathbf{w}^i \quad (1.51)$$

In Chapter 4, I compare LocKNI to other kernel feature weighting methods on realistic and widespread datasets. In Chapter 5, I describe a method to incorporate prior knowledge when learning \mathbf{w} . I show improvements when adding reasonable prior knowledge. I also compare LocKNI to state-of-the-art methods on real and realistically simulated datasets. LocKNI shows state-of-the-art performances, and also a behavior complementary to other existing methods.

Part I

Estimation of Partial Derivatives for Network Inference

Given a model f^i for a gene i , the partial derivative $\partial f^i / \partial x^j$ should reflect the action of gene j on gene i . Indeed, if j regulates gene i , a change in gene j 's concentration would have repercussions on gene i , thus $\partial f^i / \partial x^j \neq 0$. This approach raises two questions: how should genes be modeled and how the partial derivatives should be estimated.

Concerning the first question, several characteristics of gene behavior are difficult to model; most notably, the nonlinear dynamics of concentration of gene mRNAs. Research in supervised learning developed several methods able to tackle these difficult problems, such as regression trees, neural networks or kernel-based methods. However, regression trees produce piecewise constant functions, thus their model cannot be derived, while computation of neural networks may be prohibitive for real-sized networks. On the other hand, kernel methods possess desirable properties, such as robustness to noise. Kernel methods seem therefore to be good candidates for gene modeling.

With respect to the second question, many works exist for the estimation of derivatives of univariate functions. Unfortunately, gene models will be multivariate functions, and very few works exist for this problem.

In Chapter 2, I tackle the problem of partial derivative estimation from a theoretical point of view. Given some assumptions on the observations \mathbf{x} that should be met by genes, I give sufficient conditions on a learning algorithm \mathcal{K} , taking training data \mathcal{S} to learn a model \hat{f} , so that, for any continuous linear form g , $g(\partial \hat{f} / \partial x^j)$ consistently estimates $g(\partial f / \partial x^j)$. I then show that some kernel methods meet these conditions. Finally, I test this partial derivative estimation method on toy examples.

In Chapter 3, I implement the use of partial derivatives to infer the gene regulatory network. This method is improved by an ensemble method, allowing to learn from both steady-state and dynamic modeling. Estimation of partial derivatives $\partial \hat{f}^i / \partial x^j$ can be seen as a way to interpret the importance of feature j in the model \hat{f}^i . Therefore, I compare this network inference method to other feature importance interpretations. I also compare this network inference method to state-of-the-art GRNI methods on realistic datasets. It provides state-of-the-art results.

Chapter 2

Estimation of Partial Derivatives

2.1 Introduction

Learning a relationship f between input variables \mathbf{x} and output y from past examples is a common task in machine learning. Many domains (computer vision, medical imaging, bioinformatics, etc) present such problems, fostering the development of many efficient algorithms for learning \hat{f} , an estimation of f , from observational data. Most learning methods provide of theoretical bounds for convergence and performances. One can mention linear, tree-based or kernel-based methods. However, the partial derivatives estimation problem has fewer theoretical guarantees. Recently, several fields have shown interest in learning the partial derivatives $\partial f / \partial x^i$. For the characterization of nanoparticles, quantitative features, such as the diameter of a nanoparticle, can be more accurately estimated by the derivative of a response function [83, 84, 85]. Derivatives allow to quantify the progress of a disease [86] or to infer gene regulatory networks [77]. Besides, independently of the domain, partial derivatives may quantify the relevance of an input variable for a function, thus can be used as a feature selection criterion. Several saliency measures are based on partial derivatives, see [87, 88] and references therein. Nevertheless, theoretical consistency of derivative estimation must be proved. Technical difficulties arise because differentiation is not continuous, without assumption on the studied functional space. Thus estimating f by \hat{f} through a consistent learning algorithm does not imply that $\partial \hat{f} / \partial x^i$ consistently estimate $\partial f / \partial x^i$.

Consistency of several derivative estimation methods for univariate problems has

been proven. The approach calculates empirical derivatives [89]. Given a sorted¹ training dataset $(x_i, y_i)_{i=1\dots N}$, $x_i \leq x_{i+1}$ for all i , De Brabanter et al. set the empirical derivative [89]

$$y'_i = \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_i}.$$

For more robustness to noise, Charnigo et al. calculate an averaged empirical derivative [90]

$$y'_i = \sum_k w_k \frac{y_{i+k} - y_{i-k}}{x_{i+k} - x_{i-k}}$$

with w_k a decaying weight, $\sum_k w_k = 1$, $w_{k-1} \geq w_k \geq 0$ for all k . The algorithm in [91] trains \hat{f} so that $\hat{f}(x_i)$ fits y_i and $\hat{f}'(x_i)$ fits y'_i for all i . Another approach is to model f with locally polynomial functions, such as splines [92]. Charnigo et al. [90] learn a model $\hat{f}(x) = \sum_\ell w_\ell(x) \mu_\ell(x)$, with μ_ℓ a polynomial and w_ℓ a function determining if x is in the region modeled by μ_ℓ . With $w_\ell(x) = \exp(-(x - x_\ell)/h)$, Charnigo [93] proves consistency of derivatives of order j . For multivariate problems, there is only, to our knowledge, Mosci's new kernel-based approach that has been proven to estimate $\|\partial f / \partial x_i\|_2$ consistently [94].

In this work, I firstly give sufficient conditions on a target function f and a learning algorithm to consistently estimate $g(\nabla f)$, with g any continuous linear form, and ∇f the gradient of f , $\nabla f = (\partial f / \partial x^1, \dots, \partial f / \partial x^p)$. Secondly, I prove that Gaussian kernel methods satisfy these conditions. Thirdly, I suggest to approximate the integrals by sum, for faster and simpler computation. Finally, I experiment this partial derivative estimation method on toy examples.

2.2 Consistent estimation of mean partial derivatives

In the following, (\mathbf{X}, Y) are random variables from $\Omega \times \mathcal{Y}$, with Ω a finite p -dimensional space, and \mathcal{Y} is a bounded subset of \mathbb{R} . \mathbf{X} is drawn from a distribution μ . I make the following assumptions:

- (A-1) The support of μ , noted \mathcal{X}_o , is bounded and convex. The closure of \mathcal{X}_o is denoted by \mathcal{X}_c .

¹this assumption can be made without loss of generality

(A-2) The random variable Y follows

$$y = f(\mathbf{x}) + u \quad (2.1)$$

with u a zero-mean noise. I further assume that f is continuous and continuously differentiable on \mathcal{X}_c , i.e. belongs to the space $C^1(\mathcal{X}_c, \mathcal{Y})$.

In theory, assumption (A-1) is met by many empirical distributions, notably, the truncated Gaussian distribution. In practice, many random variables are assumed to follow a Gaussian distribution and the loss of generality induced by using a truncated Gaussian distribution is very small. Besides, assuming that gene concentration is bounded seems reasonable; the convexity assumption of reachable states is more debatable. Assumption (A-2) describes the link between Y and \mathbf{X} and the additive nature of the noise. It makes mild constraints on f , namely that ∇f exists and is continuous, criteria met by many biological models. I will use the following norm notations. $\mathcal{L}^2(\Omega, \mu)$ is the space integrable function, according to distribution μ , with the norms:

$$\|f\|_{\mathcal{L}^2(\Omega, \mu)}^2 = \int_{\Omega} f(\mathbf{x})^2 \mu(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{X}_c} f(\mathbf{x})^2 \mu(\mathbf{x}) d\mathbf{x} , \quad (2.2)$$

$$\|f\|_{\infty} = \max_{x \in \mathcal{X}_c} |f(x)| . \quad (2.3)$$

The Euclidian norm in space Ω is $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^p (x^i)^2}$.

The last needed notations and assumptions concern the learning algorithm. Let \hat{f}_{ℓ} be f 's estimate by algorithm \mathcal{K} with ℓ data pairs, $(\mathbf{x}_i, y_i)_{i=1 \dots \ell}$. I assume that:

(A-3) \hat{f}_{ℓ} is a consistent estimator: for all $\epsilon, \eta > 0$, there exists an integer ℓ_0 such that, for all integer $\ell \geq \ell_0$, the probability that $\|f - \hat{f}_{\ell}\|_{\mathcal{L}^2(\Omega, \mu)} \leq \epsilon$ is greater than $1 - \eta$, and

(A-4) there exists a integer ℓ_0 such that, for all integer $\ell \geq \ell_0$, the norm of \hat{f}_{ℓ} 's gradient is bounded by a constant M .

Machine Learning research has developed many algorithm satisfying (A-3). Assumption (A-4) is satisfied by some of them, in particular kernel methods, as we shall see later.

If all assumptions are met, any continuous linear form of the derivatives of f will be consistently estimated by the one of \hat{f}_{ℓ} . This is proven using the following two theorems:

Theorem 2. Let f be a function belonging to the space $C^1(\mathcal{X}_c, \mathcal{Y})$. Assume the gradient of f is bounded by M , $\|\nabla f\|_2 < M$. Then, for any $\epsilon > 0$, there exists $C_{1,\epsilon}$, independent of f , such as:

$$\|f\|_\infty \leq C_{1,\epsilon} \|f\|_{\mathcal{L}^2(\Omega,\mu)} + \epsilon . \quad (2.4)$$

Theorem 3. Let f be a $C^1(\mathcal{X}_c, \mathcal{Y})$ function, with gradient bounded by M . Let g be a linear continuous form of $C^0(\mathcal{X}_c, \mathcal{Y})$, $g : C^0(\mathcal{X}_c, \mathcal{Y}) \rightarrow \mathbb{R}$. Then, for any $\epsilon > 0$, there exists a constant $C_{2,g,\epsilon}$ such that:

$$|g(\nabla f)| \leq C_{2,g,\epsilon} \|f\|_\infty + \epsilon \quad (2.5)$$

These two theorems are proven in the appendix, pages 128 for theorem 2 and page 129 for theorem 3). I show next that kernel-ridge regression and partial least-square regression learning algorithms both satisfy assumptions (A-3) and (A-4).

2.3 Kernel methods for partial derivative estimation

There exist *universal* kernels, *i.e.* kernels whose RKHS \mathcal{H} is the whole space of continuous functions from \mathcal{X}_c to \mathbb{R} (or \mathbb{C}) [95]. Using consistency of kernel-ridge [96] or partial least-square [97] regression and a universal kernel, I obtain a learning algorithm \mathcal{K} satisfying assumption (A-3). I give sufficient conditions on a universal kernel k ensuring that the associated learning algorithm \mathcal{K} satisfies (A-4):

Lemma 4. Let \mathcal{H} be the RKHS of universal kernel k . If, for all $\mathbf{x} \in \Omega$

- the kernel is constant on the set of points (\mathbf{x}, \mathbf{x}) , $k(\mathbf{x}, \mathbf{x}) = c$,
- at point (\mathbf{x}, \mathbf{x}) , the gradient of the kernel is null, $\nabla_{\mathbf{z}} k(\mathbf{x}, \mathbf{z})|_{\mathbf{z}=\mathbf{x}} = 0_p$, and
- at point (\mathbf{x}, \mathbf{x}) , the Hessian matrix $H(\mathbf{x}, \mathbf{z})_{ij} = \frac{\partial^2 k(\mathbf{x}, \mathbf{z})}{\partial z^i \partial z^j}$ has eigenvalues bounded by a constant M , $|\langle u, H(\mathbf{x}, \mathbf{x}) u \rangle| \leq M \|u\|^2$ for all $\mathbf{x} \in \mathcal{X}$ and all $u \in \mathbb{R}^p$

then, for all $f \in \mathcal{H}$ and for all $\mathbf{x} \in \mathcal{X}_c$, the gradient of f satisfies:

$$\|\nabla f(\mathbf{x})\|_2 \leq \sqrt{M} \|f\|_{\mathcal{H}} . \quad (2.6)$$

Sketch of proof. The RKHS property and Cauchy-Schwarz inequality give:

$$|f(\mathbf{x}) - f(\mathbf{x} + \mathbf{h})| = \langle f, \phi(\mathbf{x}) - \phi(\mathbf{x} + \mathbf{h}) \rangle_{\mathcal{H}} \quad (2.7)$$

$$\leq \|f\|_{\mathcal{H}} \|\phi(\mathbf{x}) - \phi(\mathbf{x} + \mathbf{h})\|_{\mathcal{H}}, \quad (2.8)$$

while we also have:

$$|f(\mathbf{x}) - f(\mathbf{x} + \mathbf{h})| = |\langle \nabla f, \mathbf{h} \rangle + o(\|\mathbf{h}\|)| \quad (2.9)$$

Using the polarization identity, the distance $\|\phi(\mathbf{x}) - \phi(\mathbf{x} + \mathbf{h})\|_{\mathcal{H}}^2$ can be expressed with scalar products. Using the kernel trick, I can express this distance with function k . Using a Taylor expansion and conditions of lemma 4:

$$\|\phi(\mathbf{x}) - \phi(\mathbf{x} + \mathbf{h})\|_{\mathcal{H}}^2 = k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x} + \mathbf{h}, \mathbf{x} + \mathbf{h}) - 2k(\mathbf{x}, \mathbf{x} + \mathbf{h}) \quad (2.10)$$

$$= \mathbf{h}' H(\mathbf{x}, \mathbf{x}) \mathbf{h} + o(\|\mathbf{h}\|^2) \quad (2.11)$$

$$\|\phi(\mathbf{x}) - \phi(\mathbf{x} + \mathbf{h})\|_{\mathcal{H}} \leq \sqrt{M} + o(\|\mathbf{h}\|) \quad (2.12)$$

Equations (2.8) and (2.12) gives the result. A detailed proof is given in the appendix, page 130.

The conditions of lemma 4 are satisfied by the Gaussian kernel.

Lemma 5. *The Gaussian kernel of bandwidth σ satisfies the hypothese of lemma 4.*

The universality of the Gaussian kernel is proven in [95]. The other conditions are proven using calculus. This is done in the appendix, page 131.

Consistent estimation of continuous linear forms of partial derivatives with Gaussian kernel methods is obtained by lemma 5 then lemma 4, then theorem 2 and, finally, theorem 3 can be applied, giving us the following theorem:

Theorem 6. *With samples $(\mathbf{x}_i, y_i)_{i=1 \dots \ell}$ i.i.d., for any ϵ and $\eta > 0$, and for any continuous linear form g , $g : C^0(\mathcal{X}_c, \mathcal{Y}) \rightarrow \mathbb{R}$, there exists ℓ_0 such that if $\ell \geq \ell_0$, then, with probability greater than $1 - \eta$,*

$$\left| \int_{\mathcal{X}_c} g(\nabla \hat{f}_\ell(\mathbf{x})) - g(\nabla f(\mathbf{x})) d\mathbf{x} \right| \leq \epsilon, \quad (2.13)$$

where \hat{f}_ℓ is the estimator of f based on Gaussian kernel ridge regression or Gaussian partial least-square regression.

A detailed proof is given in the appendix, page 132. Now, the term $\int_{\mathcal{X}_c} g(\nabla \hat{f}_\ell(\mathbf{x})) d\mathbf{x}$ may be hard to compute. I give below a simple and fast approximation of this integral.

2.4 Approximation of integrals

An analytical formula for the integral of the estimate \hat{f} over \mathcal{X} may not be available. To overcome this difficulty, I suggest a simple method to approximate this integral by a sum. By the central limit theorem:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \nabla \hat{f}(\mathbf{x}_i) \xrightarrow{\ell \rightarrow +\infty} \int_{\mathcal{X}_c} \nabla \hat{f}(\mathbf{x}) \mu(\mathbf{x}) d\mathbf{x} \quad (2.14)$$

Moreover, as the gradient $\nabla \hat{f}$ is bounded, the central limit theorem gives bounds on the difference between the sum and the integral, provided in the appendix, page 133.

If one wants to estimate a linear form different from equation (2.14), one can use the following sum, assuming that the distribution μ is known:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \frac{h(\mathbf{x}_i)}{\mu(\mathbf{x}_i)} \nabla \hat{f}(\mathbf{x}_i) \xrightarrow{\ell \rightarrow +\infty} \int_{\mathcal{X}_c} \nabla \hat{f}(\mathbf{x}) h(\mathbf{x}) d\mathbf{x} \quad (2.15)$$

In most cases, the distribution μ is unknown, but several accurate methods exist to estimate it. In particular, multivariate kernel density estimation [98] provides nonparametric and consistent estimation of μ . Noting $\kappa(\mathbf{u}) = (2\pi)^{-p/2} \exp(-\|\mathbf{u}\|_2^2/2)$, a_ℓ a sequence such that $a_\ell \xrightarrow{\ell \rightarrow +\infty} 0$ and $\ell a_\ell \xrightarrow{\ell \rightarrow +\infty} +\infty$ (for example, $a_\ell = \ell^{-1/2}$), Simonoff defines:

$$\hat{\mu}_\ell(\mathbf{x}) = \frac{1}{\ell a_\ell^p} \sum_{i=1}^{\ell} \kappa\left(\frac{\mathbf{x} - \mathbf{x}_i}{a_\ell}\right) \quad (2.16)$$

This estimator $\hat{\mu}$ satisfies:

$$\|\hat{\mu}_\ell - \mu\|_\infty \xrightarrow{\ell \rightarrow +\infty} 0 \quad (2.17)$$

Proofs and bounds can be found in [99]. Thus, one can estimate the integral according to:

$$\int_{\mathcal{X}_c} \nabla \hat{f}(\mathbf{x}) h(\mathbf{x}) d\mathbf{x} \approx \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{h(\mathbf{x}_i)}{\hat{\mu}_\ell(\mathbf{x}_i)} \nabla \hat{f}(\mathbf{x}_i) \quad (2.18)$$

2.5 Numerical simulations

In this section, I illustrate the consistency of kernel-ridge regression for the estimation of continuous linear forms of partial derivatives. I build toy examples with input variables \mathbf{x} drawn independently and uniformly in $[-1, 1]^p$. The output variable y is a function of the following form:

$$y = f(\mathbf{x}) + g\epsilon = \tanh(\mathbf{x}^T A \mathbf{x} + \mathbf{b}^T \mathbf{x}) \sin(\mathbf{c}^T \mathbf{x}) + g\epsilon \quad (2.19)$$

with ϵ a normal noise, $\epsilon \sim \mathcal{N}(0, 1)$, uncorrelated between samples, and A, \mathbf{b} and \mathbf{c} independently chosen random parameters:

$$A \sim \mathcal{U}([0, 1]^{p \times p}) \quad (2.20)$$

$$\mathbf{b} \sim \mathcal{U}([0, 1]^p) \quad (2.21)$$

$$\mathbf{c} \sim \mathcal{U}([0, 2\pi]^p) \quad (2.22)$$

See Figure 2.1 for examples of functions given by equation (2.19).

In order to measure the error in partial derivative estimation, I define E , the error in partial derivative estimation for one simulation as follows:

$$E = \sum_{m=1}^p \left(\int_{\mathcal{X}_c} \frac{\partial f^*}{\partial x_m} \mu(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{X}_c} \frac{\partial \hat{f}^*}{\partial x_m} \mu(\mathbf{x}) d\mathbf{x} \right)^2 \quad (2.23)$$

For fixed parameters (number of training points N , dimension of input variables p , noise to signal ratio σ , with $g = \sigma \sqrt{\text{var}(f(\mathbf{x}))}$), I simulate 100 models following equation (2.19) and plot the mean and variance of the E in box plots. Figure 2.2 shows that the error decreases with the number of training points. In these examples, the problem dimension p is 2. The noise to signal ratio is 0% in the top left plot, 100% in the top right plot. The bottom plots show the evolution of the error with the noise to signal ratio σ (plot at the bottom left), and with the input dimension p (plot at the bottom right). As expected, the error increases with the noise-to-signal ratio. Besides, the estimator is very sensitive to the dimension of input variables. When p increases, the error greatly increases.

We also dispose of a partial derivative estimation to estimate other linear forms of the partial derivatives. To simulate this, I draw \mathbf{x} following a centered and reduced Gaussian distribution,

$$X \sim \mathcal{N}(0, I_p), \quad (2.24)$$

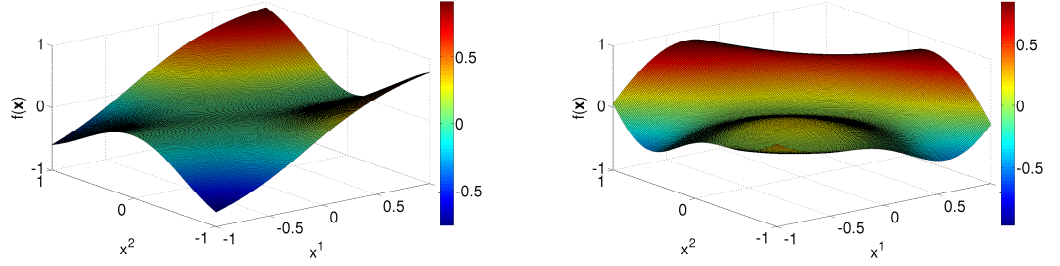


Figure 2.1: Examples of functions of the form given by equation (2.19) with dimension $p = 2$

and I estimate the integral of f 's derivatives in the cube $[-1, 1]^p$,

$$E = \sum_{m=1}^p \left(\int_{[-1,1]^p} \frac{\partial f^*}{\partial x_m} d\mathbf{x} - \int_{[-1,1]^p} \frac{\partial \hat{f}^*}{\partial x_m} d\mathbf{x} \right)^2. \quad (2.25)$$

Results are shown in Figure 2.3, with $p = 3$ and $\sigma = 25\%$. These results illustrate the property that kernel-ridge regression consistently estimates partial derivatives.

2.6 Conclusion

In this chapter, I tackled the partial derivative estimation problem. Observing the scarcity of results for the multi-dimensional case, their specificity to the learning algorithm used, I showed that, under a few assumptions on the learning algorithm, linear forms of partial derivatives were consistently estimated. Furthermore, I showed that kernel methods met these assumptions. In addition, if a kernel method is proved not to require the *i.i.d.* assumption to learn f , this results and the theorems in this chapter can be used to prove consistency of the estimates of partial derivatives without the *i.i.d.* assumption.

Finally, the proposed method to estimate derivatives was tested on toy examples. We saw that this method gives good quality results in the low-dimensional case, but its performance decreases quickly with the dimension of the input. In the following chapter, I will use partial derivatives to infer gene regulatory networks.

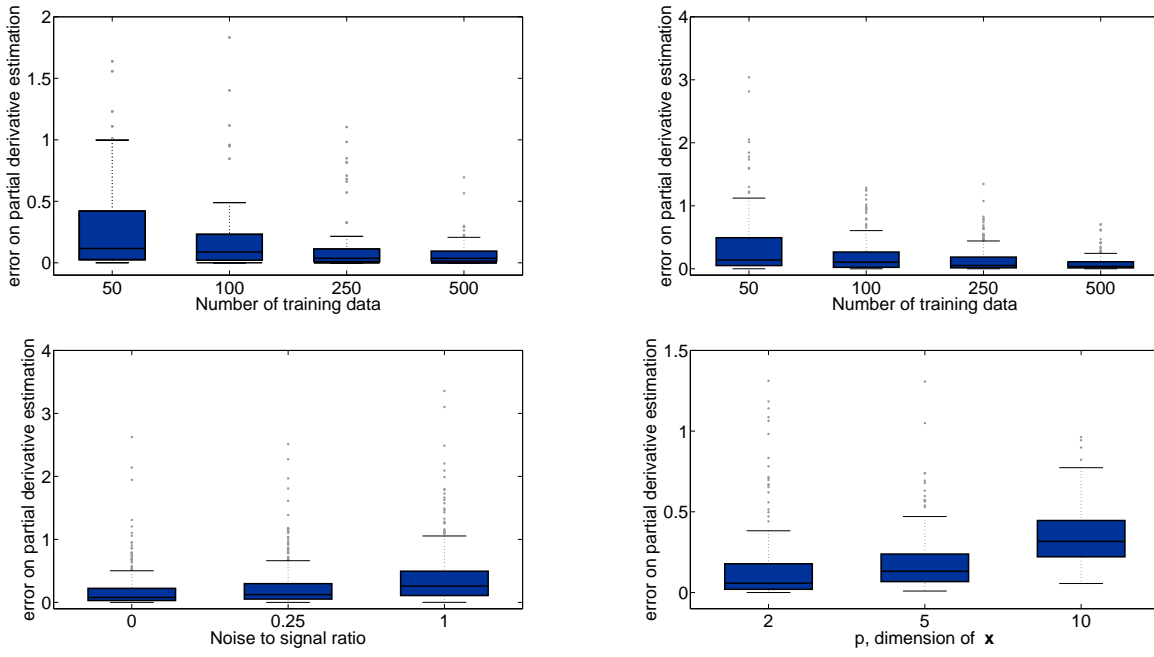


Figure 2.2: Mean and variance of $\sum_{m=1}^p (g(\partial f^*/\partial x_m) - g(\partial \hat{f}/\partial x_m))^2$. The top plots show the error in function of the number of training samples N for noise to signal ratio 0% (top left) and 100% (top right), dimension $p = 2$. The bottom plots show the error in function of the signal-to-noise ratio σ (bottom left - with $p = 3, N = 250$) and the dimension of \mathbf{x} p (bottom right $\sigma = 25\%, N = 250$)

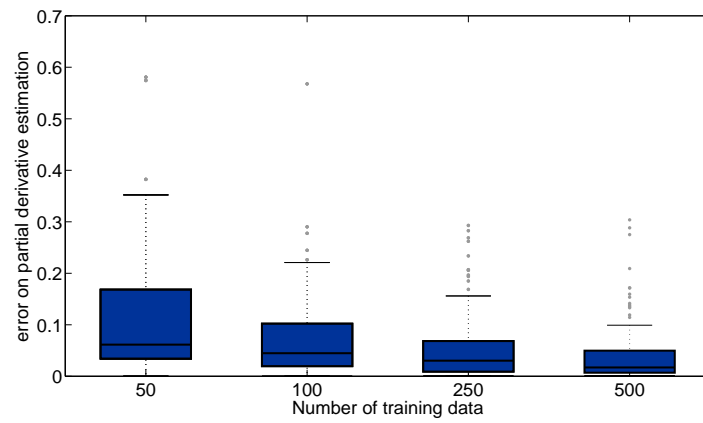


Figure 2.3: Error in the estimation of partial derivatives when input variables \mathbf{x} are drawn from a Gaussian distribution, and the integral is estimated on the cube $[-1, 1]^p$

Chapter 3

Network inference using partial derivatives of kernel-based models

3.1 Introduction

As shown in the first chapter, many model-driven approaches have been developed for gene regulatory network inference. Although kernel methods provide efficient nonparametric modeling, they have rarely been used for gene network inference, mainly due to their “black box” behavior. However, we saw, in the previous chapter, that partial derivatives of a regression model based on a universal kernel, typically a Gaussian kernel, can be used to provide consistent estimates of the mean of the partial derivatives of the target function. Therefore, I propose here to use the partial derivatives to interpret a kernel-based regression *a posteriori*. Calculating the empirical mean of the partial derivative for a given feature on data is seen as an importance measure of this feature. This chapter presents a study of this new GRN inference method. To improve this measure and to deal with large dimensions, this approach is extended to an ensemble of randomized kernel-based models. The whole approach is presented in Section 2. Then, related works about other measures of feature importance are reviewed. Section 4 describes methods for hyper-parameters selection. Section 5 is devoted to the numerical experiments which involve results on small-scale realistic networks and large scale, real-sized biological networks. \widehat{Jac} , the GRN inference method using partial derivative, is compared to other importance measures applied on kernel-based models.

3.2 Partial derivatives of kernel-based models

3.2.1 Jacobian matrix estimation for GRN inference

Several works have studied the importance of a feature through partial derivatives, see [87, 88, 94] and references therein. Assume that the behavior of a gene regulatory network is governed at steady-state by the following p models f^i :

$$\forall i \in \{1, \dots, p\}, \ell \in \{1, \dots, N\}, \quad x_\ell^i = f^i(\mathbf{x}_\ell) + \epsilon_\ell^i, \quad (3.1)$$

where $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ are the observed dataset and f^i is the true model governing the expression level of gene i in function of the expression levels of the other genes. If gene j excites (resp. inhibits) gene i , then its action should be observed in the partial derivatives: $\partial f^i / \partial x^j$ should be positive (resp. negative). Note that the effect of a gene j on gene i may be visible only on some of the system's possible states \mathcal{X} . For example, gene j may influence gene i only if its concentration is above a threshold θ_j , so that $\partial f^{*,i} / \partial x^j = 0$ if $x^j \leq \theta_j$. Similarly, beyond another concentration γ_j , gene i 's receptors to gene j 's action may saturate, and $\partial f^{*,i} / \partial x^j = 0$ if $x^j \geq \gamma_j$. Therefore the mean partial derivative $\partial f^{*,i}$ should be estimated on all possible states \mathcal{X} . Let μ be the distribution of the system state, I want to estimate the average value of the $(i, j)^{th}$ entry of the Jacobian matrices:

$$Jac(f)_{i,j} = \int_{\mathcal{X}} \frac{\partial f^i(\mathbf{x})}{\partial x^j} \mu(\mathbf{x}) d\mathbf{x}, \quad (3.2)$$

As seen in Chapter 2, the empirical mean of the partial derivatives of the estimated models \hat{f}^i consistently estimate $Jac(f)$:

$$\widehat{Jac}(\hat{f})_{i,j} = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \frac{\partial \hat{f}^i}{\partial x^j}(\mathbf{x}). \quad (3.3)$$

Similarly, the temporal behavior of the gene regulatory network may be assumed to be governed by an autoregressive model that decomposes into p models f_τ^i :

$$\forall i \in \{1, \dots, p\}, \ell \in \{0, \dots, N-1\}, \quad x^i(t_\ell + \tau) = f^i(\mathbf{x}(t_\ell)) + \epsilon_{t_\ell}^i, \quad (3.4)$$

with $\mathcal{D}_\tau = \{\mathbf{x}(t_0), \dots, \mathbf{x}(t_N)\}$, $t_\ell = t_0 + \ell\tau$ and $\epsilon_{t_\ell}^i$ are i.i.d. realization of a gaussian noise. Then

$$\widehat{Jac}(\hat{f}_\tau)_{i,j} = \frac{1}{N} \sum_{t \in \{t_1, \dots, t_N\}} \frac{\partial \hat{f}_\tau^i(\mathbf{x}(t))}{\partial x^j(t)}. \quad (3.5)$$

These Jacobian estimates of the learned model can be used in order to get an estimate of the binary and asymmetric target adjacency matrix of the network. One way to provide such an estimate is to threshold the absolute value of the Jacobian's coefficients given a threshold θ :

$$\hat{A}_{i,j} = H(|\widehat{Jac}(\hat{f})_{i,j}| - \theta), \quad (3.6)$$

with $H : \mathbb{R} \rightarrow \{0, 1\}$, the indicator function of \mathbb{R}^+ .

In the remaining part of the chapter, I will consider the problem of the estimation of the i^{th} row in adjacency matrix A . For the sake of simplicity, let i , the index of a target gene, be fixed and consider the following base model that satisfies the following equations:

$$y_\ell = f(\mathbf{z}_\ell) + \epsilon_\ell \quad (3.7)$$

with $\mathcal{S} = \{(\mathbf{z}_\ell, y_\ell), \ell = 1, \dots, N\}$ and ϵ_ℓ i.i.d. realization of a Gaussian noise. Input variables have finite dimension $\mathbf{z} \in \mathbb{R}^p$

Both these estimates would greatly suffer from high-dimensionality of the input variable \mathbf{z} . To overcome this difficulty, I propose to learn functions f that uses few dimensions of the variables. With an ensemble method, many functions f are learned using Gaussian kernel-ridge regression, but each function relies on a small subset of features of \mathbf{z} —another approach is changing the base learned to produce a sparse Jacobian matrix, and will be developed in Chapters 4 and 5. Moreover, note that other partial derivatives open many possibilities to infer the GRN: for example, one may consider that, if gene j regulates gene i , there must exists a state \mathbf{z} where gene j exert much its influence on gene i , and the partial derivative $\partial f^i / \partial z^j$ will reach a high value, thus estimating the GRN by the maximum value obtained by the partial derivatives. In this thesis, I was interested in the average Jacobian.

3.2.2 Ensemble of randomized kernel-based models

From a very general point of view, performances have been improved in many domains by using an ensemble method approach. Instead of using a function \hat{f} learned with one algorithm \mathcal{K} on the whole dataset \mathcal{S} , one can create B subsets of \mathcal{S} : $(\mathcal{E}_1, \dots, \mathcal{E}_B)$, and learn a prediction function h_b on each dataset \mathcal{E}_b . Using the mean function $\frac{1}{B} \sum_{b=1}^B h_b$ is usually more accurate and more robust to noise than using the single function h learned using the whole dataset (see [100, 46] for

some examples).

In this Chapter, there are many motivations to use ensemble methods. In order to estimate f in 3.7, a Gaussian kernel-based model is considered:

$$h(\mathbf{z}) = \sum_{\ell=1}^N \alpha_{\ell} k(\mathbf{z}_{\ell}, \mathbf{z}) \quad (3.8)$$

$$k(\mathbf{z}_{\ell}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{z}_{\ell} - \mathbf{z}\|^2}{2\sigma^2}\right) \quad (3.9)$$

$$(3.10)$$

Then, the partial derivatives are given by :

$$\frac{\partial h(\mathbf{z})}{\partial z^j} = \sum_{\ell=1}^N \alpha_{\ell} \frac{\partial k(\mathbf{z}_{\ell}, \mathbf{z})}{\partial z^j} \quad (3.11)$$

$$\frac{\partial k(\mathbf{z}_{\ell}, \mathbf{z})}{\partial z^j} = \frac{z_{\ell}^j - z^j}{\sigma^2} \exp\left(-\frac{\|\mathbf{z}_{\ell} - \mathbf{z}\|^2}{2\sigma^2}\right) \quad (3.12)$$

$$(3.13)$$

As the Gaussian kernel uses each input feature in the same way, the partial derivatives regarding two features j and m only differs by the factors $(z_{\ell}^j - z^j)$ and $(z_{\ell}^m - z^m)$. To increase this difference, the kernel-ridge regression is used as a base learner in an ensemble method, each base learner is trained on a random subspace of \mathbb{R}^p and a subsample of the data. Contrary to random forests, the base learner does not select further features; however, for a given size of the training data, models trained on smaller subspaces are expected to give more contrasted partial derivatives. Moreover, working in random subspaces allows to tackle a large number of input features. Finally, I suggest an ensemble method approach allowing to learn from heterogeneous datasets, here steady-state data and irregularly-observed time series.

Learning from heterogeneous data types

In this work, my objective is not obtaining the most accurate prediction function \hat{f} , but identifying the important features of gene i 's behavior. I observe gene i under different perspectives (steady-state or dynamic) in dataset (E_{stat}, E_{dyn}) , but I assume that both behaviors are the results of the same GRN. Thus, learning importance of features from both of these perspectives gives complementary information on the same network.

Dataset building

All measures can be used as static data $E_{stat} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ to learn a model $x^i = f_{stat}(\mathbf{x}^{-i}) + \epsilon_{stat}$, with ϵ_{stat} a zero-mean noise, as done by several state-of-the-art methods [78, 45]. In addition, these data include time-series, for which we also have information about time dependencies. We have at our disposal N_{dyn} data from several time-series. For a time-series u , the system is observed at times $t_{1,u}, \dots, t_{T_u,u}$. The observation of the system at time $t_{k,u}$ in time-series u is denoted $\mathbf{x}(t_{k,u}, u)$. These data can be used to learn a dynamical model of the system:

$$\mathbf{x}^i(t + \tau, u) = f_\tau(\mathbf{x}^{-i}(t, u)) + \epsilon_\tau \quad \forall t, u \quad (3.14)$$

for all $1 \leq u \leq N_u$, and with ϵ_τ a zero-mean noise.

To build a model f_τ from time-series, there must be regularly spaced time-points, which is rarely the case in biological time series. Indeed, when the time-series is the response of the system to an exogenous perturbation, most behavior will be observed soon after the perturbation. After a long time, the system has stabilized. Thus, experimentalists take many measurements right after the perturbation, and few when much time has passed.

To overcome the problem of irregularly spaced observation time, I build for all possible time steps τ the datasets $E_{dyna,\tau} = \{(\mathbf{x}(t_1, u_1), \mathbf{x}(t_1 + \tau, u_1)), \dots, (\mathbf{x}(t_{N_\tau} - \tau, u_{N_\tau}), \mathbf{x}(t_{N_\tau}, u_{N_\tau}))\}$ of all time point pairs $(t, t + \tau)$ available in the data, see Figure 3.1 for a visual example. Each of these datasets allows to learn a model f_τ , but not every dataset is interesting: some of them contain too little data, others have a time-step τ very long compared to the characteristic time of the system, and only a return to a steady-state is observed in the model f_τ . Without expert knowledge on a system, I relied on the experimentalists' competence, and assumed that the time τ^* , for which E_{τ^*} is the largest dynamical dataset ($\tau^* = \arg \max_\tau |E_\tau|$), is below the characteristic time of the system. Thus I accepted dataset E_τ if $\tau \leq \tau^*$ and $|E_\tau| \geq 30$.

Subspaces and subsamples building

From available datasets $(E_0, E_1, \dots, E_{N_e})$. I build B subsamples \mathcal{E}_ℓ , $\ell = 1 \dots B$ with *subbagging* and subsampling of variables, in a similar way to *extreme-randomization* [101]. Subsamples are built according to the following procedure:

1. Randomly choose from which dataset E_u to extract data. The probability of choosing dataset E_u is proportional to its size, $p(E_u \text{ selected}) =$

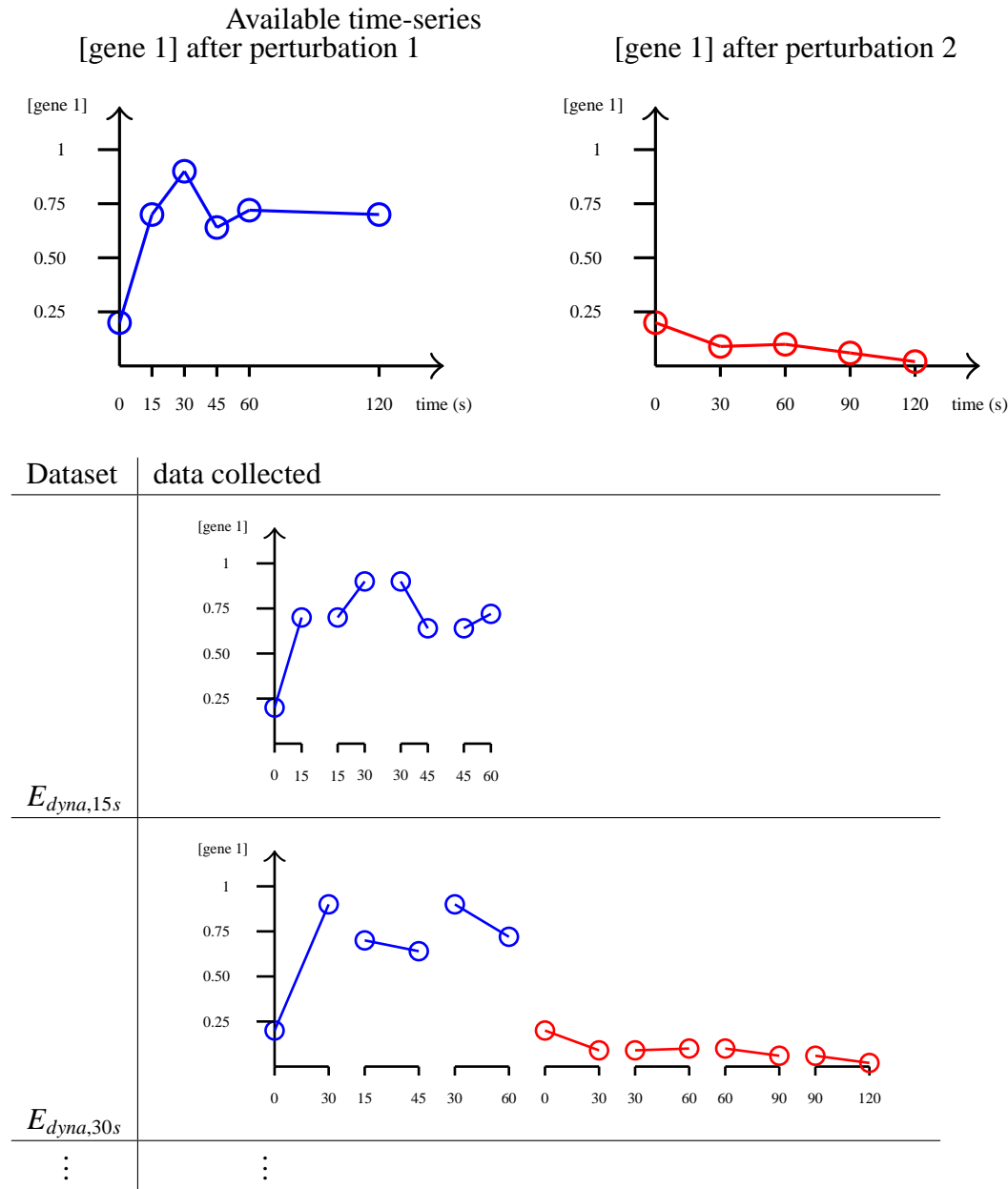


Figure 3.1: Example of irregularly spaced time-series, and the building of datasets $E_{dyna, \cdot}$.

$|E_u|/(\sum_{\ell=1}^{N_e} |E_\ell|)$. Let E_u be the selected dataset.

2. Each data vector \mathbf{x}_i of E_u has probability $p_{data,u}$ to be in subsample \mathcal{E}_ℓ . To obtain subsamples of similar sizes, $p_{data,u}$ is taken inversely proportional to E_u 's size. Assuming, without loss of generality, that E_0 is the largest dataset, I have fixed $p_{data,u} = p_{data} \frac{E_0}{E_u}$, with p_{data} a fixed hyper-parameter. To obtain test data, this probability is capped at 95%.
3. Randomly select n_{var} variables to be the potential regulators of our system. n_{var} is a hyper-parameter to fix. The set of selected variables is called \mathcal{G}_ℓ .

Let \hat{f}_ℓ^i be the model for gene i learned on subsample \mathcal{E}_ℓ . One may directly use $\widehat{Jac}(\hat{f}_\ell)$, but there is no unit nor references to assess that the value $\widehat{Jac}(\hat{f}_\ell)_{ij}$ is significantly high. The p -value statistical test will give more meaning to the observed $\widehat{Jac}(\hat{f}_\ell)$ value.

3.2.3 p -value

Several ways exist to infer a GRN from an importance measures, as $\widehat{Jac}(\hat{f})_{ij}$ can be seen as a measure of the importance of gene j to predict gene i —related to saliency measures. One can directly use the importance measure: for example, one can assume that j regulates i if the importance measure is above a threshold or if it is above a certain percentage of measured importances. I shall abbreviate this interpretation method by *dir*.

Another approach is to compute the p -value of $\widehat{Jac}(\hat{f})_{ij}$. This is the probability of observing such an importance value under the null hypothesis that gene i and j are independent. For example, one can assume that gene j regulates i if the p -value of observed measured importance of gene j for gene i is smaller than 5% or 1%. I obtain the p -values by a permutation test, theorized by Fisher [102] and Pitman [103]. This test, also called randomization test or re-randomization test, is an exact nonparametric test, making no assumptions on underlying distributions, thus it is legitimate to apply it here. Given an importance measure, *Impo*, a training set $\mathcal{S} = (\mathbf{x}_i, y_i)_{i=1\dots N}$, the test is performed as follows:

1. Randomly choose π , a permutation of $\{1, \dots, N\}$. $\mathbf{x}_{\pi(i)}$ is now independent of y_i
2. Calculate *Impo*(j) for the dataset $(\mathbf{x}_{\pi(i)}, y_i)$.

3. Repeat step (1-2) M times -in this work, I chose $M = 10000$. We now have the distribution of $Impo(j)$ with j independent from output, hence p -values.

Among the advantages of the permutation test are its theoretical soundness and the clear interpretability of its results. Its main drawback is the associated computational time. By grouping several calculations, the permutation test may be done by performing M matrix multiplications. Its full computational cost is $O(pMN^2)$, which is the same order of magnitude than the $O(pN^3)$ cost for solving each kernel-ridge regression.

3.2.4 Voting procedure

The introduction of a permutation test provides a way to calculate p -values and use it to estimate the target matrix. Let L_{im} be the indexes of subsamples allowing to learn importance of gene m for gene i , i.e. $L_{im} = \{\ell, m \in \mathcal{G}_\ell, i \notin \mathcal{G}_\ell\}$. Let V_{meth} be the matrix such as $(V_{meth})_{im}$ is the mean vote for a regulation of gene i by gene m for method $meth$:

$$\begin{aligned}
 (V_{Impo})_{im} &= \frac{1}{|L_{im}|} \left(\sum_{\ell \in L_{im}} Impo(i, m, \ell) \right) && \text{direct use of } Impo \\
 (V_{pVal})_{im} &= \frac{1}{|L_{im}|} \left(\sum_{\ell \in L_{im}} pVal(i, m, \ell) \right) && \text{use of } Impo\text{'s } p\text{-value} \\
 (V_{pVal < 5\%})_{im} &= \frac{1}{|L_{im}|} \left(\sum_{\ell \in L_{im}} (pVal(i, m, \ell) < 5\%) \right) && \text{subsample vote 1} \\
 &&& \text{iff } pVal(i, m, \ell) \leq 5\% \\
 (V_{pVal \leq 1\%})_{im} &= \frac{1}{|L_{im}|} \left(\sum_{\ell \in L_{im}} (pVal(i, m, \ell) \leq 1\%) \right) && \text{subsample vote 1 iff} \\
 &&& pVal(i, m, \ell) \leq 1\%
 \end{aligned}$$

The adjacency matrix is estimated by method $meth$ by inferring an edge from variable m to variable j if $(V_{meth})_{mj}$ exceeds a threshold $\theta \geq 0$

$$\hat{a}_{mj} = 1_{\theta((V_{meth})_{mj})} . \quad (3.15)$$

3.3 Other measures for feature importance in a model

I proposed one way to measure the importance of a feature in a model using the model's partial derivatives. Alternatively, other measures can be used and are

compared to the partial derivatives estimate: (a) information loss resulting from ignoring this feature, (b) the effect of this feature on the functional cost. Inspired by related works, I propose other *a posteriori* measures of feature importance of a (nonparametric) base model in the ensemble approach.

3.3.1 Feature relevance

Feature relevance has been defined in many ways (see [104] for various formulations and their shortcomings). In particular, Kohavi and John [104] have defined the notion of *strong relevance* in case of supervised classification. A feature j is *strongly relevant* if removal of this feature alone will result in performance deterioration of a Bayes optimal classifier. They give the formal definition:

Definition 7. (Strong relevance for supervised classification). *A feature Z^j is strongly relevant if and only if there exist some z^j, y and \mathbf{z}^{-j} such that*

$$p(Y = y | Z^j = z^j, Z^{-j} = \mathbf{z}^{-j}) \neq p(Y = y | Z^{-j} = \mathbf{z}^{-j}) \quad (3.16)$$

In case of regression, a similar definition would involve the regression function: $\mathbf{z} \rightarrow \mathbb{E}[Y|\mathbf{z}]$, being optimal according to the mean squared error criterion.

Definition 8. (Strong relevance for regression). *A feature Z^j is strongly relevant if and only if there exist some z^j, y and \mathbf{z}^{-j} such that*

$$\mathbb{E}[Y | Z^j = z^j, Z^{-j} = \mathbf{z}^{-j}] \neq \mathbb{E}[Y | Z^{-j} = \mathbf{z}^{-j}] \quad (3.17)$$

However, this optimal model is not available. Therefore, I suggest to evaluate the relevance of a feature j by the change in the mean-square error of the kernel-ridge regression based on all features, and based on all features but j . Let \hat{f} be the regression function trained on all features, and \hat{f}^{-j} the one trained on all features but j . Let \mathcal{S}_{train} be the set of data used for training \hat{f} and \hat{f}^{-j} . Let \mathcal{S}_{test} be the test data. I define an empirical relevance measure based either on training set or on test set.

$$rel_{train}(j) = \frac{1}{|\mathcal{S}_{train}|} \sum_{(\mathbf{z}, y) \in \mathcal{S}_{train}} \left[(y - \hat{f}(\mathbf{z}))^2 - (y - \hat{f}^{-j}(\mathbf{z}^{-j}))^2 \right], \quad (3.18)$$

$$rel_{test}(j) = \frac{1}{|\mathcal{S}_{test}|} \sum_{(\mathbf{z}, y) \in \mathcal{S}_{test}} \left[(y - \hat{f}(\mathbf{z}))^2 - (y - \hat{f}^{-j}(\mathbf{z}^{-j}))^2 \right]. \quad (3.19)$$

$$(3.20)$$

If feature j is relevant, there will be a loss of information; \hat{f}^{-j} should be less accurate than \hat{f} , and $rel_{train}(j)$ and $rel_{test}(j)$ will be positive. Note that, as we are not using the regression $E[Y = y|\mathbf{z}]$ optimal function, $rel_{test}(j)$ may be negative. If feature j is irrelevant, its removal from training data will reduce noise, and allow \hat{f}^{-j} to be a better regression function than \hat{f} .

3.3.2 Sensitivity analysis

Le Cun et al. [105] suggest that the sensitivity of a learning algorithm be measured for each feature. In their work, a feature's importance is the change in the functional cost caused by the removal of the feature. Following Guyon et al. [106], I measure the sensitivity as follows:

- Train a regression function $\hat{f}(\mathbf{z}) = \sum_{\mathbf{z}_i \in \mathcal{S}_{train}} \hat{\alpha}_i k(\mathbf{z}_i, \mathbf{z})$ with algorithm \mathcal{K} . Calculate the kernel-ridge functional cost

$$\mathcal{L}(0) = \frac{1}{|\mathcal{S}_{train}|} \sum_{(\mathbf{z}_i, y_i) \in \mathcal{S}_{train}} \left(y_i - \sum_{\mathbf{z}_j \in \mathcal{S}_{train}} \hat{\alpha}_j k(\mathbf{z}_j, \mathbf{z}_i) \right)^2 \quad (3.21)$$

$$+ \lambda \sum_{(\mathbf{z}_i, \mathbf{z}_j) \in \mathcal{S}_{train}^2} \hat{\alpha}_j \hat{\alpha}_i k(\mathbf{z}_j, \mathbf{z}_i) \quad (3.22)$$

- With the same coefficients $\hat{\alpha}$, calculate the functional cost with kernel k_{-j} , the adaptation of kernel k to use all features but feature j

$$\mathcal{L}(j) = \frac{1}{|\mathcal{S}_{train}|} \sum_{(\mathbf{z}_i, y_i) \in \mathcal{S}_{train}} \left(y_i - \sum_{\mathbf{z}_j \in \mathcal{S}_{train}} \hat{\alpha}_j k_{-j}(\mathbf{z}_j, \mathbf{z}_i) \right)^2 \quad (3.23)$$

$$+ \lambda \sum_{(\mathbf{z}_i, \mathbf{z}_j) \in \mathcal{S}_{train}^2} \hat{\alpha}_j \hat{\alpha}_i k_{-j}(\mathbf{z}_j, \mathbf{z}_i) \quad (3.24)$$

- Define the sensitivity of algorithm to feature j by:

$$sens(j) = \mathcal{L}(0) - \mathcal{L}(j) \quad (3.25)$$

The value of $sens(j)$ quantifies how feature j affects the functional cost using all features $\mathcal{L}(0)$. Note that this measure follows [106], but one may also consider its normalized variant $(\mathcal{L}(0) - \mathcal{L}(j))/\mathcal{L}(0)$.

These two importance measures will also be implemented with the previously described ensemble methods.

3.4 Hyperparameter selection

Several parameters need to be fixed. The kernel function used is the Gaussian kernel

$$k(z, z') = \exp\left(-\frac{\|z - z'\|_2^2}{2\sigma^2}\right) \quad (3.26)$$

since it is a very popular kernel, with good performance in practice and universal consistency in theory. There remains four hyper-parameters to choose:

- For the subsampling parameters, cross-validation would be a biased procedure. If a model was trained on more data, it would have lower test error. Moreover, by using fewer variables, many models would not have any relevant feature as explanatory variable, and would have a high test error. Our experiments have shown that the subsampling parameters n_{var} and p_{data} have little influence on the GRN inference performance, see Figure 3.5. I chose $n_{var} = 5$ and $p_{data} = 80\%$.
- The regularization-data fitting tradeoff λ and kernel bandwidth σ are selected through cross-validation. With subsampling, test error can be calculated by out-of-bag data, i.e. data not selected in the subsample. Performing cross-validation for M values of λ only costs M times the cost of performing the algorithm one time. σ can take the following values $\{0.1, 0.5, 1, 2, 5\}$. With E_0 the largest dataset, $p_{data}|E_0|$ is the average size of a subsample. Let $s = 1/\sqrt{p_{data}|E_0|}$. λ can take the following values $\{0.01s, 0.1s, s, 10s, 100s\}$.

I consider that $B = \frac{100 \times p}{n_{var}}$ is enough subsamples, the vote matrix has converged.

3.5 Experiments on small-scale networks

3.5.1 Data: DREAM3 Challenge

The Dialogue for Reverse-Engineering Assessment and Methods (DREAM) challenge allows researchers' team to compete on bioinformatic problems. In the 3rd edition [82, 107], the 4th challenge focused on gene regulatory network inference from realistically simulated networks. The networks are modules from known networks of *E.Coli* and *Yeast* [108]. Data were generated with GeneNetWeaver [109]. With realistic simulations, one knows the real network behind the data, thus one

can confidently compare the performances of network inference methods. The challenge provides networks of size 10, 50 and 100 genes. For each size, there are 5 networks, 2 extracted from *E. Coli* and 3 from *Yeast*. For each network, there is a knock-out and a knock-down experiment for every gene, and, for size 10 networks (resp. 50, 100), 4 (resp. 23, 46) time-series of 21 observation points. Knock-down and knock-out experiments bring a lot of information by their meta-information (which gene is perturbed). The challenge's organizers showed that best performances could be reached by only using the perturbed data and Z-score (described in section 1.4.1, page 28). As our methods do not integrate this meta-information, they under-perform with respect to methods using the complete information. Thus, I ignore the perturbed data and learn the network using only time-series data. I compare our results to the best teams of the DREAM3 challenge also using only time-series data.

3.5.2 Performance evaluation

Given a GRN inference method, one obtains a score for each interaction and assumes that there is a regulation from gene i on gene j if $\hat{A}_{j,m}$ is superior to a threshold θ . Noting R the number of predicted regulations, this prediction is compared with gold standard. By counting the number of true positives (TP), false positives ($FP = R - TP$), false negatives (FN) and true negatives (TN), performance can be calculated as precision, i.e. the percentage of real regulations in the inferred regulations ($pre = TP/(TP + FP)$), recall, i.e. percentage of all real retrieved regulations ($rec = TP/(TP + FN)$), or false positive rate, i.e. percentage of non-regulation predicted as regulation ($FPR = FP/(FP + TN)$).

These statistics depend on the threshold θ . To avoid the problem of poorly chosen θ , one draws the "Receiver Operating Characteristic" curve ($ROC: (FPR, pre) = f(\theta)$) and the precision-recall curve ($PR: (pre, rec) = f(\theta)$). The better the method, the higher the area under these curves. For example, with a perfect predictor, the area under the ROC curve will be 1. With a random predictor, it will be 0.5.

3.5.3 Performance without ensemble methods

These results are shown to complete this work, in Figure 3.6. Results are, as expected, lower if no ensemble method is used. Performances of each importance measure on 50-genes networks are shown in Figure 3.6, using static or dynamic

modeling. Overall, static modeling exhibits better performance for both AUROC and AUPR. No importance measure has good performance.

3.5.4 Analysis of p -value

The p -value is the probability of observing such an extreme value under an assumption of independence between input variable \mathbf{x} and output y . Thus, some false positive should occur. Considering that a regulation from gene j to gene i exists if the p -value of $Impo(i, j)$ is inferior or equal to θ ; calculating M importance measures, we expect θM false regulations, noted EFP, expected false positives. Noting P , for positive, the number of edges considered, the ratio P/EFP quantifies how this importance measure produces extreme values compared to values obtained with features independent from the output. If this ratio is below 1, it means that this importance measure does not produce extreme values. This does not mean that the measure does not rank features well.

We see in Table 3.1 the values of the mean of the E/EFP ratio on all 5 networks, and on how many networks this ratio was superior to 1 for the two thresholds $\theta = 5\%$ and $\theta = 1\%$. From this perspective, both sensitivity analysis and partial derivatives' integral do not produce extreme values. On the contrary, relevance on the training data seems to clearly separate independent and dependent cases. Surprisingly, relevance on test data is not as extreme. This difference may arise from the difficult nature of the data. We have few noisy and incomplete data; for example, data contains only RNAm concentrations, and no protein concentration or exogenous stimuli. This may cause the poor generalization of our model, thus the irrelevance of measuring mean-square error on test data.

The plot in Figure 3.2 shows the performance of the adjacency matrix estimate V_{meth} as a function of the number of subsamples, on the DREAM3 size 50 networks for two importance measures: relevance on training data and integral of partial derivatives. The behavior is similar for all four importance measures. For both importance measures, we observe that $V_{,dir}$ and $V_{,pVal}$ converge faster than $V_{,pVal<5\%}$ and $V_{,pVal<1\%}$. Also, with the two latter methods, we observe some gaps, which occur sometimes after many subsamples. Thus, the direct and $pVal$ methods appears to perform better, as far as speed of convergence is concerned.

| Edge threshold | $pVal \leq 5\%$ | | $pVal \leq 1\%$ | |
|-----------------|-----------------|---------------------|-----------------|---------------------|
| Method | P/EFP | $\#\{(P/EFP) > 1\}$ | P/EFP | $\#\{(P/EFP) > 1\}$ |
| \widehat{Jac} | 0.53 | 1 | 2.27 | 3 |
| rel_{test} | 1.41 | 5 | 5.71 | 5 |
| rel_{train} | 3.86 | 5 | 16.34 | 5 |
| $sens$ | 0.24 | 0 | 1.02 | 2 |

Table 3.1: Using a threshold p -value of $Impo < \theta\%$, one expects to obtain false positives, precisely $\theta\%$ times number of $Impo$ calculated. Noting EFP the number of Expected False Positives, the table shows the mean ratio of number of p -values superior to threshold divided by the EFP and gives the number of networks (out of 5) for which this ratio exceeded 1.

3.5.5 Performances on DREAM3 50-gene and 100-gene Networks

On size 50 datasets, results are shown in Table 3.4 for AUROC and in Table 3.5 for AUPR. Note that, aside from relevance on training data, using the importance measure directly or through p -values makes very little difference. For relevance on training data, p -values seem to enhance performance, but the change is very small. Figure 3.3 compares each directly used importance measure. Partial derivatives's integral clearly stands out of other methods. On four networks, this method has the highest AUROC; on three networks, it has the highest AUPR. Moreover, \widehat{Jac} has a large difference with second best methods on some networks (+0.09, +0.12 AUROC on networks Ecoli1 and 2), and it is never far from the best performing method (in the worst cases, \widehat{Jac} has -0.01 AUROC and -0.015 AUPR with respect to the best performing method).

Those results are confirmed on 100-gene networks. \widehat{Jac} has the best AUROC on four datasets, and best AUPR on all five. Note that these are difficult datasets. I compared \widehat{Jac} with other GRN inference methods: TIGRESS [45], GENIE3 [78], a dynamic Bayesian method, called G1DBN [110], and the best DREAM3 team using only the time-series data. Results are shown in Figure 3.4. \widehat{Jac} is the best of the five methods in four datasets out of five both in AUROC and AUPR. This shows that a naive implementation of estimation of partial derivatives is very competitive with state-of-the-art methods.

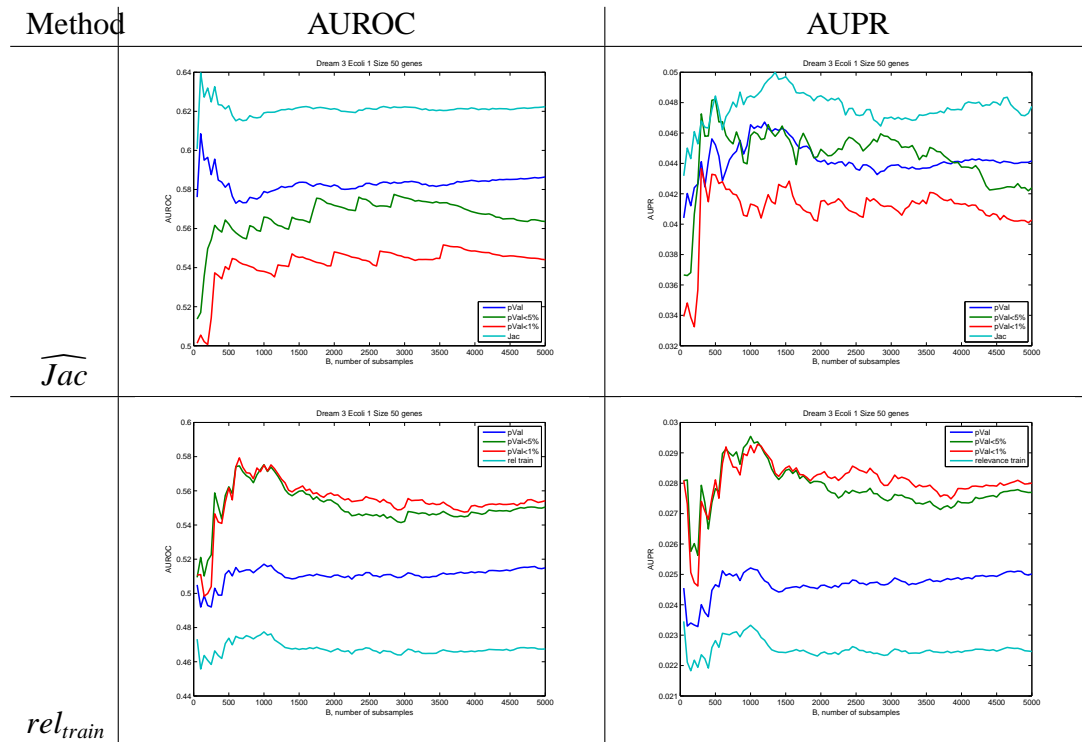


Figure 3.2: Evolution of AUROC and AUPR of V_{meth} as a function of the number of subsamples B , on DREAM3 size 50 EColi1 network

Section 3.6: Experiments on real and real-sized networks

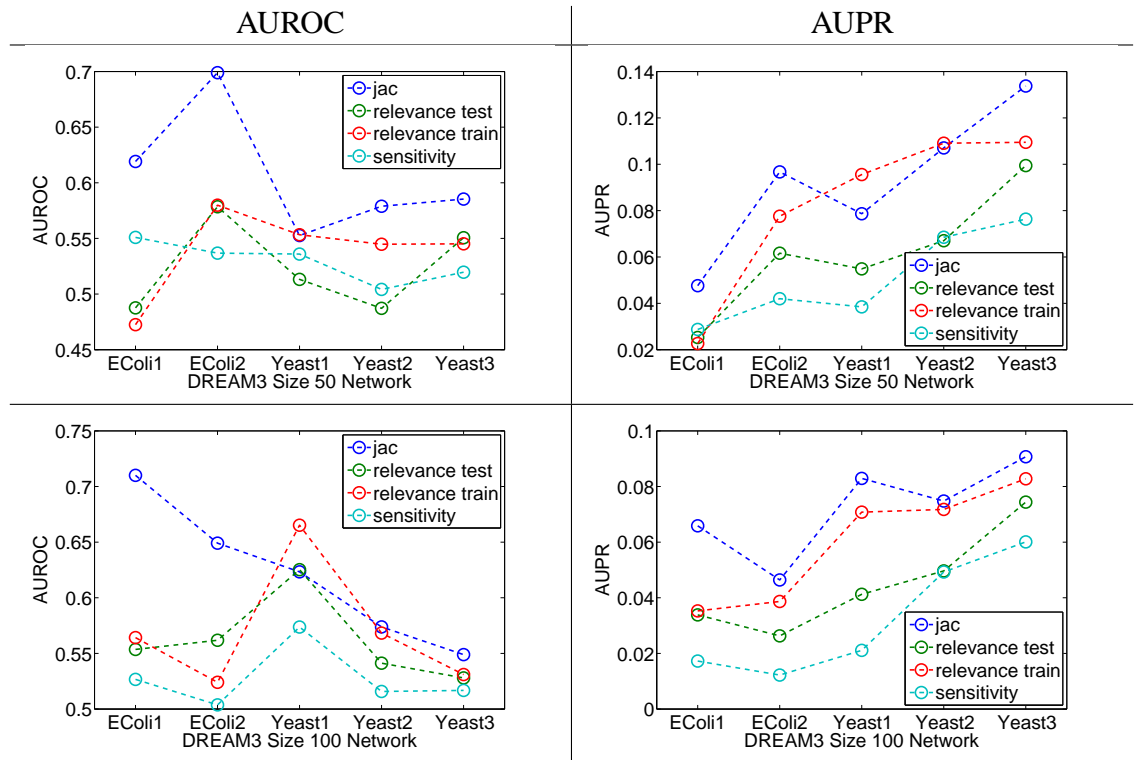


Figure 3.3: Comparison of importance measures on DREAM3 size 50 and size 100 networks

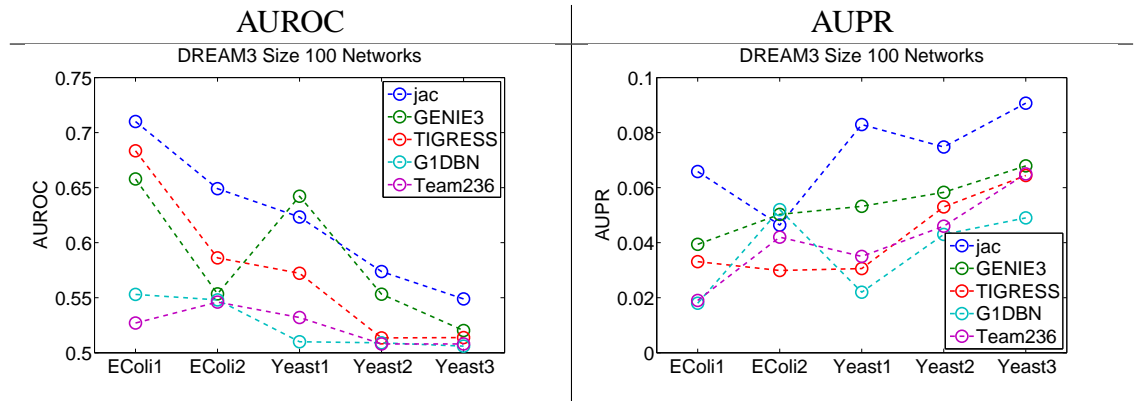


Figure 3.4: On DREAM3 size 100 networks, comparison of \widehat{Jac} against G1DBN, TIGRESS, GENIE3 and best DREAM3 performer using only time-series data.

| Network | # TF | p | N | N_{dyn} |
|--|------|------|-----|-----------|
| DREAM5 Network 1 (in-silico) | 195 | 1643 | 805 | 463 |
| DREAM5 Network 3 (<i>E.Coli</i>) | 334 | 4511 | 805 | 463 |
| DREAM5 Network 4 (<i>S.cerevisiae</i>) | 333 | 5950 | 536 | 298 |

Table 3.2: Characteristics of datasets: # TF is the number of potential transcription factors, p is the number of genes of the system, N is the number of data points, N_{dyn} is the number of data from time-series, among the N data

3.6 Experiments on real and real-sized networks

\widehat{Jac} importance measure has been evaluated on four well-referenced benchmarks. The scales and sizes of the datasets are summarized in Table 3.2.

3.6.1 Datasets

The datasets were found in the DREAM5 challenge [41]. The first dataset is a realistic simulation of genes—see [108] and [109] for a complete explanation—while the last two are real data. For each dataset, gene expression data of heterogeneous types is available: steady-state, time-series and perturbed data. We have a list of potential regulators, called “*Transcription Factors*” (TF). The gold standard is given by the DREAM organizers.

I compared \widehat{Jac} with state-of-the-art methods. DREAM5 organizers provided the contestants’ inferred networks. I compared my results to those of the best contestants.

3.6.2 Using other GRN Inference methods: A consensus approach

One talks of *wisdom of crowds* [111] when a group, using the vote of each individual, makes better decisions than any individual alone. This phenomenon is well illustrated in gene regulatory network inference. Using different methods, the mean of all inferred networks is often closer to reality than each network taken alone, as Marbach et al. have shown with the DREAM5 challenge [41].

To average each prediction, Marbach et al. sorted, for each method, which link was considered most likely. They had n_J inference methods. Method i gives a score for each interaction, score stored in $B_i \in \mathbb{R}^{p \times p}$ the score matrix of method i .

The higher the score, the more likely this interaction is, according to the method. Marbach et al. [111] ranked each interaction in the matrix R_i .

$$(R_i)_{j,m} = r \quad (3.27)$$

$$\text{s.t.} \quad \exists (j_b, m_b)_{b=1 \dots r-1} (B_i)_{j_b, m_b} \geq (B_i)_{j,m} \quad (3.28)$$

$$\exists (j_b, m_b)_{b=r+1 \dots p \times p} (B_i)_{j_b, m_b} \leq (B_i)_{j,m} \quad (3.29)$$

and then built the mean average over all network inference method,

$$R = \frac{1}{n_J} \sum_{i=1}^{n_J} R_i. \quad (3.30)$$

To obtain an adjacency matrix, one can threshold the matrix R to obtain only a given percentage of most likely interactions.

One hopes that kernel-based models infer well gene regulatory network, but also that this modelling method, as it is very different from others, will contribute well to enhance the GRN inference when used in a consensus method. For real-sized networks, I compared the performances of the consensus with and without the network inferred by the importance measures obtained in the previous subsection.

3.6.3 Performance

Results are shown in Table 3.3. While \widehat{Jac} does reach best performance for AU-ROC for two networks out of three, it shows very poor performances for AUPR. \widehat{Jac} obtains interesting performance on the difficult and interesting problem of GRN inference, but does not seem competitive with state-of-the-art methods as an individual method.

When observing the consensus, we observe a substantial increase in performance by adding \widehat{Jac} to other methods. Besides, \widehat{Jac} accounts for only 1 vote out of 30, so the performance increase is noteworthy. Marbach et al. have shown that consensus worked better if the methods gathered relied on different models, grasping different types of information. It seems that \widehat{Jac} , by its Gaussian kernel and partial derivatives approach, which are both novel, acquire an information that was not captured before.

3.7 Conclusion

In this work, I proposed a reverse-engineering GRN method not by learning an interpretable model, but by interpreting a Gaussian kernel-based model. From ex-

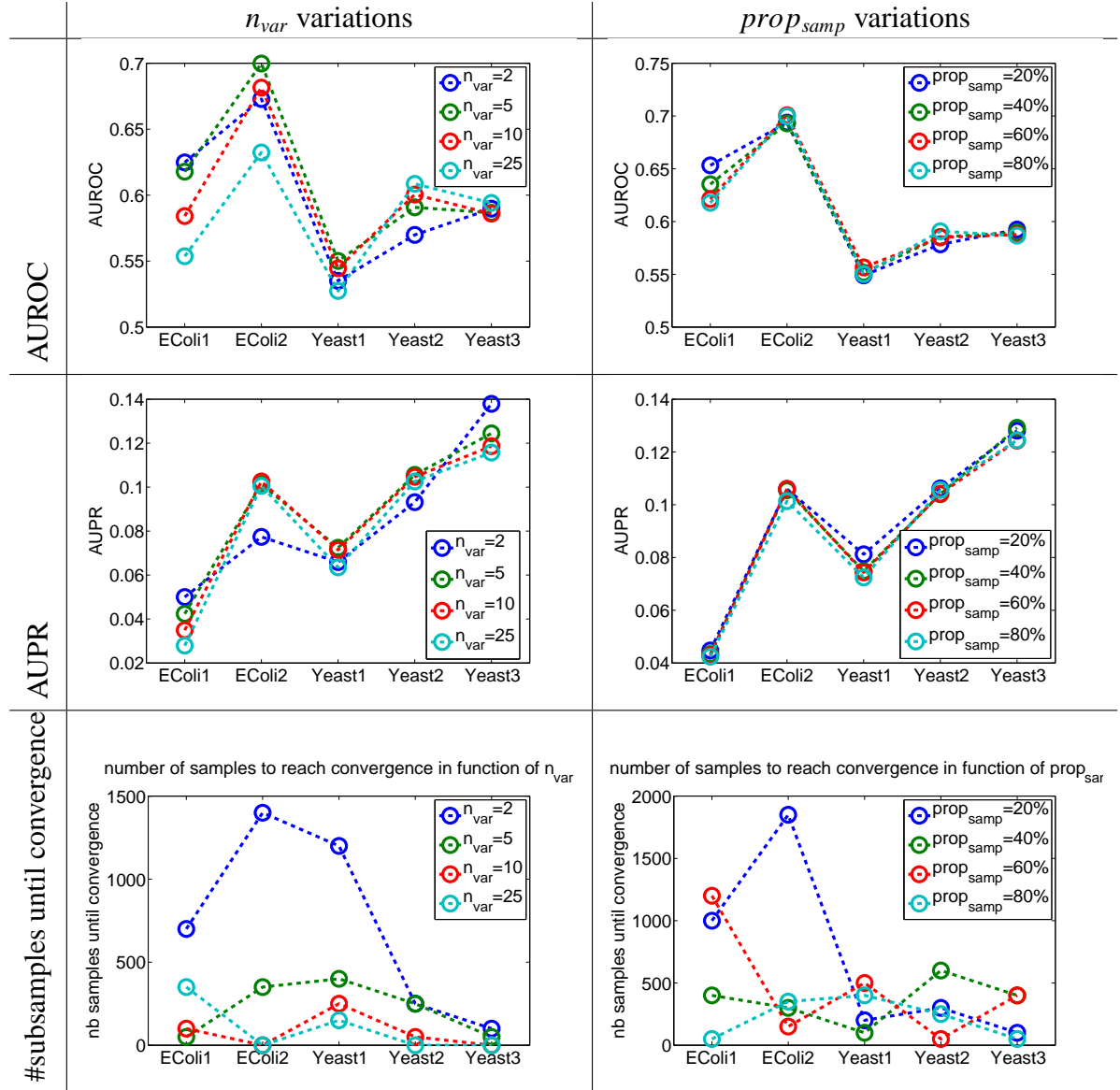


Figure 3.5: Sensivity of \widehat{Jac} to parameters p_{data} and n_{var} on DREAM3 size 50 networks

| networks | DREAM 5 N1 | | DREAM 5 N3 | | DREAM 5 N4 | |
|-------------------|-------------|--------------|--------------|------------|--------------|-------------|
| | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR |
| Individual | | | | | | |
| \widehat{Jac} | 71.89 | 8.64 | 63.19 | 3.47 | 52.57 | 2.21 |
| GENIE3 | 81.5 | 29.1 | 61.7 | 9.3 | 51.8 | 2.1 |
| TIGRESS | 78.2 | 30.1 | 59.5 | 6.9 | 51.7 | 2.0 |
| CLR | 77.3 | 25.5 | 59.0 | 7.5 | 51.6 | 2.1 |
| Consensus | | | | | | |
| Consensus | 80.89 | 32.65 | 64.94 | 8.99 | 52.02 | 2.24 |
| $C+\widehat{Jac}$ | 81.23 | 31.81 | 72.78 | 8.75 | 53.48 | 2.30 |

Table 3.3: Performance measured as AUROC and AUPR in % on DREAM5 Networks 1, 3 and 4. The results in bold font are the best ones. The first row is the consensus of other methods which are all the participating teams on DREAM5 and GENIE3, TIGRESS and CLR on *EColi* .

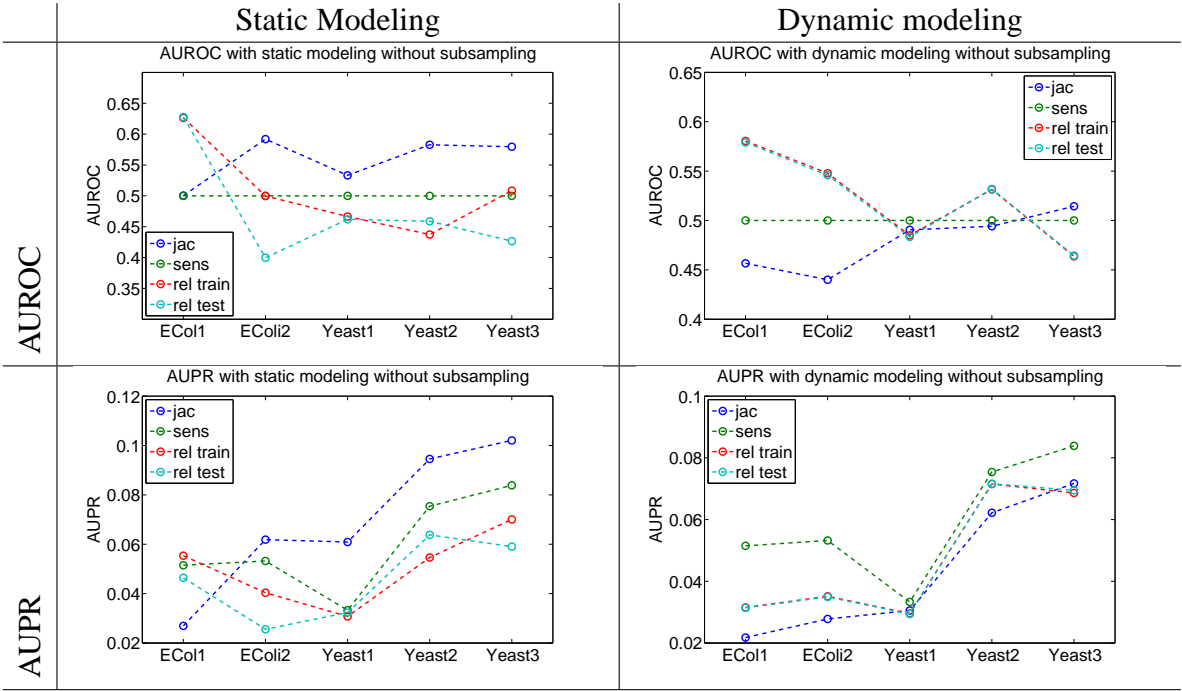


Figure 3.6: Results of importance measures without subsampling, using either static or dynamic modeling

| Network | EColi1 | EColi2 | Yeast1 | Yeast 2 | Yeast 3 |
|---|-------------|-------------|-------------|-------------|-------------|
| Importance measure: integral of partial derivatives | | | | | |
| \widehat{Jac} | 0.62 | 0.70 | 0.54 | 0.58 | 0.58 |
| pVal | 0.58 | 0.69 | 0.56 | 0.58 | 0.57 |
| pVal \leq 5% | 0.56 | 0.65 | 0.58 | 0.54 | 0.57 |
| pVal \leq 1% | 0.54 | 0.63 | 0.58 | 0.54 | 0.57 |
| Importance measure: relevance on test data | | | | | |
| rel_{test} | 0.49 | 0.55 | 0.51 | 0.49 | 0.54 |
| pVal | 0.48 | 0.5 | 0.52 | 0.42 | 0.51 |
| pVal \leq 5% | 0.47 | 0.5 | 0.51 | 0.43 | 0.52 |
| pVal \leq 1% | 0.47 | 0.5 | 0.51 | 0.43 | 0.51 |
| Importance measure: relevance on train data | | | | | |
| rel_{train} | 0.46 | 0.58 | 0.55 | 0.54 | 0.55 |
| pVal | 0.51 | 0.62 | 0.56 | 0.56 | 0.56 |
| pVal \leq 5% | 0.55 | 0.63 | 0.53 | 0.55 | 0.56 |
| pVal \leq 1% | 0.55 | 0.63 | 0.52 | 0.55 | 0.56 |
| Importance measure: sensitivity analysis | | | | | |
| pVal | 0.53 | 0.53 | 0.55 | 0.50 | 0.53 |
| pVal \leq 5% | 0.51 | 0.49 | 0.50 | 0.50 | 0.51 |
| pVal \leq 1% | 0.51 | 0.48 | 0.50 | 0.51 | 0.51 |
| $sens$ | 0.55 | 0.55 | 0.53 | 0.51 | 0.53 |

Table 3.4: AUROC results on the DREAM3 size 50 networks of presented network inference methods using a Gaussian kernel

| Network | EColi1 | EColi2 | Yeast1 | Yeast 2 | Yeast 3 |
|---|--------------|--------------|--------------|--------------|-------------|
| Importance measure: integral of partial derivatives | | | | | |
| \widehat{Jac} | 0.047 | 0.094 | 0.078 | 0.108 | 0.132 |
| pVal | 0.044 | 0.106 | 0.078 | 0.107 | 0.14 |
| pVal \leq 5% | 0.042 | 0.105 | 0.08 | 0.107 | 0.136 |
| pVal \leq 1% | 0.04 | 0.103 | 0.081 | 0.1 | 0.136 |
| Importance measure: relevance on test data | | | | | |
| rel_{test} | 0.025 | 0.055 | 0.051 | 0.068 | 0.098 |
| pVal | 0.026 | 0.033 | 0.035 | 0.052 | 0.074 |
| pVal \leq 5% | 0.025 | 0.033 | 0.034 | 0.054 | 0.073 |
| pVal \leq 1% | 0.025 | 0.033 | 0.034 | 0.054 | 0.072 |
| Importance measure: relevance on train data | | | | | |
| rel_{train} | 0.022 | 0.077 | 0.093 | 0.11 | 0.109 |
| pVal | 0.025 | 0.068 | 0.105 | 0.111 | 0.134 |
| pVal \leq 5% | 0.027 | 0.069 | 0.10 | 0.109 | 0.128 |
| pVal \leq 1% | 0.028 | 0.07 | 0.093 | 0.109 | 0.12 |
| Importance measure: sensivity analysis | | | | | |
| $sens$ | 0.029 | 0.043 | 0.039 | 0.067 | 0.077 |
| pVal | 0.027 | 0.039 | 0.050 | 0.065 | 0.078 |
| pVal \leq 5% | 0.027 | 0.035 | 0.035 | 0.070 | 0.074 |
| pVal \leq 1% | 0.027 | 0.034 | 0.035 | 0.070 | 0.074 |

Table 3.5: AUPR results on the DREAM3 size 50 networks of presented network inference methods using a Gaussian kernel

periments on realistic datasets, integrals of partial derivatives have shown promising results. This interpretation of a model \hat{f} can be used with any model type (preferably fitting the conditions of the proof).

The \widehat{Jac} importance measure has shown that it could deal with real-size networks, and provide good results. It seems that this approach captures some interactions which were overlooked by previously established methods.

For future work, one can use the \widehat{Jac} measure with regression methods that are more specific to gene modeling. One can add constraints on the functional cost to incorporate prior information, or change the regression model, for example operator-valued kernels, that learn structured output. Particular mention should be made of the work of N.Lim et al. [77] that shows very good performances on dynamic data, using partial derivatives of an operator-valued kernel model.

Part II

Kernel Feature Weighting for Network Inference

In the previous part, we saw a method to interpret a kernel-based model. In the next part, I design an interpretable kernel model, a method that can model nonlinear behavior and can estimate which genes are regulating the target gene as directly as linear model. In addition, this method produce models whose gradient is sparse, thus the kernel model may be better suited to interpretation through partial derivatives than the one obtained with kernel-ridge regression. An additional advantage of this method is its ability to incorporate prior knowledge. Such a method is made possible by the versatility of kernels. Given M different kernels k_1, \dots, k_M , a positive linear combination of these kernels, $k^* = \sum_m w_m k_m$, with $w_m \geq 0$ for all $m = 1 \dots M$, is a kernel. In particular, defining *local kernel* or *component kernel* as a kernel function that uses one feature of the input vector $k_m(\mathbf{x}, \mathbf{z}) = k(x^m, z^m)$, the weights w_m measure the importance of gene m for the target gene. Using kernel k^* on the learning set $\mathcal{S} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, the model will have the following form:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{\ell=1}^N \alpha_{\ell} \left(\sum_{m=1}^M w_m k_m(\mathbf{x}, \mathbf{x}_{\ell}) \right) \\ &= \sum_{m=1}^M w_m g_m(\mathbf{x}^m) \end{aligned}$$

The vector \mathbf{w} will be learned from the training set, using multiple kernel learning. In particular, this additive model can be seen as an extension of the linear model to additive nonlinear dynamics.

The vector \mathbf{w} can be seen as feature weighting. In Chapter 4, I describe several feature weighting or feature selection methods that can take into account nonlinear dynamics. On realistic datasets, the *local kernel* approach performs better than other feature selection methods. In Chapter 5, I compare my *local kernel* approach, called LocKNI (LOCAL Kernels for Network Inference), to state-of-the-art methods on real and realistic datasets. LocKNI gives state-of-the-art results. Besides, when used with other methods in a “wisdom of crowds” approach, LocKNI substantially enhances the performance of the consensus, hinting that *local kernels* grasp information that is not captured by other models. Finally, I provide a simple approach to incorporate prior knowledge to the multiple kernel learning approach, and use two reasonable biological sources of prior knowledge that greatly enhance LocKNI’s performance on a realistic dataset.

Chapter 4

Local Kernels and feature selection

4.1 Introduction

In our era of data abundance, the scale of supervised learning problems has changed. Whereas few domains used more than 40 variables in 1997, today, many papers explore domains with hundreds to tens of thousands of variables. For example, text categorization [112], image classification [113] or cancer prognostic from genomic data [114, 115] deal with thousands of variables for a classification task. Reducing the problem size has many benefits: from a computational point of view, it reduces data storing requirements and computational time; from a human point of view, it helps understand and visualize the problem; from a pragmatic point of view, it helps defy the curse of dimensionality [6].

In this context, researchers have developed feature selection methods. Among all the variables, many may be redundant or noisy features, see [116, 104] for a formal definition of relevant features. In many cases, the best subset of features is the one that minimizes the generalization error. Feature selection methods can be decomposed in three categories [117]:

- *Filtering methods.* They constitute a preprocessing step, independent of the choice of the learning algorithm. One evaluates the dependence between the features and the value to predict through a pre-defined metric, such as mutual information or correlation. Generally, one uses this metric to rank features, and then select the most informative features. Filters are the fastest feature selection methods. Two classical filtering examples are Fisher's linear discriminant [118] and mutual information [119]

- *Embedded methods*. These machine learning methods simultaneously select relevant features and build the prediction function. While slower than filters, these methods select the best features for a specific model. LASSO [38] and decision trees, as viewed in CART [120], are examples of such methods
- *Wrapper methods*. These methods take into account the model of prediction functions and are the most general, but also the slowest methods. They consist in selecting a subset of features F , learning a prediction function using F , and then training a prediction function on other subset of features F' until a subset F^* is found that is optimal with respect to a criterion, such as accuracy of the prediction function. Sequential Forward Selection [104] is a wrapper method.

Feature weighting -where features is given a weight from 0 to 1- can be seen as an extension of feature selection -where feature are either useful (weight is 1) or ignored (weight is 0)-. The field of gene regulatory network (GRN) inference from gene expression data is keen of feature selection or feature weighting methods. Using these methods, one can evaluate which genes are relevant for the prediction of a target gene, and consequently assume that they regulate this target gene. Filtering methods [26, 27] and linear [45, 39], Boolean [50, 48], Bayesian [66] or tree-based [78] embedded feature selection methods have been successfully used for GRN inference.

In this context, I propose a new feature weighting methods. Using the versatility of kernels, I define *local kernels*, that rely on only one feature. I propose several methods to optimize *local kernel*-based models. Moreover, I test various popular kernel feature selection or feature weighting methods along with the *local kernel* approach on realistically simulated GRN. Firstly, I describe kernel feature selection methods, and how to use them for GRN inference; Secondly, I describe hyper-parameter selection methods. Finally, I show the experiments and experimental results.

4.2 Feature weighting methods

In order to infer a GRN from gene expression data, given training data $\mathcal{S} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, N observations of the expression level of a p -gene system, one can learn a static or dynamic model. If each gene is modeled independently, a feature weighting/selection algorithm may be used to weight the importance of other genes for the prediction of a target gene. One can learn

$$\text{a static model} \quad x^i = f_{stat}^i(\mathbf{x}^{-i}) + \epsilon \quad (4.1)$$

$$\text{a dynamic model} \quad x^i(t + \tau) = f_{dyn,\tau}^i(\mathbf{x}(t)) + \epsilon \quad (4.2)$$

Using a feature weighting (resp. selection) algorithm in learning f^i , one obtains the vector of weights $\mathbf{w}_i \in \mathbb{R}_+^p$ (resp. $\mathbf{w}_i \in \{0, 1\}^p$) that evaluates the importance of each gene for the prediction of gene i . The estimated adjacency matrix is the juxtaposition of these feature weights:

$$\hat{A} = (\mathbf{w}_1, \dots, \mathbf{w}_p)^T \quad (4.3)$$

4.2.1 Local Kernel Approach

In order to have a nonlinear interpretable model, I suggested to use *local kernels* or *component kernels*, k_1, \dots, k_M , kernel k_m relying only on feature m of input variable. Then, a weighted combination k^* of those kernels is used as a kernel; $k^* = \sum_{m=1}^M w_m k_m$, with weights \mathbf{w} learned from the training set. This produces a model of the form:

$$\hat{f}(\mathbf{x}) = \sum_{\ell=1}^N \alpha_\ell \left(\sum_{m=1}^M w_m k_m(\mathbf{x}, \mathbf{x}_\ell) \right), \quad (4.4)$$

$$= \sum_{m=1}^M w_m g_m(\mathbf{x}^m), \quad (4.5)$$

with a sparse vector \mathbf{w} . This additive model has several advantages: firstly, it is able to model nonlinear dynamics in order to reflect the true behavior of biological systems. Secondly, by controlling the sparsity of \mathbf{w} , the sparsity of the partial derivatives of \hat{f} is controlled ($\partial \hat{f} / \partial x^m = 0$ if $w_m = 0$). Thirdly, extraction of regulators is as direct as in linear models, vector \mathbf{w} quantifying each feature's importance. This model can be seen as an extension of linear models.

The vector \mathbf{w} is learned from data, using Multiple kernel learning (MKL). MKL was introduced in [121] for protein function prediction. MKL has improved state-of-the-art classification performances in various domains, such as bioinformatics [121, 122], image classification [123, 113], or sound localization [124]. For a review of MKL algorithm, I recommend [125]. In this chapter, I will use three MKL algorithms: SimpleMKL [126], $\ell_2 - MKL$, developed in this thesis, and Support vector Parsimonious ANOVA (SUPANOVA) [127].

SimpleMKL In this algorithm, one obtains a linear combination of kernels by minimizing a functional cost with an ℓ_1 -constraint on the kernel weights \mathbf{w} : $\mathbf{w}_m \geq 0$ for all m and $\sum_m w_m = 1$. These constraints produce sparse weights \mathbf{w} , thus giving interpretable feature selection.

To define the functional cost, we notice that each kernel k_m has a reproducing kernel Hilbert space \mathcal{H}_m . For any function $f \in \mathcal{H}_m$, one has $\langle f, k_m(\mathbf{x}, \cdot) \rangle_{\mathcal{H}_m} = f(\mathbf{x})$. In particular, for any function of the following type $f(\mathbf{x}) = \sum_i \alpha_i k_m(\mathbf{x}_i, \mathbf{x})$, one has $\|f\|_{\mathcal{H}_m}^2 = \sum_{i,j} \alpha_i \alpha_j k_m(\mathbf{x}_i, \mathbf{x}_j) = \alpha^T K_m \alpha$. A. Rakotomamonjy [126] considers the following optimization cost:

$$\mathcal{L}(f_1, \dots, f_M, \mathbf{w}) = \sum_i l(y_i, \sum_{m=1}^M f_m(\mathbf{x}_i)) + \lambda \sum_{m=1}^M \frac{1}{w_m} \|f_m\|_{\mathcal{H}_m}^2 \quad (4.6)$$

$$\text{s.t.} \quad w_m \geq 0 \forall m, \sum_m w_m = 1 \quad (4.7)$$

If $l(\cdot, \cdot)$ is the hinge loss or the ϵ -insensitive loss, Rakotomamonjy et al. prove that solution \hat{f}_m has the form $\hat{f}_m(\mathbf{x}) = \sum_i \alpha_i w_m k_m(\mathbf{x}_i, \mathbf{x})$. With convention $\frac{0}{0} = 0$, the functional cost is define for all $\mathbf{w} \in [0, 1]^M$.

Rakotomamonjy et al. proves that the functional cost (4.6) is convex, so it has a unique minimum, which is global. They reformulate the functional cost:

$$\min_{\mathbf{w}} [J(\mathbf{w})] = \min_{\mathbf{w}} \left[\min_{f_1, \dots, f_M} \mathcal{L}(f_1, \dots, f_M, \mathbf{w}) \right] \quad (4.8)$$

$$\text{s.t.} \quad w_m \geq 0 \forall m, \sum_m w_m = 1 \quad (4.9)$$

Existence and computation of derivatives of $J(\cdot)$ are done in [126]. They obtain the minimizer of $J(\cdot)$ by a reduced gradient method, which converges for such function [128]. Once ∇J is calculated, they use a descent direction $\nabla_{red} J$ assuring that $\mathbf{w} + s \nabla_{red} J$ still satisfies the constraints (4.9). They perform line-search to find the optimal descent-step.

These reduced gradient-descent will give $\hat{\mathbf{w}}$, the minimizer of $J(\cdot)$.

$\ell_2 - MKL$ I use the MKL framework of SimpleMKL, but I change the loss function for the square error loss function $l(y_i, \sum_m f_m(\mathbf{x}_i)) = (y_i - \sum_m f_m(\mathbf{x}_i))^2$. I constrain f_m to have the form $f_m(\cdot) = \sum_i \alpha_i w_m k(\mathbf{x}_i, \cdot)$. I minimize the following functional cost:

$$\mathcal{L}(f_1, \dots, f_M, \mathbf{w}) = \sum_i (y_i - \sum_{m=1}^M f_m(\mathbf{x}_i))^2 + \lambda \sum_{m=1}^M \frac{1}{w_m} \|f_m\|_{\mathcal{H}_m}^2 \quad (4.10)$$

$$\text{s.t.} \quad f_m(\mathbf{x}) = \sum_i \alpha_i w_m k_m(\mathbf{x}_i, \mathbf{x}), \quad \forall m \quad (4.11)$$

$$w_m \geq 0 \quad \forall m, \quad \sum_m w_m = 1 \quad (4.12)$$

I alternatively optimize (a) over α with \mathbf{w} fixed and (b) over \mathbf{w} with α fixed. For the (a) step, I have a closed-form solution:

$$\alpha = \left(\sum_m K_m + \lambda Id \right)^{-1} \mathbf{y} \quad (4.13)$$

For (b), I do a reduced-gradient descent. The functional cost $\mathcal{L}(f_1, \dots, f_M, \mathbf{w})$ decreases at each iteration and is lowerly bounded by 0, thus it converges to a local minimum.

SUPANOVA In this article [127], sparsity of \mathbf{w} is not assured by an ℓ_1 -constraint, but by an ℓ_1 -regularization term. S.Gunn et al. minimize the following functional cost:

$$\mathcal{L}(\alpha, \mathbf{w}) = \sum_i \left(y_i - \sum_j \alpha_j \left(\sum_m w_m k_m(\mathbf{x}_j, \mathbf{x}_i) \right) \right)^2 \quad (4.14)$$

$$+ \lambda_1 \alpha^T \left(\sum_m w_m K_m \right) \alpha + \lambda_2 \|\mathbf{w}\|_1 \quad (4.15)$$

$$\text{s.t.} \quad w_m \geq 0 \quad \forall m \quad (4.16)$$

They kept the positivity constraint on \mathbf{w} , so $\sum_m w_m k_m(\cdot, \cdot)$ is still a semi-definite positive function. Functional cost (4.14) is a quadratic problem in α and in \mathbf{w} . They alternatively solve it for α with \mathbf{w} fixed and for \mathbf{w} with α fixed. For optimal α , they have closed-form formula:

$$\alpha = \left(\sum_m w_m K_m + \lambda_1 Id \right)^{-1} \mathbf{y} \quad (4.17)$$

For \mathbf{w} , they use an optimization algorithm for quadratic functions under constraints [129].

S.Gunn and J.S. Kandola [127] proves convergence to a global minimum by this argument : “In the quadratic case the second order partial derivatives with respect to α and \mathbf{w} are always positive ensuring that every slice is convex. This fact combined with the knowledge that the solution is finite in α and \mathbf{w} should ensure convergence to the global minimum”.

4.2.2 Filtering methods

Many filtering methods are based on correlation and mutual information, see [130, 131] for comparative study of filtering methods. I will not experiment with correlation or mutual information-based filters, as they have been used and improved a lot for GRN inference, see section 1.4.1, page 26. In the following, I will test kernel target alignment, a feature selection method which tries to find optimal feature weights for the classification or regression task. This method should take into account multivariate dependencies. I also implement RReliefF, a filter capable to identify nonlinear dependency, and that was not tested for GRN inference, to my knowledge.

Kernel target alignment (KA)

This measure was first published in [132]. A classification or regression problem would be easy to solve if the input variable \mathbf{x}_i were only the label to predict $y_i = \mathbf{x}_i$ for all i from 1 to N . In this case, the perfect predictor would simply be the identity $f^*(\mathbf{x}_i) = f^*(y_i) = y_i$. Using a linear kernel, one would have the perfect Gram-matrix $K_{linear}^* = \mathbf{y}\mathbf{y}^T$ whose entries are $(K_{linear}^*)_{ij} = \langle y_i, y_j \rangle$. Cristianini et al. look for the best feature weighting to make the Gram-matrix $K_{\mathbf{w}}$ close to the perfect Gram-matrix. With feature weights \mathbf{w} , the Gram-matrix $K_{\mathbf{w}}$ take the values $(K_{\mathbf{w}})_{ij} = k(\mathbf{w} \circ \mathbf{x}_i, \mathbf{w} \circ \mathbf{x}_j)$, where \circ denotes the Hadamard product, $\mathbf{w} \circ \mathbf{x}_i = (w^1 x_i^1, \dots, w^p x_i^p)$. Noting $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ the Euclidian scalar product between two matrices of size $N \times N$:

$$\langle A, B \rangle_{\mathcal{F}} = \sum_{i,j=1}^N A_{ij} B_{ij} \quad , \quad (4.18)$$

Cristianini defines the alignment between two matrices, K and K^* , as:

$$\mathcal{A}(K, K_{linear}^*) = \frac{\langle K, K_{linear}^* \rangle_{\mathcal{F}}}{\|K\|_{\mathcal{F}} \|K_{linear}^*\|_{\mathcal{F}}}, \quad (4.19)$$

which can be interpreted as the cosine between the two matrices. If the employed Gram-matrix is aligned with the perfect Gram-matrix, the prediction function should be easy to find. For the classification case, Cristianini, in [132], shows that the generalization error is bounded by $1 - \mathcal{A}(K, K_{linear}^*)$ and that it has the concentration property¹.

Kernel alignment has been used for regression, in [133], with centered output $y'_i = y_i - \bar{y}$, with \bar{y} the mean of \mathbf{y} . In the regression case, kernel alignment still has the concentration property.

In my experiments, I shall use a weighted Gaussian kernel:

$$K_{\mathbf{w}}(\mathbf{x}, \mathbf{z}) = \exp\left(-\sum_{i=1}^p w_i (x_i - z_i)^2\right) \quad (4.20)$$

and optimize \mathbf{w} in order to align $K_{\mathbf{w}}$ with K_{linear}^* . I also suggest to align $K_{\mathbf{w}}$ with K_{Gauss}^* , the Gaussian kernel that would be obtained with perfect input:

$$(K_{Gauss}^*)_{ij} = \exp\left(-\frac{(y_i - y_j)^2}{2}\right) \quad (4.21)$$

I follow the optimization procedure in [134], called *Scaled Alignment method*. Starting from $\mathbf{w}_0 = [1, \dots, 1]^T$, I perform a gradient descent until I have found a local minimum, see Algorithm 1.

RReliefF

This filter is employed to measure the quality of an attribute for the task of prediction. First developed for binary classification [135], "Relief" randomly selects an instance \mathbf{r} of the training set \mathcal{S} . It looks at the nearest instance of the same class (nearest hit, \mathbf{h}) and of the opposite class (nearest miss, \mathbf{m}). It evaluates the distance between two instances \mathbf{a} and \mathbf{b} according to attribute i with the function *dist*:

¹the probability of the empirical estimate deviating from its mean can be bounded as an exponentially decaying function of that deviation

Algorithm 1 Scaled Alignment Selection

Starting values $\mathbf{w}_0 = (1, \dots, 1)^T$, $\mathcal{A}_0 = 0$, $n = 0$
 Choose $K^* = K_{linear/Gaussian}^*$
while $|\mathcal{A}_n - \mathcal{A}_{n-1}| \geq \epsilon$ **do**
 Compute gradient $g_{n+1} = \nabla_{\mathbf{w}_n} \mathcal{A}(K_{\mathbf{w}}, K^*)$
 Obtain descent-step by line search $\eta_{n+1} = \arg \max_{\eta \in [0,1]} \mathcal{A}(K_{\mathbf{w}_n + \eta g_{n+1}}, K^*)$
 $\mathbf{w}_{n+1} = \mathbf{w}_n + \eta_{n+1} g_{n+1}$
 $\mathcal{A}_{n+1} = \mathcal{A}(K_{\mathbf{w}_{n+1}}, K^*)$
 $n \leftarrow n + 1$
end while

$$dist(i, \mathbf{a}, \mathbf{b}) = \begin{cases} 0 & \text{if } a_i = b_i \\ 1 & \text{otherwise} \end{cases} \quad \text{if attribute } i \text{ is discrete,} \quad (4.22)$$

$$dist(i, \mathbf{a}, \mathbf{b}) = \frac{|a_i - b_i|}{\max_{x \in S} x_i - \min_{x \in S} x_i} \quad \text{if attribute } i \text{ is continuous.} \quad (4.23)$$

A good attribute will have a large distance between the selected instance \mathbf{r} and its nearest miss, and a small distance with its nearest hit. To measure the quality of each attribute in the vector \mathbf{w} , Kim et al. initialize \mathbf{w} at 0_p , randomly select n instances in the training set and for each selected instance \mathbf{r} with nearest hit \mathbf{h} and nearest miss \mathbf{m} , increment \mathbf{w} according to:

$$w_i \leftarrow w_i + \frac{1}{n} (dist(i, \mathbf{r}, \mathbf{m}) - dist(i, \mathbf{r}, \mathbf{h})) \quad (4.24)$$

This algorithm is given in Algorithm 2. An improvement of "Relief" by selecting the k nearest hit and miss, called "ReliefF" [136], is more robust to noise and can be applied to multiclass problems. Furthermore, it can be interpreted probabilistically. Noting $y_{\mathbf{r}}$ the label of instance \mathbf{r} , if all attributes are discrete, "ReliefF" is an approximation of the difference between probabilities:

$$w_i = P(dist(i, \mathbf{r}, \mathbf{m}) | y_{\mathbf{r}} \neq y_{\mathbf{m}}) - P(dist(i, \mathbf{r}, \mathbf{h}) | y_{\mathbf{r}} = y_{\mathbf{h}}) \quad (4.25)$$

Noting the probability of label difference $P_{diffY} = P(y_{\mathbf{r}} \neq y_{\mathbf{m}})$, the probability of attribute distance $P_{diffi} = P(dist(i, \mathbf{r}, \mathbf{m}))$ and the probability of label difference knowing attribute distance $P_{diffY|diffi} = P(y_{\mathbf{r}} \neq y_{\mathbf{m}} | dist(i, \mathbf{r}, \mathbf{m}))$, equation (4.25) may be extended following Bayes's rule according to:

$$w_i = \frac{P_{diffY|diff_i} P_{diff_i}}{P_{diffY}} - \frac{(1 - P_{diffY|diff_i}) P_{diff_i}}{1 - P_{diffY}} \quad (4.26)$$

Inspired by equation (4.26), "ReliefF" has been extended to regression problems under the name "RReliefF" [137]. In a regression problem, the notions of nearest hit or miss do not exist. Given a selected instance \mathbf{r}_0 with label y_0 and its k nearest neighbors $\mathbf{r}_1, \dots, \mathbf{r}_k$ with labels y_1, \dots, y_k , (a) P_{diffY} evaluates how much the labels vary (b) P_{diff_i} how much each attribute varies and (c) $P_{diffY|diff_i}$ how much the labels vary with a variation of the attribute. Robnik-Sikonja et al. [137] suggest to weight the influence of the j^{th} nearest neighbor with function I :

$$I(j) = \frac{I_1(j)}{\sum_{\ell=0}^k I_1(\ell)} \quad \text{where } I_1 \text{ is a method to weight the influence,} \quad (4.27)$$

$$I_1(j) = 1 \quad \text{uniform weight,} \quad (4.28)$$

$$I_1(j) = \exp\left(-\left(\frac{j}{\sigma}\right)\right) \quad \text{exponentially decreasing weight,} \quad (4.29)$$

Similarly to "ReliefF", they randomly select n instances of the training set. They set $P_{diffY} = 0$, and, for all i , $P_{diff_i} = 0$ and $P_{diffY|diff_i} = 0$. Then, for each selected instance \mathbf{r}_0 , they identify the k nearest neighbors $\mathbf{r}_1, \dots, \mathbf{r}_k$ and perform the following updates:

$$P_{diffY} \leftarrow P_{diffY} + \sum_{j=1}^k I(j) |y_{\mathbf{r}_0} - y_{\mathbf{r}_j}| \quad (4.30)$$

$$P_{diff_i} \leftarrow P_{diff_i} + \sum_j I(j) \text{dist}(i, \mathbf{r}_0, \mathbf{r}_j) \quad (4.31)$$

$$P_{diffY|diff_i} \leftarrow P_{diffY|diff_i} + \sum_j I(j) |y_{\mathbf{r}_0} - y_{\mathbf{r}_j}| \text{dist}(i, \mathbf{r}_0, \mathbf{r}_j) \quad (4.32)$$

Once these computations have terminated, they evaluate the weights of each attribute according to:

$$w_i = \frac{P_{diffY|diff_i}}{P_{diffY}} - \frac{P_{diff_i} - P_{diffY|diff_i}}{n - P_{diffY}}. \quad (4.33)$$

This algorithm is described in Algorithm 3. For a study of theoretical and empirical performances of "RReliefF", see [138].

Algorithm 2 Relief algorithm

Given learning set \mathcal{S}
 Starting $\mathbf{w} = \mathbf{0}_p$
for $\ell = 1$ to n **do**
 Randomly select instance $\mathbf{r} \in \mathcal{S}$
 Find nearest hit \mathbf{h} and nearest miss \mathbf{m} in \mathcal{S}
 for $i = 1$ to p **do**
 $w_i = w_i + \text{dist}(i, \mathbf{r}, \mathbf{m}) - \text{dist}(i, \mathbf{r}, \mathbf{h})$
 end for
end for
 Output \mathbf{w}

Algorithm 3 RReliefF algorithm

Given learning set \mathcal{S}
 Starting $P_{diffY} = 0, P_{diff_i} = 0, P_{diffY|diff_i} = 0$
for $\ell = 1$ to n **do**
 Randomly select instance $\mathbf{r}_0 \in \mathcal{S}$
 Find k nearest instances $\mathbf{r}_1, \dots, \mathbf{r}_k$ in \mathcal{S}
 $P_{diffY} = P_{diffY} + \sum_{j=1}^k I(j)|y_{\mathbf{r}_0} - y_{\mathbf{r}_j}|$
 for $i = 1$ to p **do**
 $P_{diff_i} = P_{diff_i} + \sum_j I(j)\text{dist}(i, \mathbf{r}_0, \mathbf{r}_j)$
 $P_{diffY|diff_i} = P_{diffY|diff_i} + \sum_j I(j)\text{dist}(i, \mathbf{r}_0, \mathbf{r}_j)|y_{\mathbf{r}_0} - y_{\mathbf{r}_j}|$
 end for
end for
 $w_i = \frac{P_{diffY|diff_i}}{P_{diffY}} - \frac{P_{diff_i} - P_{diffY|diff_i}}{n - P_{diffY}}$
 Output \mathbf{w}

4.2.3 Embedded methods

Recursive Feature Elimination

One feature selection method iteratively eliminates the least informative feature [104]. Starting from F_0 , the set of all features, one creates nested subsets of features $F_0 \supset F_1 \supset \dots \supset F_{\text{final}}$ until F_{final} is an optimal subset of features. To identify the least informative feature, one can learn, for each feature i , a predictor ignoring this feature i , but this has a high computational cost. To avoid running the learning algorithm many times, Le Cun et al. [105] suggest the *Optimal Brain Damage* algorithm (OBD). A prediction function $f_{\hat{\alpha}}$, characterized by parameter $\hat{\alpha}$, is learned by minimizing a cost function $J(\hat{\alpha})$. To learn the importance of each feature i , Le Cun et al. calculate the change in objective function J for a change in parameter $\alpha_i = \hat{\alpha}_i + h_i$:

$$J(\hat{\alpha} + \mathbf{h}) = J(\hat{\alpha}) + \sum_i h_i \frac{\partial J}{\partial \alpha_i} + \sum_i \frac{h_i^2}{2} \frac{\partial^2 J}{\partial \alpha_i^2} + \sum_{j \neq i} \frac{h_i h_j}{2} \frac{\partial^2 J}{\partial \alpha_i \partial \alpha_j} + O(\|\mathbf{h}\|^3), \quad (4.34)$$

They suggest the following simplifications:

- the "extremal" approximation: they assume $\hat{\alpha}$ is an optimum of J , therefore $\frac{\partial J}{\partial \alpha_i} = 0$
- the "diagonal" approximation: they assume that a change in J by deleting several parameters is close to the sum of changes caused by individually deleting parameters. Therefore, they neglect cross-terms $\frac{\partial^2 J}{\partial \alpha_i \partial \alpha_j}$ if $j \neq i$
- The "quadratic" approximation: the functional cost is nearly quadratic, so they neglect the last term of the equation

Equation (4.34) thus reduces to:

$$J(\hat{\alpha} + \mathbf{h}) - J(\hat{\alpha}) \approx \sum_i \frac{h_i^2}{2} \frac{\partial^2 J}{\partial \alpha_i^2}. \quad (4.35)$$

A change $h_i = -\alpha_i$ corresponds to removing feature i . They denote by:

$$D_i J = \frac{\hat{\alpha}_i^2}{2} \frac{\partial^2 J}{\partial \alpha_i^2} \quad (4.36)$$

the importance of feature i .

Guyon et al. [106] apply this methodology to SVM with SVM-RFE, meaning SVM Recursive Feature Elimination. In particular, for linear SVM, they obtain a prediction function $f_\alpha(\mathbf{x}) = \langle \alpha, \mathbf{x} \rangle + b$ and $D_i J = \alpha_i^2$. Starting from the set of all feature F , they train a linear SVM. They eliminate the n_F features i with smallest parameter α_i , n_F being a number of features. They continue (a) learning a linear SVM on the remaining features and (b) removing the least informative features until they have found an optimal subset of features. The corresponding algorithm is described in Algorithm 4. I present later in this section ways to determine the optimal subset of features using several criteria.

For non-linear kernel methods, I. Guyon et al. compute K_F , the Gram-matrix on all features, and they have a prediction function of the form:

$$f_{\alpha, F}(\mathbf{x}) = \sum_{i=1}^n \alpha_i k_F(\mathbf{x}_i, \mathbf{x}) \quad (4.37)$$

with k_F the kernel function calculated with all features of set F . With α fixed, they calculate the cost function J if feature i was removed, with kernel $k_{F \setminus \{i\}}$. If feature i is not important, the functional cost J will have little change. The algorithm is described in Algorithm 5.

In this chapter, I will use RFE with kernel-ridge regression using linear and Gaussian kernel.

Algorithm 4 SVM-RFE for linear SVM

Input: training examples $X_0 = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ with labels $\mathbf{y} = [y_1, \dots, y_n]$

Initialize remaining features $F = [1, \dots, p]$, feature ranked list $R = []$

while $F \neq []$ **do**

 Remaining training features $X = X_0(F, :)$

 Train the classifier $\mathbf{w} = \text{SVM_train}(X, \mathbf{y})$

 Find the least informative feature $f = \arg \min_i w_i^2$

$R = [F(f), R]$

$F = [F(1 : f - 1), F(f + 1 : \text{end})]$

end while

Output: feature ranked list R

Algorithm 5 SVM-RFE for non-linear SVM

Input: training examples $X_0 = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ with labels $\mathbf{y} = [y_1, \dots, y_n]$
Initialize remaining features $F = [1, \dots, p]$, feature ranked list $R = []$
while $F \neq []$ **do**
 Compute the Gram matrix K_F
 Train the classifier $\mathbf{w} = \text{SVM_nonlinear_train}(K_F, \mathbf{y})$
 Find least informative feature $f = \arg \min_i J(K_{F \setminus \{i\}}, \mathbf{w})$
 $R = [F(f), R]$
 $F = [F(1 : f - 1), F(f + 1 : \text{end})]$
end while
Output: feature ranked list R

KerNel Iterative Feature Extraction - KNIFE

This algorithm performs feature selection by weighting each feature in the kernel. One chooses a kernel function k . Two instances \mathbf{x}, \mathbf{z} are compared with the weighted kernel $k_{\mathbf{w}}(\mathbf{x}, \mathbf{z}) = k(\mathbf{w} \circ \mathbf{x}, \mathbf{w} \circ \mathbf{z})$, where $(\mathbf{w} \circ \mathbf{x})_i = w_i x_i$. This feature selection idea was described in [139, 140]. In the KNIFE algorithm [141], the following cost is minimized:

$$\min_{\alpha, \mathbf{w}} \mathcal{L}(\alpha, \mathbf{w}) = \underbrace{L(\mathbf{y}, K_{\mathbf{w}}\alpha)}_{\text{loss function}} + \underbrace{\lambda_1 \alpha^T K_{\mathbf{w}}\alpha}_{\alpha \text{ regularization}} + \underbrace{\lambda_2 \|\mathbf{w}\|_1}_{\mathbf{w} \text{ regularization}} \quad (4.38)$$

$$\text{subject to} \quad 0 \leq w_j \leq 1, \text{ for all } j = 1 \dots p. \quad (4.39)$$

The ℓ_1 -regularization produces sparsity in the feature-weights \mathbf{w} . The functional is minimized by alternatively optimizing α with \mathbf{w} fixed and optimizing \mathbf{w} with α fixed. Optimal α is found via standard SVM algorithms. To optimize \mathbf{w} , Allen et al. [141] consider the linear approximation of the Gram-matrix around current feature-weight estimation $\mathbf{w}^{(t)}$, $\tilde{K}_{ij} = \tilde{k}_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j)$ where $\tilde{k}_{\mathbf{w}}(\mathbf{x}, \mathbf{x}') = k_{\mathbf{w}^{(t)}}(\mathbf{x}, \mathbf{x}') + \langle \nabla_{\mathbf{w}} k_{\mathbf{w}^{(t)}}(\mathbf{x}, \mathbf{x}'), \mathbf{w} - \mathbf{w}^{(t)} \rangle$. They minimize a convex surrogate cost function:

$$\min_{\mathbf{w}} \tilde{\mathcal{L}}(\alpha, \mathbf{w}) = L(\mathbf{y}, \tilde{K}_{\mathbf{w}}, \alpha) + \lambda_1 \alpha^T \tilde{K}_{\mathbf{w}}\alpha + \lambda_2 \|\mathbf{w}\|_1 \quad (4.40)$$

$$\text{s.t.} \quad 0 \leq w_j \leq 1, \text{ for all } j = 1 \dots p \quad (4.41)$$

Allen proves that this algorithm converges to a local minimum for several loss functions: the hinge loss (support vector machine), squared error loss (kernel-

ridge regression) and binomial deviance loss (logistic regression). I shall use this algorithm with the squared error loss and the Gaussian kernel. Note that, since the algorithm scales the inputs with \mathbf{w} , the kernel bandwidth σ is no longer an hyper-parameter. There remains only two hyper-parameters, the regularization parameters λ_1 and λ_2 .

Algorithm 6 KNIFE algorithm

Input: training examples $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ with labels $\mathbf{y} = [y_1, \dots, y_n]$
 Randomly initialize weights $0 \leq \mathbf{w}^{(0)} < 1$, set $t = 0$
while $\|\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}\|_1 \geq \epsilon$ **do**
 Compute, with a SVM algorithm, $\alpha^{(t)} = \arg \min_{\alpha} L(\mathbf{y}, K_{\mathbf{w}^{(t)}}) + \lambda_1 \alpha^T K_{\mathbf{w}^{(t)}} \alpha$
 $t = t + 1$
 Compute, by gradient descent, $\mathbf{w}^{(t)} = \arg \min_{\mathbf{w}} \tilde{\mathcal{L}}(\alpha, \mathbf{w})$
 under constraints $0 \leq \mathbf{w} \leq 1$
end while
 Output: weight of all features \mathbf{w}

4.3 Hyper-parameter selection method

The described feature selection methods often require some hyper-parameters, be it the Gaussian kernel bandwidth σ or a regularization parameter $\lambda_1, \lambda_2, \epsilon$. I present several methods to determine hyper-parameter values such that the trained prediction function would generalize well. Then, I will not verify that the prediction function generalize well, I check if the feature selection method selected the genes that regulate a target gene.

Given a feature selection method with c hyper-parameters, I define a finite set of values Λ_i that hyper-parameter i can take, noting $\mu \in (\Lambda_1 \times \dots \times \Lambda_c)$ a value of hyper-parameters for the feature selection method.

4.3.1 Cross-Validation (CV)

Given a value μ for the hyper-parameters, the goal is not to measure the quality of the prediction function on the training set but on new unlabeled data, usually by means of the generalization error. Cross-validation is one of the most popular methods to evaluate the generalization error [142]. To estimate the generalization error, I perform k -fold cross-validation. The training data \mathcal{S} is partitionned in

k equal size sets, the subsamples $\mathcal{E}_1, \dots, \mathcal{E}_k$. One subsample \mathcal{E}_i is kept as the validation data to test. Denoting \hat{f}_{-i} to be the prediction function trained on the $k - 1$ remaining subsamples, the mean square error (MSE) is evaluated according to:

$$MSE = \frac{1}{|\mathcal{E}_i|} \sum_{(\mathbf{x}, y) \in \mathcal{E}_i} (y - \hat{f}_{-i})^2 . \quad (4.42)$$

For each choice μ of hyperparameters, the MSE is obtained on each subsample, using a function trained with hyper-parameter μ on the other subsamples. The value of μ for which the MSE is the lowest is selected for the hyper-parameter.

4.3.2 Stability

Breiman [143] points out that, since cross-validation selects a hyper-parameter value on the training data, the MSE may be greatly underestimated by this procedure when the dataset is not large enough. He suggests aiming for a stable model if we have few data - which is typically the case in gene regulation network inference. If a small variation in the training data leads to significant changes in the prediction function, the model is unstable, hence it is not reliable.

To evaluate the stability of a method with hyperparameter value μ , I follow the procedure in [144]. I create $n_s = 50$ subsamples of the training data $\mathcal{S}: \mathcal{E}_1, \dots, \mathcal{E}_{n_s}$. Each subsample uses a fraction $f = 80\%$ of the training data. Denoting \mathbf{w}_i be the feature weights of the model trained on subsample \mathcal{E}_i . The stability of an algorithm with hyper-parameter μ is evaluated at the mean alignment of feature weights, whose definition is in equation (4.19):

$$stab = \frac{2}{n_s(n_s - 1)} \sum_{i=1}^{n_s} \sum_{j=i+1}^{n_s} \mathcal{A}(\mathbf{w}_i, \mathbf{w}_j) \quad (4.43)$$

I select the value of the hyperparameter μ for which the feature selection method has the highest stability.

4.3.3 Block-Stability

Politis et al. [145, 146] observe that this stability evaluation treats data from training set as if they were independent. When the data is a time-series, the data

samples are not independent. To get around this limitation, they introduced a block-stability procedure, that takes into account the dependence of data. After having selected a block-size b and randomly chosen an integer $r \in [1, n]$, I build subsample $\mathcal{E}_i = (\mathbf{x}_r, \mathbf{x}_{r+1}, \dots, \mathbf{x}_{(r+b) \bmod n})$ of the training set $\mathcal{S} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$. Given subsamples $\mathcal{E}_1, \dots, \mathcal{E}_{n_s}$ built with the block-stability procedure, stability is then calculated as in the previous method.

4.3.4 Bayesian and Akaike Information Criterion

The fewer parameters a model has, the less likely it is to overfit the data. From this observation, Schwarz [147] developed a criterion for selecting hyperparameters, consisting of the likelihood function and a penalty term for the number of parameters in the model. Assuming that the model errors are independent and distributed according to a centered normal distribution, and denoting:

$$\hat{\sigma}_E^2 = \frac{1}{n-1} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \quad (4.44)$$

the empirical variance of the model errors, he measures the likelihood of observing y_1, \dots, y_N given the model $f(\mathbf{x}_1, \dots, \mathbf{x}_N)$ by:

$$p(\mathbf{y}|f) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi\hat{\sigma}_E^2}} \exp\left(-\frac{(y_i - f(\mathbf{x}_i))^2}{2\hat{\sigma}_E^2}\right) \quad (4.45)$$

He looks for a model that maximizes the probability of observations \mathbf{y} under model f while minimizing the number of free parameters. In my case, the number of free parameters is the number of features used in the model. Denoting k the number of used features, *i.e.* features for which $w_i > 0$, I aim to minimize the Bayesian Information Criterion:

$$BIC = -2 \log(p(\mathbf{y}|f)) + k \log(N) \quad (4.46)$$

$$= N \log(\hat{\sigma}_E^2) + k \log(N) + \text{Constant} \quad (4.47)$$

The Akaike Information Criterion [148] relies on the same idea of maximizing the log likelihood of observed data while minimizing the number of free parameters, and is given by this formula:

$$AIC = 2k - 2\log(p(\mathbf{y}|f)) \quad (4.48)$$

$$= 2k + 2N\log(\hat{\sigma}_E) + Constant \quad (4.49)$$

4.4 Experiments

The experiments are carried out using the DREAM3 Challenge 4 data, described in Chapter 3. Performance will be evaluated for AUROC and AUPR, defined in Chapter 3.

4.4.1 Hyperparameter selection

The methods have been tested with hyper-parameters in the following sets:

- For SimpleMKL, regularization tradeoff $C \in \{0.01, 0.1, 1, 10, 100\}$, ϵ -insensitive loss $\epsilon \in \{0.01, 0.05, 0.15\}$ and kernel bandwidth $\sigma \in \{0.5, 1, 2\}$.
- For $\ell_2 - MKL$, regularization tradeoff $\lambda \in \{0.01, 0.1, 1, 10, 100\}$ and kernel bandwidth $\sigma \in \{0.5, 1, 2\}$.
- For SUPANOVA, α regularization tradeoff $\lambda_1 \in \{0.01, 0.1, 1, 10, 100\}$, \mathbf{w} regularization tradeoff $\lambda_2 \in \{0.01, 0.1, 1, 10, 100\}$ and kernel bandwidth $\sigma \in \{0.5, 1, 2\}$.
- For KNIFE, α regularization tradeoff $\lambda_1 \in \{0.01, 0.1, 1, 10, 100\}$ and \mathbf{w} regularization tradeoff $\lambda_2 \in \{0.01, 0.1, 1, 10, 100\}$.
- For kernel target alignment and RReliefF, there is no hyper-parameter to set.
- For RFE, regularization tradeoff $\lambda \in \{0.01, 0.1, 1, 10, 100\}$ and the number of used features to model target gene $k \in [1, \dots, p]$. For the number of used features, stability is irrelevant, it will always use all the features. Thus the selection of hyper-parameters must be performed through cross-validation, BIC or AIC for RFE methods.

The performance of a feature selection method with a hyper-parameter selection criterion greatly varies with the network studied, see Figure 4.1. For each feature selection method and for each size 10 network, I compute the correlation

Section 4.4: Experiments

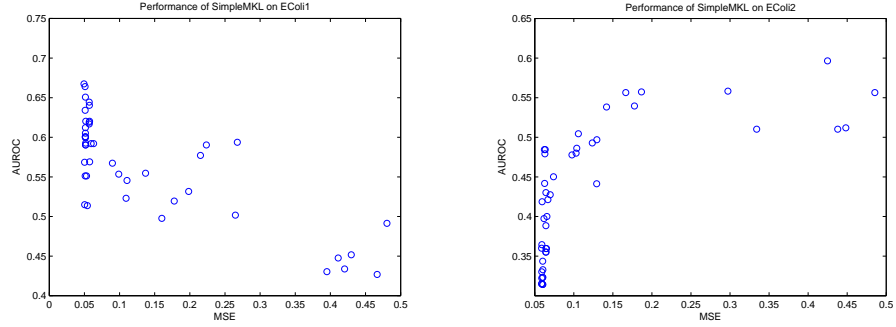


Figure 4.1: For SimpleMKL, values of the points (MSE evaluated by cross-validation, AUROC) for the different values of the hyper-parameter μ in the pre-defined set. The AUROC appears to depend more on the dataset than the hyper-parameters. Second, the shape of the distribution of the points varies greatly from one network to another. Cross-validation does not seem to be a good hyper-parameter selection method when SimpleMKL is used for the task of network inference.

between the performance of the feature selection method and the value of the hyper-parameter criterion. The average of this correlation, across all networks, is shown in Tables 4.1 and 4.2. MSE , BIC and AIC should be negatively correlated with $AUROC$ or $AUPR$; stability and block-stability should be positively correlated.

First, we note that too few examples are available to be able to conclude that any hyper-parameter selection criterion is good. We can conclude that most criteria give poor results, no better than what would be obtained with an educated guess. Second, the correlation is highly unstable. For $\ell_2 - MKL$ and for AUROC, one would rather use stability than cross-validation: stability has lower correlation (0.44 against -0.54)², but the standard deviation of this correlation is much lower (0.16 against 0.34). Third, for RFE, the BIC criterion seems more relevant than other methods, even though the results have a high variance.

²Note that stability should be positively correlated to performance, and MSE negatively correlated. So a correlation of -0.54 for MSE is a better result than a correlation of 0.44 for stability.

| FS Method | CV MSE | Stability | Block Stability | BIC | AIC |
|----------------|-------------------|------------|-----------------|-------------------|------------|
| SimpleMKL | 0.1±0.48 | -0.35±0.15 | -0.49±0.08 | -0.06±0.14 | -0.02±0.14 |
| $\ell_2 - MKL$ | -0.54±0.34 | 0.44±0.16 | 0.43±0.1 | 0.47±0.36 | 0.45±0.37 |
| SUPANOVA | -0.23±0 | -0.06±0.1 | -0.02±0.07 | 0.22±0 | -0.22±0 |
| KNIFE | 0.24±0.18 | -0.04±0.5 | -0.21±0.5 | -0.01±0.42 | -0.01±0.42 |
| RFE linear | 0.14±0.93 | - | - | -0.4±0.35 | -0.09±0.81 |
| RFE Gaus | 0.01±0.7 | - | - | -0.27±0.51 | 0.25±0.53 |

Table 4.1: Correlation between various hyper-parameter selection criteria and the AUROC for six feature selection methods for network inference. The values displayed are the mean \pm standard deviation over size 10 networks. The best hyper-parameter selection criterion for a feature selection method is shown in bold. For SimpleMKL and KNIFE, the criterion is either wrongly correlated with the feature selection’s performance or the correlation is close to 0. We conclude that none of the considered hyper-parameter selection criteria works well in this context.

| FS Method | CV MSE | Stability | Block Stability | BIC | AIC |
|----------------|------------|------------|-----------------|-------------------|------------|
| SimpleMKL | 0±0.36 | -0.3±0.22 | -0.45±0.16 | -0.05±0.18 | -0.08±0.26 |
| $\ell_2 - MKL$ | -0.03±0.41 | -0.06±0.32 | -0.03±0.27 | -0.06±0.43 | -0.07±0.43 |
| SUPANOVA | 0±0.04 | 0.05±0.17 | 0.06±0.12 | -0.02±0.05 | -0.01±0.03 |
| KNIFE | 0.19±0.21 | -0.09±0.62 | -0.25±0.58 | -0.08±0.4 | -0.08±0.4 |
| RFE linear | 0.27±0.97 | - | - | -0.48±0.68 | 0.05±0.96 |
| RFE Gaus | -0.19±0.63 | - | - | -0.28±0.55 | 0.19±0.57 |

Table 4.2: Correlation between various hyper-parameter selection criteria and the AUPR for six feature selection methods. For every method but RFE, the criterion is either wrongly correlated with the feature selection’s performance or the correlation is close to 0. For RFE, the BIC criterion seems the most relevant, but is also highly unstable.

| AUROC | EColi1 | EColi2 | Yeast1 | Yeast2 | Yeast3 | Avg time (sec) |
|----------------|-------------|-------------|-------------|-------------|-------------|----------------|
| SimpleMKL | 0.66 | 0.36 | 0.56 | 0.54 | 0.51 | 2726.17 |
| $\ell_2 - MKL$ | 0.65 | 0.41 | 0.49 | 0.57 | 0.63 | 200.78 |
| SUPANOVA | 0.5 | 0.5 | 0.5 | 0.5 | 0.65 | 104.83 |
| KNIFE | 0.61 | 0.57 | 0.65 | 0.45 | 0.33 | 3842.98 |
| KA Gaus | 0.58 | 0.54 | 0.43 | 0.62 | 0.53 | 0.45 |
| KA Linear | 0.36 | 0.44 | 0.47 | 0.48 | 0.34 | 0.54 |
| RReliefF | 0.5 | 0.53 | 0.54 | 0.49 | 0.54 | 0.86 |
| RFE Linear | 0.36 | 0.58 | 0.48 | 0.61 | 0.57 | 0.39 |
| RFE Gaus | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 9.02 |
| Team 236 | 0.62 | 0.65 | 0.65 | 0.44 | 0.49 | - |
| Team 190 | 0.57 | 0.52 | 0.63 | 0.58 | 0.60 | - |

Table 4.3: Results in AUROC of nine eleven feature selection methods and two DREAM competing methods on Size 10 networks. The best performing method on a given network is highlighted in bold. The last column gives the average computational time on one network for the whole testing process (*i.e.*, computing each hyper-parameter selection criterion and calculating model for each selected hyper-parameter)

4.4.2 Performance on size 10 networks

The performance is assessed by AUROC, AUPR and the computational time. For SimpleMKL, SUPANOVA, KNIFE, hyper-parameters have been selected with cross-validation, as no method stands out. Stability has been used for $\ell_2 - MKL$; BIC for RFE.

For AUROC, results are shown in Table 4.3. No method is competitive on all networks. Many methods underperform random guessing (AUROC= 0.50) on some dataset. For AUPR, results are shown in Figure 4.2. The SUPANOVA and RFE Gaussian methods stand out. We also see that these methods are competitive on these few experiments.

Computational time is given in Table 4.3. Given a network, the time given is the sum of the time to calculate every hyper-parameter selection criterion, and compute the prediction function f for the selected hyper-parameters. Filter methods (KA, RReliefF) are considerably faster, as expected, but the RFE methods are competitive. $\ell_2 - MKL$ and SUPANOVA are executed rather quickly, and can be applied to size 50 networks. The computational time of SimpleMKL and KNIFE is too large to be able to apply them to size 50 networks in this study.

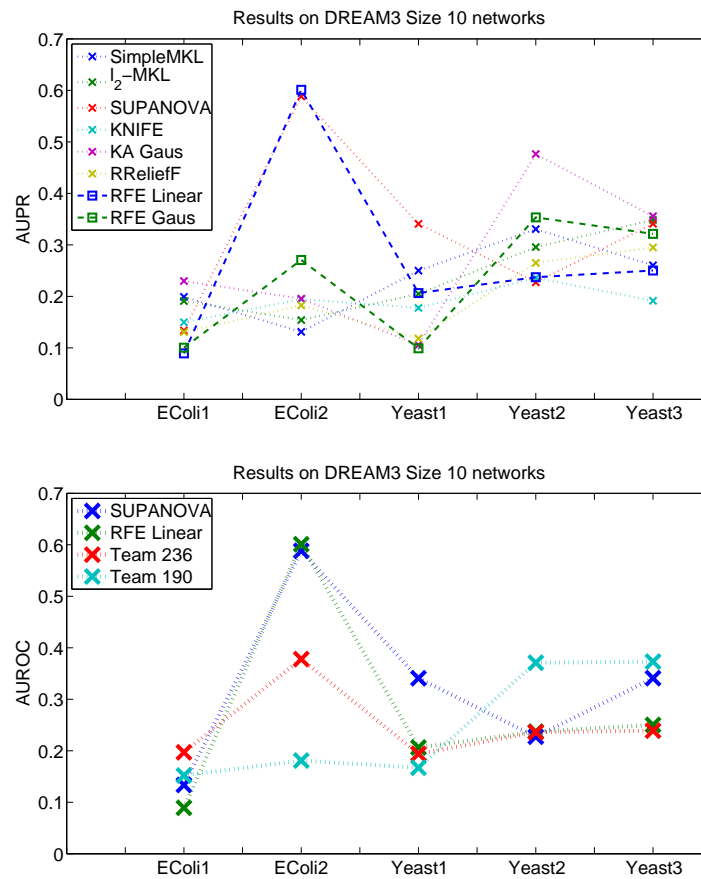


Figure 4.2: Top: Comparison of the AUPR of described methods on the five size 10 networks. Bottom: Comparison of the AUPR of RFE Gaussian and SUPANOVA to best competing teams in DREAM3 Challenge

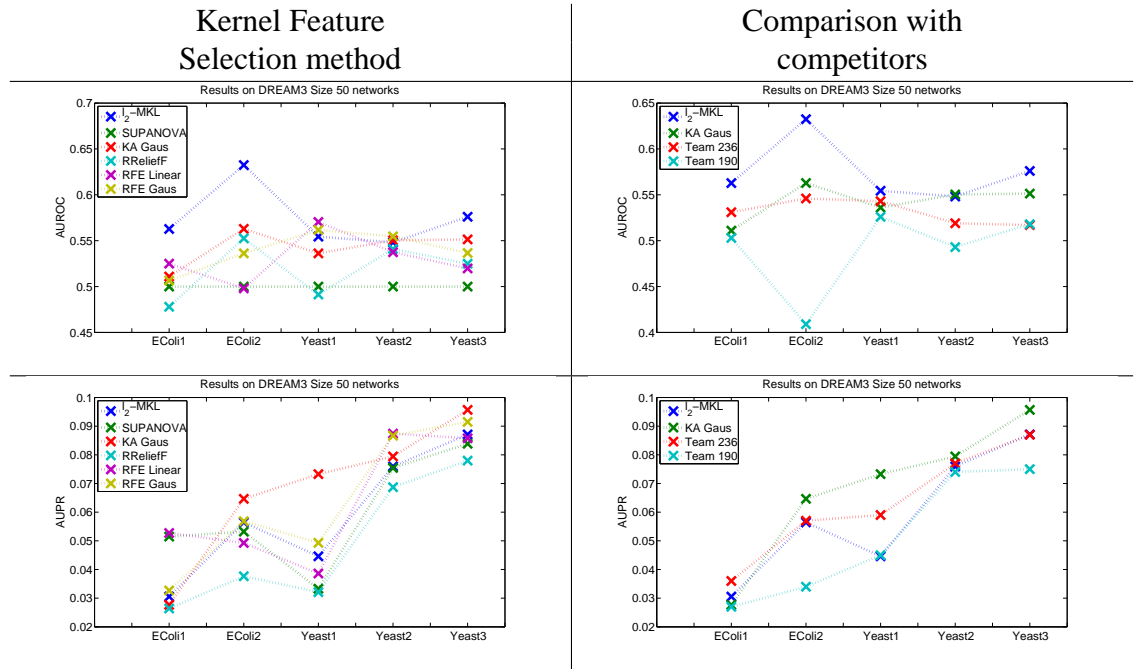


Figure 4.3: Comparison of kernel feature selection methods on DREAM 3 Size 50 Networks

4.4.3 Performance on size 50 networks

AUROC and AUPR on all networks are displayed in Figure 4.3. Also shown in this figure are the two best kernel feature selection methods against the two best competitors of the DREAM3 challenge. We see that $\ell_2 - MKL$ is the best performer in AUROC on all five 50-gene networks. In AUPR, the Gaussian kernel alignment prevails. Performance is still poor, and computation time will be a burden for $\ell_2 - MKL$ on size 100 networks. As we shall see in Chapter 5, ensemble methods will enable the use of $\ell_2 - MKL$ for networks of size 100 or more.

4.5 Conclusion

Gene regulatory network inference from gene expression data is an active research field, that will increasingly benefit from the abundance of data in the coming years. Many approaches to GRNI have been explored, but kernel feature selection methods has so far been neglected. In this work, I tested nine

kernel feature selection methods, and five hyper-parameter selection methods on realistic datasets. The results show the limitations of these methods. No hyper-parameter selection method seems efficient, except BIC for RFE methods, which would need further testing to be validated. Some methods are too time-consuming to be directly applicable to real-size networks. This comparative study also gave some positive indications for constructing methods that would give better results. $\ell_2 - MKL$ and Gaussian kernel alignment appeared to extract more information for inference of GRN. This motivated the enhancement of the $\ell_2 - MKL$ method, in particular its enhancement by ensemble methods. Learning the network on many subsamples of data usually yields better result than learning one network on all data [41]. In the following chapter, I will show that an enhanced $\ell_2 - MKL$ method yields state-of-the-art results on real networks.

Chapter 5

LocKNI: Local Kernel for Network Inference

5.1 Introduction

Local kernels were motivated by several reasons. First, kernel methods deal with several difficulties of gene modeling, since they are robust-to-noise, they provide a nonlinear model, and the versatility of the loss function allows incorporation of prior knowledge. We saw, in the previous chapter, that $\ell_2 - MKL$ using *local kernels* provided some of the most interesting results in GRN inference among kernel feature weighting or feature selection methods, therefore, this method will be implemented with the ensemble method described in Chapter 3. Even on small scale networks, ensemble methods will increase the performance of the GRNI method. I shall call LocKNI the alliance of $\ell_2 - MKL$, *local kernels* and the double randomization scheme.

Original contributions can be made when using *local kernels* in the regular $\ell_2 - MKL$ framework. A simple modification of the loss function introduces prior knowledge in the regularization framework, thus improving LocKNI's results. Besides, since they rely on a few features, *local kernel*-based models have sparse gradients, thus have more interpretable partial derivatives. The drawback of these local models is that they are no longer universal, hence they may be unable to approximate well the partial derivatives of a perfect model. In this chapter, I first describe $\ell_2 - MKL$ and its modification to incorporate prior knowledge. Second, as in Chapter 3, I use randomized $\ell_2 - MKL$ models with *local kernel* in an ensemble, exploiting a double scheme of randomization both on variables

and on data. In the fourth section, I explain the chosen hyperparameter selection method. In the fifth section, I compare, on real and real-sized networks, LocKNI without prior knowledge with state-of-the-art methods. I also evaluate the partial derivatives of LocKNI as a GRNI method. There will also be a discussion of LocKNI's strength and weaknesses: how does it behave compared to other GRNI methods ? What types of error does LocKNI make ? In the fifth section, I propose two reasonable priors that may be incorporated in LocKNI. I will test, on simulated data, the improvement in LocKNI's performance with incorporation of prior knowledge.

5.2 $\ell_2 - MKL$ and Prior Knowledge Incorporation

5.2.1 $\ell_2 - MKL$

We saw, in the previous chapters, that two types of training data $\mathcal{S} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$ may be available, to learn either a static or a dynamic model:

$$x^i = f_{stat}^i(\mathbf{x}^{-i}) + \epsilon^i \quad \text{static model,} \quad (5.1)$$

$$x^i(t + \tau) = f_{dyn,\tau}^i(\mathbf{x}^{-i}(t)) + \epsilon^i(t) \quad \text{dynamic model.} \quad (5.2)$$

These models can be estimated with supervised learning tools on a training set. With the general notation $\mathcal{S} = ((\mathbf{z}_1, y_1), \dots, (\mathbf{z}_N, y_N))$, \mathbf{z} being input variables, y being an output variable to predict, a function $y = f(\mathbf{z}) + \epsilon$ is learned on the training set. This notation will be used to describe the learning algorithm.

In the previous chapter, a multiple kernel approach was presented, using *local kernels*, i.e. kernels that use only one component of an input vector, $k_m(\mathbf{z}_1, \mathbf{z}_2) = k(z_1^m, z_2^m)$. We saw how a linear combination of these kernels, $k^* = \sum_m w_m k_m$, could be learned, by optimizing the weight \mathbf{w} . The function f and weights \mathbf{w} are found by minimizing the following functional cost:

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{i=1}^N (y_i - \sum_{m=1}^M f_m(\mathbf{z}_i))^2 + \lambda \sum_{m=1}^M \frac{1}{w_m} \|f_m\|_{\mathcal{H}_m}^2 \quad (5.3)$$

$$s.t. \quad (C) : \sum_{m=1}^M w_m = 1, \quad w_m \geq 0, \quad \forall m \in \{1, \dots, M\} \quad (5.4)$$

$$f_m(\mathbf{z}) = \sum_i \alpha_i w_m k_m(\mathbf{z}_i, \mathbf{z}) \quad (5.5)$$

This function can be optimized by alternately optimizing $\mathcal{L}(\mathbf{w}, \alpha_{fixed})$ over \mathbf{w} when α is fixed and optimizing $\mathcal{L}(\mathbf{w}_{fixed}, \alpha)$ over α when \mathbf{w} is fixed. With \mathbf{w} fixed, the minimum is obtained in closed-form as:

$$\hat{\alpha} = (K_* + \lambda I_N)^{-1} \mathbf{y} \quad (5.6)$$

$$\text{with } K_* = \sum_m w_m K_m \quad (5.7)$$

and I_N is the $N \times N$ identity matrix. With α fixed, \mathbf{w} is optimized through reduced gradient descent [128].

5.2.2 Prior knowledge incorporation

As kernel-based models result from the minimization of a unique global loss function, they benefit from the possibility of taking prior knowledge into account in the definition of the loss function. In order to incorporate an assumption of the existence of a regulator, I propose to relax the sparsity constraint imposed on the weight w_m by dividing λ by a coefficient λ_m . If prior knowledge hints that gene m should be a regulator gene i , λ_m would be set larger than 1; if prior suggests that m should not be a regulator, $\lambda_m < 1$:

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{\ell=1}^N (y_\ell - \sum_{m=1}^M f_m(\mathbf{z}_\ell))^2 + \sum_{m=1}^M \frac{\lambda}{\lambda_m w_m} \|f_m\|_{\mathcal{H}_m}^2 \quad (5.8)$$

$$\text{s.t. } \sum_{m=1}^M w_m = 1, w_m \geq 0, \forall m \in \{1, \dots, M\} \quad (5.9)$$

$$f_m(\mathbf{z}) = \sum_i \alpha_i w_m k_m(\mathbf{z}_i, \mathbf{z}) \quad (5.10)$$

As in Section 5.2.1, this function can be optimized by alternately optimizing $\mathcal{L}(\mathbf{w}, \alpha_{fixed})$ over \mathbf{w} with α fixed and optimizing $\mathcal{L}(\mathbf{w}_{fixed}, \alpha)$ over α with \mathbf{w} fixed. With \mathbf{w} fixed, the minimum can be expressed in closed-form as:

$$\hat{\alpha} = (K_*^2 + \lambda K_*^{prior})^{-1} (K_* \mathbf{y}) \quad (5.11)$$

$$\text{with } K_* = \sum_m w_m K_m \quad (5.12)$$

$$\text{and } \tilde{K}_*^{prior} = \sum_m \frac{w_m}{\lambda_m} K_m. \quad (5.13)$$

With α fixed, \mathbf{w} is optimized through reduced gradient descent.

5.3 Ensemble method

We saw, in Chapter 3, an ensemble method to learn the GRN from several heterogeneous datasets. From available datasets $(E_0, E_1, \dots, E_{N_e})$, I build B subsamples \mathcal{E}_ℓ , $\ell = 1 \dots B$, with *subbagging* and subsampling of variables, in a way similar to *extreme-randomization* [101]. Subsamples are built according to the following procedure:

1. Randomly choose from which dataset E_u to extract data. The probability $p(E_u)$ of choosing dataset E_u is proportional to its size, $p(E_u) = |E_u| / (\sum_{\ell=1}^{N_e} |E_\ell|)$. Let E_u be the selected dataset.
2. Each data vector \mathbf{x}_i of E_u has probability $p_{data,u}$ to be in subsample \mathcal{E}_ℓ . To obtain subsamples of similar sizes, $p_{data,u}$ is taken inversely proportional to E_u 's size. Assuming, without loss of generality, that E_0 is the largest dataset, I have fixed $p_{data,u} = p_{data} \frac{E_0}{E_u}$, with p_{data} a fixed hyper-parameter.
3. Randomly select n_{var} variables to be the potential regulators of the system. The number n_{var} is a hyper-parameter to fix. The set of selected variables is called \mathcal{G}_ℓ .

To learn the importance of genes in \mathcal{G}_ℓ for all the other genes i on subsample \mathcal{E}_ℓ , the mean vote for a regulation of gene i by gene m is computed as the mean weight w_m^i of feature m when predicting gene i with $\ell_2 - MKL$ and *local kernels*. Mean votes are stored in matrix B :

$$B_{im} = \frac{1}{n_{im}} \left(\sum_{\ell, m \in \mathcal{G}_\ell, i \notin \mathcal{G}_{\ell l}} w_m^i \right), \quad (5.14)$$

with n_{im} the number of subsamples where gene m was a potential regulator and gene i was not. The estimated adjacency matrix can then be obtained by thresholding matrix B :

$$\hat{A}_{im} = H(B_{im} - \theta) \quad (5.15)$$

where H is the Heaviside step function.

5.4 Experimental results

5.4.1 Hyperparameter choice

The chosen kernel function is the Gaussian kernel

$$k(z, z') = \exp\left(-\frac{(z - z')^2}{2\sigma^2}\right), \quad (5.16)$$

which has the advantage of being widely used, with good performances in practice and universal consistency. Four hyperparameters remain to be set:

- I select the kernel bandwidth $\sigma = 1$, as I have normalized the empirical variance of the data to 1, and the Gram matrix achieves maximum entropy when σ matches the empirical standard deviation.
- Experiments have shown that the subsampling parameters, n_{var} and p_{data} , have little influence on the capacity of LocKNI to recognize edges. Since LocKNI is faster with smaller subsamples, I set $n_{var} = 5$ and $p_{data} = 0.20$.
- For the tradeoff λ between regularization and data fitting, I also use a heuristic. With a sample \mathcal{D} of size $N_{\mathcal{D}}$, it can be shown that kernel-ridge regression is consistent if $\lambda = 1/\sqrt{N_{\mathcal{D}}}$ [96].

We can consider that convergence has been reached with $L = \frac{100 \times p}{n_{var}}$ subsamples. For each regression, computational complexity is $O(|\mathcal{E}_l|^3)$ for each variable of each subsample. There are L subsamples and p variables, computational complexity is $O(Lp(p_{data}N)^3)$.

5.4.2 Datasets

The evaluation of LocKNI will be two-fold. First, it is evaluated without integration of prior knowledge. LocKNI will be compared to state-of-the-art methods on four real and real-sized networks. The datasets are summarized in Table 5.1. The \widehat{Jac} measure will be used on LocKNI's model as a second approach to GRNI. LocKNI's strength and weaknesses will be studied on simulated data. LocKNI's error will be categorized. Then, LocKNI is evaluated with prior information on a simulated dataset. The purpose of this assessment is to demonstrate the benefit of incorporation of prior knowledge to the reverse-engineering approach. For evaluation against state-of-the-art methods, the first three datasets come from

Section 5.4: Experimental results

| Network | # TF | p | N | N_{dyn} |
|--|------|------|-----|-----------|
| DREAM5 Network 1 (in-silico) | 195 | 1643 | 805 | 463 |
| DREAM5 Network 3 (<i>E.Coli</i>) | 334 | 4511 | 805 | 463 |
| DREAM5 Network 4 (<i>S.cerevisiae</i>) | 333 | 5950 | 536 | 298 |
| <i>E.Coli</i> | 169 | 4297 | 466 | 186 |

Table 5.1: Characteristics of the datasets. # TF is the number of potential regulators, p is the number of target genes, N is the number of data in the training set, N_{dyn} is the number, among these N data, that are part of a time-series.

| networks | DREAM 5 N1 | | DREAM 5 N3 | | DREAM 5 N4 | | <i>EColi</i> | |
|--------------------------------|-------------|-------------|-------------|------------|--------------|-------------|--------------|-------------|
| | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR |
| LocKNI | 78.7 | 25.54 | 65.6 | 7.68 | 51.43 | 1.98 | 65.11 | 8.56 |
| \widehat{Jac} | 71.89 | 8.64 | 63.19 | 3.47 | 52.57 | 2.21 | 63.5 | 5.75 |
| $\widehat{Jac}(\text{LocKNI})$ | 52.22 | 9.33 | 60.97 | 4.16 | 55.99 | 2.32 | 62.77 | 5.70 |
| GENIE3 | 81.5 | 29.1 | 61.7 | 9.3 | 51.8 | 2.1 | 64.04 | 6.04 |
| TIGRESS | 78.2 | 30.1 | 59.5 | 6.9 | 51.7 | 2.0 | 64.54 | 3.98 |
| CLR | 77.3 | 25.5 | 59.0 | 7.5 | 51.6 | 2.1 | 63.26 | 6.59 |

Table 5.2: Performance of GRN inference methods measured by AUROC and AUPR in percentages on DREAM5 Network 1, 3 and 4, and on *EColi*, for which the data is from *M3D* and the gold standard is from RegulonDB. The best results are shown in bold font.

the DREAM5 challenge [41], as described in Chapter 3 and the fourth from the Many Microbe Microarray Database (*M3D*) [149] (*EColi* version 4 build 6). For *EColi* from *M3D*, the gold standard is the currently known network available in RegulonDB v8.1 [150].

For the DREAM5 datasets the network inferred by LocKNI is compared to the best networks inferred in the challenge. For the *EColi* dataset from *M3D*, LocKNI is compared to three of the best methods in DREAM5, namely GENIE3 [78], CLR [27] and TIGRESS [45], with their MATLAB implementation and their default parameters.

Table 5.2 shows the performance of LocKNI compared to those of GENIE3, TIGRESS and CLR, and other methods presented in this thesis— \widehat{Jac} from Chapter 3 and $\widehat{Jac}(\text{LocKNI})$, the GRN inference through the partial derivatives of LocKNI. On network N4 from the DREAM5 challenge all the methods perform equally and just slightly better than a random guess in terms of AUROC. Such a result suggests that this network inference problem is too difficult given the avail-

able data. \widehat{Jac} (LocKNI) has better performance on this dataset, but 55.99 AUROC is close to random guess. Overall, \widehat{Jac} (LocKNI) performs poorly. As LocKNI has a performant feature selection, LocKNI's model should give more weight to the partial derivative of relevant features than a Gaussian kernel-ridge regression. We see that, in practice, this is not the case. It seems that much information about partial derivatives is lost by the additive model.

Results on the other networks are of greater interest. All methods perform significantly better than a random guess (e.g. an AUROC of 50%). On Network 3 and *E.coli*, LocKNI achieves the best performance in terms of AUROC, even outpacing GENIE3 in terms of AUPR for *E.coli*. On network 1, GENIE 3 leads over all other methods for the AUROC, while TIGRESS performs the best according to AUPR. Overall, LocKNI achieves state-of-the-art performance and improves it in two cases. From these comparisons follows a new question: does the prediction of LocKNI differ from the other methods? Can we benefit from that by including LocKNI in a consensus method that will gather predictions of the whole set of four methods?

5.4.3 Network inference by consensus of methods

As in Chapter 3, consensus of GRNI inference methods is used: one consensus (C) consists of all competing methods, a second consensus (C+L) consists of LocKNI and all competing methods, a third one (C+J) consists of \widehat{Jac} and competing methods, and a fourth consensus (C+L+J) consists of all competing methods, LocKNI and \widehat{Jac} together. On the DREAM5 datasets, the competing methods are the 30 best contestants' submissions; on the *E.Coli* dataset, the competing methods are TIGRESS, GENIE3 and CLR. Results are shown in Table 5.3. Interestingly, AUROC is always improved by including LocKNI in the ensemble. However this improvement is small, approximately 1%, except for Network 3 of the DREAM5 challenges. In this case LocKNI outperforms all the other methods and also provides improvements when combined with the consensus models. This improvement is obtained while being one vote out of thirty, which indicates that LocKNI grasps different information than the other methods.

Note that consensus with LocKNI is usually better than consensus with \widehat{Jac} . This is coherent as LocKNI is a better method on these datasets. Nevertheless, on *E.Coli*, the network where LocKNI is the best in AUROC and AUPR, consensus with \widehat{Jac} outperforms the one with LocKNI. This shows that complementary methods are necessary for the consensus approach. Besides, using all methods (C+L+J) shows little improvement over adding LocKNI or \widehat{Jac} to the consensus.

Section 5.4: Experimental results

| networks | DREAM 5 N1 | | DREAM 5 N3 | | DREAM 5 N4 | | <i>EColi</i> | |
|---------------------------|-------------|--------------|--------------|------------|--------------------|--------------------|--------------|-------------|
| Method | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR |
| Consensus | | | | | | | | |
| C | 80.89 | 32.65 | 64.94 | 8.99 | 52.02 | 2.24 | 66.06 | 5.94 |
| C+L | 81.31 | 31.82 | 73.11 | 8.68 | 53.64 | 2.29 | 66.47 | 5.76 |
| C+J | 81.23 | 31.81 | 72.78 | 8.75 | 53.58 | 2.30 | 67.93 | 5.99 |
| C+L+J | 81.30 | 31.83 | 73.03 | 8.72 | 53.44 | 2.25 | 67.94 | 5.96 |
| Best-of-Single algorithms | 81.5 | 31.5 | 65.6 | 9.3 | 55.99 | 2.32 | 65.11 | 8.56 |
| Best algorithm | G3 | TIG | L | G3 | $\widehat{Jac}(L)$ | $\widehat{Jac}(L)$ | L | L |

Table 5.3: AUROC and AUPR in %. Performance of Consensus Method first with competing methods (C)—competing methods are GENIE3, TIGRESS and CLR on *EColi*, all DREAM5 contestants on the DREAM5 dataset—, second, also including LocKNI (C+L), third with competing methods and \widehat{Jac} on Gaussian kernel-ridge regression (C+J), and, finally, consensus with competing methods, LocKNI and \widehat{Jac} (C+L+J). The best results are shown in bold font. In the “best algorithm” row, “G3” stands for GENIE3, “TIG” for TIGRESS, “L” for LocKNI and $\widehat{Jac}(L)$ for the network inferred by the partial derivatives of the LocKNI model.

It seems that, individually, LocKNI and \widehat{Jac} bring complementary results to other methods, but little seems to be learned by using both LocKNI and \widehat{Jac} . The contribution of LocKNI and \widehat{Jac} is similar, probably because kernels would capture information that wasn’t identified before.

Following Marbach et al. [41], I perform PCA on the contestants’ predictions to see which methods were close to each other (see Figure 5.1). LocKNI stands out as yielding different results from other approaches. The improvements in AUROC are obtained nearly without degrading AUPR.

5.4.4 Error analysis on Network N1

As LocKNI differs from the other network inference methods, the different types of errors are studied here, following the categorization introduced in [45]:

- a *reverse edge* means that a regulation from i to j is inferred instead of the regulation $j \rightarrow i$,
- *siblings*: LocKNI infers a link from i to j because those genes are related through a third gene z . Gene i may be a “*grand-father*” of j ($i \rightarrow z \rightarrow j$),

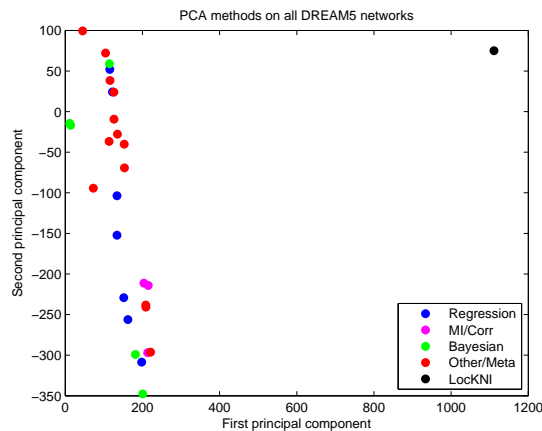


Figure 5.1: PCA on the prediction vector of all methods. For all DREAM5 networks, one method’s prediction is the rank it gives to each interaction, from most likely to least likely.

a “grand-son” ($j \rightarrow z \rightarrow i$), a “brother” ($i \leftarrow z \rightarrow j$) or i and j may be a couple ($i \rightarrow z \leftarrow j$),

- *great-siblings*: genes i and j are connected through two other genes,
- *others*: all links $i \rightarrow j$ which do not belong to any of the previous categories.

LocKNI’s errors were inspected on the simulated dataset (DREAM5 Network 1), where the class of each error can be determined. The ROC curve of all types of targets is shown in Figure 5.2. For a category, such as *reverse edges*, I look at precision and false positive rate of LocKNI’s prediction if I consider that *reverse edges*, without true edges, were the links to find. If LocKNI regularly made one type of mistake, then the area under the ROC curve for this category would be high. Figure 5.2 shows that LocKNI’s errors are rather well balanced. It has a small bias towards selecting *siblings* rather than other edges. LocKNI seems as sensitive to *great-siblings* as it is to *others*. Note that LocKNI appears to identify directionality well, as *reverse edges* are rarely selected.

5.4.5 Incorporation of prior knowledge

As shown in section 5.2.2, in addition to being able to retrieve networks from the experimental data, LocKNI inherits from the regularization framework the ability

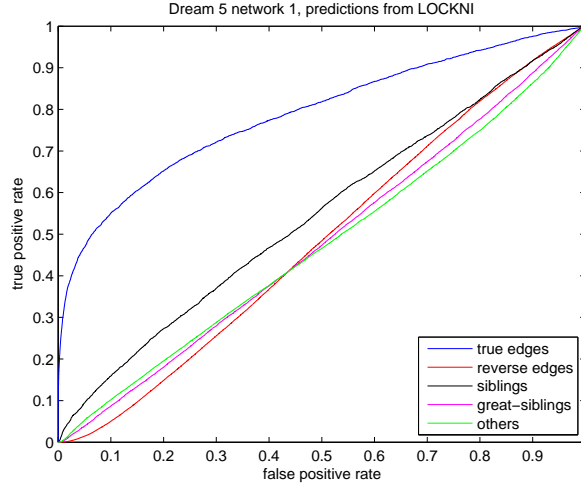


Figure 5.2: Types of links inferred by LocKNI for DREAM5 network 1.

to incorporate prior knowledge. A simple way to include evidence about a regulation $j \rightarrow i$ consists of setting the hyperparameter λ_j in Eq. 5.8 according to the available information.

In knock-out or knock-down experiments information about existence of connections between genes are close to being elucidated. One can infer that gene j regulates gene i if gene i is very perturbed in gene j knock-out experiments. This can be measured by Z-scores [37]. Let μ_{wt}^i be the mean of gene i 's expression level in wild type experiments, σ_{wt}^i be its standard deviation in these experiments and μ_j^i be its mean concentration in gene j 's perturbation experiments. Pinna et al. [37] define the metric:

$$W_{ij} = \frac{\mu_j^i - \mu_{wt}^i}{\sigma_{wt}^i}, \quad (5.17)$$

which has given very good results for network inference on realistically simulated datasets [82]. For prediction of target gene i , I normalize the sparsity constraint by dividing λ by the weight $\lambda_j = |W_{ij}|$ if the experiment with gene j knocked-out is available, $\lambda_j = 1$ otherwise.

I examine the algorithm behavior with prior knowledge on a limited size DREAM3 network of 50 genes, where knock-out's are systematically provided. The data set consists of 21 time point measurements and of steady-state measurements. In the first experiment, I measure average AUROC and AUPR obtained

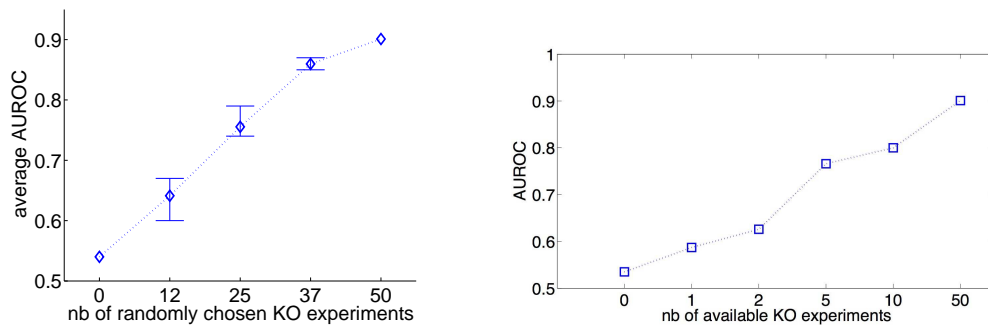


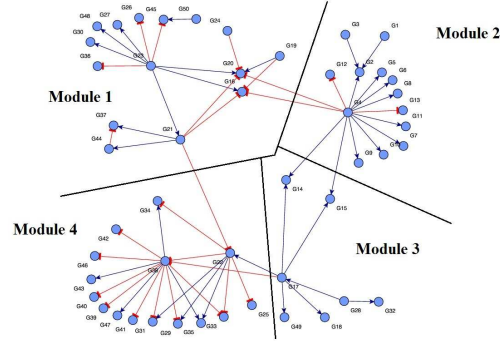
Figure 5.3: (a) Average AUROC versus the number of KO experiments, using *E. coli* 1, in DREAM3 challenge. (b) AUROC versus the number of chosen KO experiments, using *E. coli* 1, in DREAM3 challenge.

| DREAM3 N1 Size 50 | 1 module (no prior) | 2 modules | 4 modules |
|-------------------|---------------------|-----------|-----------|
| AUROC | 52.8 | 70.2 | 79.5 |
| AUPR | 2.91 | 3.96 | 7.1 |

Table 5.4: AUROC and AUPR when prior knowledge about module existence is given (DREAM3 network *E. Coli* size 50).

by LocKNI trained on available time courses and enhanced by 12, 25, 37 and 50 uniformly drawn single KOs out of the 50 total knock-outs. Incorporation of prior knowledge significantly improves the performance. However this experiment is very different from what a biologist would do. A biologist, expert in the studied biological system, would choose carefully the KO experiments to run by giving preference to hubs. Figure 5.3 shows the results for an increasing number of KO experiments chosen according the number of known targets of transcription factors of the system. Contrary to the previous experiment, which provides average results, this study shows how a biologist can improve drastically performance of a network inference algorithm by producing a very few well chosen KOs. Only 5 well chosen single KO experiments are needed to make AUROC reach 78%.

Another aspect of networks is their decomposition into modules, where modules correspond to groups of genes strongly connected in the regulation graph. Such an assumption is reasonably realistic and may be obtained in some cases from gene ontology devoted to biological processes. In order to integrate this hypothesis, the sparsity constraint may be modulated by differentiating two kinds of regulation weights in the models: the weights that concern intra-module edges

Figure 5.4: Network *E.Coli1* size 50 from DREAM3

and those that concern inter-module edges. For each subtask \mathcal{P}_i trained on a given subset and a target gene i , the sparsity constraint is relaxed by dividing the hyper-parameter λ by a parameter λ_j : edges within a module are encouraged and for them, λ_j is chosen above 1 (for instance, 2 in the numerical results) to reduce the effect of the sparsity constraint; edges between two modules are not encouraged and sparsity is imposed with a larger strength by setting $\lambda_j = 1$. The network of DREAM3 Size 50 *E.Coli1* and its decomposition in four modules is shown in Figure 5.4. To decompose this network in two modules, I consider modules 1 and 2 to be one module, modules 3 and 4 to be the other module. Table 5.4 shows the drastic improvement provided when prior knowledge about module decomposition become more precise: when using the assumption that the network can be decomposed into four modules, AUROC reaches 79.5% and the AUPR doubles when a rough knowledge about modules (two modules) is replaced by a more accurate one (four modules).

5.5 Conclusion

I have proposed a new model-driven network inference method, LocKNI, that learns sparse nonlinear models and then extracts an estimate of the target regulation graph matrix from estimated models. Interestingly, this kernel-based method shares some features with linear approaches, such as the regularization framework, and some features with ensemble-based methods such as randomization on both variables and individuals. Compared to tree-based algorithms based on the greedy and incremental minimization of a local loss, kernel-based models are

derived from the minimization of a global loss function under constraints, whose choice offers some versatility. I show that the sparsity constraint can be modulated according to the prior knowledge about existing edges. From probabilistic and Bayesian points of view, this is similar to choosing different variances in Laplace priors of the model. In practice, evidence given by knock-out data can be used to relax the sparsity constraint on potential edges. Another kind of prior knowledge, about the decomposition of the target network into modules, can be easily incorporated by imposing a lower degree of sparsity within a module and encouraging sparsity between modules. Another feature of LocKNI is the versatility it inherits from the property that kernels can be built using a convex combination of base kernels. Using 1D projection kernels, as presented in Chapter 4, provides a way to select regulators. This can be further extended to joint regulations by considering 2D projection kernels, that involve pairs of regulators.

Conclusion

In this thesis, I studied the gene regulatory network inference problem, from a bioinformatic point of view. I attempted to develop machine learning tools to extract information about the gene regulatory network (GRN), from gene expression data. This work started by noticing that many methods had been tried for this problem, but kernel methods had received little attention, although they have both relevant theoretical advantages—they are nonparametric, robustness-to-noise; they can estimate any function—and practical advantages—they have low computational costs and give good results in practice on many applications—. I have made two contributions: from a theoretical point of view, I have shown that the mean of partial derivatives is estimated consistently by some Gaussian kernel methods; from a practical point of view, I introduced and developed LocKNI, an interpretable kernel method.

On real and realistically simulated datasets, I have obtained interesting results with \widehat{Jac} , a new partial derivative estimation method, and state-of-the-art results with LocKNI. Currently, no method prevails in GRN inference, a notoriously difficult problem. The best way to infer a GRN is to average networks inferred by several methods. In my opinion, in this “large p , small N ” framework with noisy data, combining various methods will remain the best methodology. Even if one had the perfect model, it would not be well learned on such a hard dataset. By using various models, each one makes error in its own way. Gathering base learners that make independent mistakes would give a predictor that makes fewer errors. This thesis was initiated by the idea that kernels grasped an information that could not be seen by other methods—for example, linear methods cannot understand nonlinear behavior; tree-based models provide piece-wise constant functions, and not the smooth functions that may be built using kernels. \widehat{Jac} and LocKNI substantially enhanced the network inferred by the consensus of other methods. Besides, even if LocKNI appears more accurate than \widehat{Jac} , their

contribution to the consensus are very similar, hinting that the main information gain was not obtained by the way kernels were used, but by the use of kernels.

The methods introduced in this thesis may be applied to other problems. For example, these methods could be useful in *biomarker discovery for breast cancer prognosis*. Breast cancer prognosis is an important challenge, as better prognosis can save lives. Studies have shown that gene expression is relevant to detect subclasses of breast cancer. Scientists have searched for a *signature*, *i.e.* a list of genes that contain prognostic power for breast cancer. Using expression data from healthy and ill patients, LockNI could identify a signature, and be added to other classification methods used on breast cancer prognosis. Another example is the *differentially networking* problem. One wants to identify subparts of the regulatory network that change between healthy and disease-affected tissues. One could infer two networks using LockNI, one on healthy tissues, another on disease-affected ones. The two networks could be compared.

Several improvements may be introduced. For \widehat{Jac} , I used a very general regression method: kernel-ridge regression. One may use a regression better suited to gene modeling. As long as this regression asymptotically finds the true model f and its partial derivatives are bounded asymptotically, the method will consistently estimate any continuous linear form of the partial derivatives of f . For example, N.Lim [77] uses partial derivatives of operator-valued kernels. His regression model learns a structured output, better adapted to gene dynamic modeling.

LockNI can be improved by adding prior information, as shown in Chapter 5, using Z-score to weight potential regulators of a target gene. Knowing that perturbational data are more informative than others, one may change LockNI to use Z-score weighting if available, or to give additional weight to perturbational data. Because LockNI has a functional cost and a simple optimization scheme (solving a quadratic problem on α and doing a gradient descent on \mathbf{w}), it can easily be modified to take into account extra information.

Besides, this manuscript also contains several failed attempts, such as using p -values to estimate the adjacency matrix, or trying unsuccessfully many hyper-parameters selection methods. These results should not be overlooked, as they may provide information to further work on GRNI. The hyper-parameter selection problem is strategic. Many GRN inference methods choose beforehand

their hyper-parameters. Performances can be increased or degraded by choosing the right or wrong hyper-parameters. Yet, the most common hyper-parameter selection methods does not seem relevant for GRN inference. Maybe they need adjustments. Maybe a new metric has to be created. Computational cost is not a limiting issue when using ensemble methods and evaluation on out-of-bag samples, the computational burden is “only” multiplied by the number of hyper-parameters to try. This is affordable with several state-of-the-art GRN inference methods, and may substantially upgrade the methods.

Finally, many mathematical models have been tried for GRN inference. Many tools are available freely online (GENIE3, TIGRESS, CLR, ANOVA, LockNI, etc), in a ready-to-use format. An important improvement will come: more data will be available, thus increasing the size of the learning sets and methods will be able to achieve a better identification of their optimal model. Maybe we will then reach the limits of the simplified view, modeling the network only with genes, and scientists will have to incorporate all actors of gene regulation (mRNA, protein, microRNA, etc). This seems far away. For the near future, biologists have at hand powerful tools, and the significant improvements should occur by gathering data and using theses tools.

Conclusion

Appendices

Bias-variance tradeoff Considering that the loss function is the squared-loss one, the expected true risk of \hat{f} according to the distribution of the learning set \mathcal{S} is given by:

$$\mathbb{E}_{\mathcal{S}}(R(\hat{f})) = \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(y - \hat{f}(\mathbf{x}))^2 \right] \right] \quad (18)$$

$$= \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(\epsilon + f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \right] \quad (19)$$

$$= \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} [\epsilon^2] \right] + \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \right] \quad (20)$$

$$+ 2\mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - \hat{f}(\mathbf{x})) \epsilon \right] \right] \quad (21)$$

As $\mathbb{E}_{\mathcal{P}}[\epsilon] = 0$ and ϵ is independent from \mathbf{x} , $\mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - \hat{f}(\mathbf{x})) \epsilon \right] = 0$. Let σ^2 be the variance of the noise $\sigma^2 = \mathbb{E}_{\mathcal{P}}[\epsilon^2]$. f^* is the minimizer of the true risk, thus $\mathbb{E}_{\mathcal{S}}[\hat{f}(\mathbf{x})] = f^*(\mathbf{x})$.

$$\mathbb{E}_{\mathcal{S}}(R(\hat{f})) = \sigma^2 + \mathbb{E}_{\mathcal{S}} \left[\mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}) + f^*(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \right] \quad (22)$$

$$= \sigma^2 + \mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}))^2 \right] + \mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f^*(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \quad (23)$$

$$+ 2\mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}))(f^*(\mathbf{x}) - \hat{f}(\mathbf{x})) \right] \quad (24)$$

One can invert the integration over \mathcal{S} and \mathcal{P} , so $\mathbb{E}_{\mathcal{S}} [\mathbb{E}_{\mathcal{P}} [.]] = \mathbb{E}_{\mathcal{P}} [\mathbb{E}_{\mathcal{S}} [.]] = \mathbb{E}_{\mathcal{S}, \mathcal{P}} [.]$. f and f^* do not depend on the learning set \mathcal{S} , and $\mathbb{E}_{\mathcal{S}} [f^*(\mathbf{x}) - \hat{f}(\mathbf{x})] = 0$, so

$$\mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}))(f^*(\mathbf{x}) - \hat{f}(\mathbf{x})) \right] = \mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x})) \underbrace{\mathbb{E}_{\mathcal{S}} [f^*(\mathbf{x}) - \hat{f}(\mathbf{x})]}_{=0} \right] \quad (25)$$

$$= 0 \quad (26)$$

So

$$\mathbb{E}_{\mathcal{S}}(R(\hat{f})) = \mathbb{E}_{\mathcal{P}, \mathcal{S}}[\epsilon^2] + \mathbb{E}_{\mathcal{P}} \left[(f(\mathbf{x}) - f^*(\mathbf{x}))^2 \right] + \mathbb{E}_{\mathcal{S}, \mathcal{P}} \left[(f^*(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 \right] \quad (27)$$

$$= \sigma^2 + (\text{biais})^2 + (\text{variance}) \quad (28)$$

Theorem 1 Representer theorem.

Let \mathcal{X} be a nonempty set and k a positive-definite real-valued kernel on $\mathcal{X} \times \mathcal{X}$ with corresponding reproducing kernel Hilbert space \mathcal{H} . Given a training sample $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in (\mathcal{X} \times \mathbb{R})^N$, a strictly monotonically increasing real-valued function $\Omega : [0, \infty) \rightarrow \mathbb{R}$, and an arbitrary empirical risk function $L : (\mathbb{R} \times \mathbb{R})^N \rightarrow \mathbb{R} \cup \{\infty\}$, then for any $f^* \in \mathcal{H}$ satisfying

$$\hat{f} = \arg \min_{f \in \mathcal{H}} L((y_1, f(\mathbf{x}_1)), \dots, (y_N, f(\mathbf{x}_N))) + \Omega(\|f\|) \quad (29)$$

\hat{f} admits a representation of the form:

$$\hat{f}(\cdot) = \sum_{i=1}^N \alpha_i k(\cdot, \mathbf{x}_i) \quad (30)$$

with $\alpha_i \in \mathbb{R}$ for all $1 \leq i \leq N$

Proof. Given a kernel function k , thus an RKHS \mathcal{H} and a feature map ϕ (not unique). Let E be the linear span of the mappings $\phi(\mathbf{x}_i)$ in the RKHS \mathcal{H} , $E = \text{span}(\{\phi(\mathbf{x}_i)\}_{i=1 \dots N}) \subset \mathcal{H}$. Let E^T be its orthogonal complement, $E^T \oplus E = \mathcal{H}$. Let $f \in \mathcal{H}$:

$$f = v_E + v_{E^T} \quad \text{Decomposition of } f \text{ in } v_E \in E \text{ and } v_{E^T} \in E^T \quad (31)$$

$$f = \sum_{i=1}^N \alpha_i \phi(\mathbf{x}_i) + v \quad \text{Definition of } E \quad (32)$$

$$f(\mathbf{x}_j) = \langle v_E + v_{E^T}, \phi(\mathbf{x}_j) \rangle_{\mathcal{H}} \quad \text{Because } \mathcal{H} \text{ is the RKHS of } k \quad (33)$$

$$f(\mathbf{x}_j) = \langle v_E, \phi(\mathbf{x}_j) \rangle_{\mathcal{H}} \quad \text{Because } E^T \text{ is the orthogonal complement} \quad (34)$$

$$\text{of } E, \text{ and } \phi(\mathbf{x}_j) \in E \quad (35)$$

$$f(\mathbf{x}_j) = v_E(\mathbf{x}_j) \quad \forall j \in \{1, \dots, N\} \quad (36)$$

$$(37)$$

Thus a function $f \in \mathcal{H}$ will have the same value on the training set than its projection in E . Then comes equality of the loss functions

$$L((y_1, f(\mathbf{x}_1)), \dots, (y_N, f(\mathbf{x}_N))) = L((y_1, v_E(\mathbf{x}_1)), \dots, (y_N, v_E(\mathbf{x}_N))) \quad (38)$$

But the projection on E of function f will have smaller norm, as is shown:

$$\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} \quad (39)$$

$$= \langle v_E + v_{E^T}, v_E + v_{E^T} \rangle_{\mathcal{H}} \quad (40)$$

$$= \|v_E\|_{\mathcal{H}}^2 + \|v_{E^T}\|_{\mathcal{H}}^2 + 2 \underbrace{\langle v_{E^T}, v_E \rangle_{\mathcal{H}}}_{=0 \text{ because } E \text{ and } E^T \text{ orthogonal}} \quad (41)$$

$$= \|v_E\|_{\mathcal{H}}^2 + \|v_{E^T}\|_{\mathcal{H}}^2 \quad (42)$$

$$\geq \|v_E\|_{\mathcal{H}}^2 \quad (43)$$

So function f has same risk function as v_E . $\|f\| \geq \|v_E\|$, and Ω is a strictly increasing function, so the minimum of the risk function plus regularization term is reached in E , so the function f^* has the form:

$$f^* = \sum_{i=1}^N \alpha_i \phi(\mathbf{x}_i) \quad (44)$$

$$f^*(\mathbf{x}) = \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad (45)$$

$$(46)$$

□

Theorem 2 Let f be a function in space $C^1(\mathcal{X}_c, \mathcal{Y})$. Assume f has gradient bounded by M , $\|\nabla f\|_2 < M$. Then, for any $\epsilon > 0$, there exists $C_{1,\epsilon}$ such as:

$$\|f\|_\infty \leq C_{1,\epsilon} \|f\|_{\mathcal{L}^2(\Omega,\mu)} + \epsilon \quad (47)$$

and $C_{1,\epsilon}$ is independant of f .

Proof. Consider the infinite set of open balls $(B(\mathbf{x}, \frac{\epsilon}{2M}))_{\mathbf{x} \in \mathcal{X}_c}$. Clearly this set covers \mathcal{X}_c . By the Borel-Lebesgue theorem, we can extract a finite set of those balls $(B_i)_{i=1 \dots N}$ that will cover \mathcal{X}_c .

As \mathcal{X}_c is compact and f continuous on \mathcal{X}_c , $\|f\|_\infty$ is reached in a point $\mathbf{x}^* \in \mathcal{X}_c$. Let B_{i^*} be a ball of the finite set that contains \mathbf{x}^* . For all \mathbf{x} in B_{i^*} , we have $\|\mathbf{x} - \mathbf{x}^*\|_2 \leq \epsilon/M$ because they both belong to a ball of radius $\epsilon/(2M)$.

$$|f(\mathbf{x}) - f(\mathbf{x}^*)| = \langle \nabla f(c), \mathbf{x} - \mathbf{x}^* \rangle \quad \text{for some } c, \text{ by the mean value theorem} \quad (48)$$

$$\leq M \|\mathbf{x} - \mathbf{x}^*\|_2 \quad \text{Cauchy-Schwarz and bounded gradient} \quad (49)$$

$$\leq \epsilon \quad (50)$$

Thus we have:

$$\int_{B_{i^*} \cap \mathcal{X}_c} |f(x)| \mu(x) d\mathbf{x} \geq \int_{B_{i^*} \cap \mathcal{X}_c} (\|f\|_\infty - \epsilon) \mu(x) d\mathbf{x} \quad (51)$$

$$\geq (\|f\|_\infty - \epsilon) \mu(B_{i^*} \cap \mathcal{X}_c) \quad (52)$$

Noting $\mu(B_{i^*} \cap \mathcal{X}_c) = \int_{B_{i^*} \cap \mathcal{X}_c} \mu(\mathbf{x}) d\mathbf{x}$. Using a Cauchy-Schwarz theorem, we have:

$$\int_{B_{i^*} \cap \mathcal{X}_c} |f(x)| \mu(x) d\mathbf{x} \leq \sqrt{\int_{B_{i^*} \cap \mathcal{X}_c} f(x)^2 \mu(x) d\mathbf{x}} \sqrt{\int_{B_{i^*} \cap \mathcal{X}_c} 1 \mu(x) d\mathbf{x}} \quad (53)$$

$$\leq \|f\|_{\mathcal{L}^2(\Omega,\mu)} \sqrt{\mu(B_{i^*} \cap \mathcal{X}_c)} \quad (54)$$

Using equation (52) and (54), we have:

$$(\|f\|_\infty - \epsilon) \mu(B_{i^*} \cap \mathcal{X}_c) \leq \|f\|_{\mathcal{L}^2(\Omega,\mu)} \sqrt{\mu(B_{i^*} \cap \mathcal{X}_c)} \quad (55)$$

$$\|f\|_\infty \leq \frac{\|f\|_{\mathcal{L}^2(\Omega,\mu)}}{\sqrt{\mu(B_{i^*} \cap \mathcal{X}_c)}} + \epsilon \quad (56)$$

The serie $(\mu(B_i \cap \mathcal{X}_c))_i$ is finite, so there is a minimum $\mu(B^*)$. As \mathcal{X}_c is the closure of an open set, $\mu(B_i \cap \mathcal{X}_c)$ is stricly greater than 0 for all i , so $\mu(B^*)$ is strictly greater than 0. So

$$\|f\|_\infty \leq \frac{\|f\|_{\mathcal{L}^2(\Omega, \mu)}}{\sqrt{\mu(B^*)}} + \epsilon \quad (57)$$

□

Theorem 3 *Let f be a $C^1(\mathcal{X}_c, \mathcal{Y})$ function, with gradient bounded by M . Let g be a linear continuous form of $C^0(\mathcal{X}_c, \mathcal{Y})$, $g : C^0(\mathcal{X}_c, \mathcal{Y}) \rightarrow \mathbb{R}$. Then, for any $\epsilon > 0$, there exists $C_{2,g,\epsilon}$ such that:*

$$|g(\nabla f)| \leq C_{2,g,\epsilon} \|f\|_\infty + \epsilon \quad (58)$$

Proof. We are going to prove this in one dimension, without loss of generality:

$$\left| g\left(\frac{\partial f}{\partial x^j}\right) \right| \leq C_{2,g,\epsilon} \|f\|_\infty + \epsilon \quad (59)$$

As we are in a finite-dimensional space, this suffices to prove for $g(\nabla f)$. By Fréchet-Riezs'theorem, there exists $h \in C^0(\mathcal{X}_c, \mathcal{Y})$ such that:

$$g\left(\frac{\partial f}{\partial x^j}\right) = \int_{\mathcal{X}_c} h(\mathbf{x}) \frac{\partial f}{\partial x^j}(\mathbf{x}) d\mathbf{x} \quad (60)$$

By Heine's theorem, as \mathcal{X}_c is compact, h is uniformly continuous on \mathcal{X}_c . Thus, there exists δ such that, for all $(\mathbf{x}, \mathbf{x}')$ with $\|\mathbf{x} - \mathbf{x}'\|_2 < \delta$, we have

$$|h(\mathbf{x}) - h(\mathbf{x}')| \leq \frac{\epsilon}{MV(\mathcal{X}_c)} \quad (61)$$

with $V(\mathcal{X}_c) = \int_{\mathcal{X}_c} 1 d\mathbf{x}$. As we did in the proof of theorem (2), we can cover \mathcal{X}_c with a set of balls $(B_i)_{i=1 \dots N}$ of radius δ . We note $P_i = B_i \cap \mathcal{X}_c$.

$$\left| g\left(\frac{\partial f}{\partial x^j}\right) \right| = \left| \sum_i \int_{P_i} \frac{\partial f}{\partial x^j}(\mathbf{x}) (\underbrace{h(\mathbf{x}) - h_i}_{\leq \epsilon/(MV(\mathcal{X}_c))} + h_i) d\mathbf{x} \right| \quad (62)$$

with $h_i = h(\mathbf{b}_i)$, with \mathbf{b}_i the center of ball B_i

$$\left| g\left(\frac{\partial f}{\partial x^j}\right) \right| \leq \left| \sum_i \int_{P_i} \frac{\partial f}{\partial x^j}(\mathbf{x}) \frac{\epsilon}{MV(\mathcal{X}_c)} d\mathbf{x} + \int_{P_i^{-j}} \int_{P_i^j} \frac{\partial f}{\partial x^j}(\mathbf{x}) h_i dx^j d\mathbf{x}^{-j} \right| \quad (63)$$

$$\leq \epsilon + \left| \sum_i h_i \int_{P_i^{pj}} f(\mathbf{x}_{\max j}) - f(\mathbf{x}_{\min j}) d\mathbf{x}_{-j} \right| \quad (64)$$

With $\mathbf{x}_{\max j} = \arg \max_{u \in P_i} \{u^j, u^{-j} = x^{-j}\}$ and $\mathbf{x}_{\min j} = \arg \min_{u \in P_i} \{u^j, u^{-j} = x^{-j}\}$.

$$\left| g\left(\frac{\partial f}{\partial x^j}\right) \right| \leq \epsilon + \sum_i 2 \|f\|_\infty |h_i V(P_i^{-j})| \quad (65)$$

$$\leq \epsilon + 2V(\mathcal{X}_c^{-j}) \|h\|_\infty \|f\|_\infty \quad (66)$$

□

Lemma 4 *Let \mathcal{H} be the RKHS of universal kernel k . If, for all $\mathbf{x} \in \Omega$*

- *The kernel is constant $k(\mathbf{x}, \mathbf{x}) = c$*
- *On point (\mathbf{x}, \mathbf{x}) , the gradient of the kernel is null, $\nabla_{\mathbf{z}} k(\mathbf{x}, \mathbf{z})|_{\mathbf{z}=\mathbf{x}} = 0_p$*
- *On point (\mathbf{x}, \mathbf{x}) , the Hessian matrix $H(\mathbf{x}, \mathbf{z})_{ij} = \frac{\partial^2 k(\mathbf{x}, \mathbf{z})}{\partial z^i \partial z^j}$ has eigenvalues bounded by a constant M , $|\langle u, H(\mathbf{x}, \mathbf{x})u \rangle| \leq M\|u\|^2$ for all $\mathbf{x} \in \mathcal{X}$ and all $u \in \mathbb{R}^p$*

Then, for all $f \in \mathcal{H}$, for all $\mathbf{x} \in \mathcal{X}_c$:

$$\|\nabla f(\mathbf{x})\|_2 \leq \sqrt{M} \|f\|_{\mathcal{H}} \quad (67)$$

Proof. Let $f \in \mathcal{H}$. We have:

$$|f(\mathbf{x}) - f(\mathbf{x}')| = |\langle f, K_{\mathbf{x}} - K_{\mathbf{x}'} \rangle_{\mathcal{H}}| \quad (68)$$

$$\leq \|f\|_{\mathcal{H}} \|K_{\mathbf{x}} - K_{\mathbf{x}'}\|_{\mathcal{H}} \quad (69)$$

Using the property of reproducing kernel and Cauchy-Schwarz's theorem. Using polarization identity, the kernel trick, Taylor expansion and lemma (4)'s assumptions, we have:

$$\|K_{\mathbf{x}} - K_{\mathbf{x}+\mathbf{h}}\|_{\mathcal{H}}^2 = \|K_{\mathbf{x}}\|_{\mathcal{H}}^2 + \|K_{\mathbf{x}+\mathbf{h}}\|_{\mathcal{H}}^2 - 2 \langle K_{\mathbf{x}}, K_{\mathbf{x}+\mathbf{h}} \rangle_{\mathcal{H}} \quad (70)$$

$$= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x} + \mathbf{h}, \mathbf{x} + \mathbf{h}) - 2k(\mathbf{x}, \mathbf{x} + \mathbf{h}) \quad (71)$$

$$= 2c - 2c - 2 \langle \nabla_{\mathbf{x}} k(\mathbf{x}, \mathbf{x}), \mathbf{h} \rangle \quad (72)$$

$$= - \langle \mathbf{h}, H(\mathbf{x}, \mathbf{x}) \mathbf{h} \rangle + o(\|\mathbf{h}\|_2^2) \quad (73)$$

$$= - \langle \mathbf{h}, H(\mathbf{x}, \mathbf{x}) \mathbf{h} \rangle + o(\|\mathbf{h}\|_2^2) \quad (74)$$

From equations (69,74), we have:

$$|f(\mathbf{x}) - f(\mathbf{x} + \mathbf{h})| = | \langle \nabla_{\mathbf{x}} f, \mathbf{h} \rangle_{\Omega} + o(\|\mathbf{h}\|_2) | \quad (75)$$

$$\leq \|f\|_{\mathcal{H}} (\|\mathbf{h}\| \sqrt{M} + o(\|\mathbf{h}\|_2)) \quad (76)$$

Thus

$$\|\nabla f\|_2 \leq \sqrt{M} \|f\|_{\mathcal{H}} \quad (77)$$

□

Lemma 5 *The Gaussian kernel of bandwidth σ satisfies the hypothese of lemma (4).*

Proof. This result is obtained by a few calculations with the Gaussian kernel:

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|_2^2}{2\sigma^2}\right) \quad (78)$$

$$k(\mathbf{x}, \mathbf{x}) = 1 \quad (79)$$

$$\frac{\partial k}{\partial z^i}(\mathbf{x}, \mathbf{z}) = \frac{-(z^i - x^i)}{\sigma^2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|_2^2}{2\sigma^2}\right) \quad (80)$$

$$= 0 \text{ if } \mathbf{x} = \mathbf{z} \quad (81)$$

$$\frac{\partial^2 k}{\partial z^i \partial z^j}(\mathbf{x}, \mathbf{z}) = \frac{(z^i - x^i)(z^j - x^j)}{\sigma^4} \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|_2^2}{2\sigma^2}\right) \quad (82)$$

$$= 0 \text{ if } \mathbf{x} = \mathbf{z} \quad (83)$$

$$\frac{\partial^2 k}{\partial (z^i)^2}(\mathbf{x}, \mathbf{z}) = \left(\frac{(z^i - x^i)^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|_2^2}{2\sigma^2}\right) \quad (84)$$

$$= \frac{-1}{\sigma^2} \text{ if } \mathbf{z} = \mathbf{x} \quad (85)$$

We have the Hessian matrix $H(\mathbf{x}, \mathbf{x}) = -Id/\sigma^2$, which has eigenvalues bounded by $1/\sigma^2$.

□

Theorem 6 *With samples $(\mathbf{x}_i, y_i)_{i=1\dots\ell}$ i.i.d., for any $\epsilon, \eta > 0$, any continuous linear form g of $C^0(\mathcal{X}_c, \mathcal{Y})$, $g : C^0(\mathcal{X}_c, \mathcal{Y}) \rightarrow \mathbb{R}$, there exists ℓ_0 such as if $\ell \geq \ell_0$, then, with probability greater than $1 - \eta$:*

$$\left| \int_{\mathcal{X}_c} g(\nabla \hat{f}_\ell(\mathbf{x})) - g(\nabla f(\mathbf{x})) d\mathbf{x} \right| \leq \epsilon \quad (86)$$

with \hat{f}_ℓ the estimator of f based on a Gaussian kernel ridge regression or Gaussian partial least-square regression.

Proof. From [96], we have consistency of kernel-ridge regression. From [97], we have consistency of partial least-square regression. In both articles, authors prove that the difference between the estimator and the true function is bounded in both the norms $\|\cdot\|_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{L}^2(\Omega, \mu)}$.

Using lemma (4) and bound on $\|\hat{f}_\ell - f\|_{\mathcal{H}}$, we obtain that, for ℓ sufficiently large, we have, with probability greater than $1 - \eta$:

$$\|\nabla \hat{f}_\ell(\mathbf{x}) - \nabla f(\mathbf{x})\|_2 \leq 1 \text{ for all } \mathbf{x} \in \mathcal{X}_c \quad (87)$$

Let g be the linear form. By Fréchet-Riezs' theorem, there exists $\nu \in C^0(\mathcal{X}_c, \mathcal{Y})$ such that::

$$g : f \mapsto \int_A f(\mathbf{x}) \nu(\mathbf{x}) d\mathbf{x} \quad (88)$$

Using theorem (3):

$$\left| g\left(\frac{\partial \hat{f}_\ell - f}{\partial x_j}\right) \right| \leq C_{2,g,\epsilon} \|\hat{f}_\ell - f\|_\infty + \frac{\epsilon}{3} \quad (89)$$

Using theorem (2):

$$\|\hat{f}_\ell - f\|_\infty \leq C_{1,\epsilon} \|\hat{f}_\ell - f\|_{\mathcal{L}^2(\Omega, \mu)} + \frac{\epsilon}{3C_{2,g,\epsilon}} \quad (90)$$

Using consistency result, for ℓ sufficiently large, we have, with probability greater than $1 - \eta$:

$$\|\hat{f}_\ell - f\|_{L^2(\Omega, \mu)} \leq \frac{\epsilon}{C_{1,\epsilon} C_{2,g,\epsilon} 3} \quad (91)$$

Combining the 3 equations (89,90,91), we obtain the result:

$$\left| \int_A \nabla \hat{f}_\ell(\mathbf{x}) - \nabla f(\mathbf{x}) \nu(\mathbf{x}) d\mathbf{x} \right| \leq \epsilon \quad (92)$$

□

Bounds for integral estimation *Let f be a $C^1(X_c, \mathcal{Y})$ function whose partial derivatives are bounded by M . Let $(\mathbf{x}_i)_{i=1,\dots,\ell}$ be independently and identically distributed random variables, drawn from a distribution μ . The difference between the empirical mean of a partial derivative*

$$S_l = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\partial f}{\partial x^j}(\mathbf{x}_i)$$

and its true mean

$$e = \mathbb{E} \left(\frac{\partial f}{\partial x^j} \right) = \int_{X_c} \frac{\partial f}{\partial x^j}(\mathbf{x}) \mu(\mathbf{x}) d\mathbf{x}$$

follows the normal distribution $\mathcal{N}(0, \ell^{-1} \sigma^2)$, with $\sigma \leq M$. Thus we have

$$P \left(\frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\partial f}{\partial \mathbf{x}^j}(\mathbf{x}_i) - \int_{X_c} \frac{\partial f}{\partial \mathbf{x}^j} \mu(\mathbf{x}) d\mathbf{x} \leq m \right) \leq \Phi \left(\frac{mM}{\sqrt{\ell}} \right) \quad (93)$$

with $\Phi(m)$ the cumulative distribution of $\mathcal{N}(0, 1)$

Proof. Firstly, I show that the variance of $\frac{\partial f}{\partial x^j}$ is lower or equal to M^2 . Then I verify that I meet the central limit theorem's assumption, and apply it to find the previously stated bounds.

$$\text{Var}\left(\frac{\partial f}{\partial x^j}\right) = \mathbb{E}\left(\left(\frac{\partial f}{\partial x^j} - e\right)^2\right) \quad (94)$$

$$= \int_{\frac{\partial f}{\partial x^j}(x) \geq e} \left(\frac{\partial f}{\partial x^j} - e\right)^2 \mu(\mathbf{x}) d\mathbf{x} + \int_{\frac{\partial f}{\partial x^j}(x) < e} \left(\frac{\partial f}{\partial x^j} - e\right)^2 \mu(\mathbf{x}) d\mathbf{x} \quad (95)$$

$$\leq \int_{\frac{\partial f}{\partial x^j}(x) \geq e} (M - e)^2 + \int_{\frac{\partial f}{\partial x^j}(x) < e} (-M - e)^2 \quad (96)$$

Let $p = P\left(\frac{\partial f}{\partial x^j}(x) \geq e\right)$. In the worst-case scenario where the partial derivative only takes the values $\{-M, M\}$, $e = M(2p - 1)$.

$$\text{Var}\left(\frac{\partial f}{\partial x^j}\right) \leq p(M - M(2p - 1))^2 + (1 - p)(-M - M(2p - 1))^2 \quad (97)$$

$$\leq M^2(p(2 - 2p)^2 + (1 - p)(2p)^2) \quad (98)$$

$$\leq 4M^2(p(1 - p)) \quad (99)$$

And $p(1 - p) \leq 1/4$, minimum reached in $p = 1/2$. Thus:

$$\text{Var}\left(\frac{\partial f}{\partial x^j}\right) \leq M^2 \quad (100)$$

As \mathbf{x}_i are i.i.d., $\frac{\partial f(\mathbf{x}_i)}{\partial x^j}$ are i.i.d.. Mean and variance of $\frac{\partial f(\mathbf{x}_i)}{\partial x^j}$ are defined and finite, thus, I can apply the central limit theorem. Noting $S_\ell = \sum_{i=1}^{\ell} \frac{\partial f(\mathbf{x}_i)}{\partial x^j}$, and σ the standard deviation of $\frac{\partial f(\mathbf{x}_i)}{\partial x^j}$, the theorem implies that:

$$\frac{S_\ell - \ell e}{\sigma \sqrt{\ell}} \xrightarrow{\ell \rightarrow +\infty} \mathcal{N}(0, 1) \quad (101)$$

or

$$P\left(\frac{S_\ell}{\ell} - e \leq m\right) = \Phi\left(\frac{m\sigma}{\sqrt{\ell}}\right) \quad (102)$$

□

Bibliography

- [1] Sarah Leavitt and DeWitt Stetten Jr. *Deciphering the Genetic Code: Marshall Nirenberg*. Office of NIH History, 2004.
- [2] H. Pearson. Genetics: what is a gene? *Nature*, 441(7092):398–401, 2006.
- [3] K.A. Baggerly, K.R. Coombes, and J.S. Morris. An introduction to high-throughput bioinformatics data. 2006.
- [4] V.A. Huynh-Thu. Machine learning-based feature ranking: Statistical interpretation and gene network inference. 2012.
- [5] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.
- [6] R. Bellman. *Adaptive control processes: a guided tour*, volume 4. Princeton university press Princeton, 1961.
- [7] K-R Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*, 12(2):181–201, 2001.
- [8] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.
- [9] B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.
- [10] A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

- [11] C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.
- [12] M.G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
- [13] L.A. Goodman and W.H. Kruskal. Measures of association for cross classifications*. *Journal of the American Statistical Association*, 49(268):732–764, 1954.
- [14] A. S. Butte and I. S. Kohane. *Relevance networks: A first step toward finding genetic regulatory networks within microarray data*. Springer, 2003.
- [15] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
- [16] I. Farkas, H. Jeong, T. Vicsek, A.-L. Barabási, and Z.N. Oltvai. The topology of the transcription regulatory network in the yeast, *saccharomyces cerevisiae*. *Physica A: Statistical Mechanics and its Applications*, 318(3):601–612, 2003.
- [17] A.-L. Barabási and Z.N. Oltvai. Network biology: understanding the cell’s functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.
- [18] R. Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.
- [19] M.P.H. Stumpf, C. Wiuf, and R.M. May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(12):4221–4224, 2005.
- [20] R. Khanin and E. Wit. How scale-free are biological networks. *Journal of computational biology*, 13(3):810–818, 2006.
- [21] H. Agrawal. Extreme self-organization in networks constructed from gene expression data. *Physical review letters*, 89(26):268702, 2002.
- [22] G. Chen, P. Larsen, E. Almasri, and Y. Dai. Rank-based edge reconstruction for scale-free genetic regulatory networks. *BMC bioinformatics*, 9(1):75, 2008.

- [23] J. Cohen. Eta-squared and partial eta-squared in fixed factor anova designs. *Educational and Psychological Measurement*, 1973.
- [24] Robert Küffner, Tobias Petri, Pegah Tavakkolkhah, Lukas Windhager, and Ralf Zimmer. Inferring gene regulatory networks by anova. *Bioinformatics*, 28(10):1376–1382, 2012.
- [25] A. S. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing*, pages 418–429, 2000.
- [26] A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano. Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, (7), 2006.
- [27] J.J. Faith, B. Hayete, J.T. Thaden, et al. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biology*, 5(1):e8, 2007.
- [28] R. Opgen-Rhein and K. Strimmer. Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.
- [29] C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley symposium on mathematical statistics and probability*, volume 1, pages 197–206, 1956.
- [30] J. Whittaker. *Graphical models in applied multivariate statistics*. Wiley Publishing, 2009.
- [31] A.P. Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.
- [32] D. Kollar and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.
- [33] I.M. Johnstone and B.W. Silverman. Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences. *The Annals of Statistics*, 32(4):1594–1649, 2004.

- [34] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. von Rohr, L. Thiele, et al. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biol*, 5(11):R92, 2004.
- [35] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300, 1995.
- [36] H. Li and J. Gui. Gradient directed regularization for sparse gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7(2):302–317, 2006.
- [37] A. Pinna, N. Soranzo, and A. de la Fuente. From knockouts to networks: Establishing direct cause-effect relationships through graph analysis. *PLoS ONE*, 5(10), 10 2010.
- [38] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [39] M. Gustafsson, M. Hornquist, and A. Lombardi. Constructing and analyzing a large-scale gene-to-gene regulatory network-lasso-constrained inference and biological validation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2:254–261, 2005.
- [40] A. Shojaie, S. Basu, and G. Michailidis. Adaptive thresholding for reconstructing regulatory networks from time-course gene expression data. *Statistics in Biosciences*, 4:66–83, 2012.
- [41] D. Marbach, J. C. Costello, R. Küffner, et al. Wisdom of crowds for robust gene network inference. *Nature Methods*, 2012.
- [42] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2005.
- [43] J. Chiquet, Y. Grandvalet, and C. Charbonnier. Sparsity with sign-coherent groups of variables via the cooperative-lasso. *The Annals of Applied Statistics*, 6(2):795–830, 2012.

- [44] Y. Wang, T. Joshi, X.-S. Zhang, D. Xu, and L. Chen. Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, 22(19):2413–2420, 2006.
- [45] A.-C. Haury, F. Mordelet, P. Vera-Licona, and J.-P. Vert. Tigress: Trustful inference of gene regulation using stability selection. *BMC Systems Biology*, (6:145), 2012.
- [46] N. Meinshausen and P. Bühlmann. Stability selection (with discussion). *Journal of the Royal Statistical Society: Series B*, (72):pp. 417–473, 2010.
- [47] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.
- [48] T. Akutsu, S. Miyano, S. Kuhara, et al. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Pacific Symposium on Biocomputing*, volume 4, pages 17–28, 1999.
- [49] S. Liang, S. Fuhrman, R. Somogyi, et al. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific symposium on biocomputing*, volume 3, page 2, 1998.
- [50] I. Shmulevich and W. Zhang. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18(4):555–565, 2002.
- [51] I. Tabus, R. Jorma, and J. Astola. *Normalized Maximum Likelihood Models for Boolean Regression with Application to Prediction and Classification in Genomics*. Springer US, 2006.
- [52] S. Huang. Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *Journal of Molecular Medicine*, 77:469–480, 1999. 10.1007/s001099900023.
- [53] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437 – 467, 1969.
- [54] D.M. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks: search methods and experimental results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, pages 569–595, 1995.

- [55] D.M. Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.
- [56] N. Friedman, M. Linial, I. Nachman, and D. Pe’er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.
- [57] N. Friedman, I. Nachman, and D. Peér. Learning bayesian network structure from massive datasets: the sparse candidate algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI’99, pages 206–215, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [58] A.J. Hartemink, D.K. Gifford, T. Jaakkola, and R.A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific symposium on biocomputing*, volume 6, page 266, 2001.
- [59] T.S. Jaakkola and M.I. Jordan. Variational probabilistic inference and the qmr-dt network. *Journal of Artificial Intelligence Research*, 10:291–322, 1999.
- [60] H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc., 1999.
- [61] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.
- [62] M. Zou and S.D. Conzen. A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.
- [63] S. Imoto, K. Sunyong, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, and S. Miyano. Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. In *Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society*, pages 219–227. IEEE, 2002.

- [64] S. Kim, S. Imoto, and S. Miyano. Dynamic bayesian network and non-parametric regression for nonlinear modeling of gene networks from time series gene expression data. *Biosystems*, 75(1):57–65, 2004.
- [65] E.R. Morrissey, M.A. Juárez, K.J. Denby, and N.J. Burroughs. Inferring the time-invariant topology of a nonlinear sparse gene regulatory network using fully bayesian spline autoregression. *Biostatistics*, 12(4):682–694, 2011.
- [66] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. dAlche Buc. Gene networks inference using dynamic bayesian networks. *Bioinformatics*, 19(suppl 2):ii138–ii148, 2003.
- [67] C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D.L. Wild, and F. Falciani. Modeling t-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, 2004.
- [68] Z. Wang, X. Liu, Y. Liu, J. Liang, and V. Vinciotti. An extended kalman filtering approach to modeling nonlinear dynamic gene regulatory networks via short gene expression time series. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(3):410–419, 2009.
- [69] L. Qian, H. Wang, and E.R. Dougherty. Inference of noisy nonlinear differential equation models for gene regulatory networks using genetic programming and kalman filtering. *Signal Processing, IEEE Transactions on*, 56(7):3327–3339, 2008.
- [70] A. Noor, E. Serpedin, M. Nounou, and H. Nounou. Inferring gene regulatory networks via nonlinear state-space models and exploiting sparsity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(4):1203–1211, 2012.
- [71] R. Thomas, S. Mehrotra, E.T. Papoutsakis, and V. Hatzimanikatis. A model-based optimization framework for the inference on gene regulatory networks from dna array data. *Bioinformatics*, 20(17):3221–3235, 2004.
- [72] N. Noman and H. Iba. Inference of gene regulatory networks using s-system and differential evolution. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 439–446. ACM, 2005.

- [73] S.I. Ao and V. Palade. Ensemble of elman neural networks and support vector machines for reverse engineering of gene regulatory networks. *Applied Soft Computing*, 11(2):1718–1726, 2011.
- [74] M. Grimaldi, R. Visintainer, and G. Jurman. Regnann: Reverse engineering gene networks using artificial neural networks. *PloS one*, 6(12):e28646, 2011.
- [75] T. Äijö and H. Lähdesmäki. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22):2937–2944, 2009.
- [76] R. Bonneau, D.J. Reiss, P. Shannon, M. Facciotti, L. Hood, N.S. Baliga, V. Thorsson, et al. The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol*, 7(5):R36, 2006.
- [77] N. Lim, Y. Şenbabaoğlu, G. Michailidis, and F. dAlché Buc. Okvar-boost: a novel boosting algorithm to infer nonlinear dynamics and interactions in gene regulatory networks. *Bioinformatics*, 29(11):1416–1423, 2013.
- [78] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. Inferring regulatory networks from expression data using tree-based methods. *PlosOne*, (5(9):e12776), 2010.
- [79] F. Emmert-Streib, G.V. Glazko, G. Altay, and R. de Matos Simoes. Statistical inference and reverse engineering of gene regulatory networks from observational expression data. *Frontiers in genetics*, 3, 2012.
- [80] V. Narendra, N. I Lytkin, C.F. Aliferis, and A. Statnikov. A comprehensive assessment of methods for de-novo reverse-engineering of genome-scale regulatory networks. *Genomics*, 97(1):7–18, 2011.
- [81] G. Stolovitzky, DON Monroe, and A. Califano. Dialogue on reverse-engineering assessment and methods. *Annals of the New York Academy of Sciences*, 1115(1):1–22, 2007.
- [82] D. Marbach, Prill R.J., T. Schaffter, C. Mattiussi, Floreano D., and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences USA*, 107(14):6286–6291, 2010.

- [83] R. Charnigo, M. Francoeur, P. Kenkel, M. Pinar Mengüç, B. Hall, and C. Srinivasan. Estimating quantitative features of nanoparticles using multiple derivatives of scattering profiles. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 112(8):1369–1382, 2011.
- [84] M. Francoeur, P.G. Venkata, and M.P. Mengüç. Sensitivity analysis for characterization of gold nanoparticles and agglomerates via surface plasmon scattering patterns. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 106(1):44–55, 2007.
- [85] R. Charnigo. Estimating multiple derivatives simultaneously: What is optimal? *Journal of Biometrics & Biostatistics*, 2011.
- [86] J.O. Ramsay and B.W. Silverman. *Functional data analysis*. Springer-Verlag, 1997.
- [87] J.M. Steppe and K.W. Bauer Jr. Feature saliency measures. *Computers & Mathematics with Applications*, 33(8):109–126, 1997.
- [88] P. Leray and P. Gallinari. Feature selection with neural networks. *Behaviormetrika*, 26:145–166, 1999.
- [89] K. De Brabanter, J. De Brabanter, and B. De Moor. Nonparametric derivative estimation. In *Proc. of the 23rd Benelux Conference on Artificial Intelligence*, pages 75–81, 2011.
- [90] R. Charnigo, B. Hall, and C. Srinivasan. A generalized c p criterion for derivative estimation. *Technometrics*, 53(3):238–253, 2011.
- [91] R. Khemchandani, S. Chandra, et al. Regularized least squares support vector regression for the simultaneous learning of a function and its derivatives. *Information Sciences*, 178(17):3402–3414, 2008.
- [92] V. Rondonotti, J.S. Marron, and C. Park. Sizer for time series: a new approach to the analysis of trends. *Electronic Journal of Statistics*, 1:268–289, 2007.
- [93] R. Charnigo and C. Srinivasan. Self-consistent estimation of mean response functions and their derivatives. *Canadian Journal of Statistics*, 39(2):280–299, 2011.

- [94] S Mosci, L. Rosasco, S. Villa, M. Santoro, and A. Verri. Nonparametric sparsity and regularization. *Journal of Machine Learning Research*, 14:1665–1714, 2013.
- [95] C.A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *The Journal of Machine Learning Research*, 7:2651–2667, 2006.
- [96] A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.
- [97] Gilles Blanchard and Nicole Krämer. Kernel partial least squares is universally consistent. *arXiv preprint arXiv:0902.4380*, 2009.
- [98] J.S. Simonoff. *Smoothing methods in statistics*. Springer, 1996.
- [99] E. Giné and A. Guillou. Rates of strong uniform consistency for multivariate kernel density estimators. In *Annales de l’Institut Henri Poincaré (B) Probability and Statistics*, volume 38, pages 907–921. Elsevier, 2002.
- [100] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [101] A.-C. Haury. Feature selection for gene expression data: molecular signatures for breast cancer outcome prediction and gene regulatory network inference. phd thesis, école nationale des mines et des ponts,. Technical report, 2012.
- [102] R.A. Fisher. The design of experiments. 1935.
- [103] E.J.G. Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130, 1937.
- [104] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.
- [105] Y. LeCun, J.S. Denker, S.A. Solla, R.E. Howard, and L.D. Jackel. Optimal brain damage. In *NIPs*, volume 2, pages 598–605, 1989.
- [106] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

- [107] R.J. Prill, D. Marbach, J. Saez-Rodriguez, P.K. Sorger, L.G. Alexopoulos, X. Xue, N. D Clarke, G. Altan-Bonnet, and G. Stolovitzky. Towards a rigorous assessment of systems biology models: the dream3 challenges. *PloS one*, 5(2):e9202, 2010.
- [108] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.
- [109] T. Schaffter, D. Marbach, and D. Floreano. Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011.
- [110] S Lèbre. Inferring dynamic genetic networks with low order independencies. *Statistical applications in genetics and molecular biology*, 8(1):1–38, 2009.
- [111] J. Surowiecki. The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business. *Economies, Societies and Nations*, 2004.
- [112] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.
- [113] D. Tuia, G. Camps-Valls, G. Matasci, and M. Kanevski. Learning relevant image features with multiple-kernel classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 48(10):3780–3791, 2010.
- [114] C. Sotiriou, P. Wirapati, S. Loi, A. Harris, S. Fox, J. Smeds, H. Nordgren, P. Farmer, V. Praz, B. Haibe-Kains, et al. Gene expression profiling in breast cancer: understanding the molecular basis of histologic grade to improve prognosis. *Journal of the National Cancer Institute*, 98(4):262–272, 2006.
- [115] L. Ein-Dor, I. Kela, G. Getz, D. Givol, and E. Domany. Outcome signature genes in breast cancer: is there a unique set? *Bioinformatics*, 21(2):171–178, 2005.
- [116] G.H. John, R. Kohavi, K. Pfleger, et al. Irrelevant features and the subset selection problem. In *ICML*, volume 94, pages 121–129, 1994.

- [117] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [118] C.M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.
- [119] M. Zaffalon and M. Hutter. Robust feature selection by mutual information distributions. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 577–584. Morgan Kaufmann Publishers Inc., 2002.
- [120] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and regression trees belmont. CA: Wadsworth International Group, 1984.
- [121] G.R.G. Lanckriet, M. Deng, N. Cristianini, M.I. Jordan, W.S. Noble, et al. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the pacific symposium on biocomputing*, volume 9, page 2. World Scientific Singapore, 2004.
- [122] S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable svms for biological sequence classification. In *Research in Computational Molecular Biology*, pages 389–407. Springer, 2005.
- [123] B. Guo, S. R Gunn, R. I Damper, and J.D.B. Nelson. Customizing kernel functions for svm-based hyperspectral image classification. *Image Processing, IEEE Transactions on*, 17(4):622–629, 2008.
- [124] R. Takashima, T. Takiguchi, and Y. Ariki. Feature selection based on multiple kernel learning for single-channel sound source localization using the acoustic transfer function. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2696–2699. IEEE, 2011.
- [125] M. Gonen and E. Alpaydyn. Multiple kernel learning algorithms. *JMLR*, 12:2211–2268, 2011.
- [126] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.
- [127] S.R. Gunn and J.S. Kandola. Structural modelling with sparse kernels. *Machine learning*, 48(1-3):137–163, 2002.

- [128] D.G. Luenberger. *Linear and nonlinear programming*. Springer, 2003.
- [129] T.F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.
- [130] W. Duch. Filter methods. In *Feature Extraction*, pages 89–117. Springer, 2006.
- [131] N. Sánchez-Marono, A. Alonso-Betanzos, and M. Tombilla-Sanromán. Filter methods for feature selection—a comparative study. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007*, pages 178–187. Springer, 2007.
- [132] N. Cristianini, N. Shawe-Taylor, A. Elisseeff, and A. Kandola. On kernel target alignment. *Advances in neural information processing systems*, 14:367, 2002.
- [133] J. Kandola, J. Shawe-Taylor, and N. Cristianini. On the extensions of kernel alignment. *NeuroCOLT technical report*, 2002.
- [134] M. Ramona, G. Richard, and B. David. Multiclass feature selection with kernel gram-matrix-based criteria. *IEEE Transactions on Neural Networks*, 2012.
- [135] K. Kira and L.A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992.
- [136] I. Kononenko. Estimating attributes: analysis and extensions of relief. In *Machine Learning: ECML-94*, pages 171–182. Springer, 1994.
- [137] Marko Robnik-Šikonja and Igor Kononenko. An adaptation of relief for attribute estimation in regression. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML97)*, pages 296–304, 1997.
- [138] M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.
- [139] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *NIPS*, volume 12, pages 668–674, 2000.

Bibliography

- [140] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *Advances in neural information processing systems*, pages 553–560, 2002.
- [141] G.I. Allen. Automatic feature selection via weighted kernels and regularization. *Journal of Computational and Graphical Statistics*, 22(2):284–299, 2013.
- [142] K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41–59, 2003.
- [143] L. Breiman. Heuristics of instability and stabilization in model selection. *The annals of statistics*, 24(6):2350–2383, 1996.
- [144] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific symposium on biocomputing*, volume 7, pages 6–17, 2001.
- [145] D.N. Politis and H. White. Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23(1):53–70, 2004.
- [146] A. Patton, D.N. Politis, and H. White. Correction to automatic block-length selection for the dependent bootstrap by d. politis and h. white. *Econometric Reviews*, 28(4):372–375, 2009.
- [147] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.
- [148] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.
- [149] J.J. Faith, M.E. Driscoll, V.A. Fusaro, E.J. Cosgrove, B. Hayete, F.S. Juhn, S.J. Schneider, and T.S. Gardner. Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. *Nucleic acids research*, 36(suppl 1):D866–D870, 2008.
- [150] H. Salgado, M. Peralta-Gil, S. Gama-Castro, et al. Regulondb v8. 0: omics data sets, evolutionary conservation, regulatory phrases, cross-validated gold standards and more. *Nucleic acids research*, 41(D1):D203–D213, 2013.