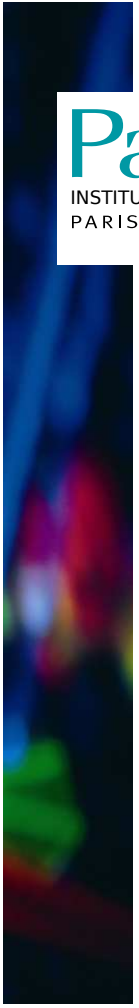
Learning from positive and unlabeled examples in biology

Fantine Mordelet



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°432 :
Sciences des métiers de l'ingénieur

Doctorat européen ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité « Bio-informatique »

présentée et soutenue publiquement par

Fantine MORDELET

le 15 décembre 2010

Méthodes d'apprentissage statistique à partir d'exemples positifs et indéterminés en biologie.

~ ~ ~

Learning from positive and unlabeled examples in biology.

Directeur de thèse : **Jean-Philippe VERT**

Jury

François DENIS, Professeur, Laboratoire d'Informatique Fondamentale, Université de Provence	Rapporteur
Yves MOREAU, Professeur, Bioinformatics Research Group, Université de Leuven (Belgique)	Rapporteur
Florence D'ALCHE-BUC, Professeur, Laboratoire IBISC, Université d'Evry-Val d'Essonne	Examinateur
Denis THIEFFRY, Professeur, Département de Biologie IBENS, Ecole Normale Supérieure	Examinateur
Jean-Philippe VERT, Maître de recherche, Centre de Bio-Informatique, Mines ParisTech	Directeur de thèse

T H È S E

MINES ParisTech
Centre de Bio-Informatique
35 rue Saint-Honoré, 77300 Fontainebleau, France

Contents

Remerciements

Je tiens tout d'abord à remercier M. Y. Moreau, F. Denis, D. Thieffry et Mme F. D'Alché-Buc pour avoir accepté d'être membres de mon jury de thèse et de faire ainsi partie de la dizaine de personnes qui aura lu cette thèse (ce qui comprend déjà mon humble personne et mon directeur de thèse). Ce qui m'amène tout naturellement à remercier le directeur de thèse sus-mentionné et Emmanuel Barillot, le directeur de l'u900 pour m'avoir accueillie dans cette belle équipe. Jean-Philippe, en vertu de tout ce que tu m'as apporté, je te souhaite d'élever plein d'autres petits CBIO, qui baigneront dès la prime enfance de leur carrière de chercheur dans la douce idéologie du grand noyau.

Ensuite, il y a mes collègues, qui ont aussi joué un rôle crucial dans mon apprentissage.. Quelle consolation de les avoir quand on doute, quand les choses ne marchent pas si bien qu'on aimerait (et ça arrive souvent!). Et puis c'est comme quand on est petit, que les vacances se terminent et qu'on est triste de retourner à l'école mais on sait quand même qu'on sera content de revoir les copains. Sauf qu'à l'époque, on ne racontait pas des blagues foireuses en buvant du café dès le matin. La liste est longue mais on va essayer de ne pas oublier de monde. Précisons aussi que certains peuvent se trouver dans plusieurs catégories même je ne cite leur nom qu'une fois. Commençons par les curistes. Il y a mes cops de Martini: Laurence, Fanny, Jenny, Loredana et les deux accolytes de mon "trio des thésardes": Cécile et Anne. Sur le plan de la logistique, elles sont super aussi, elles ont toujours le dentifrice, le maillot de sport ou le cachet qui me manque (même si le Spasfon ne fait pas grand chose contre l'appendicite). Il y a ceux de la cantine de l'ENS : Gautier, Patrick, Stéphane et Stéphane, Nicolas, Jonas, Caroline, Isabel, Joern avec qui on a bien rigolé pour oublier le goût des aliments qu'on y sert. Il y a tous ceux que j'ai harcelé pour des questions diverses et qui m'ont répondu avec gentillesse: comment marche un ordi avec Pierre et Stuart, la biologie avec Simon, Perl et bash avec Séverine et Georges (vous ne vous en souvenez pas forcément mais moi si). Et mes patients collègues de bureau qui répondent invariablement "à tes souhaits" à tous mes

éternuements et Dieu sait qu'il en faut de la constance pour tenir le rythme : Bruno, Valentina et Paola. Une mention spéciale à Pierre N. qui m'a initié au monde de la bioinfo; il a été mon professeur, puis il m'a aidée à trouver un stage de césure et enfin, il m'a fait découvrir Curie quand je cherchais un endroit où passer trois merveilleuses années de thèse, ça tombait bien, non?. Maintenant, on passe au CBIO. Précisons tout de suite que le CBIO, c'est bien et que tout le monde devrait en être convaincu. Il y a les grands pontes de la secte (euh, du labo) qui ont toujours de bons conseils, Ãă savoir, "un bon chercheur doit toujours être un peu malheureux, avoir un peu faim et un peu froid" et toutes ces choses indispensables pour bien commencer une carrière : Véronique, Yoshi et Jean-Philippe. Il y a mes "grands frères" de thèse : Laurent, Franck, Misha, Kevin et Martial. Ils m'ont appris la fenêtre noire (enfin, le terminal comme il paraît qu'il faut dire), le cluster, me servir d'un Mac, la régularisation ("plus C est grand, plus... c'est quoi déjà?"), vénérer notre chef et tellement d'autres choses. Et tout ça m'a servi ensuite pour accueillir ma "petite soeur" de thèse : Anne-Claire, mais l'échange a eu lieu dans l'autre sens aussi de bien des manières.

Passons à mes amis, on ne peut pas dire, il en faut dans la vie! Pour suivre assidûment les péripéties, les amours, les emmerdes de sa Cocotte depuis près de 10 ans, il y a Poulette (ou encore Nathalie) qui me soutient à coups de cocktails ou séances de Desperate Housewives avec tisane et chocolat, selon l'humeur. Pour remettre un peu de sport dans cette vie de dépravation, tout en papotant bien sûr, il y a Célinou (qui ne recule pas devant un petit cocktail non plus). Elle appartient aussi à la catégorie "copains thésards", ceux qui savent de quoi il retourne et qui peuvent mieux que quiconque apprécier les joies et les peines de cette situation : Maxime, Tim , Magali, Rémi et Matthieu. Cependant, il serait injuste de ne pas mentionner les autres, qui suivent mes progrès malgré tout et qui sont sont présents quand il le faut, tout simplement: Nadia et Laurent, les Despos, les Saint-Louisards, les Ensae, et enfin mes amis de lycée, collège et primaire que j'ai retrouvés tout récemment.

A présent, place à une catégorie toute particulière : ma famille. Ils ont eu d'autant plus de mérite de me soutenir qu'ils n'avaient la plupart du temps aucune idée de ce que je faisais, avouons-le! Je voudrais donc exprimer ma reconnaissance et ma tendresse à Mamie de Bretagne, à mes oncles et tantes : Patrick, Laurence et Thierry à mes cousins et cousinettes : Caroline, Emilie, Victor et Jérémie. Ensuite bien sûr, à mes soeurs, Manon et Oriane, qui elles aussi défendent vaillament les couleurs des Mordelet dans la jungle des études supérieures. Même si techniquement, ils ne sont pas de ma famille, ils le sont à mes yeux, merci à Alain et Laurence, à mon parrain adoré Denis

et sa femme Agnès chez qui je peux toujours recharger mes batteries. A ce stade, et sans pour autant verser dans Zola, il est temps d'adresser une pensée affectueuse à mon père, qui je l'espère aurait été fier de moi. Enfin, last but not least, Môman! Elle a toujours été là dans mes grands moments de doute, pour me donner confiance et y croire à ma place. Ca a commencé la veille de ma rentrée en CP: "J'y arriverai jamais Maman... C'est trop dur le CP!" et ça a continué ainsi jusqu'à la thèse, c'est dire si elle en a de la patience... Pour la toute fin, je vous ai réservé quelqu'un de très spécial : vous savez tout de lui sauf son vrai prénom! Je vais essayer de vous faire deviner. Il me supporte au quotidien et assure la logistique dans les moments les plus durs. Il est beau et fort, il a un super ordi et des gros pouces de pied. Il m'offre des fleurs, du rire et des sushis quand ça ne va pas, de l'affection, des consolations et bien plus encore. Vous ne savez toujours pas qui c'est? J'ai nommé ... Chouchou!

Abstract

In biology there are many problems for which traditional laboratory techniques are overwhelmed. Whether they are time consuming, expensive, error-prone or low throughput, they struggle to bring answers to these many questions that are left unanswered. In parallel, biotechnologies have evolved these past decades giving rise to mass production of biological data. High-throughput experiments now allow to characterize a cell at the genome-scale, raising great expectations as for the understanding of complex biological phenomenons. The combination of these two facts has induced a growing need for mathematicians and statisticians to enter the field of biology. Not only are bioinformaticians required to analyze efficiently the tons of data coming from high-throughput experiments in order to extract reliable information but their work also consists in building models for biological systems that result into useful predictions. Examples of problems for which a such expertise is needed encompass among others regulatory network inference and disease gene identification. Regulatory network inference is the elucidation of transcriptional regulation interactions between regulator genes called transcription factors and their gene targets. On the other hand, disease gene identification is the process of finding genes whose disruption triggers some genetically inherited disease. In both cases, since biologists are confronted with thousands of genes to investigate, the challenge is to output a prioritized list of interactions or genes believed to be good candidates for further experimental study. The two problems mentioned above share a common feature: they are both prioritization problems for which positive examples exists in small amounts (confirmed interactions or identified disease genes) but no negative data is available. Indeed, biological databases seldom report non-interactions and it is difficult not to say impossible to determine that a gene is not involved in the development process of a disease. On the other hand, there are plenty of so-called unlabeled examples like for instance genes for which we do not know whether they are interacting with a regulator gene or whether they are related to a disease. The problem of learning from positive and unlabeled examples, also called PU learning, has been studied in

itself in the field of machine learning. The subject of this thesis is the study of PU learning methods and their application to biological problems. In the first chapter we introduce the baggingSVM, a new algorithm for PU learning, and we assess its performance and properties on a benchmark dataset. The main idea of the algorithm is to exploit by means of a bagging-like procedure, an intrinsic feature of a PU learning problem, which is that the unlabeled set is contaminated with hidden positive examples. Our baggingSVM achieves comparable performance to the state-of-the-art method while showing good properties in terms of speed and scalability to the number of examples. The second chapter is dedicated to SIRENE, a new method for supervised inference of regulatory network. SIRENE is a conceptually simple algorithm which compares favorably to existing methods for network inference. Finally, the third chapter deals with the problem of disease gene identification. We propose ProDiGe, an algorithm for Prioritization Of Disease Genes with PU learning, which is derived from the baggingSVM. The algorithm is tailored for genome-wide gene search and allows to integrate several data sources. We illustrate its ability to correctly retrieve human disease genes on a real dataset.

Résumé

En biologie, il est fréquent que les techniques de laboratoire traditionnelles soient inadaptées à la complexité du problème traité. Une raison possible à celà est que leur mise en oeuvre requiert souvent beaucoup de temps et/ou de moyens financiers. Par ailleurs, certaines d'entre elles produisent des résultats peu fiables ou à trop faible débit. C'est pourquoi ces techniques peinent parfois à apporter des réponses aux nombreuses questions biologiques non résolues. En parallèle, l'évolution des biotechnologies a permis de produire massivement des données biologiques. Les expériences biologiques à haut débit permettent à présent de caractériser des cellules à l'échelle du génome et sont porteuses d'espoir pour la compréhension de phénomènes biologiques complexes. Ces deux faits combinés ont induit un besoin croissant de mathématiciens et de statisticiens en biologie. La tâche des bioinformaticiens est non seulement d'analyzer efficacement les masses de données produites par les expériences à haut débit et d'en extraire une information fiable mais aussi d'élaborer des modèles de systèmes biologiques menant à des prédictions utiles. L'inférence de réseaux de régulation et la recherche de gènes de maladie sont deux exemples parmi d'autres, de problèmes où une expertise bioinformatique peut s'avérer nécessaire. L'inférence de réseaux de régulation consiste à identifier les relations de régulation transcriptionnelle entre des gènes régulateurs appelés facteurs de transcription et des gènes cibles. Par ailleurs, la recherche de gènes de maladie consiste à déterminer les gènes dont les mutations mènent au développement d'une maladie génétiquement transmise. Dans les deux cas, les biologistes sont confrontés à des listes de milliers de gènes à tester. Le défi du bioinformaticien est donc de produire une liste de priorité où les interactions ou gènes candidats sont rangés par ordre de pertinence au problème traité, en vue d'une validation expérimentale. Les deux problèmes mentionés plus haut partagent une caractéristique commune: ce sont tous les deux des problèmes de priorisation pour lesquels un petit nombre d'exemples positifs est disponible (des interactions connues ou gènes de maladie déjà identifiés) mais pour lesquels on ne dispose pas de données négatives. En effet, les bases de données biologiques ne reportent

que rarement les paires de gènes non interactives. De même, il est difficile voire impossible de déterminer à coup sûr qu'un gène n'est pas impliqué dans le développement d'une maladie. Par ailleurs, des nombreux exemples indéterminés existent qui sont par exemple des gènes dont on ne sait pas si ils interagissent avec un facteur de transcription ou encore des gènes dont on ne sait pas s'ils sont causaux pour une maladie. Le problème de l'apprentissage à partir d'exemples positifs et indéterminés (PU learning en anglais) a été étudié en soi dans le domaine de l'apprentissage automatique (machine learning). L'objet de cette thèse est l'étude de méthodes de PU learning et leur application à des problèmes biologiques. Le premier chapitre présente le baggingSVM, un nouvel algorithme de PU learning et évalue ses performances et propriétés sur un jeu de données standard. L'idée principale de cet algorithme est d'exploiter au moyen d'une procédure voisine du bagging, une caractéristique intrinsèque d'un problème de PU learning qui est que l'ensemble des exemples indéterminés contient des positifs cachés. Le baggingSVM atteint des performances comparables à l'état de l'art tout en faisant preuve de bonnes propriétés en termes de rapidité et d'échelle par rapport au nombre d'exemples. Le deuxième chapitre est consacré à SIRENE, une nouvelle méthode supervisée pour l'inférence de réseaux de régulation. SIRENE est un algorithme conceptuellement simple qui donne de bons résultats en comparaison à des méthodes existantes pour l'inférence de réseaux. Enfin, le troisième chapitre décrit ProDiGe, un algorithme pour la priorisation de gènes de maladie à partir d'exemples positifs et indéterminés. Cet algorithme, issu du baggingSVM, peut gérer la recherche de gènes de maladies à l'échelle du génome et permet d'intégrer plusieurs sources de données. Sa capacité à retrouver correctement des gènes de maladie a été démontrée sur un jeu de données réel.

Chapter 1

Context

In this chapter, we expose the basic notions which are necessary to understand the developments of this thesis. Section 1.1 is dedicated to the biological aspects of gene regulation and disease gene hunting which are two topics of interest of our work. Then, section 1.2 focuses on the machine learning aspects of this work.

1.1 Biological context

1.1.1 Gene regulation

1.1.1.1 General mechanisms

This section aims to give an overview of the gene regulation process within a living cell. The genome of an individual is defined as the hereditary information encoded in its genetic material. Therefore, it contains all information needed by the organism to develop and live. The physical support of the genome is a molecule called DesoxyriboNucleic Acid (DNA). The general aspect and localisation of DNA is illustrated on figure 1.1.

A single DNA strand is a sequence of nucleotides, which are composed of a phosphate group, a sugar and a base. There are four types of bases which are thymine (T), adenine (A), guanine (G) and cytosine (C). Then, a double DNA strand is made of two single DNA strands which form complementary bonds in the following way : a T base is always complementary with an A and a G base is complementary with a C. This double strand of complementary base pairs has the shape of an helix. It is coiled around proteins called histones and then super-coiled to form a chromosome, which is located in the nucleus of the cell. In humans, there are 23 pairs of chromosomes.

All cells in an organism contain the totality of the genome, that is to say,

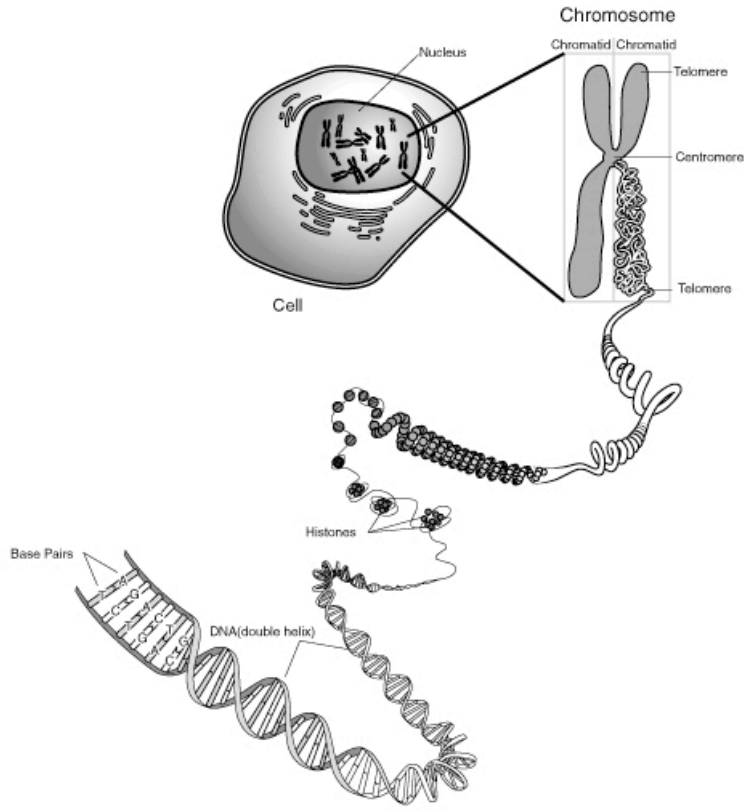


Figure 1.1: General aspect and localisation of DNA in a eukaryotic cell

the whole DNA sequence. The central dogma of biology (cf figure 1.2) states that coding genes on DNA are transcribed into messenger RNA (mRNA) that transport themselves outside the nucleus, to be translated into proteins. Each protein has a specific function to achieve inside the cell. Some have cellular functions: they might serve for cell motility, for maintaining the cell structure, for communication with the outside, for transporting molecules... Some proteins, called enzymes have a biochemical function, meaning that they serve as catalyzers to chemical reactions. In a word, proteins are the main actors of a cell's activity.

However, the cells of an organism are not all the same. Their activity, and therefore the type and quantity of proteins they produce, varies according to their location and to the environmental conditions they are facing. For instance, a liver cell and a blood cell do not behave identically. Likewise,

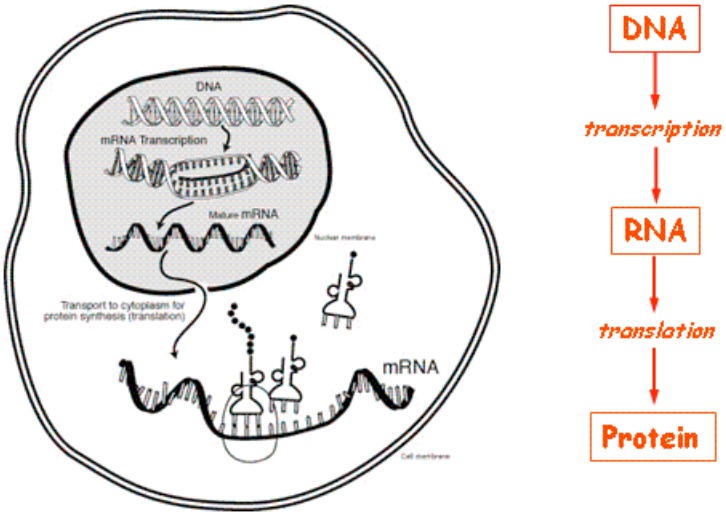


Figure 1.2: The central dogma of biology

variations in the environment like a change of temperature or the presence of some chemical compounds, do result in an adaptive behaviour of a cell. As a consequence, a cell does not express all coding parts of the genome, in the sense that there are genes that do no result into functional proteins into that cell. This also means that a cell is able to modulate the production of its proteins to satisfy its needs. This process is called gene regulation. There are several levels of gene expression regulation:

- DNA level regulation
  In order to allow for transcription, the structure of DNA must be modified. Some enzymes are responsible for DNA methylation or histone modifications that strengthen DNA packing, thus preventing the transcription machinery from accessing the sequence to be transcribed.

- Transcriptional regulation
  Among the proteins produced by a cell, some are called transcription factors (TF) and their function is precisely to regulate gene expression. A single TF has a certain number of gene targets. RNA polymerase is an enzyme complex responsible for transcription initiation. It binds to particular DNA sequences named promoters. By recognizing a target's binding site, a transcription factor increases (or decreases) the affinity of RNA polymerase for that gene's promoter, thus allowing (or disabling) its transcription into mRNA. In the first case, the TF is an activator,

whereas it is called an inhibitor in the second case. Figure 1.3 gives an example case of the activating regulation of gene A on gene B. In condition 1, gene A is transcripted and translated into an activator TF protein, which binds to the operator (O) of gene B, allowing RNA polymerase to bind the promoter (P), which initiates transcription of gene B into mRNA, which is then translated into a triangular protein. Then, under a second condition, which could be an external stimulus or that it has been produced in a sufficient quantity, the triangular protein is not needed anymore. The TF becomes inactive (see the change of shape), it cannot bind anymore to the operator, RNA polymerase is not recruited and transcription does not happen.

- Post-transcriptional regulation
  Being transcribed is a necessary but not sufficient step for being translated into a functional protein. Between transcription and translation, the transcripts undergo various processing mechanisms. They are stabilized (capping and polyadenylating) and spliced. Splicing is a mechanism that removes introns (non-coding regions). But, they can also be sequestated and degradated. At this stage, gene silencing and degradation are processes which participate to a regulation system named RNA interference whose main actors are micro RNAs (miRNA) and small interfering RNAs (siRNA).

- Translational regulation
  At last, translation itself can be disabled mainly at the beginning of the translation by the ribosome.

In this thesis, we focus on the particular case of transcriptional regulation. In particular, we neglect the last two types of regulation and assume that measuring mRNA quantities is sufficient to measure gene expression.

1.1.1.2 Why study gene regulation?

Elucidating the regulatory structure of a cell is a long-studied issue to which many efforts have been devoted. If we were able to understand a cell's functional mechanisms, we could predict its behaviour under different stimuli, like for instance the response to a drug. Another practical application in biomedicine is the search for new therapeutic targets. Trying to cure some disease, one often wishes to be able to inhibit or enhance some particular functional pathway. For instance, one could imagine a novel cancer therapy which would stop cancerous cells from proliferating while not being too toxic for surrounding tissues. As another example, Plasmodium falciparum

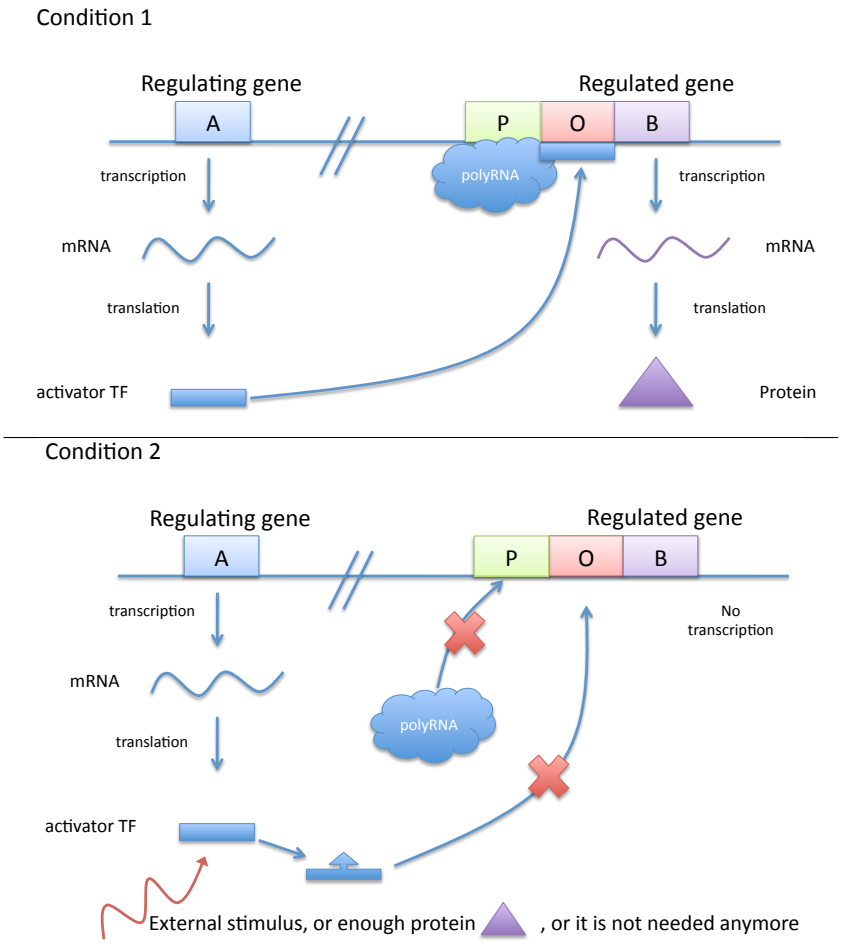


Figure 1.3: Example case of an activating regulation interaction.

is a parasite responsible for malaria. Some strains of this parasite are now resistant to known treatments. Unraveling the defense mechanisms of this parasite could allow to disable them and to design a new efficient treatment.

1.1.1.3 Experimental characterization

Low-throughput assays Historically, the first attempts to discover transcriptional regulatory interactions were carried out by experimental biologists. A regulatory interaction is, as we have described above, a binding reaction between a protein (the transcription factor) and a DNA fragment (the target gene's binding site). We review here the main experimental techniques used for assessing protein-DNA interactions.

- Gel shift analysis
  A DNA fragment, marked with a fluorescent marker is put on a gel, which has previously been impregnated with a protein. Since binding slows down the fragment, the difference in speed at which the DNA fragment moves along the gel with and without a protein, indicates whether the protein and DNA bind.

- DNase footprinting
  This technique exploits the fact that a protein binding to a DNA fragment tends to protect this fragment from cleavage. A DNA fragment of interest is amplified and labeled at one end. The obtained fragments are separated in two portions. The first portion is incubated with the protein of interest and DNase, a DNA cleaving enzyme. The second portion is incubated with DNase but without the protein of interest. The amount of DNase is calibrated so that each fragment is cut once. Both samples are then separated by gel electrophoresis [1] and the cleavage pattern in the presence of the protein is compared to the pattern in the absence of the protein.

- Nitrocellulose filter binding assay
  This assay aims to estimate the strength of interaction between a protein ligand and a DNA fragment by measuring the binding constant as a function of the protein quantity. Nitrocellusose paper, used as a filter, is ideal for immobilizing proteins, whereas DNA will not stick to it unless being bound to a protein. Radioactively labeled DNA is incubated with a known amount of protein and the mixture is spilled

[1] A technique for separating molecules placed on a gel matrix, by subjecting them to the action of an electric field.

on a nitrocellulose filter, placed above a vacuum which pumps the liquid downward. This way, all DNA which is not bound to a protein is removed. Then, radioactivity is measured, reflecting the quantity of bound DNA. This is repeated for various amounts of protein, so that we obtain a function plot of the amount of bound DNA against the amount of protein. A rapidly increasing curve shows a high affinity (cf figure 1.4).

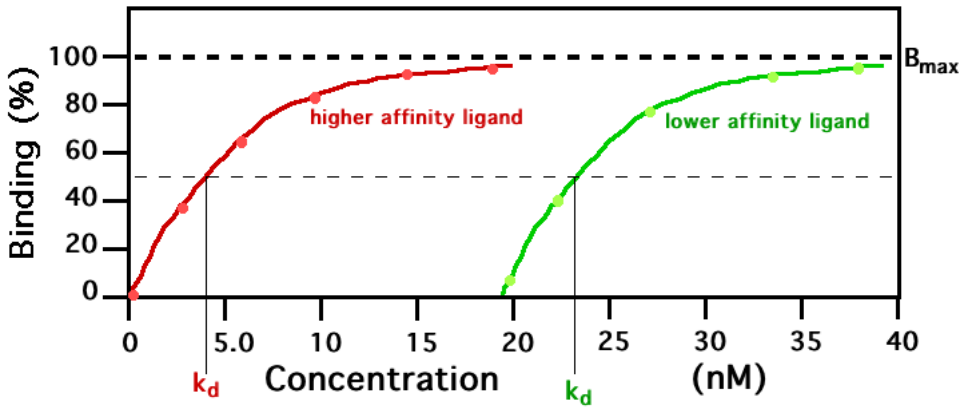


Figure 1.4: Results of a filter binding assay.

High throughput technologies A major drawback of the methods presented so far is their cost in terms of time and money. Indeed, one has to replicate manually the experiment for each pair of protein-DNA fragment one wants to test. That is why high-throughput experimental assays have been designed, which allow to test for many pairs at the same time. Here we focus on the ChIP-on-chip technology, also know as genome-wide localization analysis. The ChIP-on-chip technique, which stands for Chromatin ImmunoPrecipitation on chip, is a widely used technique for testing DNA-protein interactions. More specifically, such an experiment aims to identify the binding sites of a DNA-binding protein, such as a transcription factor. Figure 1.5 illustrates the experiment which we roughly describe. The protein is linked to the DNA in an *in vivo* environment. Then, DNA is chunked and the protein is targeted with a specific antibody, which allows to retrieve complexes made of a protein and a DNA fragment bound to it (a step called chromatin immuno-precipitation). These fragments are separated from the protein of interest and they are spotted on a DNA microarray (a chip)[2] in

[2]The microarray technology is explained later.

order to identify them. The ChIP-Seq technology is the same, except that the identification of the fragments is made by sequencing and mapping them on a reference genome. It is more sensitive than ChIP-on-chip and less prone to saturation effects of the signal.

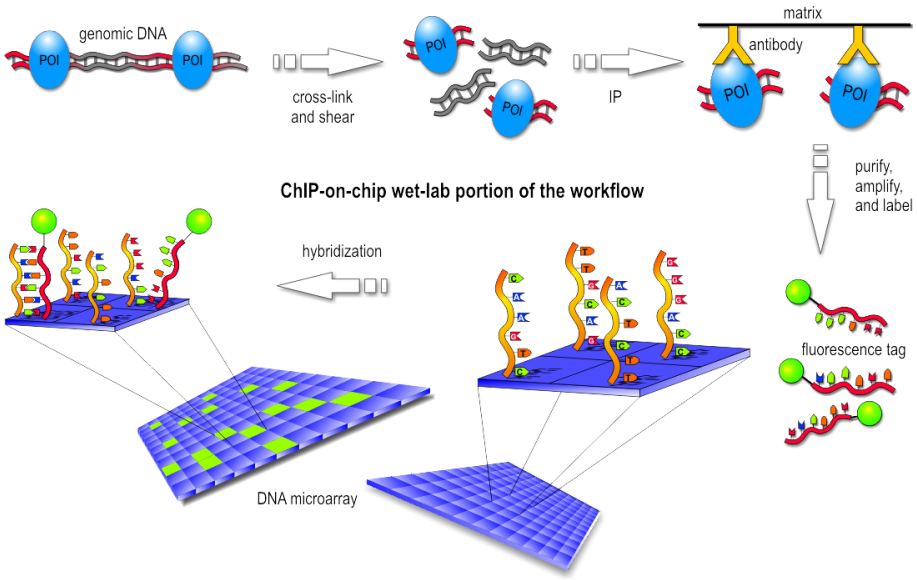


Figure 1.5: The ChIP-on-chip experiment.

1.1.1.4 *In silico* inference

Switching to the systems biology paradigm Even though high throughput assays have alleviated the experimental burden, a ChIP-on-chip experiment is still restricted to a single TF protein. Moreover, results still lack reliability. For instance, there are interactions which remain difficult to characterize by a ChIP-on-chip assay because it is not possible to design an antibody to recognize the protein of interest, or because an antibody exists but is not specific enough to this protein. In a general perspective, the great complexity of cellular mechanisms, generated by the numerous interactions existing in the cell between genes or proteins, makes it a grueling (and expensive) work to elucidate biological networks *in situ* i.e through experimental means. For those reasons, whether they are reliability or cost issues, *in silico* (computational) methods are increasingly appealed to for the inference of regulatory interactions.

The problem of regulatory interaction inference as we have depicted it, typically fits into the scheme of systems biology. We have already pointed out that cells could have very complex behaviors. Classical biology studies these behaviors by dissecting the cell and zooming on a few of its components. In opposition to that point of view, systems biology claims that one cannot understand the way a cell functions unless one sees it as a system, where any biological function is the result of thousands of components interacting, rather than the product of the activity of some isolated components. The two main problems that have caught the attention of systems biologists are network inference and model validation. In the context of transcriptional regulation, network inference consists in the elucidation of regulatory interactions between the components of the cell. This task is often referred to as "reverse engineering" in literature. The model validation part deals with the search for some general design principles underlying the regulatory structures. Fitting a model to an inferred network structure, we are allowed to make predictions which in turn, help to gain insight on the mechanistic of the cell . A common way to proceed is to reduce the complex machinery to a more simple model, whose parameters and/or shape allow one to test some hypotheses. Then, the model is usually validated by carrying out simulations. While reverse engineering operates at a very global scale, model validation often necessitates to focus on smaller networks, first because fitting a model on a too large network is difficult and also for interpretation purposes (some particular pathways of interest can be studied more precisely).

It is worth noting at this stage that the goal of systems biology is not to discard experimental biology. Both approaches are meant to complement each other in an iterative scheme. Whenever a computational algorithm is trained to infer an interaction network, high confidence interactions need to be experimentally validated. Likewise, whenever a model is found to be plausible, it needs to be checked by experimental means. In any case, the objective of computational methods is to guide experimentalists who would otherwise have to move blind. In turn, it is important that systems biologists can rely on the biologists' experience and knowledge, to build meaningful and relevant models. A crucial remark we can make about network inference is that, in this iterative scheme, it seems important to produce some confidence score for each interaction, so that only the most likely TF/gene pairs are submitted to experimental validation. In other words, it seems that the job expected from computational methods is above all a prioritization job.

Learning the edges between the nodes Under the systems biology vision, it is convenient to view the regulatory machinery as a graph. Naturally, the nodes of the graph are be genes, proteins or any acting component one wants to study and an edge represents an interaction between two of these components. Figure 1.6 gives a representation example of a regulatory network. In this particular example, nodes are heterogeneous and edges are directed. Red nodes are genes coding for transcription factor proteins (which we can directly call transcription factor as well) and blue nodes are other non-TF coding genes. A directed edge can only be drawn from a red node, meaning that the corresponding TF regulates the expression of the corresponding target gene.

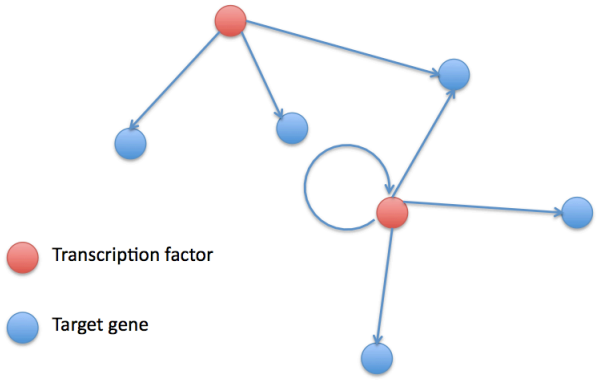


Figure 1.6: A graph representation of a regulatory network.

Having represented the transcriptional machinery as a graph, the task of is equivalently stated as "learning the edges between the nodes of the graph". A common way to proceed is to start from the output of a network, namely the expression levels of genes in order to get back to what has generated these expression patterns, that is the original structure of the network. DNA microarrays are devices that provide expression level measures (see section 4.5). A gene is featured by an expression profile that describes the activity of that gene under various conditions (steady state data) or at various time points (time-series data). We now give a brief review of methods which infer regulatory networks *de novo* from gene expression data. Mainly, these methods differ in the interpretation of the graph representation. We detail below the mathematical formalism they use, their specificities and the assumptions they make.

- Clustering methods [Eisen et al., 1998, Amato et al., 2006]
  Genes are clustered in groups of genes having similar expression profile,

indicating co-expression. Indeed, co-expressed genes are likely to participate to the same pathways and therefore, to be functionally related. Therefore, an edge between two genes on the graph is not directed and cannot be interpreted as a regulatory interaction but rather as a functional relationship. Steady-state as well as time-series data can be used.

- Correlation-based and information-theoretic methods [Butte and Kohane, 2000, Margolin et al., 2006, Faith et al., 2007]
  The rationale behind these approaches is that an interaction between a TF-coding gene and one of its targets can be detected through the dependence relationship that is induced between their expression levels. Information-theoretic methods use the mutual information measure, which can be seen as a measure of departure from independence. Contrary to correlation-based methods, information-theoretic methods are able to detect more complex forms of dependence than just linear dependence. An undirected edge between two nodes on the graph stands for a (linear) dependency between the expression of the two genes. If correlation is used, one can distinguish between activating and inhibiting regulation, according to the sign of correlation. However, causality cannot be determined through correlation only and therefore, an edge might just represent an indirect regulation. To avoid this, the concept of partial correlation has been proposed, which computes correlation between two variables conditional to all other variables [Rice et al., 2005, Liang and Wang, 2008].

- Boolean network inference [Akutsu et al., 2000]
  In that particular setting, the expression level of a gene i is encoded as a Boolean variable X_i: $X_i = 1$ means "expressed" as opposed to a value of 0. To do so, expression profiles are discretized into binary vectors. The value of the Boolean variable X_i depends on the values of the nodes pointing to it through a logical function. We note that this type of inference produces qualitative rather than quantitative relationships. The logical function for one node can vary over time or not, depending on which type of data is used.

- Bayesian network inference Friedman et al. [2000], Beal et al. [2005], Yu et al. [2004b]
  A graphical model represents conditional independence relationships of a set of random variables by means of a graph. A Bayesian network is a particular type of graphical model where the graph is a directed acyclic graph. Figure 1.7 gives an example of such a graph. Each

variable is represented by a node and each node is independent to all other nodes conditionally to its parents. For instance, figure 1.7 shows that conditionally to variables 5 and 7, variable 11 is independent from variables 2, 3, 8 , 9 and 10. Given the conditional dependency graph

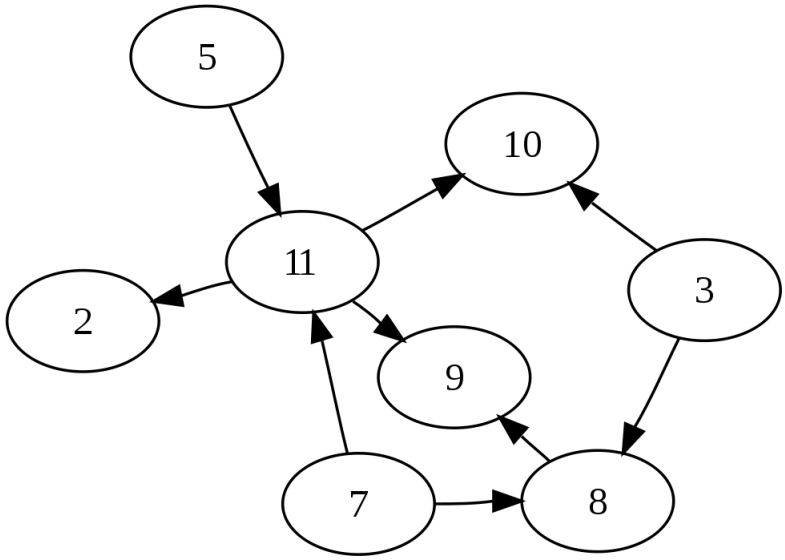


Figure 1.7: A directed acyclic graph.

G, data D and a model for the conditional likelihood (gaussian for instance), one can compute the posterior probability of that graph given the data D and maximize it with respect to G. In practice, since this optimization is an NP-hard problem, Bayesian inference methods resort to heuristic searches of the optimal graph. Bayesian networks have been extended to dynamic bayesian networks which additionally take into account the evolution over time of the regulatory network.

- Ordinary differential equations (ODE) methods [Gardner et al., 2003, di Bernardo et al., 2005, Bansal et al., 2006]
  While previous methods consider mainly statistical dependencies between the nodes, ODE-based methods are deterministic by nature and they focus on causal relationships. Following the principles of chemical kinetics, the expression variation of each gene is related to the level expression of all other genes and possibly to an external signal via a differential equation. The parameters of that equation are estimated by fitting the data to the model. These parameters determine the nature of the edges. A basic example is when the variation of the expression

of a gene x_j is chosen to be linearly related to the expression level of other genes $\{x_i\}_{i \neq j}$

$$\dot{x_j} = \sum_{i \neq j} a_i * x_i$$

Coefficients a_i are computed to fit the data and they are interpreted as weights on edges leaving from x_j to the $x_i's$. In essence, ODE-based methods are well suited for the analysis of time-series data, but they can also be used with steady-state expression profiles, in which case $\dot{x_j} = 0 \; \forall j$.

In silico methods are meant to investigate a very wide search space. To handle the level of complexity one is faced with when adopting a systemic point of view, one has to resort to simplifying assumptions. Linearity or discrete node value are examples of simplifying assumptions. We see that the approaches we have presented above adopt different formalisms corresponding to different assumptions on the data and on the way interactions might be captured. Some take into account the nature of the interactions (is the TF an activator or an inhibitor) while others do not, some do consider their dynamics (different interactions may be active at different time points). At last, some, like ODE-based methods, are able to infer synergistic effects while others are not. Besides, the complexity of the networks to be learnt might raise underdetermination issues, meaning that the available data might be insufficient to determine a single solution. To overcome this issue, one can also encode prior beliefs in the algorithm in order to restrict the search space and to guide the inference process. These priors beliefs often come as evolutionary constraints [Marbach et al., 2009] and are used to guide the inference towards biologically plausible networks. For instance, many computational methods enforce some sparsity constraints on the network structure and claim that gene networks tend to be parsimonious for the robustness property this confers them [Leclerc, 2008]. A possible way to proceed is to restrict the in-degree of the nodes [Akutsu et al., 2000]. Indeed, it is a widely accepted fact that each gene is regulated by a small set of regulators. This is related to the scale-free property which states that the networks we want to infer contain a few densely connected nodes called "hubs", while the rest of the nodes have very few connections. Moreover, studying the topology of some known regulatory networks, it has been noticed that some particular motifs would appear more frequently than what would be expected by chance. Again, this feature can be used to enforce the algorithm to output realistic networks.

1.1.2 The disease gene hunting problem

In this section, we introduce the problem of disease gene hunting. Then, we briefly review the traditional approaches that were used to identify disease genes.

1.1.2.1 Disease gene discovery

Having discovered the path which leads from genes to proteins and therefore to a corresponding phenotype, the task of finding which genes are responsible for the appearance of a given phenotype has attracted great attention in the genetic scientists community. In particular, a burning topic during the past 20 years has been to identify those genes whose disruption lead to acquired Mendelian diseases. Mendelian diseases are genetically inherited diseases caused by a mutation in a single gene. They are sometimes called monogenic diseases. For instance, cystic fibrosis (or mucoviscidosis) is a Mendelian disease caused by a mutation in the gene for the protein cystic fibrosis transmembrane conductance regulator (CFTR). The Online Mendelian Inheritance in Men (OMIM) database [McKusick, 2007] was initially created to gather knowledge on Mendelian disease/gene associations. However, efforts have been progressively shifting to the harder task of finding genes associated with polygenic or complex diseases. Indeed, the disruption of a single gene is sometimes not enough to trigger a disease phenotype, and it is thought that instead, the disease is caused by the simultaneous action of several genes and possibly by environmental factors. Yet, the mode of action of this set of genes is unknown and if environmental factors are playing a role in the development of the disease, it is uneasy to distinguish the environmental basis from the genetic causes. Alzheimer's disease is an example of a complex disease. The main reason why researchers concentrate their efforts on the disease gene hunting is that many diseases remain misunderstood. The identification of novel causal genes would give additional insight on the mechanisms which are at the origin of these poorly known diseases and above all, could open the way for the design of new therapies to cure them.

1.1.2.2 Experimental approaches for disease gene identification

Traditional strategies for finding disease genes are mixed strategies involving both molecular biology and genetic (statistical) techniques. There are mainly three ways to proceed.

Candidate gene approach This approach requires prior knowledge of the diseases, such as the type of function that is perturbed by the disease or the

tissues that are affected [Kwon and Goate, 2000]. Candidate genes are prioritized using his knowledge and then tested in association studies. These studies gather a set of individuals, some of whom are affected by the disease, some of whom are healthy people. The candidate gene is statistically tested for segregating polymorphisms among this population of individuals. Namely, if a given allele of that gene is significantly inherited more frequently in subjects having the disease than in healthy subjects, it is acknowledged as a disease gene. We can make a distinction here between population-based studies and family-based studies. Population-based studies allow the inclusion of any subject in the population and is similar to a classical epidemiology study. Genotypes are looked for that are more present in the case population than in the control population of healthy subjects. The inclusion plan of such a study must be made carefully, since the genotype frequency also varies between subjects of different geographical and/or ethnic origin. Otherwise, the success of the study might be hampered by what is called a stratification confounding effect. On the other hand, in family-based studies, healthy parents are used as controls for their affected offspring. They avoid the confounding stratification effect but since members from the same family live in the same environmental conditions, the role played by the candidate gene is not clearly separated from that of environmental hidden factors. A general drawback of the candidate gene approach is that it might fail if the disease pathophysiology is currently misunderstood or our knowledge of it is incomplete, which is often the case.

Genome-wise Association Studies Single Nucleotide Polymorphisms are variations on a single base pair of the genome between individuals of the same species. They represent around 90% of human genetic variations. SNP arrays are a special type of DNA microarrays (see section 4.5) which are designed to detect such polymorphisms on the whole genome for a given population. They have marked the advent of genome-wise association studies(GWAS). GWAS are association studies such as described in the previous paragraph except they examine systematically all SNPs across the genome. They are different from candidate gene studies because they do not use any knowledge of the disease to focus on particular genes but rather proceed to extensive statistical tests for association between a phenotype and genomic variations.

Genomic screen approach Contrary to the previous approach, genomic screening is a systematic method which operates without any prior knowledge on the studied disease. It relies on the use of "genetic markers" and

"genetic linkage". Genetic markers are highly polymorphic[3] DNA sequences (possibly genes) whose location on the genome is known. They are evenly distributed on the genome and usually serve as identifiers of the genome (like kilometer markers on a road). Genomic linkage could be defined as an inheritance property. During meiosis, DNA segments are exchanged between the chromosome inherited from the father and that inherited from the mother. This process, called recombination, allows to create new trait combinations and participates to genetic diversity. The consequence of that crossing-over is that alleles that would have been inherited together otherwise, can now be split up in different daughter cells. Moreover, it has been observed that the chances of being split-up by a recombination event were higher for alleles further away from each other. Therefore, if a marker is observed to be frequently inherited along with the phenotype of interest, this indicates that the gene responsible for that phenotype is likely to be close to it. The genetic distance between two markers is defined as the recombination frequency, i.e the proportion of recombining meiosis. A recombination frequency of 1% corresponds to a genetic map unit and is called a centiMorgan (cM). The corresponding physical distance (in number of bases) varies according to the localization on the genome. There are statistical tests which allow to identify two markers delimiting a candidate region, likely to contain the responsible gene. Figure 1.8 is extracted from [Silver, 1995] and illustrates the whole process: in this example, the phenotype of interest is "green eyes". Two markers D3Ab34 and D3Ab29 are found to have a recombination frequency with the phenotype of 2 over 400, one marker, D3Xy12, had a frequency of 1 over 400 and D3Xy55 was concordant in all 400 cases (panel A). Therefore, the candidate region was delimited by markers D3Ab34 and D3Ab29 (panel B). Then, fine mapping of this region can then be achieved by a technique called positional cloning. This comprises a time consuming procedure called chromosome walking, which consists in moving forward from the first delimiting marker towards the second, getting each time closer to the gene of interest. The process is aided by hybridizing a series of overlapping clones from a library to the region. The ordered series of such clones is called a contig, which can be seen as a physical representation of the region. These clones are easily identifiable. At each step of the walk, specific tests (including co-segregation tests) can determine whether the current clone belongs to the gene that is looked for. In panel E of figure 1.8, we see the results of co-segregating tests for the different clones. 1R and 10R show one recombining event (resp. in individuals number 156 and 332), while 6L and 2R fragments are concordant

[3]A gene is said to be polymorphic if it comes into different shapes or "alleles" in a population, yielding different versions of the same phenotype.

with the phenotype. Therefore, localization of the "green eyes" locus is enhanced and restricted to a narrower region between 1R and 10R. Although the initial interval is narrowed down, the output of positional cloning is usually not a gene but rather an interval, whose size depends on the resolution limit (on the order of 1-10 cM). Candidate genes in that interval must then be examined successively. Note that the evolution of sequencing technologies now allows to save a lot of times. Nevertheless, both approaches necessitate recruiting a large number of families and collecting DNA samples from the members of these families, which is still a very long and heavy process.

Prioritization using *in silico* methods Considering the inherent difficulty of traditional techniques of disease gene identification, and the fact that researchers were often left with a remaining list of thousands of genes to investigate, there were a growing need for automated techniques. That is why this area of research has turned towards computer scientists and mathematicians to develop computational techniques. Similarly to the gene regulation case, what is expected from *in silico* methods in the case of disease gene discovery is rather a prioritization job than a classification job. That is to say the desired output of an algorithm is an ordered list of genes, where the top genes are believed to be the better disease gene candidates.

1.1.3 Data resources

The increasing need for automated methods is a general phenomenon in biology, that goes far beyond the two particular problems presented in this thesis. It is true that for many biological problems, experimental techniques are time consuming and sometimes very expensive but this fact by itself is insufficient to explain the growing success of computational biology nor why mathematicians, statisticians and computer scientists have got interested in the biological issues. This phenomenon takes a lot more sense if we place under the light of what can be termed a "data revolution". This revolution has been triggered by the completion of the Human Genome Project in 2003. The goal of this project was to determine the sequence of base pairs which constitute human DNA and to identify all human genes. This achievement was a considerable step towards understanding molecular mechanisms in general and was chosen to mark the advent of the post-genomic era. Since then, biotechnologies have evolved at a fast pace and advances in that field have generated an unprecedented variety of data, describing living cells at a very high precision level. Among them, high-throughput techniques have enabled researchers, for the first time in history, to characterize biological samples through a high number of quantitative measures at the genome scale. As a

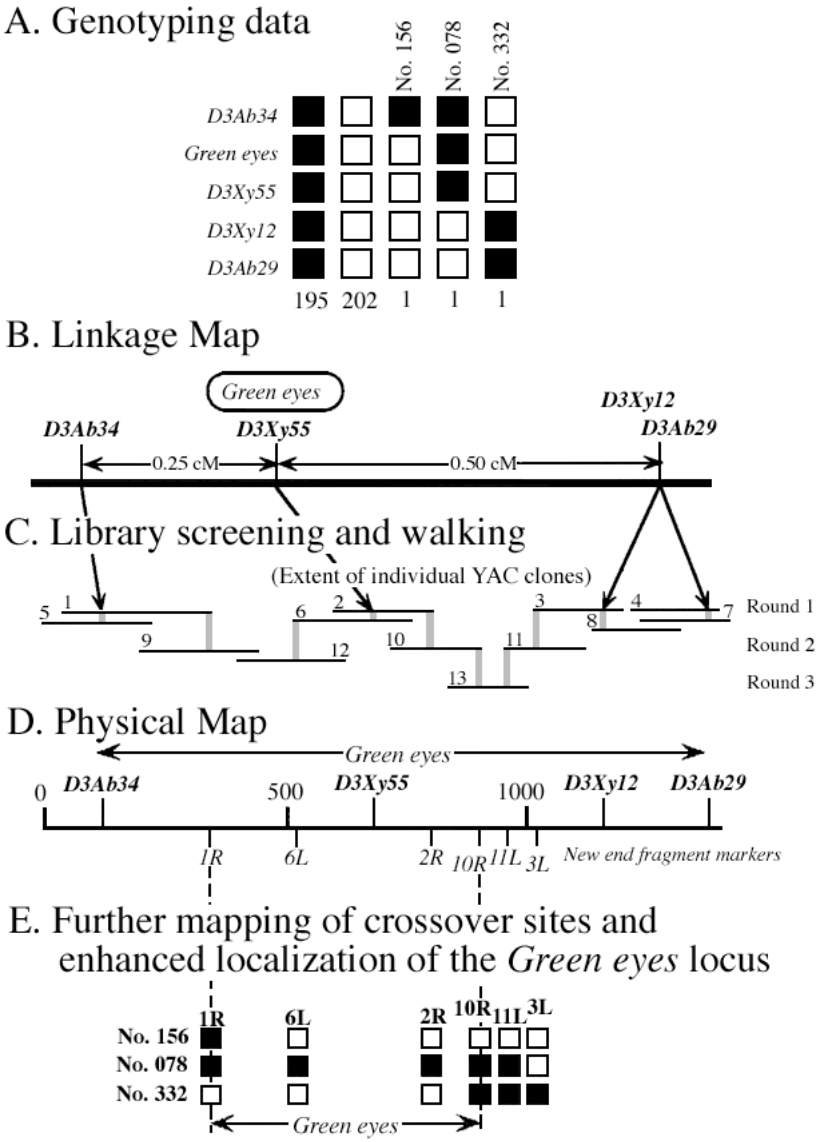


Figure 1.8: The different steps of positional cloning.

practical consequence, data analysis and mathematical/statistical modeling have been increasingly integrated at the core of biological research. Besides, the analysis of datasets having the size and complexity of those produced by high-throughput technologies raise challenging methodological issues that require appropriate statistical treatment.

In the two problems we have introduced earlier, the principal object to be described is a gene. Therefore, we now review the different types of data that are now available to characterize genes or equivalently the proteins they code for. Figure 1.10 illustrates further a few types of data that are available to describe a gene and gives a better idea of the kind of quantitative information they provide.

1.1.3.1 Transcriptomics data

The most well known type of high-throughput data to describe genes is probably transcriptomics data, also known as microarray expression data. Microarrays are devices for measuring the level of expression of all genes in a tissue in a particular condition or time point. The expression level of a gene is quantified via the abundance of messenger RNA after transcription of that gene. We mentioned in section 1.1.1 that this was a simplification, since measuring the quantity of proteins would be more sensible to quantify expression. The corresponding data are called proteomics data. However, proteomics techniques are far less mature than transcriptomics and researchers implicitely consider that transcriptome is correlated enough with proteome to provide a good approximation of the actual expression of a gene. For a broad explanation of the technical details, a microarray is a solid suppport, also called a "chip", with an arrayed series of spots, each of which contains a fixed amount of probes. A probe is a short sequence designed to match a specific mRNA. The target sample is fluorescently labeled and hybridized to the probes on the array. A labeled mRNA matching a probe generates a light signal whose intensity ideally depends on its quantity (see figure 1.9). The intensity signal is captured (see left picture on figure 1.10) and processed to yield a single quantitative measure for each gene. Affymetrix and Agilent are two companies whose microarray technology is widely spread in the genomic research community. Finally, a single microarray produces a snapshot of the activity of a tissue sample in a particular condition. Generally, this process is repeated in different conditions (steady-state data) or at different time points (time series data). Different conditions usually means either different samples (from normal individuals versus patients with a disease, or different tissues) or different environmental conditions (oxydative stress, drug, temperature

change, PH change...). Time series experiments may be designed to observe how expression evolves in time after a given event (for instance, knock out of a gene or subjecting the sample to a drug). In both cases, one eventually obtains an expression profile or each gene which is a vector of measures that characterizes the gene in question.

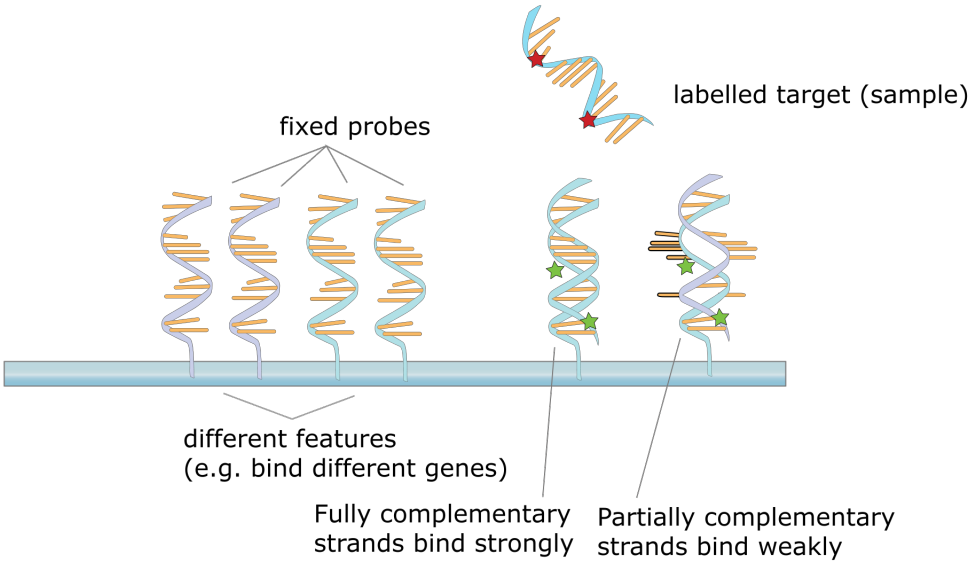


Figure 1.9: Target samples are hybridized to probes on a microarray.

A related type of expression data, which is perhaps less popular, is Expressed Sequence Tag (EST) data. EST are short sequenced fragments of complementary DNA (cDNA) obtained from mRNA bits sampled from a given tissue. These fragments are indicators of expression of a gene. Therefore, they can be used to identify yet unknown genes by physical mapping. But even when the corresponding gene is already known, the information about the tissues and the conditions in which the gene was found to expressed is still valuable and can be used to describe this gene. For instance, in the candidate approach for disease gene discovery we have been reviewing previously, if one was looking for a gene responsible for a disease affecting the brain, she would start by investigating those genes known to be expressed in this area of the body.

1.1.3.2 Subcellular localization data

An important feature for a gene is the localization within the cell of the protein it codes for. Knowing where a protein lives enables one to know more about its function, about it interacting partners. It can also be a valuable information for who would like to target this protein. Laboratory techniques encompass immuno-fluorescence, fluorescent microscopy, fractionation combined with mass spectometry.

1.1.3.3 Sequence data

The sequencing technologies are probably those that have known the most spectacular development since their invention in the late seventies (they were independently discovered by Sanger's team in the UK and Maxam and Gilbert's team in the US). Nowadays, it has become straightforward to characterize a gene by its nucleotide sequence. A lot of work has been dedicated to sequence analysis, sequence alignment and sequence comparison. Sequence analysis has allowed to identify particular structures, recurrent patterns or motifs associated with specific types of sequences. An important application of sequence alignment and comparison has been to find homologous proteins. Homologous proteins are coded by genes deriving from a common ancestor. Going a step further, one can distinguish between two types of homologous proteins:

- Paralogous proteins belong to a single species and result from a duplication event.

- After a speciation event, the two copies of the same gene in the two newly formed species are brought to evolve separately and therefore their sequence and possibly function diverge. They are called orthologous.

However, even after billions of years, homologous genes might have kept a certain degree of resemblance, both because their sequences are similar and because their function might be conserved. Homology is often detected by means of sequence similarity. Detection of homologous proteins is the ground of comparative genomics. or instance, when looking for the function of a gene (protein), a fruitful approach is to look for informations from similar proteins either in other better known species, via its orthologous proteins or possibly within the same species, via its paralogous proteins.

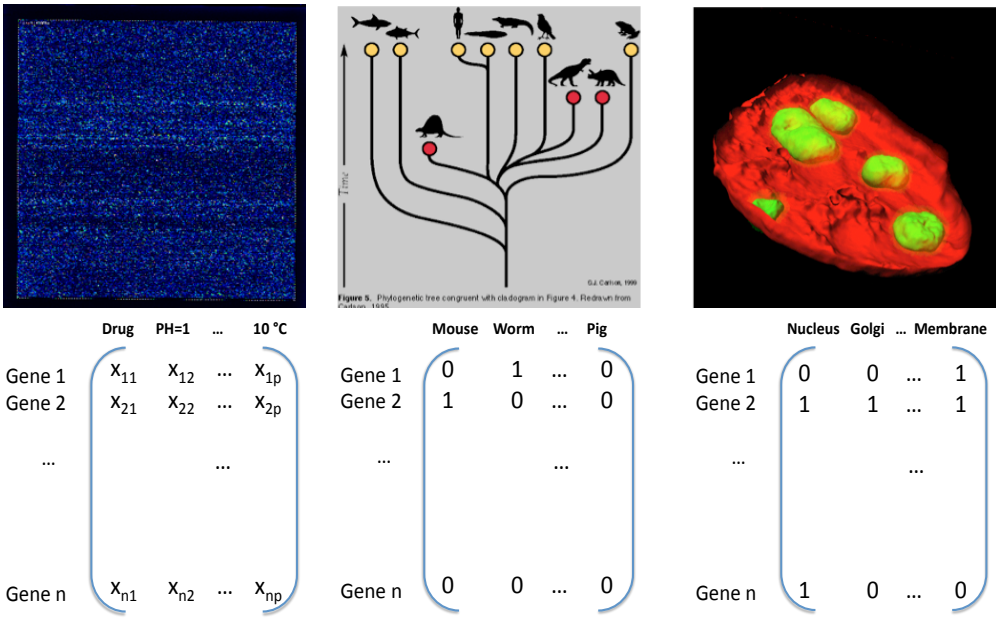


Figure 1.10: Different data representation for genes.

1.1.3.4 Annotation data

Annotation is a generic term that refers to the process of attaching biological knowledge to a gene or a protein. This biological knowledge covers many types of information. Here is a non-exhaustive list.

Phylogenetic profiling Phylogenetic data are typically produced by comparative genomics methods. For each gene A in a given species (let's say a human gene) one can look for orthologous genes in a list of other species (for instance, mouse, worm and pig). As mentioned above, this is done by measuring sequence similarity. BLAST (Basic Local Alignment Tool) is a popular tool to output such measures [Altschul et al., 1990]. By thresholding the BLAST score, one can get a binary vector of size 3 (the number of other species) with 1 indicating that your gene has an orthologous gene in the corresponding species. Comparing genes using such phylogenetic profiles makes sense since it is believed that genes conserved in the same species are likely to be functionally related.

Functional annotation Functional annotation is meant to describe what is known about the function of a gene. For instance, Gene Ontology (GO)

database [Ashburner et al., 2000] structures this information via a controlled vocabulary of terms, embedded into a hierarchical tree. For instance, the term "intracellular organelle" has for parent term "intracellular part" and for children "cytoplasmic vesicle". The list of GO terms associated with a gene describes that gene at the functional level. Besides molecular functions performed, these terms can describe biological processes (or pathways) in which the gene is involved and cellular components in which it is active. Similarly, a gene can be attached a list of pathways it participates to. Pathway membership can be found for instance in the Kyoto Encyclopedia of Genes and Genome (KEGG) database [Kanehisa et al., 2004]. KEGG works following the same type of organization. As an example, pathway hsa05218 is named "Melanoma-Homo sapiens" and belongs to the class of "Human diseases;cancers". Both types of data can presented as binary matrix (like localization or phylogenetic data, see figure 1.10), each column corresponding to a GO or KEGG term. Likewise, literature contains vast amounts of biological information, which be be automatically searched with text mining tools, like TXTGate [Glenisson et al., 2004].

Functional relatedness is even more directly reported in physical interaction data (Protein-protein interaction or PPI), which we therefore mention here, even though they are not exactly annotation data. PPI data can be found in numerous databases, such as Bind, Biogrid, Dip, Human Protein Reference Database (HPRD), IntNetDb, IntAct or String [Bader et al., 2003, Stark et al., 2006, Xenarios et al., 2002, Mishra et al., 2006, Xia et al., 2006, Aranda et al., 2010, Jensen et al., 2009].

Structural annotation When translated from a messenger RNA, a protein is folded so as to acquire a particular structure, this is illustrated in figure 1.11. As can be seen on this picture, the structure of a protein is really difficult to describe. At the level of the primary structure, the description concerns the sequence of amino-acids. Sequence analysis can also be performed directly at the gene level. As briefly mentioned, it deals with identifying motifs that are recurrent patterns in the chain of amino-acids (or nucleotides). Examining co-occurrence of identical motifs in two gene sequences is a possible way to compare them. At a higher level, there are parts of the protein that can evolve independently from the rest of the chain. They are called domains and they are widely used to annotate proteins. The presence of common domains in two proteins is a clue for functional relatedness. InterPro is an example of a database which might contain such information on a protein.

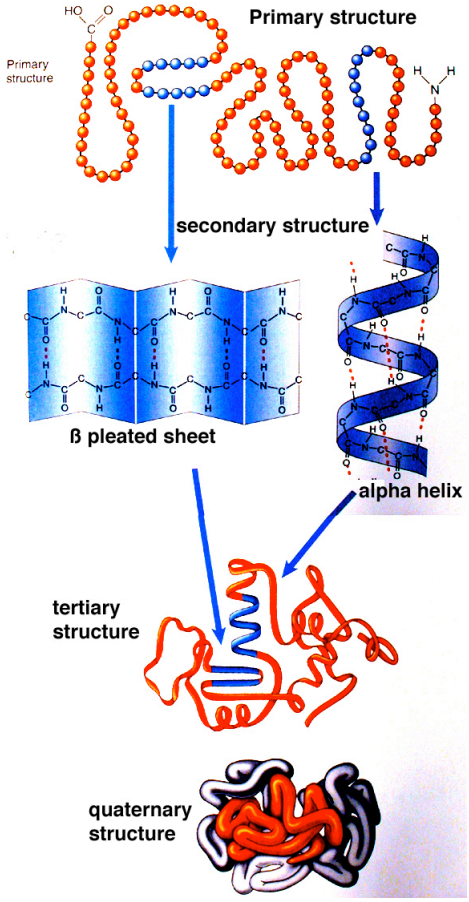


Figure 1.11: The several successive levels of structure acquired by a protein.

1.1.3.5 Data fusion

As a conclusion of this long and yet incomplete list of data types, there are many ways to characterize a gene. Each data representation has its own specificities and all of them carry very different information with respect to the problem at hand. As a result, it is highly probable that the results one can obtain from a single data type are incomplete but complementary to those obtained with different views. Besides, it is likely that the relationship between different data contain some information as well. To get the whole picture, a natural idea is therefore to incorporate many data sources in the inference process. This is currently referred to as data fusion. There are many ways to proceed. One that is quite straightforward is to combine the results obtained from different data sources *a posteriori*. For instance, in the context of regulatory network inference, you could learn a first set of edges $\{\mathcal{E}_{expr}\}$ using gene expression data, a second set $\{\mathcal{E}_{phylo}\}$ using phylogenetic data and take as a final result either the union or intersection of these two sets of edges. Intuitively, taking the intersection reduces the set of predicted edges, hence drastically decreasing sensitivity, all the more so as the sets have a small overlap. For that matter, it is a known issue that inference methods often lack robustness and that data perturbation results in weakly overlapping predictions. In that case, taking the union tends to hinder precision. Furthermore, this solution does not fit in a prioritization perspective, since it cannot handle properly ranked lists of predictions. A second option is to integrate the data sources *a priori*, i.e to find a way to combine them first and to give this as input to the learning process to finally output a ranked list of predictions. We subsequently indicate a way to do this in section 1.2.

1.2 Machine learning context

We have already stressed the need for automatized methods to tackle complex biological problems. Among scientific disciplines, machine learning is particularly well adapted to that framework. It can be defined as a subdomain of artificial intelligence which designs algorithmic techniques aimed at giving a machine the ability to learn. It deals with the automatic extraction of information from input data, aided by computational and statistical methods. Examples of topics that typically appeal to machine learning are natural language processing, computer vision, written digit recognition, robot locomotion or video games. In this thesis, we show that they can be particularly useful in biology. This section is dedicated to a brief introduction to the main machine learning tools we have used, that are kernel methods and

more specifically Support Vector Machines (SVM).

1.2.1 Learning from data

In a very general context, machine learning algorithms need to be fueled with data in order to produce an output. To formalize this, consider that data are actually a collection of objects or individuals, each described by p random variables or "features". We denote by $\mathbf{X} = (X_1, \ldots, X_p)$ these features. We assume that their distribution is unknown and that, however, we want to learn something from these features.

1.2.1.1 Unsupervised versus supervised learning

A major distinction can be made regarding the kind of information one wants to retrieve. If one wants to learn anything about the data distribution itself, he will perform "unsupervised learning". This includes density estimation or learning about the topographical organization of the distribution (via clustering or manifold inference, for instance). The learner will be aided by a training set which consists in observations of the p variables (actual measurements of these variables) for n objects or individuals:

$$\mathbf{x}_i \in \mathbb{R}^p, i \in (1, \ldots, n)$$

On the other hand, if the output is some other variable $\mathbf{Y} = (Y_1, \ldots, Y_n)$, we are dealing with "supervised" learning. In this case, the inference is guided and the goal is to guess the relationship between the input variable $\mathbf{X}$ and the output variable $\mathbf{Y}$, under the shape of a generic function which links them: $\mathbf{Y} = f(\mathbf{X})$. This time, as a training set, the learner is given a vector of values of $\mathbf{Y}$ for the n objects: $[y_i]_{i\in(1\ldots n)}$ in addition to the matrix of measurements $[\mathbf{x}_i]$. A typical strategy is to define a loss function that quantifies the cost of incorrectly predicting the output. Then, the algorithm's objective will be to minimize the expectation of that loss. Since this expectation is unknown, the theoretical objective criterion will be estimated by an empirical criterion. If we denote the loss function by L, the algorithm boils down to the following optimization problem

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} L\big(f(\mathbf{x}_i), y_i\big)\,. \tag{1.1}$$

1.2.1.2 The bias-variance trade-off

To start with, the function f is generally searched for in some determined Hilbert function space $\mathcal{H}$. A major challenge is to determine how complex

this function must be. A primary reflex might be to pick one as complex as possible in order to fit the training set. This typically corresponds to However, the objective is wider than just fitting the training set, it is to find a function that is true *in general*. If new observations of $\mathbf{X}$ were produced, the learner would then be able to predict correctly the output $\mathbf{Y}$ for these new objects. This is called the generalization property. Most of times, achieving both goals at the same time is not possible and this dilemma is usually referred to as the bias-variance trade-off.

A related difficulty lies in the amount of data available. In statistics, in general, the more observations one has, the better. In the biological context, a critical issue is that the number of observations n is frequently much smaller than the number of variables p, creating a gap between the number of observations required to infer a highly complex function and the actual amount of available observation points. Again, in this case, many functions can fit the training set perfectly but generalize poorly. This phenomenon, known as overfitting, can be walked around by a regularization process, which consists in practice to limit the search of function f to a restricted class of functions $\mathcal{F} \subset \mathcal{H}$. The larger the function class (or the higher its capacity), the more complex function f is allowed to be. The theoretical criterion to be optimized is called the risk and is defined as

$$R_L(f) = \mathbb{E}[L(f(\mathbf{X}), \mathbf{Y})]\,.$$

Then, the empirical version this risk is

$$R_L^n(f) = \frac{1}{n}\sum_{i=1}^{n} L\big(f(\mathbf{x}_i, y_i)\big)\,.$$

Let us assume that

$$\begin{aligned} f^* &= \operatorname*{argmin}_{f\in\mathcal{H}} R_L(f)\,, \\ f_{\mathcal{F}} &= \operatorname*{argmin}_{f\in\mathcal{F}} R_L(f)\,, \\ \text{and}\quad \hat{f}_n &= \operatorname*{argmin}_{f\in\mathcal{F}} R_L^n(f)\,. \end{aligned}$$

The difference between the optimal risk and the estimated empirical risk can be decomposed as

$$\mathbb{E}\big(R_L^n(\hat{f}_n) - R_L(f^*)\big) = \underbrace{\mathbb{E}\big(R_L^n(\hat{f}_n) - R_L(f_{\mathcal{F}})\big)}_{\text{Estimation error}} + \underbrace{R_L(f_{\mathcal{F}}) - R_L(f^*)}_{\text{Approximation error}}\,.$$

Let us consider a particular class of functions $\mathcal{F}_B = \{f \in \mathcal{H},\ \|f\|_{\mathcal{H}} \leq B\}$. A classical result states that the estimation error can be upper bounded by term which decreases with n the number of data examples and increases with B, the capacity of the function class $\mathcal{F}_B$. On the other hand, the approximation error decreases with B. To minimize the risk, one finally has to find the ideal value of B which minimizes the upper bound. If the function class is too large, the approximation error will be small but the number of points will be insufficient to estimate the over-complex function $f_{\mathcal{F}_B}$. On the other hand, if it is too small, there is a high chance that function $f_{\mathcal{F}_B}$, which we estimate, is actually too far from the ideal function f^*.

In practice, for the optimization problem, constraints are added to discard functions which are overly complex in the sense that they vary too much. Following the risk minimization perspective presented in the previous paragraph, this is equivalent to restricting the function space $\mathcal{H}$ and is generally done by penalizing the norm of the function in that space (a small norm or small B indicating low variations). The criterion in (1.1) gets an additional term and the optimization problem becomes:

$$\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}}\ \|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} L\big(f(\mathbf{x}_i), y_i\big)\,. \tag{1.2}$$

Note that parameter C orchestrates the bias-variance trade-off and that it usually needs to be optimized.

If the output variable $\mathbf{Y}$ is continuous, the problem of finding function f is called regression while it is called classification if the output is a discrete variable. In this case, the values of $\mathbf{Y}$ represent class labels. We have underlined the fact that our goal in network inference as well as disease gene discovery was prioritization rather than classification. However, any classification method that produces a score function f reflecting our confidence in the predictions is well suited for prioritization. Indeed, it is the case of most of them but this score is usually thresholded in order to obtain a classification rule. In section 1.2.2, we focus on a particular classification method called the Support Vector Machine (SVM).

1.2.1.3 Some variants of supervised learning

In this subsection, we present a few situations when the training information is reduced, that have led to specific developments in machine learning.

Semi-supervised learning A general problem in supervised learning is the lack of labeled samples (both positive and negative). On the other hand, it frequently happens that plenty of examples are available for which we do not know whether they belong to the positive or to the negative class, these are called "unlabeled" examples. A possible reason for this established fact is that labeled examples are difficult to obtain because it takes time, money or expertise to identify them, while unlabeled examples are easy to get. To overcome this problem, it has been suggested that unlabeled data could be useful. Indeed, it seems logical to think that some information is contained in the distribution of the huge amount of unlabeled examples which are available. Methods built on that premise are called "semi-supervised" methods. In this perspective, let us mention co-training, a method developped by Blum and Mitchell [1998]. This method makes up for the lack of labeled samples by using two distinct views of the data. Two learning algorithms are trained separately on each view and their predictions are used so that they mutually learn from each other. Other authors choose to consider unlabeled samples as missing data and have proposed the use of the Expectation-Maximization algorithm, see for instance Nigam et al. [2000]. At last, it is worth mentioning the work of Joachims [1999] on transductive SVM. Transductive learning is based on Vapnik's principle [Vapnik, 1998] : "When solving a problem of interest, do not solve a more general problem as an intermediate step." Thus, tranductive methods focus on the prediction of the labels of the unlabeled set only. This is all the more relevant if one is not interested in building a general rule but rather in predicting the label vector $\mathbf{y}^* = (y_1^*, \ldots, y_{N_U}^*)' \in \{-1, 1\}^{N_U}$ for a given unlabeled set with features $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_{N_U}^*\}$. Let the set of labeled examples be $\{(y_1, \mathbf{x}_1), \ldots, (y_{N_L}, \mathbf{x}_{N_L})\}$. The transductive problem minimizes a criterion in both function f and label vector $\mathbf{y}^*$:

$$\min_{f \in \mathcal{H},\ \mathbf{y}^* \in \{-1,1\}^{N_U}} \left\{ \|f\|_{\mathcal{H}}^2 + \sum_{i=1}^{N_L} L(y_i, f(\mathbf{x}_i)) + C \sum_{j=1}^{N_U} L(y_j^*, f(\mathbf{x}_j^*)) \right\}.$$

This expression can be transformed successively into :

$$\min_{f \in \mathcal{H}} \left\{ \min_{y^*} \left\{ \|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{N_L} L(y_i, f(\mathbf{x}_i)) + C \sum_{j=1}^{N_U} L(y_j^*, f(\mathbf{x}_j^*)) \right\} \right\},$$

$$\min_{f \in \mathcal{H}} \left\{ \|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{N_L} L(y_i, f(\mathbf{x}_i)) + C \min_{y^* \in \{-1,1\}^{N_U}} \left\{ \sum_{j=1}^{N_U} L(y_j^*, f(\mathbf{x}_j^*)) \right\} \right\},$$

$$\min_{f \in \mathcal{H}} \left\{ \|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{N_L} L(y_i, f(\mathbf{x}_i)) + C \sum_{j=1}^{N_U} \min_{y_j^* \in \{-1,1\}} \left\{ L(y_j^*, f(\mathbf{x}_j^*)) \right\} \right\}.$$

This shows that transductive learning can be seen as a functional optimization problem alike to equation (1.2) using a particular loss function for the unlabeled examples. Let's denote this loss function by L^* :

$$L^*(f(\mathbf{x}^*)) = \min\left(L(1, f(\mathbf{x}_i^*)), L(-1, f(\mathbf{x}_i^*))\right).$$

Then, we eventually come to

$$\min_{f\in\mathcal{H}}\left\{\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{N_L} L(y_i, f(\mathbf{x}_i)) + C\sum_{j=1}^{N_U} L^*(f(\mathbf{x}_j^*))\right\}. \tag{1.3}$$

For more details on these methods and many others, we refer the reader to Chapelle et al. [2006].

Learning from positive examples only We now discuss another particular type of supervised problem, in which only positive examples are available. Sometimes, negative labeled examples are particularly hard, not to say impossible, to obtain. A straightforward approach consists in using the positive set to determine some area in space $\mathcal{X}$ which would likely contain the positive class examples. This amounts to estimating the support of the positive distribution or its density level sets. One-class SVM is a popular method to achieve this [Schölkopf et al., 2001, Manevitz and Yousef, 2001, De Bie et al., 2007]. It can also be formulated as functional optimization problem with a regularization term:

$$\min_{f\in\mathcal{H}}\left\{\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{N_P}\max(0, f(\mathbf{x}_i))\right\}. \tag{1.4}$$

Nevertheless, these methods easily suffer from overfitting, are hardly applicable in high dimensional problems and therefore, require a large amount of examples to learn correctly. Since many datasets are characterized by a small positive set $\mathcal{P}$, these methods are often unsuitable.

Learning from positive and unlabeled examples Once more, in many practical cases and despite the absence of negative examples, a huge amount of unlabeled examples are available in addition to the positive set. Learning from positive and unlabeled examples is often referred to as "PU learning". A theoretical study of Probably Approximately Correct learning from positive and unlabeled examples has been provided by Denis [1998] showing that any functional class learnable under the statistical query model of Kearns

[1993] such as the k-DNF class, is learnable from positive and unlabeled examples. Inspired from that result, Letouzey et al. [2000] have proposed an algorithm using a modified decision tree algorithm based on the statistical query model. Following the same idea of estimating statistical queries over positive and unlabeled examples, Denis et al. [2002] have also noticeably worked on a Naive Bayes classifier. They use a probabilistic trick to estimate conditional negative probabilities from the positive examples. Nevertheless, an important drawback is that an estimate of the positive class probability is required. Later, they have successfully used co-training to enhance their classifier's performance in Denis et al. [2003]. PU learning is also a particular case of learning with class-conditional classification noise or CCCN learning [Decatur, 1997]. In this kind of learning problems, the true labels of the training examples are known up to some noise. Namely, the labels are flipped with some probability which quantifies the level of noise. This probability may depend on the class of the considered example. In PU learning particularly, if unlabeled examples are negatively labeled, the noise on negative labels (the probability of a negative label being flipped) is null while it is strictly positive for positive labels. In other words, positive examples are trustworthy while negative examples contain false negatives. Stempfel [2009] and Stempfel and Ralaivola [2009] have studied this problem thoroughly, and in particular they have proposed an algorithm to learn SVMs efficiently from sloppily labeled data. However, the value or an estimate of the classification noise is needed.

Since it is sometimes difficult to provide an estimate of the positive weight, another natural idea is to identify a set of reliable negative examples. This idea is implemented by the S-EM method for text classification in Liu et al. [2002], the PEBL method for Webpage classification in Yu et al. [2004a] and by others, reviewed in Liu et al. [2003]. This leads to iterative procedures where labels might be allowed to vary or not from step to step.

An obvious drawback of these methods is that they rely on the quality of the initially identified negative subset. Convergence is not ensured because a few errors might propagate very quickly. Therefore, other methods propose to directly discriminate the unlabeled examples against the positive ones [Liu et al., 2003, Lee and Liu, 2003, Elkan and Noto, 2008]. They are based on the theoretical justification provided in Liu et al. [2002] that minimizing the number of unlabeled examples classified as positives while keeping positive labeled examples classified correctly gives functions with small expected error if the number of positive and unlabeled examples are large enough. To account for noise, errors on the positive labeled examples might be allowed but to a smaller extent than errors on the unlabeled set. The latter idea can

be formulated as :

$$\min_f \left\{ \|f\|^2_{\mathcal{H}} + w_P * C \sum_{i=1}^{N_P} L(1, f(\mathbf{x}_i)) + w_U * C \sum_{j=1}^{N_U} L(-1, f(\mathbf{x}^*_j)) \right\}.$$

In particular, Liu et al. [2003] propose to use the hinge loss (cf. section 1.2.2) and to optimize the weights w_P and w_U. This method, called the BiasedSVM, is currently the state-of-art PU learning method. Another possible choice for the weights is derived from a common practice when dealing with unbalanced data: $w_P = \frac{N_U}{N}$ and $w_U = 1 - w_P$.

1.2.2 The Support Vector Machine

The Support Vector Machine (SVM) is a supervised classification method which was developed in the nineties by Boser et al. [1992] on the grounds of Vapnik-Chervonenkis statistical learning theory [Vapnik, 1998]. Their ability to tackle high dimensional problems and their good performance in many applications have made SVMs a popular technique.

1.2.2.1 A geometrical intuition

Introducing SVMs from a geometrical point of view provides a good intuition of the principles underlying this method. To give a formal framework to our introduction, suppose the input data is a random variable $\mathbf{X}$ which takes its value in $\mathcal{X}$ [4], while $\mathbf{Y} \in \{-1, 1\}$. The training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ consists in n observations drawn from $\mathcal{P}$, the unknown joint distribution of $(\mathbf{X}, \mathbf{Y})$. Figure 1.12 gives an example of a pattern recognition problem. Here, an instance $y_i = 1$ means example i is a pear while $y = -1$ means the example is an apple. Variable $\mathbf{X}$, which features the pictures, might be the set of its pixels for instance.

Assume you want to find a linear separation, i.e a hyperplane to discriminate your two classes. There might be several hyperplanes that are suitable for this. A natural idea is to represent each class by its centroid and to draw the bisecting plane of these two points. Figure 1.13 shows that this is not always a successful strategy, even in the case where the classes are linearly separable.

Adopting a different perspective, Vapnik introduced the large margin separation paradigm. He proposed to consider a hyperplane class defined in a

[4] For the moment, let's consider $\mathcal{X} \subseteq \mathbb{R}^p$.

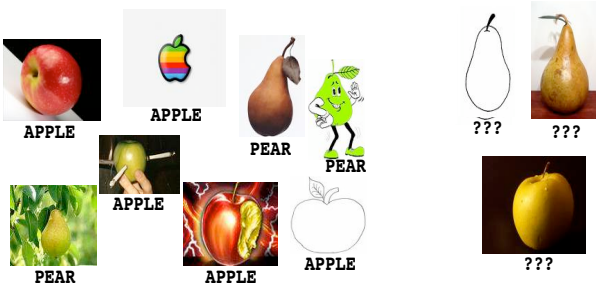


Figure 1.12: A pattern recognition problem.

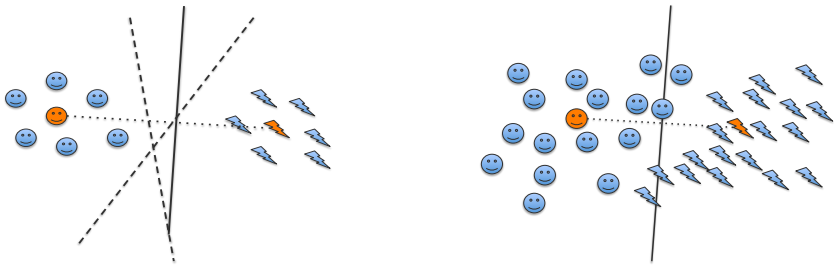


Figure 1.13: A straighforward separating hyperplane is not always convenient.

Hilbert space $\mathcal{H}$ by

$$\left\{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\right\}.$$

The optimal hyperplane separates the two classes while satisfying

$$\max_{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}} \min \left\{ \|\mathbf{x} - \mathbf{x}_i\|, \mathbf{x} \in \mathcal{X}, \langle \mathbf{w}, \mathbf{x} \rangle + b = 0, i \in \{1, \ldots, n\} \right\}.$$

Figure 1.14 shows an example for $\mathcal{X} \subset \mathbb{R}$. A first remark is that the hyperplane remains unchanged if both b and $\mathbf{w}$ are muliplied by the same constant. Thus, to set up a scale, the constant c is conventionally fixed to 1. The resulting canonical hyperplane is at equal distance from the two points with opposite labels $\mathbf{x}_1$ and $\mathbf{x}_2$ and the margin is equal to $\frac{2}{\|\mathbf{w}\|}$. This hyperplane can be obtained by maximizing the margin under the constraint that the points are correctly separated, which amounts to solving the following convex optimization problem

$$\begin{cases} \min\limits_{\mathbf{w}, b} \|\mathbf{w}\|^2 \\ \text{st } y_i * (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \ \forall i \in \{1, \ldots, n\}. \end{cases}$$

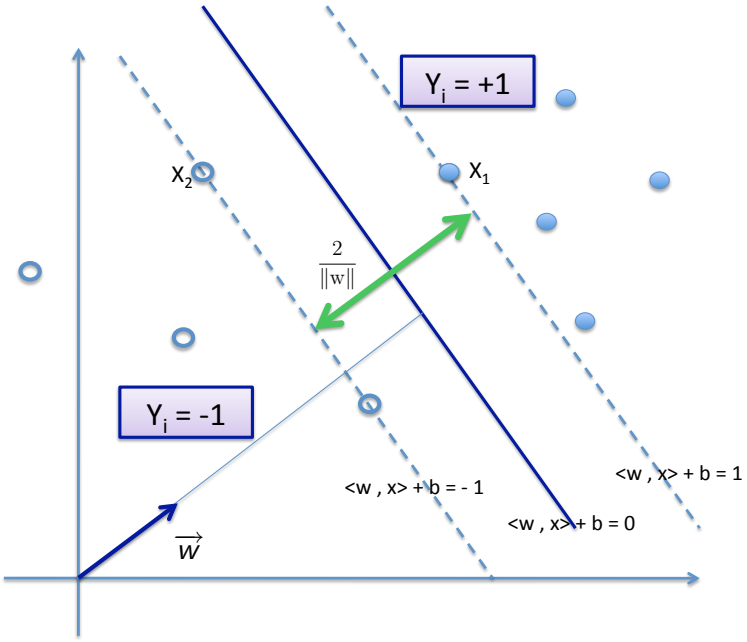


Figure 1.14: The large margin separating hyperplane.

At this point, there are a few remarks to make:

- As no error is tolerated, it can happen that no solution exists for the optimization problem.

- From a robustness point of view, a solution where an outlier can affect the hyperplane dramatically is not desirable.

- We have restricted ourselves to linear classifiers but in real situations, the examples are seldom linearly separable.

1.2.2.2 Soft margin SVMs

To circumvent the two first issues, Cortes and Vapnik [1995] have introduced the soft margin SVM. The idea was to add "slack variables" that represent errors. Figure 1.15 represents these slack variables ξ_i.

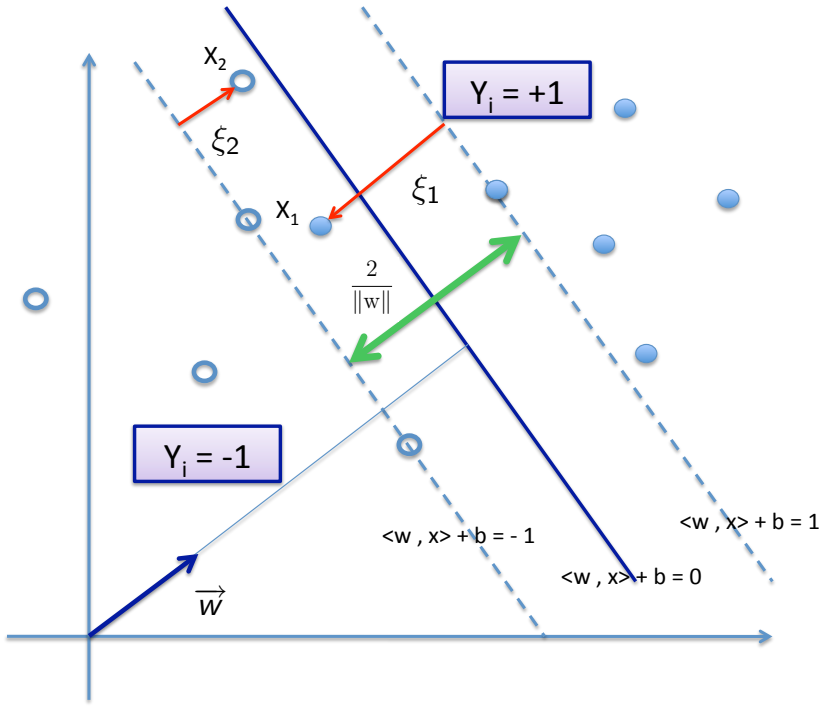


Figure 1.15: Soft margin hyperplane: introducing slack variables.

The soft margin hyperplane is obtained by minimizing the margin and correctly separating the two classes up to errors as small as possible. The

corresponding optimization problem is now:

$$\begin{cases} \min\limits_{\mathbf{w},b} \|\mathbf{w}\|^2 + C\sum\limits_{i=1}^{n} \xi_i \\ \text{st } y_i * (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \forall i \in \{1, \dots, n\} \\ \quad \xi_i \geq 0 \qquad\qquad\qquad\qquad \forall i \in \{1, \dots, n\}\,. \end{cases} \tag{1.5}$$

Parameter $C \in \mathbb{R}$ in equation (1.5) plays exactly the same role as in equation (1.2), that is controlling the bias-variance trade-off. When C is large, the second terms prevails and the optimizer will strive to fit the data to reduce the bias at the detriment of the variance of the predictor. On the contrary, when C is small, it will endeavor to minimize the first term, allowing more errors (more bias) but yielding a predictor with better generalization ability (less variance). Note that slack variables can be introduced similarly for transductive and one-class SVM in equations (1.3) and (1.4).

Equation (1.5) can equivalently be re-written:

$$\begin{cases} \min\limits_{f} \|f\|^2 + C\sum\limits_{i=1}^{n} \xi_i \\ \text{st } \xi_i \geq \max\left(1 - y_i * (f(\mathbf{x}_i) + b), 0\right) \quad \forall i \in \{1, \dots, n\}\,. \end{cases} \tag{1.6}$$

which is equivalent to

$$\min_{f} \|f\|^2 + C\sum_{i=1}^{n} \max\left(1 - y_i * (f(\mathbf{x}_i) + b), 0\right), \tag{1.7}$$

where $f(\mathbf{x}_i) = \langle \mathbf{w}, \mathbf{x}_i \rangle$ and $\|f\| = \|\mathbf{w}\|$. This makes the link with equation (1.2) for the choice of a particular loss function known as the hinge loss (see figure 1.16). If we note $h(\mathbf{x}) = f(\mathbf{x}) + b$, the distance from $\mathbf{x}$ to the hyperplane, the hinge loss is defined as

$$L(h(\mathbf{x}), y) = \max(1 - y * h(\mathbf{x}), 0)\,.$$

It is small when y and $h(\mathbf{x})$ have the same sign and zero when $h(\mathbf{x})$ is additionally bigger than 1. When $0 \leq y * h(\mathbf{x}) \leq 1$, the corresponding point is between the margin hyperplanes but correctly classified so the loss is positive but small. At last, $y * h(\mathbf{x}) < 0$ means the point is incorrectly classified (on the wrong side of the separating hyperplane) and this error is penalized by a larger loss value. Function h is the score function, it can be used directly for prioritization purposes. If one wishes to produce a classification rule instead, one can decide to assign each point $\mathbf{x}$ to the positive or negative class according to the sign of $h(\mathbf{x})$. Note that the hinge loss is also used for transductive SVM (see equation (1.3)). For one-class SVM, the loss function is $\max(0, f(\mathbf{x}))$.

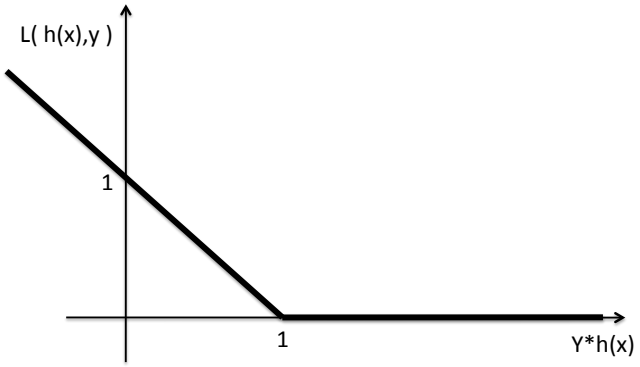


Figure 1.16: The hinge loss function.

1.2.2.3 Non-linear SVMs

In practice, the idea of error tolerance has to be combined with the search for separation boundaries that are more general than linear boundaries. In order to obtain an elliptic separation boundary like on figure 1.17, one feels that it makes sense to define it mathematically by:

$$w_1x^2 + w_2y^2 + 2w_3xy + w_4x + w_5y + b = 0\,. \tag{1.8}$$

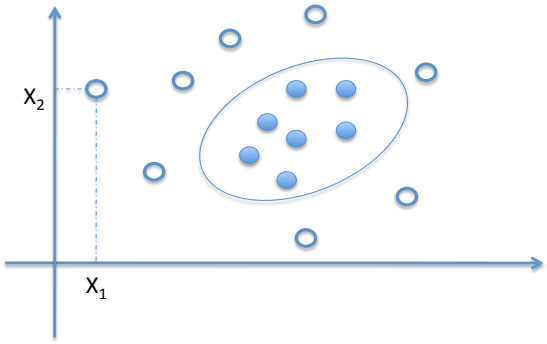


Figure 1.17: Looking for a non-linear separation boundary.

If we define $\mathbf{w} = (w_1, \ldots, w_5)$ and the transforming function ϕ as

$$\begin{array}{rccc} \phi: & \mathbb{R}^2 & \longrightarrow & \mathbb{R}^5 \\ & \mathbf{x} = (x_1, x_2) & \longmapsto & (x_1^2, x_2^2, 2x_1x_2, x_1, x_2)\,, \end{array}$$

we see that equation (1.8) can be re-written as a linear equation defining a hyperplane in $\mathbb{R}^5$.

$$\mathbf{w}^T\phi(\mathbf{x}) + b = 0\,.$$

More generally, one can proceed in the following manner:

- Apply a non-linear transformation ϕ to the input data to map them from $\mathcal{X}$ to a higher-dimensional space $\mathcal{H}$.
- Build a linear hyperplane in $\mathcal{H}$.

This hyperplane will define a non-linear separation in the original feature space $\mathcal{X}$. All one needs to do is replace $\mathbf{x}$ by $\phi(\mathbf{x})$ in all the minimization problems above. Nevertheless, a critical issue is the choice of function ϕ. In the following section, we introduce the reader to kernel methods, which allow to choose ϕ "implicitely" and we present further non-linear SVMs.

1.2.3 Kernels methods

1.2.3.1 Motivations

Learning algorithms all use a certain data representation, such as those we have reviewed in section 4.5. However, it is not always obvious to describe data efficiently. It might be unclear which representation to choose, and if this representation makes sense. In addition, if descriptors are too numerous, computations might become difficult. When looking at the different representations we have proposed in section 4.5, we see that many of them are rather justified by what they imply in terms of comparison between two genes and not so much by their ability to feature intrinsically these genes. Kernel methods offer a framework for representing any kind of object through pairwise comparisons.

1.2.3.2 Definitions

The central tool of these methods is the kernel function. It is indeed a comparison function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ which transforms the input data

$$\mathbf{x}_i \in \mathcal{X}, i \in (1, \ldots, n)$$

into a $n \times n$ matrix K defined by:

$$[K]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)\,.$$

This matrix is called the Gram matrix. More precisely, we restrict ourselves to a specific type of kernels which are positive definite kernels.

Definition A positive definite (p.d) kernel on $\mathcal{X}$ is a symmetrical function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}),$$

which satisfies, for all $N \in \mathbb{N}$, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$ and $(a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

Lemma 1.2.1 *Let $\mathcal{X}$ be some input space and $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$. Then, function $K : \mathcal{X}^2 \mapsto \mathbb{R}$ defined as follows is p.d:*

$$\forall \, (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K \, (\mathbf{x}, \mathbf{x}') = \langle \Phi \, (\mathbf{x}) \, , \Phi \, (\mathbf{x}') \rangle_{\mathbb{R}^d} \, .$$

1.2.3.3 The kernel trick

Lemma 1.2.1 states that an inner product defines a positive definite kernel. To show the converse, one needs to introduce the notion of a reproductive kernel Hilbert space (RKHS) and to expound Aronszajn's theorem [Aronszajn, 1950].

Definition Let $\mathcal{X}$ be some set and $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ a class of functions forming a Hilbert space endowed with an inner product $\langle ., . \rangle_{\mathcal{H}}$. Function $K : \mathcal{X}^2 \mapsto \mathbb{R}$ is called a reproducing kernel (r.k) of $\mathcal{H}$ if

1. $\mathcal{H}$ contains all functions of the form

$$\forall \mathbf{x} \in \mathcal{X}, \; K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) \, .$$

2. $\forall \mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$, the reproducing property is satisfied :

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} \, .$$

If a r.k exists, then $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS).

Theorem 1.2.2 *(Aronszajn, 1950)*

1. *If a r.k exists, it is unique.*
2. *A r.k exists iff $\forall \boldsymbol{x} \in \mathcal{X}$, the mapping $f \mapsto f(\boldsymbol{x})$ (from $\mathcal{H}$ in $\mathbb{R}$) is continuous.*

3. *A r.k is positive definite.*

4. *Conversely, if K is a d.p kernel, then a RKHS exists whose reproducing kernel is K.*

The consequence of this theorem is that a positive definite kernel is an inner product:

Corollary 1.2.3 *K is a positive kernel on $\mathcal{X}$ if and only if there is a RKHS $\mathcal{H}$ and a mapping*

$$\Phi : \mathcal{X} \mapsto \mathcal{H} ,$$

such that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}} .$$

A major practical application of this corollary is called the "kernel trick". The idea is the following: if an algorithm handles finite dimension vectors and can be expressed using only inner products of these vectors, replacing them all by a positive kernel function K is implicitly equivalent to applying this algorithm to the vectors mapped in the RKHS $\mathcal{H}_K$. Therefore, it is neither necessary to manipulate the transformed vectors $\Phi(\mathbf{x})$ nor to formulate function Φ. However, note that when n is large, computing the Gram matrix might be computationally heavy, notwithstanding memory and storage issues.

A straighforward way to define a kernel when a vectorial representation of the data is available, is to compute the Gram matrix as the matrix of dot products. This is the linear kernel. In the vectorial cases, there are also several direct definitions of "ready-made" kernels such as the polynomial and gaussian radial basis function (RBF).

Definition

- Polynomial: $K(x, x') = (xx' + 1)^d$.
- Gaussian RBF: $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$.

1.2.3.4 A kernelized SVM

We now explain how to solve the SVM optimization problem using the reproducing kernel theory. Let $\mathcal{H}_K$ be a RKHS, the problem can be stated as

$$\min_{f \in \mathcal{H}_K, b \in \mathbb{R}} \|f\|^2_{\mathcal{H}_K} + C \sum_{i=1}^{n} \max\left(1 - y_i * (f(\mathbf{x}_i) + b), 0\right) . \tag{1.9}$$

We now present the representer theorem which states that f admits a particular shape that will help solve the optimization problem in (1.9).

Theorem 1.2.4

- *Let $\mathcal{X}$ be a set with a p.d kernel K, $\mathcal{H}_K$ the associated RKHS and let $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$ be a finite set of points of $\mathcal{X}$*

- *Let $\psi : \mathbb{R}^{n+1} \mapsto \mathbb{R}$ be a function of $n+1$ variables, strictly increasing with respect to the last variable.*

- *Then, any solution of the optimization problem*

$$\min_{f \in \mathcal{H}_K} \psi(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}_K})$$

admits a representation of the form

$$\forall \mathbf{x} \in \mathcal{X},\ f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i)\,.$$

Using the representer theorem, the dual problem comes as

$$\begin{cases} \max\limits_{\alpha \in \mathbb{R}^n} -\alpha^T K \alpha + 2\alpha^T \mathbf{y} \\ \text{st } 0 \leq \alpha_i y_i \leq C,\ \forall i \in \{1, \ldots, n\}\,. \end{cases} \tag{1.10}$$

Since this is a convex optimization problem, the Karush-Kuhn and Tucker (KKT) conditions hold and provide some interpretation of coefficients α_i on the data points:

1. $\alpha_i = 0 \Rightarrow \xi_i = 0$ and $y_i(f(x_i) + b) > 1$
   The first constraint is not active while the second is. The point is well classified and beyond the marginal hyperplanes.

2. $0 < \alpha_i y_i < C \Rightarrow \xi_i = 0$ and $y_i(f(x_i) + b) = 1$
   Both constraints are active. The point is well classified and lies exactly on the marginal hyperplane.

3. $\alpha_i y_i = C \Rightarrow \xi_i > 0$ and $y_i(f(x_i) + b) < 1$
   The first constraint is active while the second is not. This point lies on the wrong side of its marginal hyperplane and can even be misclassified.

Figure 1.18 illustrates the different cases. The points of the training set are circled with a different color according to their α coefficient and the corresponding location with respect to the marginal hyperplanes (represented with dashed lines). The points for which $\alpha_i \neq 0$ are called the "support vectors" and have given its name to the method. Only these points in all the training set play a role in the score function and are therefore determinant to classify or score a new test point via

$$\forall \mathbf{x} \in \mathcal{X}, h(\mathbf{x}) = \sum_{i \in SV} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\,.$$

The solution is sparse in the input which leads to fast algorithms. Besides, scoring a new point only requires a few kernel evaluations.

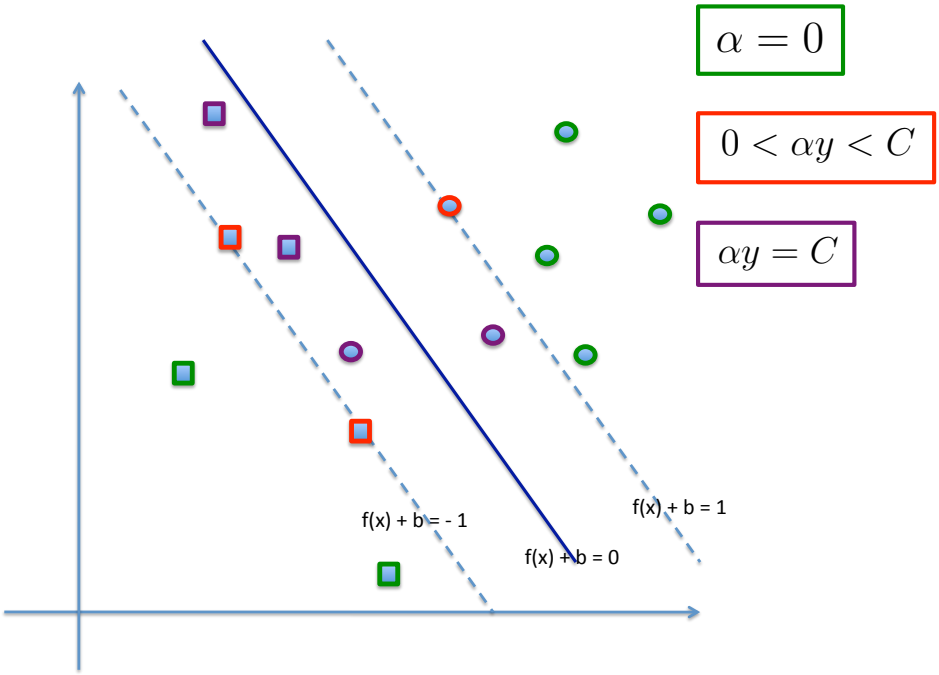


Figure 1.18: A geometrical interpretation of the α coefficients.

1.2.3.5 Data fusion with kernels

We now present a proposition which allows to perform data fusion in an elegant manner with kernels. Suppose you have two data sources and you have built two positive definite kernels K_1 and K_2.

Proposition 1.2.5 *Let $\mathcal{X}$ be a set and K_1, K_2 two positive definite kernels defined on $\mathcal{X} \times \mathcal{X}$.*
For all positive coefficients λ_1, λ_2, the function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined by

$$(\mathbf{x}, \mathbf{x}') \longmapsto K(\mathbf{x}, \mathbf{x}') = \lambda_1 K_1(\mathbf{x}, \mathbf{x}') + \lambda_2 K_2(\mathbf{x}, \mathbf{x}')$$

is a positive definite kernel.

As a consequence, performing any kernel learning method with kernel K amounts to integrating simply two data sources. Actually, if we denote by $\Phi_1 : \mathcal{X} \to \mathcal{H}_{K_1} \subseteq \mathbb{R}^{d_1}$ (resp. Φ_2) the mapping function associated with K_1 (resp. K_2), we can show that the mapping function Φ of kernel K is defined by

$$\begin{array}{rccl} \Phi: & \mathcal{X} & \longrightarrow & \mathcal{H}_K \subseteq \mathbb{R}^{d_1+d_2} \\ & \mathbf{x} & \longmapsto & (\sqrt{\lambda_1} * \Phi_1(\mathbf{x})^T, \sqrt{\lambda_2} * \Phi_2(\mathbf{x})^T)^T \,. \end{array}$$

Therefore, the transformed features are simply weighted and concatenated. A learner can choose the weights if he has a prior idea on the relative importance of the data sources. Equal weights mean that all data sources equally contribute to the score function f. Suppose there are d positive definite kernels $\{K_1, K_2, \ldots, K_d\}$. Replacing K with the sum of these kernels in equation (1.10) gives the following problem:

$$\left\{ \begin{array}{l} \max\limits_{\alpha \in \mathbb{R}^n} -\sum\limits_{j=1}^{d} \alpha^T K_j \alpha + 2\alpha^T \mathbf{y} \\ \text{st } 0 \leq \alpha_i y_i \leq C, \ \forall i \in \{1, \ldots, n\} \,, \end{array} \right. \tag{1.11}$$

which is the dual of a minimization problem, which looks for a function f of the form $f = f_1 + \ldots + f_d$, with $(f_1, \ldots, f_d) \in \mathcal{H}_{K_1} \times \ldots \times \mathcal{H}_{K_d}$:

$$\min_{(f_1,\ldots,f_d), b \in \mathbb{R}} \sum_{j=1}^{d} \|f_j\|^2_{\mathcal{H}_{K_j}} + C \sum_{i=1}^{n} L\Big(\sum_{j=1}^{d} f_j(\mathbf{x}_i) + b, y_i \Big) \,. \tag{1.12}$$

Otherwise, the weights can also be learnt from the data thanks to Multiple Kernel Learning (MKL) [Bach et al., 2004, Lanckriet et al., 2004c]. We further assume that all kernels are normalized to have a 1-diagonal. The optimal weights should solve the following min-max problem

$$\left\{ \begin{array}{l} \min\limits_{\mu \in \mathbb{R}^d} \max\limits_{\alpha \in \mathbb{R}^n} -\sum\limits_{j=1}^{d} \mu_j * \alpha^T K_j \alpha + 2\alpha^T \mathbf{y} \\ \text{st } 0 \leq \alpha_i y_i \leq C, \ \forall i \in \{1, \ldots, n\} \\ \quad \|\mu\|_1 = 1 \text{ and } \mu_j \geq 0, \ \forall j \in \{1, \ldots, d\} \,. \end{array} \right. \tag{1.13}$$

This is in turn equivalent to

$$\begin{cases} \max\limits_{t\in\mathbb{R}} \max\limits_{\alpha\in\mathbb{R}^n} -t + 2\alpha^T \mathbf{y} \\ \text{st } 0 \leq \alpha_i y_i \leq C,\ \forall i \in \{1,\dots,n\} \\ \quad t \geq \alpha^T K_j \alpha,\ \forall j \in \{1,\dots,d\}\,. \end{cases} \tag{1.14}$$

Finally, Bach et al. [2004] have shown that problem (1.14) is the dual of the Support Kernel Machine (SKM) problem:

$$\min_{(f_1,\dots,f_d),b\in\mathbb{R}} \sum_{j=1}^{d} \|f_j\|_{\mathcal{H}_{K_j}} + C \sum_{i=1}^{n} L\Big(\sum_{j=1}^{d} f_j(\mathbf{x}_i) + b, y_i\Big)\,. \tag{1.15}$$

We see that only the functional norm term has changed with respect to equation (1.12).

In this section, we have provided a unified view of some machine learning methods, in terms of functional optimization. This should allow the reader to see clearer in the landscape of methods and also to compare them more easily.

1.3 Contributions of this thesis

In this section, we give an overview of the contributions of this thesis. There are mainly three contributions, one methodological and two of applicative nature. Each of them is the topic of a dedicated chapter where it is further developped. Note that each chapter corresponds to a published or submitted paper. For the sake of consistency, we did not follow any chronological order in the presentation.

1.3.1 A bagging SVM to learn from positive and unlabeled examples

In chapter 2, we first consider the problem of learning a binary classifier from a training set of positive and unlabeled examples, both in the inductive and in the transductive setting. This problem, aforementioned as *PU learning* in section 1.2.1.3, differs from the standard supervised classification problem by the lack of negative examples in the training set. It corresponds to an ubiquitous situation in many applications such as information retrieval or gene ranking, when we have identified a set of data of interest sharing a particular property, and we wish to automatically retrieve additional data sharing the

same property among a large and easily available pool of unlabeled data. A first contribution was to propose a conceptually simple method, akin to bagging, to approach both inductive and transductive PU learning problems, by converting them into series of supervised binary classification problems discriminating the known positive examples from random subsamples of the unlabeled set. We empirically demonstrated the relevance of the method on simulated and real data, where it performs at least as well as existing methods while being faster.

1.3.2 Regulatory network inference

Chapter 3 is dedicated to the problem of regulatory network inference. As explained in section 1.1.1, living cells are the product of gene expression programs that involve the regulated transcription of thousands of genes. The elucidation of transcriptional regulatory networks in thus needed to understand the cell's working mechanism, and can for example be useful for the discovery of novel therapeutic targets. Although several methods have been proposed to infer gene regulatory networks from gene expression data, a recent comparison on a large-scale benchmark experiment revealed that most current methods only predict a limited number of known regulations at a reasonable precision level. A second contribution of this thesis is to propose SIRENE, a new method for the inference of gene regulatory networks from a compendium of expression data. The method decomposes the problem of gene regulatory network inference into a large number of local binary classification problems, that focus on separating target genes from unlabeled genes for each TF. SIRENE does not need any assumption on the data and it introduces a new paradigm of inference, based on a biologically meaningful idea: if a gene A is regulated by a TF B, then a gene A' showing similarity with gene A is likely to be regulated by TF B. SIRENE is thus conceptually simple and computationally efficient. We have tested it on a benchmark experiment aimed at predicting regulations in *E. coli*, and shown that it retrieves of the order of 6 times more known regulations than other state-of-the-art inference methods.

1.3.3 Identification of disease genes with PU learning

Finally, chapter 4 deals with the application of PU learning to the disease gene identification problem. Recall that elucidating the genetic basis of human diseases is a central goal of genetics and molecular biology. While traditional linkage analysis and modern high-throughput techniques often provide long lists of tens or hundreds of disease gene candidates, the identification of

disease genes among the candidates remains time-consuming and expensive. As stressed in section 1.1.2.2, efficient computational methods are therefore needed to prioritize genes within the list of candidates, by exploiting the wealth of informations available about the genes in various databases. As a third contribution of this thesis, we propose ProDiGe, a novel algorithm for Prioritization of Disease Genes. ProDiGe implements a machine learning strategy based on learning from positive and unlabeled examples, which allows to integrate various sources of information about the genes, to share information about known disease genes across diseases, and to perform genome-wide searches for new disease genes. Experiments on real data show that ProDiGe outperforms state-of-the-art methods for the prioritization of genes in human diseases.

Chapter 2

A bagging SVM to learn from positive and unlabeled examples

2.1 Résumé

Ce premier chapitre est consacré au problème d'apprentissage d'un classifieur binaire à partir d'un ensemble d'entraînement constitué d'exemples positifs et indéterminés (non étiquetés). On s'intéresse tant au cadre inductif que transductif. Ce problème, appelé PU learning en anglais, diffère du cadre d'apprentissage supervisé standard par le manque d'exemples négatifs dans l'ensemble d'entraînement. Il correspond à une situation omniprésente dans des applications aussi variées que la recherche d'information ou la priorisation de gènes, où l'on a identifié un ensemble de données d'intérêt ayant en commun une propriété particulière et où l'on souhaite identifier automatiquement d'autres objets partageant cette propriété parmi un ensemble large et facilement accessible d'exemple indéterminés. Dans ce chapitre, nous proposons une méthode conceptuellement simple, voisine du bagging (bootstrap aggregating) pour traiter à la fois l'approche inductive et transductive du problème, en le convertissant en une série de problèmes de classification binaire supervisés discriminant les exemples positifs contre des sous-échantillons aléatoires de l'ensemble des indéterminés. La pertinence de cette méthode est démontrée empiriquement sur des données simulées puis sur des données réelles, où l'on observe qu'elle a au moins d'aussi bons résultats que les méthodes existantes tout en étant plus rapide.

2.2 Introduction

In many applications, such as information retrieval or gene ranking, one is given a finite set of data of interest sharing a particular property, and wishes to find other data sharing the same property. In information retrieval, for example, the finite set can be a user query, or a set of documents known to belong to a specific category, and the goal is to scan a large database of documents to identify new documents related to the query or belonging to the same category. In gene ranking, the query is a finite list of genes known to have a given function or to be associated to a given disease, and the goal is to identify new genes sharing the same property [Aerts et al., 2006]. In fact this setting is ubiquitous in many applications where identifying a data of interest is difficult or expensive, e.g., because human intervention is necessary or expensive experiments are needed, while unlabeled data can be easily collected. In such cases there is a clear opportunity to alleviate the burden and cost of interesting data identification with the help of machine learning techniques.

More formally, let us assign a binary label to each possible data: positive $(+1)$ for data of interest, negative (-1) for other data. Unlabeled data are data for which we do not know whether they are interesting or not. Denoting $\mathcal{X}$ the set of data, we assume that the "query" is a finite set of data $\mathcal{P} = \{x_1, \ldots, x_m\} \subset \mathcal{X}$ with positive labels, and we further assume that we have access to a (possibly large) set $\mathcal{U} = \{x_{m+1}, \ldots, x_n\}$ of unlabeled data. Our goal is to learn, from $\mathcal{P}$ and $\mathcal{U}$, a way to identify new data with positive labels, a problem often referred to as *PU learning*. More precisely we make a distinction between two flavors of PU learning:

- *Inductive PU learning*, where the goal is to learn from $\mathcal{P}$ and $\mathcal{U}$ a function $f : \mathcal{X} \to \mathbb{R}$ able to associate a score or probability to be positive $f(x)$ to any data $x \in \mathcal{X}$. This may typically be the case in an image or document classification system, where a subset of the web is used as unlabeled set $\mathcal{U}$ to train the system, which must then be able to scan any new image or document.

- *Transductive PU learning*, where the goal is estimate a scoring function $s : \mathcal{U} \to \mathbb{R}$ from $\mathcal{P}$ and $\mathcal{U}$, i.e., where we are just interested is finding positive data in the set $\mathcal{U}$. This is typically the case in the disease gene ranking application, where the full set of human genes is known during training and split between known disease genes $\mathcal{P}$ and the rest of the genome $\mathcal{U}$. In that case we are only interested in finding new disease genes in $\mathcal{U}$.

Several methods for PU learning, reviewed in Section 2.3 below, reduce the problem to a binary classification problem where we learn to discriminate $\mathcal{P}$ from $\mathcal{U}$. This can be theoretically justified, at least asymptotically, since the log-ratio between the conditional distributions of positive and unlabeled examples is monotonically increasing with the log-ratio of positive and negative examples [Elkan and Noto, 2008, Scott and Blanchard, 2009], and has given rise to state-of-the-art methods such as *biased support vector machine (biased SVM)* [Liu et al., 2003] or weighted logistic regression [Lee and Liu, 2003]. Although this reduction suggests that virtually any method for (weighted) supervised binary classification can be used to solve PU learning problems, we put forward in this chapter that some methods may be more adapted than others in a non-asymptotic setting, due to the particular structure of the unlabeled class. In particular, we investigate the relevance of methods based on aggregating classifiers trained on artificially perturbed training sets, in the spirit of bagging [Breiman, 1996]. Such methods are known to be relevant to improve the performance of unstable classifiers, a situation which, we propose, may occur particularly in PU learning. Indeed, in addition to the usual instability of learning algorithms confronted to a finite-size training sets, the content of a random subsample of unlabeled data in positive and negative examples is likely to strongly affect the classifier, since the contamination of $\mathcal{U}$ in positive examples makes the problem more difficult. Variations in the contamination rate of $\mathcal{U}$ may thus have an important impact on the trained classifier, a situation which bagging-like classifiers may benefit from.

Based on this idea, we propose a general and simple scheme for inductive PU learning, akin to an asymetric form of bagging for supervised binary classification. The method, which we call *bagging SVM*, consists in aggregating classifiers trained to discriminate $\mathcal{P}$ from a small random subsample of $\mathcal{U}$, where the size of the random sample plays a specific role. This method can naturally be adapted to the transductive PU learning framework. We demonstrate on simulated and real data that bagging SVM performs at least as well as existing methods for PU learning, while being often faster in particular when $|\mathcal{P}| << |\mathcal{U}|$.

This chapter is organized as follows. After reviewing related work in Section 2.3, we present the bagging SVM for inductive PU learning in Section 2.4, and its extension to transductive PU learning in Section 2.5. Experimental results are presented in 2.6, followed by a Discussion in Section 2.7.

2.3 Related work

A growing body of work has focused on PU learning recently. The fact that only positive and unlabeled examples are available prevents a priori the use of supervised classification methods, which require negative examples in the training set. A first approach to overcome the lack of negative examples is to disregard unlabeled examples during training and simply learn from the positive examples, e.g., by ranking the unlabeled examples by decreasing similarity to the mean positive example [Joachims, 1997] or using more advanced learning methods such as 1-class SVM [Schölkopf et al., 2001, Manevitz and Yousef, 2001, Vert and Vert, 2006, De Bie et al., 2007]

Alternatively, the problem of inductive PU learning has been studied on its own from a theoretical viewpoint [Denis et al., 2005, Scott and Blanchard, 2009], and has given rise to a number of specific algorithms. Several authors have proposed two-step algorithms, heuristic in nature, which first attempt to identify negative examples in the unlabeled set, and then estimate a classifier from the positive, unlabeled and likely negative examples [Manevitz and Yousef, 2001, Liu et al., 2002, Li and Liu, 2003, Liu et al., 2003, Yu et al., 2004a]. Alternatively, it was observed that directly learning to discriminate $\mathcal{P}$ from $\mathcal{U}$, possibly after rebalancing the misclassification costs of the two classes to account for the asymetry of the problem, leads to state-of-the-art results for inductive PU learning. This approach has been studied, with different weighting schemes, using a logistic regression or a SVM as binary classifier [Liu et al., 2003, Lee and Liu, 2003, Elkan and Noto, 2008]. Inductive PU learning is also related to and has been used for novelty detection, when $\mathcal{P}$ is interpreted as "normal" data and $\mathcal{U}$ contains mostly positive examples [Scott and Blanchard, 2009], or to data retrieval from a single query, when $\mathcal{P}$ is reduced to a singleton [Shah et al., 2008].

Transductive PU learning is arguably easier than inductive PU learning, since we know in advance the data to be screened for positive labels. Many semi-supervised methods have been proposed to tackle transductive learning when both positive and negative examples are known during training, including transductive SVM [Joachims, 1999], or many graph-based methods, reviewed by Chapelle et al. [2006]. Comparatively little effort has been devoted to the specific transductive PU learning problem, with the notable exception of Liu et al. [2002], who call the problem *partially supervised classification* and proposes an iterative method to solve it, and Pelckmans and Suykens [2009] who formulate the problem as a combinatorial optimization problem over a graph. Finally, Sriphaew et al. [2009] recently proposed a bagging approach which shares similarities with ours, but is more complex and was only tested on a specific application.

2.4 Bagging for inductive PU learning

Our starting point to learn a classifier in the PU learning setting is the observation that learning to discriminate positive from unlabeled samples is a good proxy to our objective, which is to discriminate positive from negative samples. Even though the unlabeled set is contaminated by hidden positive examples, it is generally admitted that its distribution contains some information which should be exploited. That is for instance, the foundation of semi-supervised methods.

Indeed, let us assume for example that positive and negative examples are randomly generated by class-conditional distributions $\mathbb{P}_+$ and $\mathbb{P}_-$ with densities h_+ and h_-. If we model unlabeled examples as randomly sampled from $\mathbb{P}_+$ with probability γ and from $\mathbb{P}_-$ with probability $1-\gamma$, then the distribution of unlabeled has a density

$$h_u = \gamma h_+ + (1-\gamma)h_- \,. \tag{2.1}$$

Now notice that

$$\frac{h_u(x)}{h_+(x)} = \gamma + (1-\gamma)\frac{h_-(x)}{h_+(x)} \,, \tag{2.2}$$

showing that the log-ratio between the conditional distributions of positive and unlabeled examples is monotonically increasing with the log-ratio of positive and negative examples [Elkan and Noto, 2008, Scott and Blanchard, 2009]. Hence any estimator of the conditional probability of positive vs. unlabeled data should in theory also be applicable to discriminate positive from negative examples. This is the case for example of logistic regression or some forms of SVM [Steinwart, 2003, Bartlett and Tewari, 207]. In practice it seems useful to train classifiers to discriminate $\mathcal{P}$ from $\mathcal{U}$ by penalizing more false negative than false positive errors, in order to account for the fact that positive examples are known to be positive, while unlabeled examples are known to contain hidden positives. Using soft margin SVM while giving high weights to false negative errors and low weights to false positive errors leads to the biased SVM approach described by Liu et al. [2003], while the same strategy using a logistic regression leads to the weighted logistic regression approach of Lee and Liu [2003]. Both methods, tested on text categorization benchmarks, were shown to be very efficient in practice, and in particular outperformed all approaches based on heuristic identifications of true negatives in $\mathcal{U}$.

Among the many methods for supervised binary classification which could be used to discriminate $\mathcal{P}$ from $\mathcal{U}$, bootstrap aggregating or "bagging" is an interesting candidate [Breiman, 1996]. The idea of bagging is to estimate a

series of classifiers on datasets obtained by perturbing the original training set through bootstrap resampling with replacement, and to combine these classifiers by some aggregation technique. The method is conceptually simple, can be applied in many settings, and works very well in practice [Breiman, 2001, Hastie et al., 2001]. Bagging generally improves the performance of individual classifiers when they are not too correlated to each other, which happens in particular when the classifier is highly sensitive to small perturbations of the training set. For example, Breiman [2001] showed that the difference between the expected mean square error (MSE) of a classifier trained on a single bootstrap sample and the MSE of the aggregated predictor increases with the variance of the classifier.

We propose that, by nature, PU learning problems have a particular structure that leads to instability of classifiers, which can be advantageously exploited by a bagging-like procedure which we now describe. Intuitively, an important source of instability in PU learning situations is the empirical contamination $\hat{\gamma}$ of $\mathcal{U}$ with positive examples, i.e., the percentage of positive examples in $\mathcal{U}$ which on average equals γ in (2.1). If by chance $\mathcal{U}$ is mostly made of negative examples, i.e., has low contamination by positive examples, then we will probably estimate a better classifier than if it contains mostly positive examples, i.e., has high contamination. Moreover, we can expect the classifiers in these different scenarii to be little correlated, since intuitively they estimate different log-ratios of conditional distribution. Hence, in addition to the "normal" instability of a classifier trained on a finite-size sample, which is exploited by bagging in general, we can expect an increased instability in PU learning due to the sensitivity of the classifier to the empirical contamination $\hat{\gamma}$ of $\mathcal{U}$ in positive examples. In order to exploit this sensitivity in a bagging-like procedure, we propose to randomly subsample $\mathcal{U}$ and train classifiers to discriminate $\mathcal{P}$ from each subsample, before aggregating the classifiers. By subsampling $\mathcal{U}$, we hope to vary in particular the empirical contamination between samples. This will induce a variety of situations, some lucky (small contamination), some less lucky (large contamination), which eventually will induce a large variability in the classifiers that the aggregation procedure can then exploit.

In opposition to classical bagging, the size K of the samples generated from $\mathcal{U}$ may play an important role to balance the accuracy against the stability of individual classifiers. On the one hand, larger subsamples should lead on average to better classifiers, since any classification method generally improves on average when more training points are available. On the other hand, the empirical contamination varies more for smaller subsamples. More precisely, let us denote by $\hat{\gamma}$ the true contamination rate in $\mathcal{U}$, that is, the true proportion of positive examples hidden in $\mathcal{U}$. Whenever a bootstrap sample

$\mathcal{U}_t$ of size K is drawn from $\mathcal{U}$, its empirical number of positive examples is a binomial random variable $\sim B(K, \hat{\gamma})$, leading to a contamination rate $\hat{\gamma}_t$ with mean and variance:

$$\mathbb{E}(\hat{\gamma}_t) = \hat{\gamma} \text{ and } \mathbb{V}(\hat{\gamma}_t) = \frac{1}{K}\hat{\gamma}(1-\hat{\gamma})\,.$$

Smaller values of K therefore increase the proportion of "lucky" subsamples, and more generally the variability of classifiers, a property which is beneficial for the aggregation procedure. Finally this suggests that the size K of subsample is a parameter whose effect should be studied and perhaps tuned.

In summary, the method we propose for PU learning is presented in Algorithm 1. We call it bagging SVM when the classifier used to discriminate $\mathcal{P}$ from a random subsample of $\mathcal{U}$ is a biased SVM. It is akin to bagging to learn to discriminate $\mathcal{P}$ from $\mathcal{U}$, with two important specificities. First, only $\mathcal{U}$ is subsampled. This is to account for the fact that elements in $\mathcal{P}$ are known to be positive, and moreover that the number of positive examples is often limited.Second, the size of subsamples is a parameter K whose effect needs to be studied. If an optimal value exists, then this parameter may need to be adjusted.

The number T of bootstrap samples is also a user-defined parameter. Intuitively, the larger T the better, although we observed empirically little improvement for T larger than 100. Finally, although we propose to aggregate the T classifiers by a simple average, other aggregation rules could easily be used. On preliminary experiments on simulated and real data, we did not observed significant differences between the simple average and majority voting, another popular aggregation method.

2.5 Bagging SVM for transductive PU learning

We now consider the situation where the goal is only to assign a score to the elements of $\mathcal{U}$ reflecting our confidence that these elements belong to the positive class. Liu et al. [2002] have studied this same problem which they call "partially supervised classification". Their proposed technique combines Naive Bayes classification and the Expectation-Maximization algorithm to iteratively produce classifiers. The training scores of these classifiers are then directly used to rank $\mathcal{U}$. Following this approach, a straightforward solution to the transductive PU learning problem is to train any classifier to discriminate between $\mathcal{P}$ and $\mathcal{U}$ and to use this classifier to assign a score to the unlabeled data that were used to train it. Using SVMs this amounts

to using the biased SVM training scores. We will subsequently denote this approach by transductive biased SVM.

However, one may argue that assigning a score to an unlabeled example that has been used as negative training example is problematic. In particular, if the classifier fits too tightly to the training data, a false negative x_i will hardly be given a high training score when used as a negative. In a related situation in the context of semi-supervised learning, Zhang et al. [2009] showed for example that unlabeled examples used as negative training examples tend to have underestimated scores when a SVM is trained with the classical hinge loss. More generally, most theoretical consistency properties of machine learning algorithms justify predictions on samples outside of the training set, raising questions on the use of all unlabeled samples as negative training samples at the same time.

Alternatively, the inductive bagging PU learning lends itself particularly well to the transductive setting, through the procedure described in Algorithm 2. Each time a random subsample $\mathcal{U}_t$ of $\mathcal{U}$ is generated, a classifier is trained to discriminate $\mathcal{P}$ from $\mathcal{U}_t$, and used to assign a predictive score to any element of $\mathcal{U} \setminus \mathcal{U}_t$. At the end the score of any element $x \in \mathcal{U}$ is obtained by aggregating the predictions of the classifiers trained on subsamples that did not contain x (the counter $n(x)$ simply counts the number of such classifiers). As such, no point of $\mathcal{U}$ is used simultaneously to train a classifier and to test it. In practice, it is useful to ensure that all elements of $\mathcal{U}$ are not too often in $\mathcal{U}_t$, in order to average the predictions over a sufficient number of classifiers.

Algorithm 1 Inductive bagging PU learning

INPUT : $\mathcal{P}$, $\mathcal{U}$, K = size of bootstrap samples, T = number of bootstraps
OUTPUT : a function $f : \mathcal{X} \to \mathbb{R}$
for $t = 1$ to T **do**
  Draw a subsample $\mathcal{U}_t$ of size K from $\mathcal{U}$.
  Train a classifier f_t to discriminate $\mathcal{P}$ against $\mathcal{U}_t$.
end for
Return

$$f = \frac{1}{T} \sum_{t=1}^{T} f_t$$

Algorithm 2 Transductive bagging PU learning

INPUT : $\mathcal{P}, \mathcal{U}$, K = size of bootstrap samples, T = number of bootstraps
OUTPUT : a score $s : \mathcal{U} \to \mathbb{R}$
Initialize $\forall x \in \mathcal{U}, n(x) \leftarrow 0, f(x) \leftarrow 0$
for $t = 1$ to T **do**
  Draw a bootstrap sample $\mathcal{U}_t$ of size K in $\mathcal{U}$.
  Train a classifier f_t to discriminate $\mathcal{P}$ against $\mathcal{U}_t$.
  For any $x \in \mathcal{U} \setminus \mathcal{U}_t$, update:

$$f(x) \leftarrow f(x) + f_t(x)\,,$$
$$n(x) \leftarrow n(x) + 1\,.$$

end for
Return $s(x) = f(x)/n(x)$ for $x \in \mathcal{U}$

2.6 Experiments

In this section we investigate the empirical behavior of our bagging algorithm on one simulated dataset (Section 2.6.1) and two real applications: text retrieval with the 20 newsgroup benchmark (Section 5.2), and reconstruction of gene regulatory networks (Section 5.3). We compare the new bagging SVM to the state-of-the-art biased SVM, and also add in the comparison for real data two one-class approaches, namely, ranking unlabeled examples by decreasing mean similarity to the positive examples (called *Baseline* below), and the one-class SVM [Schölkopf et al., 2001]. Both bagging and biased methods involve an SVM with asymetric penalties C_+ and C_- for the positive and negative class, respectively. By default we always set them to ensure that the total penalty is equal for the two classes, i.e., $C_+ n_+ = C_- n_-$, where n_+ and n_- are the number of positive and negative examples fed to the SVM, and optimized the single parameter $C = C_+ + C_-$ over a grid. We checked on all experiments that this choice was never significantly outperformed by other penalty ratio C_+/C_-.

2.6.1 Simulated data

A first series of experiments were conducted on simulated data to compare our bagging procedure to the biased approach in an inductive setting. We consider the simple situation where the positive examples are generated from an isotropic Gaussian distribution in $\mathbb{R}^p$: $\mathcal{P} \sim \mathbb{P}_+ = \mathcal{N}(0_p, \sigma * I_p)$, with $p = 50$ and $\sigma = 0.6$, while the negative examples are generated from another Gaussian distribution with same isotropic covariance and a different mean,

of norm 1. We replicate the following iteration 50 times for different values of γ :

- Draw a sample $\mathcal{P}$ of 5 positives examples, and a sample $\mathcal{U}$ of 50 unlabeled examples from $\gamma * \mathbb{P}_+ + (1-\gamma) * \mathbb{P}_-$.
- Train respectively the biased and bagging logit (with 200 bootstraps)[1].
- Compare their performance on a test set of 1000 examples containing 50% positives.

For K, we tested equally spaced values between 1 and 50, and we varied γ on the interval $[0; 0.9]$. The performance is measured by computing the area under the Receiving Operator Characteristic curve (AUC) on the independent test set. Figure 2.1 (left) shows the performance of bagging logit for different levels of contamination of $\mathcal{U}$, as a function of K, the size of the random samples. The uppermost curve thus corresponds to $\gamma = 0$, i.e., the case where all unlabeled data are negative, while the bottom curve corresponds to $\gamma = 0.8$, i.e., the case where 80% of unlabeled data are positive. Note that $K = 50$ corresponds to classical bagging on the biased logit classifier, i.e., to the case where all unlabeled examples are used to train the classifier.

We observe that in the classical setting of supervised binary classification where $\mathcal{U}$ is not contaminated by positive samples ($\gamma = 0$), the bagging procedure does not improve performance, whatever the size of the bootstrap samples. On the other hand, as contamination increases, we observe an overall decrease of the performance, confirming that the classification problem becomes more difficult when contamination increases. In addition, the bagging logit always succeeds in reaching at least the same performance for some value of K below 50, even for high rates of contamination. Figure 2.1 (right) shows the evolution of AUC as γ increases, for both methods. For the bagging logit we report the AUC reached for the best K value. We see that bagging logit slightly outperforms biased logit method.

To further illustrate the assumption that motivated bagging SVM, namely that decreasing K would decrease the average performance of single classifiers but would increase their variance due to the variations in contamination, we show in Figure 2.2 a scatter plot of the AUC of individual classifiers as a function of the empirical contamination of the bootstrap sample $\hat{\gamma}$, for two values of K (10 and 40). Here the mean contamination was set to $\gamma = 0.2$. Obviously, the variations of $\hat{\gamma}$ are much larger for $K = 10$ (between 0 and 0.5) than for $K = 40$ (between 0.1 and 0.25). The correlation coefficient between

[1]The bagging logit corresponds to the procedure described above, when the classifier is a logistic regression. This is the same for the biased logit, see also[Lee and Liu, 2003]

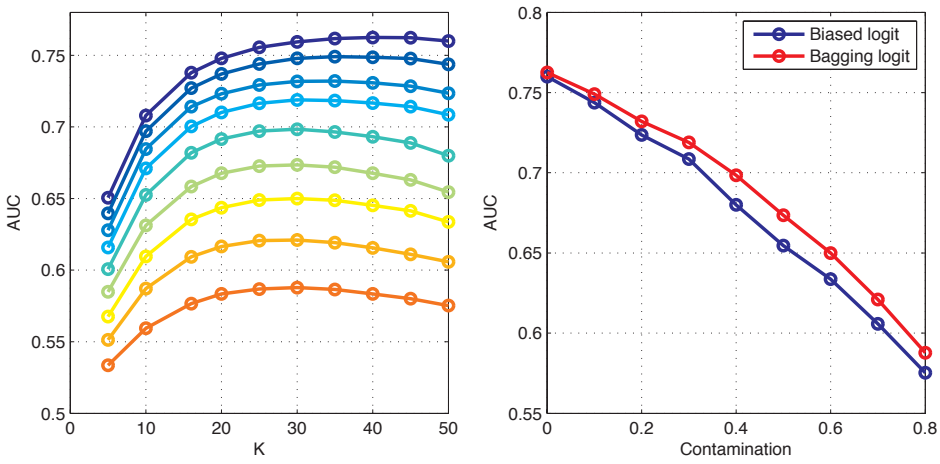


Figure 2.1: Results on simulated data. *Left:* AUC of the bagging logit as a function of K, the size of the bootstrap samples, on simulated data. Each curve, from top to bottom, corresponds to a contamination level $\gamma \in \{0; 0.1; 0.2; \ldots; 0.8\}$. *Right* Performance of two methods as a function of γ, the contamination level, on simulated data. The performance of bagging logit was taken at the optimal K value.

$\hat{\gamma}$ and the performance (reported above each plot) is strongly negative, in particular for smaller K. It is quite clear that less contaminated subsamples tend to yield better classifiers, and that the variation in the contamination is an important factor to increase the variance between individual predictors, which aggregation can benefit from.

2.6.2 Newsgroup dataset

The 20 Newsgroup benchmark is widely used to test PU learning methods. The version we used is a collection of 11293 articles partitioned into 20 subsets of roughly the same size (around 500)[2], corresponding to post articles of related interest. For each newsgroup, the positive class consists of those ∼500 articles known to be relevant, while the negative class is made of the remainder. After pre-processing, each article is represented by a 8165-dimensional vector, using the TFIDF representation over a dictionnary of 8165 words [Joachims, 1997].

To simulate a PU learning problem, we applied the following strategy. For a given newsgroup, we created a set $\mathcal{P}$ of known positive examples by

[2]We used the Matlab pre-processed version available at `http://renatocorrea.googlepages.com/ng2011293x8165itrn.mat`

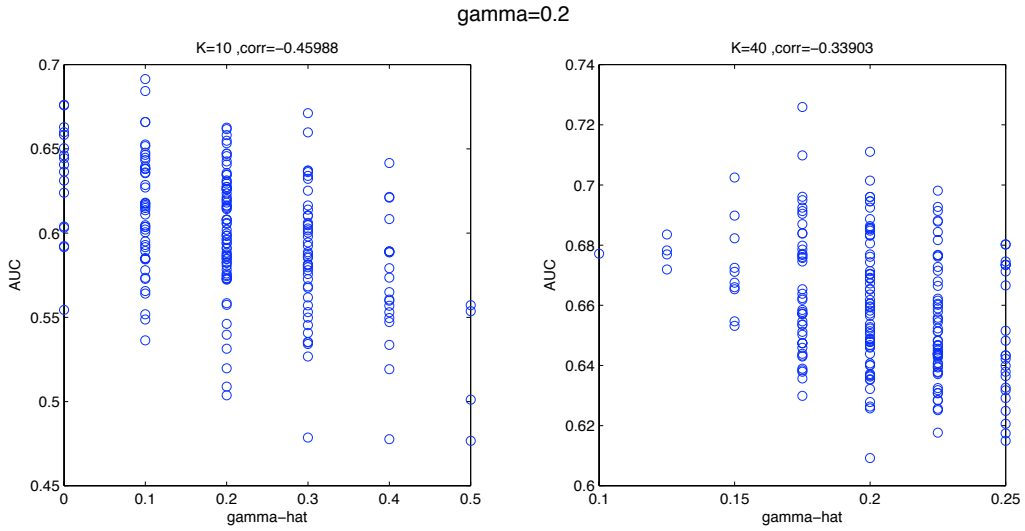


Figure 2.2: Distribution of AUC and $\hat{\gamma}$ over the 500 iterations of one bootstrap loop on the simulated dataset, $\gamma = 0.2$.

randomly selecting a given number of positive examples, while $\mathcal{U}$ contains the non-selected positive examples and all negative examples. We varied the size NP of $\mathcal{P}$ in $\{5, 10, 20, 50, 100, 200, 300\}$ to investigate the influence of the number of known positive examples. For each newsgroup and each value of NP, we train all 4 methods described above (bagging SVM, biased SVM, baseline, one-class SVM) and rank the samples in $\mathcal{U}$ by decreasing score (transductive setting). We then compute the area under the ROC curve (AUC), and average this measure over 10 replicates of each newsgroup and each value of NP. For bagging and biased SVM, we varied the C parameter over the grid $[\exp(-12 : 2 : 2)]$, while we vary parameter ν in $[0.1 : 0.1 : 0.9]$ for 1-class SVM. We only used the linear kernel.

We first investigated the influence of T. Figure 2.3 shows, for the first newsgroup, the performance reached as a function of T, for different settings in NP and K. As expected we observe that in general the performance increases with T, but quickly reaches a plateau beyond which additional bootstraps do not improve performance. Overall the smaller K, the larger T must be to reach the plateau. From these preliminary results we set $T = 35$ for $K \leq 20$, and $T = 10$ for $K > 30$, and kept it fix for the rest of the experiments. To further clarify the benefits of bagging, we show in Figure 2.6.2 the performance of the bagging SVM versus the performance of a SVM trained on a single bootstrap sample ($T = 1$), for different values of K and

a fixed number of positives $NP = 10$. We observe that, for K below 200, aggregating classifiers over several bootstrap subsamples is clearly beneficial, while for larger values of K it does not really help. This is coherent with the observation that SVM usually rarely benefit from bagging: here the benefits come from our particular bagging scheme. Interestingly, we see that very good performance is reached even for small values of K with the bagging.

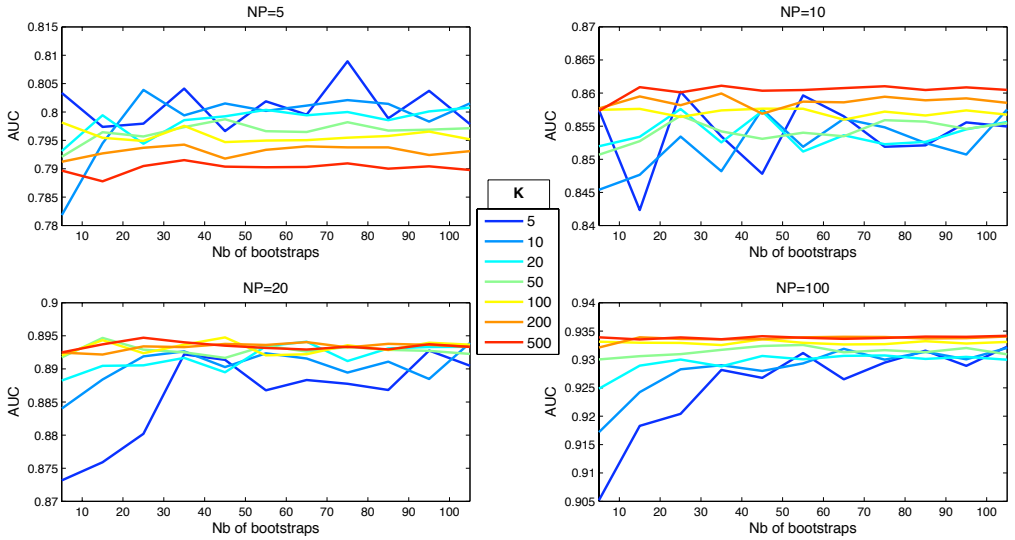


Figure 2.3: Performance on one newsgroup as a function of the number of boostraps T, for different values of NP and K.

Figure 2.5 shows the mean AUC averaged over the 10 folds and the 20 newsgroups for bagging SVM as a function of K, and compares it to that of the biased SVM. More precisely, each point on the curve corresponds to the performance averaged over the 20 Newsgroups after choosing a posteriori the best C parameter for each newsgroup. This is equivalent to comparing optimal cases for both methods. Contrary to what we observed on simulated date, we observe that K has in general very little influence on the performance. The AUC of the bagging SVM is similar to that of the biased SVM for most values of K, although for NP larger than 50, a slight advantage can be observed for the biased SVM over bagging SVM when K is too small. We conclude that in practice, parameter K may not need to be finely tuned and we advocate to keep it moderate. In all cases, $K = NP$ seems to be a safe choice for the bagging SVM.

Finally, Figure 2.6 shows the average AUC over the 20 newsgroups for all four methods, as a function of NP. Overall all methods are very similar, with the Baseline slightly below the others. In details, the bagging SVM

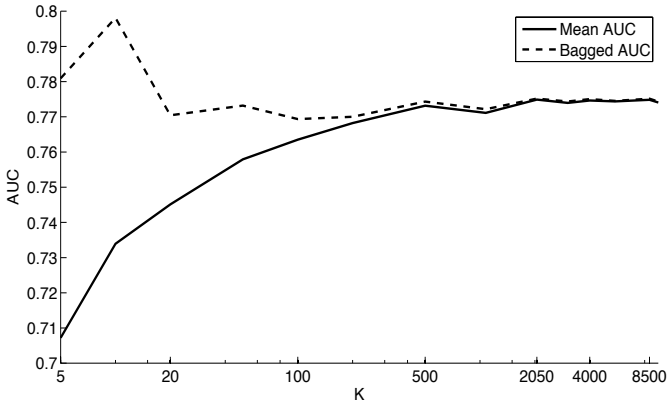


Figure 2.4: Performance on one newgroup of bagging SVM (*bagging AUC*) vs a SVM trained on a single bootstrap sample (*mean AUC*), for different values of K.

curve dominates all other methods for $NP \geq 20$, while the 1-class SVM is the one which dominates for smaller values of NP. Although the differences in performance are small, the bagging SVM outperforms the biased SVM significantly for $NP > 20$ according to a Wilcoxon paired sample test (at 5% confidence). For small values of NP however, no significant difference can be proven in either way between bagging SVM and 1-class SVM, which remains a very competitive method.

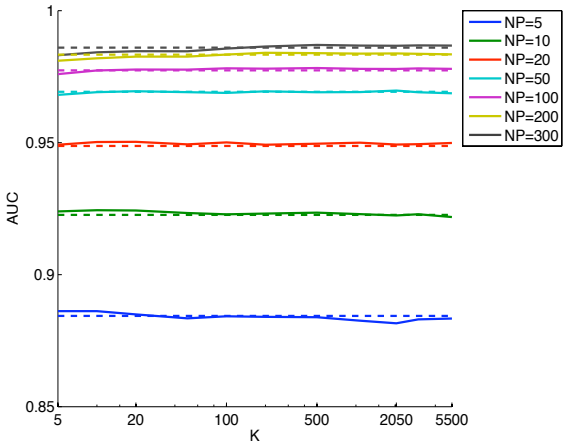


Figure 2.5: Macro averaged performance of the bagging SVM as a function of K. The dashed horizontal lines show the AUC level of the biased SVM. The curves are plotted for different values of NP, the size of the positive set.

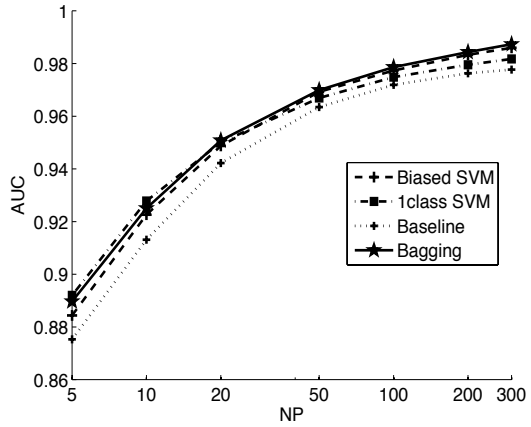


Figure 2.6: Performance of the baseline method, the 1-class SVM, the biased SVM and the newly proposed bagging SVM methods on the 20 Newsgroups dataset. Each curve shows how the mean AUC varies with the number of positive training examples NP. For each value of NP, the performance of bagging SVM is computed at the optimal value for K, as shown in Figure 2.5.

2.6.3 *E. coli* dataset : inference of transcriptional regulatory network

In this section we test the different PU learning strategies on the problem of inferring the transcription regulatory network of the bacteria *Escherichia coli* from gene expression data. The problem is, given a transcription factor (TF), to predict which genes it regulates. Following Mordelet and Vert [2008], we can formulate this problem as transductive PU learning by starting from known regulated genes (considered positive examples), and looking for additional regulated genes in the bacteria's genome.

To represent the genes, we use a compendium of microarray expression profiles provided by Faith et al. [2008], in which 4345 genes of the *E. Coli* genome are represented by vectors in dimension 445, corresponding to their expression level in 445 different experiments. We extracted the list of known regulated genes for each TF from RegulonDB [Salgado et al., 2006]. We restrict ourselves to 31 TFs with at least 8 known regulated genes.

For each TF, we ran a double 3-fold cross validation with an internal loop on each training set to select parameter C of the SVM (or ν for the 1-class SVM). Following Mordelet and Vert [2008], we normalize the expression data to unit norm, use a Gaussian RBF kernel with $\sigma = 8$, and perform a particular cross-validation scheme to ensure that operons are not split between folds.

Finally, following our previous results on simulated data and the newsgroup benchmark, we test two variants of bagging SVM, setting K successively to NP and $5 * NP$. These choices are denoted respectively by *bagging1 SVM* and *bagging5 SVM*.

Figure 2.6.3 shows the average precision/recall curves of all methods tested. Overall we observe that all three PU learning methods give significantly better results than the two methods which use only positive examples (Wilcoxon paired sample test at 5% significance level). No significant difference was found between the three PU learning methods. This confirms again that for different values of K bagging SVM matches the performance of biased SVM.

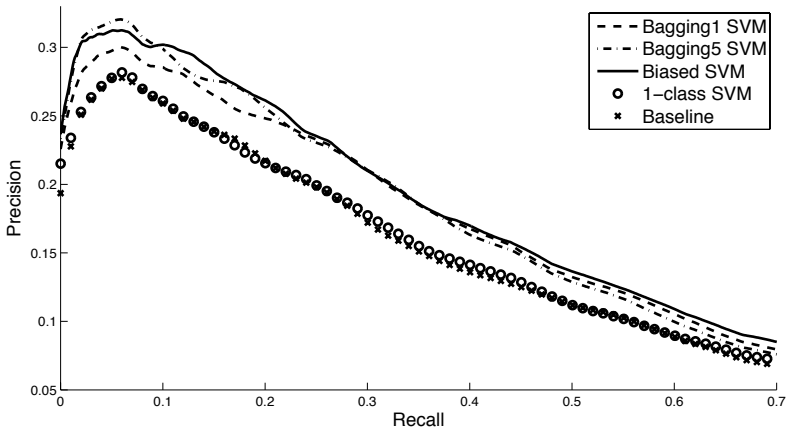


Figure 2.7: Precision-recall curves to compare the performance between the baggin1 SVM, the bagging5 SVM, the biased SVM, the 1-class SVM and the baseline method.

2.7 Discussion

The main contribution of this work is to propose a new method, bagging SVM, both for inductive and transductive PU learning, and to assess in detail its performance and the influence of various parameters on simulated and real data.

The motivation behind bagging SVM was to exploit an intrinsic feature of PU learning to benefit from classifier aggregation through a random subsample strategy. Indeed, by randomly sampling K examples from the unlabeled examples, we can expect various contamination rates, which in turn can lead to very different single classifiers (good ones when there is little contamina-

tion, worse ones when contamination is high). Aggregating these classifiers can in turn benefit from the variations between them. This suggests that K may play an important role in the final performance of bagging SVM, since it controls the trade-off between the mean and variance of individual classifiers. While we showed on simulated data that this is indeed the case, and that there can be some optimum K to reach the best final accuracy, the two experiments on real data did not show any strong influence of K and suggested that $K = NP$ may be a safe default choice. This is a good news since it does not increase the number of parameters to optimize for the bagging SVM and leads to balanced training sets that most classification algorithms can easily handle.

The comparison between different methods is mitigated. While bagging SVM outperforms biased SVM on simulated data, they are not significantly different on the two experiments with real data. Interestingly, while these PU learning methods were significantly better than two methods that learned from positive examples only on the gene regulatory network example, the 1-class SVM behaved very well on the 20 newsgroup benchmark, even outperforming the PU learning methods when less than 10 training examples were provided. Many previous works, including Liu et al. [2003] and Yu et al. [2004a] discard 1-class SVMs for showing a bad performance in terms of accuracy, while Manevitz and Yousef [2001] report the lack of robustness of this method arguing that it has proved very sensitive to changes of parameters. Our results suggest that there are cases where it remains very competitive, and that PU learning may not always be a better strategy than simply learning from positives.

Finally, the main advantage of bagging SVM over biased SVM is the computation burden, in particular when there are far more unlabeled than positive examples. Indeed, a typical algorithm, such as an SVM, trained on N samples, has time complexity proportional to N^α, with α between 2 and 3. Therefore, biased SVM has complexity proportional to $(P + U)^\alpha$ while bagging SVM's complexity is proportional $T * (P + K)^\alpha$. With the default choice $K = P$ ratio of CPU time to train the biased SVM vs the bagging SVM can therefore be expected to be $((P + U)/(2P))^\alpha / T$. Then we conclude that bagging SVM should be faster than biased SVM as soon as $U/P > 2T^{1/\alpha} - 1$. For example, taking $T = 35$ and $\alpha = 3$, bagging SVM should be faster than biased SVM as soon as $U/P > 6$, a situation very often encountered in practice where the ratio U/P is more likely to be several orders of magnitude larger. In the two real datasets, this was always the case. Table 2.7 reports CPU time and performance measure for training bagging SVM on the first fold of newsgroup 1 with C fixed at its best value a posteriori and $NP = 10$.

Bagging		CPU			AUC-AUP		
		K=10	K=50	K=200	K=10	K=50	K=200
T	35	13	39	91	0.921-0.531	0.917-0.524	0.902-0.518
	50	18	54	127	0.920-0.539	0.914-0.522	0.904-0.522
	200	72	170	473	0.918-0.539	0.910-0.528	0.904-0.511

Table 2.1: CPU time and performance measures for different settings of T and K for bagging SVM.

In comparison, the biased SVM's CPU time is 227s for AUC = 0.932 and AUP = 0.491. This confirms that for reasonable values of T and K, the bagging SVM is much faster than the biased SVM for a comparable performance.

Chapter 3

Regulatory network inference

This chapter presents the work which was published in Mordelet and Vert [2008].

3.1 Résumé

Le chapitre 3 traite le problème de l'inférence de réseaux de régulation. On rappelle que le comportement des cellules vivantes résulte d'un programme de régulation de l'expression des gènes qui implique des milliers de gènes. Elucider ces réseaux de régulation transcriptionnelle est donc essentiel pour comprendre comment la cellule fonctionne et peut s'avérer utile pour découvrir des cibles thérapeutiques nouvelles. Il existe plusieurs méthodes qui infèrent ces réseaux à partir de données d'expression des gènes. Une comparaison récente des ces méthodes sur un jeu de données de référence à grande échelle, a révélé que la plupart de ces méthodes existantes ne recouvre qu'un nombre limité de régulations connues à un niveau de précision raisonnable. Ici, nous proposons SIRENE, une nouvelle méthode supervisée pour l'inférence de réseaux de régulation à partir de données transcriptomiques. SIRENE décompose le problème global en un grand nombre de problèmes de classification binaire, dits locaux car chacun vise à identifier de nouveaux gènes cibles parmi les gènes indéterminés pour un facteur de transcription particulier. SIRENE ne nécessite aucune hypothèse sur les données et introduit un nouveau paradigme d'inférence biologiquement fondé : si un gène A est régulé par une gène B et qu'un gène A' est similaire au gène A, alors il est très probable que le gène A' soit aussi régulé par le facteur de transcription B. SIRENE est une méthode conceptuellement simple et efficace d'un point de vue calculatoire. Les tests effectués sur le jeu de données de référence pour la prédiction de régulations chez *E. coli* démontrent que SIRENE recouvre

environ 6 fois plus de régulations connues que les méthodes de l'état de l'art.

3.2 Introduction

Elucidating the structure of gene regulatory networks is crucial to understand how transcription factors (TF) regulate gene expression and allow an organism to regulate its metabolism and adapt itself to environmental changes. While high-throughput sequencing and other post-genomics technologies offer a wealth of information about individual genes, the experimental characterization of transcriptional cis-regulation at a genome scale remains a daunting challenge, even for well-studied model organisms. *In silico* methods that attempt to reconstruct such global gene regulatory networks from prior biological knowledge and available genomic and post-genomic data therefore constitute an interesting direction towards the elucidation of these networks.

Transcriptional cis-regulation directly influences the level of mRNA transcripts of regulated genes. Not surprisingly, many *in silico* methods have been proposed to reconstruct gene regulatory networks from gene expression data, produced at a fast rate by microarrays [Bansal et al., 2007]. Clustering gene expression profiles across different conditions identifies groups of genes with similar transcriptomic response, suggesting co-regulation within each group [Tavazoie et al., 1999]. Clustering methods are widely used, computationally efficient, but do not easily lead to the identification of regulators for a given set of genes. Some authors nonetheless have observed that identifying similarities, or more generally mutual information between the expression profiles of a TF and of a target gene is a good indicator of regulation [Butte et al., 2000, Faith et al., 2007]. When time series of gene expression data are available, other reverse-engineering methodologies can be applied to capture the interactions governing the observed dynamics. Different mathematical formalisms have been proposed to model such dynamics, including boolean networks [Akutsu et al., 2000] or ordinary or stochastic partial differential equations [Chen et al., 1999, Tegner et al., 2003, Gardner et al., 2003, Chen et al., 2005, Bernardo et al., 2005, Bansal et al., 2006]. Some authors have also attempted to detect causality relationships between gene expression data, be they time series or compendia of various experiments, using statistical methods such as Bayesian networks [Friedman et al., 2000]. These methods that estimate the regulatory network by fitting a dynamic or statistical model are often computationally and data demanding.

The comparison of these different approaches and of their capacity to accurately reconstruct large-scale regulatory networks has been hampered by the difficulty to assemble a realistic set of biologically validated regulatory

relationships and use it as a benchmark to assess the performance of each method. Recently, Faith et al. [2007] compiled such a benchmark, by gathering all known transcriptional cis-regulation in *Escherichia coli* and collecting a compendium of several hundreds of gene expression profiling experiments. They compared several approaches, including Bayesian networks [Friedman et al., 2000], ARACNe [Margolin et al., 2006], and the context likelihood of relatedness (CLR) algorithm, a new method that extends the relevance networks class of algorithms [Butte et al., 2000]. They observed that CLR outperformed all other methods in prediction accuracy, and experimentally validated some predictions. CLR can therefore be considered as state-of-the-art among methods that use compendia of gene expression data for large-scale inference of regulatory networks.

In this chapter we present SIRENE (Supervised Inference of REgulatory NEtworks), a new method to infer gene regulatory networks on a genome scale from a compendium of gene expression data. SIRENE differs fundamentally from other approaches in that it requires as inputs not only gene expression data, but also a list of known regulation relationships between TF and target genes. In machine learning terminology, the method is *supervised* in the sense that it uses a partial knowledge of the information we want to predict in order to guide the inference engine for the prediction of new information. The necessity to input some known regulations is not a serious restriction in many applications, as many regulations have already been characterized in model organisms, and can be inferred by homology in newly sequenced genomes. Known regulations allow us to use a natural induction principle to predict new regulations: if a gene A has an expression profile similar to a gene B known to be regulated by a given TF, then gene A is likely to be also regulated by the TF. The fact that genes with similar expression profiles are likely to be co-regulated has been used for a long time in the construction of groups of genes by unsupervised clustering of expression profiles. The novelty in our approach is to use this principle in a supervised classification paradigm. This inference paradigm has the advantage that no particular hypothesis is made regarding the relationship between the expression data of a TF and those of regulated genes. In fact, expression data for the TF are not even needed in our approach.

Many algorithms for supervised classification can be used to transform this inference principle into a working algorithm. We use in our experiments the support vector machine (SVM) algorithm, a state-of-the-art method for supervised classification. The idea to cast the problem of gene or protein networks inference as a supervised classification problem, using known interactions as inputs, has been recently proposed and investigated for the reconstruction of protein-protein interaction (PPI) and metabolic networks

[Yamanishi et al., 2004, Ben-Hur and Noble, 2005]. Bleakley et al. [2007] proposed a simple method where a local model is estimated to predict the interacting partners of each protein in the network, and all local models are then combined together to predict edges throughout the network. They showed that this method gave important improvement in accuracy compared to more elaborated methods on both the PPI and metabolic networks. Here we adapt this strategy for the reconstruction of gene regulatory networks. For each TF, we estimate a local model to discriminate, based on their expression profiles, the genes regulated by the TF from others genes. All local models are then combined to rank candidate regulatory relationships between TFs and all genes in the genome. SIRENE is conceptually simple, easy to implement, and computationally scalable to whole genomes because each local model only involves the training of a supervised classification algorithm on a few hundreds or thousands examples.

We test SIRENE on the benchmark experiment proposed by Faith et al. [2007], which aims at reconstructing known regulations within *E. coli* genes from a compendium of gene expression data. On this benchmark, SIRENE strongly outperforms the best results reported by Faith et al. [2007], with the CLR algorithm. For example, at a 60% true positive rate (precision), CLR identifies 7.5% of all known regulatory relationships (recall), while SIRENE has a recall of 44.5% at the same precision level using expression profiles.

3.3 System and Methods

3.3.1 SIRENE

SIRENE is a general method to infer new regulation relationships between known TF and all genes of an organism. It requires two types of data as inputs. First, each gene in the organism needs to be characterized by some data, in our case a vector of expression values in a compendium of expression profiles. Second, a list of known regulation relationships between known TF and some genes is needed. More precisely, for each TF, we need a list of genes known to be regulated by the TF, and if possible a list of genes known not to be regulated by it. Such lists can typically be constructed from publicly available databases of experimentally characterized regulation, e.g., RegulonDB for *E. coli* genes [Salgado et al., 2006]. While such databases usually do not contain informations about the absence of regulation, we discuss in Section 3.3.3 below how we generate negative examples.

When such data are available, SIRENE splits the problem of regulatory network inference into many binary classification subproblems, one subprob-

lem being associated to each TF. More precisely, for each TF, SIRENE trains a binary classifier to discriminate between genes known to be regulated and genes known not to be regulated by the TF, based on the data that characterize the genes (e.g., expression data). The rationale behind this approach is that, although we make no hypothesis regarding the relationship between the measured expression level of a TF and its targets, we assume that if two genes are regulated by the same TF then they are likely to exhibit similar expression patterns. In our implementation, we use a SVM to solve the binary classification problems (Section 3.3.2), but any other algorithm for supervised binary classification could in principle be used. Once trained, the model associated to a given TF is able to assign to each new gene, not used during training, a score that tends to be positive and large when it believes, based on the data that characterize the gene, that the gene is regulated by the TF. The final step is to combine all scores of the different models to rank the candidate TF-gene interactions in a unique list by decreasing score.

In summary, SIRENE decomposes the difficult problem of gene regulatory network inference into a large number of subproblems that attempt to estimate local models to characterize the genes regulated by each TF. A similar approach was proposed by Bleakley et al. [2007] to infer undirected graphs, and successfully tested on the reconstruction of metabolic and PPI networks. Here we are confronted with a slightly different problem, since the graph we wish to infer is directed and we just need to infer local models to predict genes regulated by any given TF.

3.3.2 SVM

In our implementation of SIRENE, we use a SVM to train predictors for each local model associated to a TF. SVM is a popular algorithm to solve general supervised binary classification problems which is considered state-of-the-art in many applications and is available in many free and public implementations [Vapnik, 1998, Schölkopf et al., 2004]. The basic ingredient of a SVM is a kernel function $K(x, y)$ between any two genes x and y, that can often be thought of as a measure of similarity between the genes. In our case, the similarity between genes is measured in terms of expression profiles. Given a set of n genes $x_1, \ldots, x_n$ that belong to two classes, denoted arbitrarily -1 and $+1$, a SVM estimates a scoring function for any new gene x of the form:

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

The weights α_i in this expression are optimized by the SVM to enforce as much as possible large positive scores for genes in the class $+1$ and large

negative scores for genes in the class -1 in the training set. A parameter, often called C, allows to control the possible overfitting to the training set. The scoring function $f(x)$ can then be used to rank genes with unknown class by decreasing score, from the most likely to belong to class $+1$ to the most likely to belong to class -1.

The kernel $K(x, y)$ defines the similarity measure used by the SVM to build the scoring function. In our experiments we want to infer regulations from gene expression data. Each collection of gene expression data is a vector, so we simply use the common Gaussian radial basis function kernel between vectors u and v:

$$K(u, v) = \exp\left(-\frac{||u - v||^2}{2\sigma^2}\right),$$

where $\sigma > 0$ is the bandwidth parameter of the kernel.

Each SVM has therefore two parameters, C and σ. In order to limit the risk of overfitting and positive bias in our performance evaluation that could result from an over-optimization of these parameters on the benchmark data, we simply fix them for all SVM to the unique values $C = +\infty$ and $\sigma = 8$. The value $C = +\infty$ means that we train hard-margin SVM, which is always possible with a Gaussian kernel [Vapnik, 1998]. The choice $\sigma = 8$ was based on the observation that we use expression profiles for 445 microarrays scaled to zero mean and unit standard deviation, i.e., each gene is represented by a vector of dimension 445 and of length $\sqrt{445} \sim 21$. Hence the distance between two orthogonal profiles is of the order of $\sqrt{2} \times \sqrt{445} \sim 32$. We expect that a bandwidth of the order of $\sigma = 8$, which puts two orthogonal profiles at about 4σ from each other, is a safe default choice. We performed preliminary experiments with different values of C and σ, which did not result in any significant improvement or decrease of performance, suggesting that the behaviour of SIRENE is robust to variations in its parameters around these default values. All results below were obtained with this default parameter choice.

3.3.3 Choice of negative examples

SIRENE being a supervised inference algorithm, two sets of positive and negative training examples are needed for each SVM. Although regulations reported in databases such as RegulonDB can safely be taken as positive training examples, the choice of negative examples is more problematic for two reasons. First, few information is published and archived regarding the fact that a given TF is found not to regulate a given target gene. Hence there is no systematic source of negative examples for our problem. A natural choice is then to take TF-gene pairs not reported to have regulatory

relationships in databases as negative examples, mixing both true negative and false negative. In that case, we are then confronted with the second problem which is that, once a hard-margin SVM is trained on positive and negative examples, it always predict significantly negative scores on negative examples used during training. As a result it is not possible to use the SVM score on genes used during training if we want to find TF-pairs that were wrongly assigned to the negative class.

To overcome this issue, we propose the following scheme. Let us suppose we want to predict whether genes are regulated of not by a given TF. All genes known to be regulated by this TF form a set of positive examples, and no prediction is needed for them. The other genes are split in 3 subsets of roughly equal size. Then, in turn, each subset is taken apart, and a SVM is trained with all positive examples and all genes in the two other subsets as negative examples. The SIRENE score for the genes in the subset left apart is the SVM prediction score on these genes, which were not used during SVM training. Repeating this loop 3 times, we obtain the SIRENE score for all genes with no known regulation by the TF. This process is then repeated for all other TF one by one. The advantage of this procedure is that, even though there are false negative in the training set of each SVM, the predictions on the genes not used during training can still be positive if some of these genes look similar to the positive training examples.

3.3.4 CLR

We compare the performance of SIRENE with CLR, a method for gene network reconstruction from gene expression data that was shown by Faith et al. [2007] to be state-of-the-art on a large-scale benchmark evaluation. CLR an extension of the relevance networks class of algorithm [Butte et al., 2000], which predict regulations between TF and genes when important mutual information can be detected. In the case of CLR, an adaptive background correction step is added to the estimation of mutual information. For each gene, the statistical likelihood of the mutual information score is computed within its network context. Then, for each TF-target gene pair, the mutual information score is compared to the context likelihood of both the TF and the target gene, and turned into a z-score. Putative TF-gene interactions are then ranked by decreasing z-score.

3.3.5 Experimental protocol

In order to assess the performance of SIRENE as an inference engine, and compare it with other existing methods, we test it on a benchmark of known

regulatory network. However, SIRENE being a supervised method, we adopt a cross-validation procedure to make sure that its performance is measured on prediction not used during the model training step. Consequently we adopt the following 3-fold cross validation strategy, coherent with the SIRENE protocol to make predictions explained in Section 3.3.3. Given a set of TF, a set of genes, and a set of known TF-gene regulations within these sets, we split randomly the set of genes in 3 parts, train the SVM for each TF on two of these subsets, and evaluate their prediction quality on the third subset, i.e., on the regulations of those genes that were not used during training (Figure 3.1). This process is repeated 3 times, testing successively on each subset, and the prediction qualities of all folds are averaged.

In this cross-validation procedure, a particular attention must be paid to the existence of transcription units and operons in *E. coli*. Indeed, a given TF typically regulates all genes within an operons, which moreover usually have very similar expression profiles. As a result, if genes within an operon are split between a training and a test set, then the SVM prediction is likely to be correct simply because the SVM will predict that a test gene with a profile very similar to a training gene should be in the same class. In other words, the SVM can probably easily recognize operons and make correct predictions due to the presence of operons. However we are interested here in the prediction of inference of regulations for new operons. To simulate this problem in our cross-validation setting, we make sure that all genes that belong to the same operon are in the same subset of genes, i.e., are always either in the training set or in the test set together. In our experiments below we report results both an a classical cross-validation setting, and on this particular scheme that preserves the integrity of operons in the train/test splits.

The CLR algorithm is evaluated with the same protocol. However, since CLR is unsupervised, the training set is not used in each fold, and the final ROC and precision/recall curves are equivalently obtained by computing the curves on all genes simultaneously.

To evaluate the quality of a prediction we rank all possible TF-gene regulation in the test set by decreasing score, and compute both the receiving operating characteristic (ROC) curve and the precision/recall (PR) curve. The ROC curve plots the recall, i.e., the percentage true interactions that have a score above a threshold, as a function of the false positive rate, i.e., the fraction of negative interactions that have a score above a threshold, when the threshold varies. The PR curve plots the precision, i.e., the percentage of true positive among the predictions above a threshold, as a function of recall, when the threshold varies. One ROC and PR curve is obtained in each fold of cross-validation, and these curves are averaged over the three folds to yield

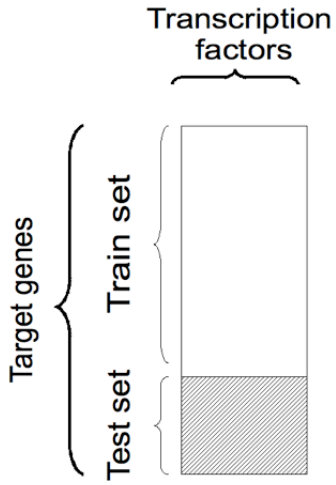


Figure 3.1: Cross validation for the transcriptional regulatory graph

the final estimated ROC and PR curve.

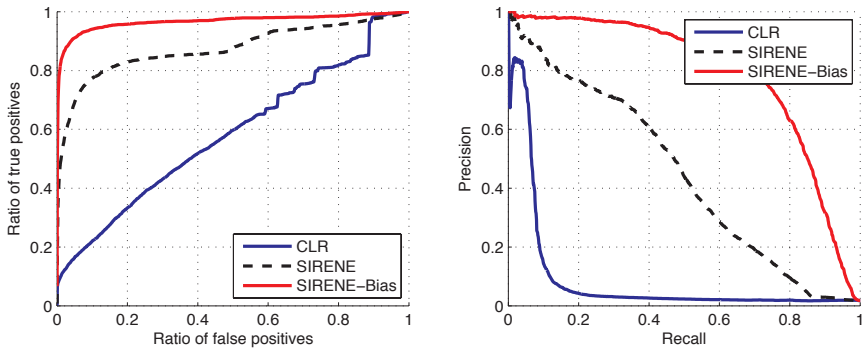
3.4 Data

We used in our experiments the expression and regulation data made publicly available by Faith et al. [2007] for *E. coli*, and downloaded from `http://gardnerlab.bu.edu/data/PLoS_2007/data_and_validation.html`. The expression data consist of a compendium of 445 *E. coli* Affymetrix Antisense2 microarray expression profiles for 4345 genes. The microarrays were collected under different experimental conditions such as PH changes, growth phases, antibiotics, heat shock, different media, varying oxygen concentrations and numerous genetic perturbations. The expression data for each gene were normalized to zero mean and unit standard deviation. The regulation data consist of 3293 experimentally confirmed regulations between 154 TF and 1211 genes, extracted from the RegulonDB database [Salgado et al., 2006].

We downloaded the list of 899 known operons in *E. coli* from RegulonDB. Each operon contains one or several genes, and each gene belongs to at most one operon. Genes not present in any of the regulonDB were considered to form an operon by themselves, resulting in a total of 3360 operons for the 4345 genes. This operon information was used to create the folds in the cross-validation procedure, as explained in Section 3.3.5.

3.5 Results

SIRENE was compared to CLR and other algorithms on the *E coli* benchmark used by Faith et al. [2007] and described in the previous section. Figure 3.2 shows the ROC and PR curves of CLR and SIRENE. The two curves for the later, labeled SIRENE and SIRENE-Bias, are respectively obtained when we use the cross-validation protocol presented in Section 3.3.5 and when we use a classical cross-validation scheme where genes within a known operon can be split between training and test sets.



(A) (B)

Figure 3.2: Comparison of performance between CLR and SIRENE. (A) ROC curves, and (B) precision/recall curves. The SIRENE curve corresponds to the SIRENE algorithm evaluated by 3-fold cross-validation, when genes within an operon are never split between the training and the test set. The SIRENE-bias curve is the same algorithm evaluated by classical 3-fold cross-validation, where genes are randomly assigned to training and test sets.

CLR scores were obtained directly from Faith et al. [2007]. The PR curve of CLR is similar to that presented by Faith et al. [2007], confirming that we use the exact same benchmark. Both for ROC and PR, SIRENE performance curves are significantly above CLR. SIRENE-bias is itself much better than SIRENE, confirming the importance of the evaluation bias if operons are split artificially between training and test sets in the cross-validation procedure. In what follows we restrict ourselves to the analysis of the results of SIRENE in the correct cross-validation protocal.

The PR curve is particularly relevant because the number of true regulations is very small compared to the total number of possible TF-gene pairs. We see that the recall obtained by SIRENE, i.e., the proportion of known regulations that are correctly predicted, is several times larger than the re-

call of CLR at all levels of precision. More precisely, Table 3.1 compares the recalls of SIRENE, CLR and several other methods at 80% and 60% precision. The other methods reported are relevance network [Butte et al., 2000], ARACNe [Margolin et al., 2006], and a Bayesian network [Friedman et al., 2000] implemented by Faith et al. [2007]. The performance of these three methods was taken directly from Faith et al. [2007].

Table 3.1: Recall of different gene regulation prediction algorithm at different levels of precision (60% and 80%). The values for relevance network, ARACNe and Bayesian network were taken from Faith et al. [2007].

Method	Recall at 60%	Recall at 80%
SIRENE	**44.5%**	**17.6%**
CLR	7.5%	5.5%
Relevance networks	4.7%	3.3%
ARACNe	1%	0%
Bayesian network	1%	0%

At 60% precision, SIRENE predicts 6 times more known regulations than CLR, which was the best among all methods tested on this benchmark by Faith et al. [2007]. With 44.5% recall at this precision level, the performance of SIRENE allows one, in principle, to retrieve almost half of all known regulations.

The main conceptual difference between SIRENE and other methods is that SIRENE is a supervised method that requires known regulations to train its models. As an attempt to understand why the performance of SIRENE was better than that of other state-of-the-art unsupervised methods, we reasoned that TF with a large number of known regulated target genes could better take advantage of the supervised setting, and therefore that predictions for these TF should in general be better than predictions for TF with few known targets. To validate this hypothesis, we computed the ROC curve for SIRENE by cross-validation, restricted to the prediction of targets for each individual TF in turn. For each TF, we then computed the area under the ROC curve (AUC) as an indicator of how well the targets of each particular TF are predicted. We did this estimation for both CLR and SIRENE, and show in Figure 3.3 the distributions of AUC scores for all TF as a function of the number of known target genes in RegulonDB, for both CLR and SIRENE. As expected, the values for SIRENE tend to be larger than those for CLR. More importantly, we observe in the SIRENE plot a trend to have better AUC values for TF trained on more known targets. This trend

is not present for CLR, which does not benefit from the knowledge of more or less targets for each TF. This result was expected and suggests that, as our knowledge expands and the number of known regulations continues to increase, so will the performance of supervised methods like SIRENE.

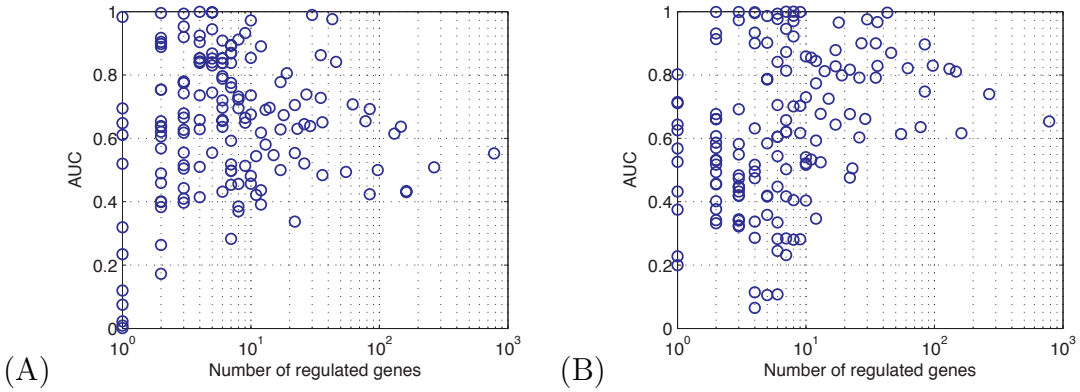


Figure 3.3: AUC per TF as a function of the number of regulated genes. (A) CLR and, (B) SIRENE

Having validated the relevance and performance of SIRENE on the regulonDB benchmark, we performed a global prediction of the *E. coli* regulatory network at 60% precision in order to predict new regulations in *E. coli*. More precisely, for each of the 154 TF with at least one known target in RegulonDB we computed the SIRENE score for all *E. coli* genes (4345 in total) that were not known targets, using the protocol described in Section 3.3.3. The RegulonDB database contained 3293 known TF-target regulations, so we assigned a score to the $4345 \times 154 - 3293 = 665837$ other candidate TF-gene pairs. From the cross-validation experiment we calibrated the level of SIRENE score threshold associated to various levels of precision. We selected all pairs with a score above a threshold of -0.41, corresponding to an estimated precision of 60%. At this threshold, 991 new regulations were predicted in addition to the 3293 known ones. Combining known and predicted regulations we obtained a regulatory network with 4284 edges involving 1688 genes.

In order to illustrate some predicted regulations, we focus now on the regulations of TF by other TF. Removing all non-TF genes of the predicted network, we obtain a graph with 131 TF and 349 interactions among them (TF with no interaction were removed). Among them, the *rpoD* gene, which codes for the RNA polymerase sigma factor, accounts alone for 85 regulations. In order to obtain a picture easier to visualize with the Cytoscape software

[Shannon et al., 2003], we removed rpoD from this graph, and only kept the main connected component which is shown in Figure 3.4. This core regulatory network involves 90 TF, and combines 196 known regulations among them with 32 predicted ones.

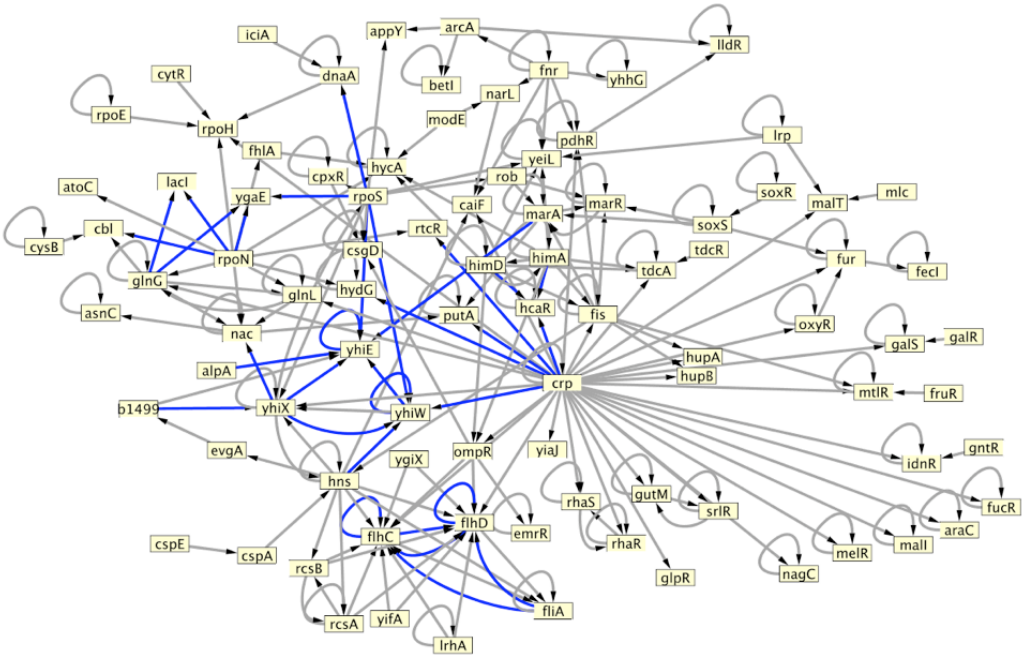


Figure 3.4: Main connected component of the predicted regulatory network among TF of *E. coli*, at an estimated 60% precision level. For clarity purpose the rpoD gene was removed from this picture. Grey arrows indicate known regulations, blue arrows indicate new predicted interactions.

Most regulations in this densely connected region of the *E. coli* regulatory network have been investigated in detail, and it not a surprise that the number of newly predicted regulations is limited. Still a quick survey of the literature can confirm some of these predictions. For example, four new regulators are predicted for *yhiW* (*crp,hns,rpoS,yhiX* and itself), which is itself predicted to regulate *yhiE*. Although these regulations were not present in the database used to train the model, they are confirmed by the literature. The GadW protein coded by *yhiW* is a regulator that participates in controlling several genes of the acid resistance system. It is indeed regulated by the proteins coded by *yhiX* and by the general proteins *crp,hns,rpoS* that control resistance to acidity through the gad system that utilizes two isoforms of glutamate decarboxylase encoded by gene regions *gadA* and *gadB* and a

putative glutamate:-aminobutyric acid antiporter encoded by *gadC* [Tucker et al., 2002, Waterman and Small, 2003, Ma et al., 2003]. Another predicted regulation that was confirmed by a literature search is the dependence of *hcaR*, a TF involved in the oxidative stress response, by a functional CAP protein encoded by the *crp* gene [Turlin et al., 2001]. Although preliminary, these first validations confirm the relevance of the approach and may suggest further experimental validations for subsystems of interest.

SIRENE is easy to implement and scales well to large-scale inference. Indeed, the main idea behind SIRENE is to decompose the network inference into a set of local binary classification problems, aimed at discriminating targets from non-targets of each TF. Although we used a SVM as a basic algorithm to solve these local problems, any algorithm for pattern recognition may be used instead. Each local problem involves at most a training set of a few thousands genes, easily manageable by most machine learning algorithms. This strategy also paves the way to the use of other genomic data to predict regulation. Indeed, local models for gene classification often improve in performance when several data, such as phylogenetic or cell subcellular localization information is available, and SVM provide a convenient framework to practically perform this data integration [Lanckriet et al., 2004a, Bleakley et al., 2007]. Another interesting features of SIRENE is its ability to predict self-regulation, that other methods have generally difficulties to deal with.

A important limitation of SIRENE is its inability to predict targets of TF with no *a priori* known target. More generally, the performance of SIRENE tends to decrease when few targets are known. Thus, for example, it can not be used to discover new transcription factors. An interesting direction of future research is therefore to extend the predictions to TF with no known target. A possible direction may be to combine the supervised approach with other non-supervised approaches in some meaningful way.

Chapter 4

Prioritization of disease genes with PU learning

4.1 Résumé

Le chapitre 4 est dédié à l'application de l'apprentissage à partir d'exemples positifs et indéterminés au problème de l'identification de gènes de maladies. On rappelle que retrouver les bases génétiques des maladies humaines est un des objectifs centraux de la génétique et de la biologie moléculaire. Les analyses de liaison traditionnelles et les techniques modernes à haut débit fournissent souvent de longues listes, de dizaines voire de centaines de gènes candidats. De ce fait, l'identification de gènes de maladies parmi ces candidats reste coûteuse et particulièrement chronophage. Comme nous l'avons souligné dans la section 1.1.2.2, des méthodes de calcul efficaces sont donc indispensables pour prioriser les gènes de la liste des candidats, tout en exploitant la mine d'informations disponibles sur ces gènes au travers des différentes bases de données génomiques. Nous proposons ProDiGe, un nouvel algorithme de priorisation de gènes de maladies. ProDiGe implémente une stratégie d'apprentissage statistique à partir d'exemples positifs et indéterminés, qui permet l'intégration de sources d'information hétérogènes. Par ailleurs, il permet de partager l'information sur les gènes causaux déjà identifiés entre différentes maladies et de rechercher de nouveaux gènes de maladie à l'échelle du génome. Les expériences, réalisées sur des données réelles, montrent que ProDiGe surpasse les méthodes existantes pour la priorisation de gènes de maladie chez l'humain.

4.2 Introduction

During the last decades, considerable efforts have been made to elucidate the genetic basis of rare and common human diseases. The discovery of so-called *disease genes*, whose disruption causes congenital or acquired disease, is indeed important both towards diagnosis and towards new therapies, through the elucidation of the biological bases of diseases. Traditional approaches to discover disease genes first identify chromosomal region likely to contain the gene of interest using, e.g., linkage analysis or study of chromosomal aberrations in DNA samples from large case-control populations. The regions identified, however, often contains tens to hundreds of candidate genes. Finding the causal gene(s) among these candidates is then an expensive and time-consuming process, which require extensive laboratory experiments. Progresses in sequencing, microarray or proteomics technologies have also facilitated the discovery of genes whose structure or activity are modified in disease samples, on a full genome scale. However, again, these approaches routinely identify long lists of candidate disease genes among which only one or a few are the causative agents of the disease process, and further biological investigations are required to identify them. In both cases, it is therefore important to select the most promising genes to be further studied among the candidates, i.e., to *prioritize* them from the most likely to be a disease gene to the less likely.

Gene prioritization is typically based on prior information we have about the genes, e.g., their biological functions, patterns of expression in different conditions, or interactions with other genes. The availability of complete genome sequences and the wealth of large-scale biological data sets now provide an unprecedented opportunity to speed up the gene hunting process [Giallourakis et al., 2005]. Integrating a variety of heterogeneous information stored in various databases and in the literature to obtain a good final ranking of hundreds of candidate genes is, however, a difficult task for human experts. Unsurprisingly many computational approaches have been proposed to perform this task automatically via statistical and data mining approaches. While some previous works attempt to identify promising candidate genes without prior knowledge of any other disease gene, e.g., by matching the functional annotations of candidate genes to the disease or phenotype under investigation [Perez-Iratxeta et al., 2002, Turner et al., 2003, Tiffin et al., 2005], many successful approaches assume that some disease genes are already known and try to detect candidate genes which share similarities with known disease genes for the phenotype under investigation [Freudenberg and Propping, 2002, Aerts et al., 2006, De Bie et al., 2007, Linghu et al., 2009,

Hwang and Kuang, 2010, Yu et al., 2010] or for related phenotypes [Freudenberg and Propping, 2002, Ala et al., 2008, Wu et al., 2008, Köhler et al., 2008, Vanunu et al., 2010, Hwang and Kuang, 2010]. These methods, which we review in more detail in Section 4.3 below, vary in the algorithm they implement and in the data they use to perform gene prioritization. For example, Endeavour and related work [Aerts et al., 2006, De Bie et al., 2007, Yu et al., 2010] use state-of-the-art machine learning techniques to integrate heterogeneous information and rank the candidate genes by decreasing similarity to known disease genes, while PRINCE [Vanunu et al., 2010] uses label propagation over a protein-protein interaction (PPI) network and is able to borrow information from known disease genes of related diseases to find new disease genes.

Here we propose ProDiGe, a new method for prioritization of disease genes, with brings several novelties compared to the state-of-the-art:

- Given a list of known disease genes, ProDiGe implements a machine learning algorithm to rank candidate genes using both the known and the candidate genes in the model building phase. This differs from approaches like those of Aerts et al. [2006], De Bie et al. [2007], Yu et al. [2010] which only use the known disease genes to build the scoring function, which is then used to rank the candidate genes. In the machine learning jargon, we formulate the problem as an instance of learning from positive and unlabeled examples (PU learning) [Liu et al., 2002, Denis et al., 2005, Mordelet and Vert, 2010], which brings improved performance by exploiting the relative functional similarity of both known and candidate disease genes.

- ProDiGe borrows information from genes known to play a role in phenotypes related to the disease of interest. This again differs from Aerts et al. [2006], De Bie et al. [2007], Yu et al. [2010], which can only process diseases one by one. It allows to rank genes even for diseases with no known gene by relying only on known disease genes of related diseases. In the machine learning jargon, we implement a *multi-task* strategy to share information between different diseases [Evgeniou et al., 2005], and weight the sharing of information by the phenotypic similarity of diseases.

- ProDiGe performs heterogeneous data integration to combine a variety of information about the genes in the scoring function, including sequence features, expression levels in different conditions, PPI interactions or presence in the scientific literature. We use the powerful

framework of *kernel methods* for data integration [Pavlidis et al., 2002, Schölkopf et al., 2004, Lanckriet et al., 2004a], akin to the work of Aerts et al. [2006], De Bie et al. [2007], Yu et al. [2010]. This differs from approaches like that of Vanunu et al. [2010], which are limited to scoring over a gene or protein network.

We test ProDiGe on real data extracted from the OMIM database [McKusick, 2007]. It is able to rank the correct disease gene in the top 5% of the candidate genes for 78% of the diseases with at least one other known disease genes, and for 67% of the diseases when no other disease genes is known, outperforming state-of-the-art methods like Endeavour and PRINCE.

The chapter is organised as follows. A first Section 4.3 is dedicated to a more extensive review of related work. In Section 4.4 we describe ProDiGe and its different variants. Section 4.5 summarizes the data we use, and Section 4.6 presents the results of our experiments. We conclude with discussion in Section 4.7.

4.3 Related work

The problem of disease gene prioritization, or gene hunting, has come into many different variants. A first series of approaches have been dedicated to the discovery of disease genes in general, i.e., genes responsible for at least one disease. Indeed these genes tend to share distinctive sequence features such as greater length of their amino acid sequence or broader phylogenetic extent [López-Bigas and Ouzounis, 2004, Calvo et al., 2007, Adie et al., 2005]. These methods, however, are designed to predict disease genes in general, but can not associate these genes with any particular disease, which is our goal in this chapter. In order to achieve this goal, many authors exploit the fact that for many diseases, genetic linkage studies have identified genomic loci which are highly susceptible to contain one or many of their causal genes [Vanunu et al., 2010, Ala et al., 2008, Adie et al., 2005, Turner et al., 2003, Franke et al., 2006, van Driel et al., 2003, Oti et al., 2008, 2006]. Though still leaving one with hundreds of genes to investigate, the prior knowledge of these loci greatly alleviates the identification task. A commonly adopted strategy consists of the two following steps. First, a score is attributed to each gene reflecting confidence that it is related to some disease. Then, for each disease, genes within its associated loci are prioritized by decreasing order of their score. For instance, Turner et al. [2003] proposed POCUS, a method based on co-occurrence of Gene Ontology (GO) terms in different loci to determine significant enrichment of IDs between loci relative to the genome.

In a similar spirit, Franke et al. [2006] attribute high scores to those genes that are functionally close to genes located on different loci (as measured by the distance on a functional linkage graph). Oti et al. [2008] compute a score measuring conserved co-expression to rank candidate disease genes. Nevertheless, such methods are not tailored to more difficult problems. For instance, it may occur that rare diseases have not yet been the object of any exhaustive study and that no region of interest is available for them. Besides, genetic linkage studies are often supported by statistical methods which are by nature error-prone and might miss positional candidates. In those cases, one needs a method which is able to scan efficiently the whole genome.

Another major distinction to be made between gene prioritization methods is whether or not they explicitly use known disease genes as guides to discover novel ones. Some methods, among which many early works, propose to infer disease genes *de novo*, that is, as if no disease gene had yet been discovered. A few methods mentioned above fall into this category [Oti et al., 2008, Turner et al., 2003, Franke et al., 2006, van Driel et al., 2003], which also includes the work of Perez-Iratxeta et al. [2002] who use a data mining technique based on fuzzy relation to compute association scores between a query disease and a gene. However, knowledge about disease genes has greatly increased recently, and most recent approaches are *supervised*, in the sense that they use as prior knowledge a training set of known disease genes. Candidate genes are then scored according to some similarity measure to this positive training set. Following this principle, Linghu et al. [2009] build a functional linkage network and prioritize genes according to their distance on this graph to training genes for a query disease, whereas Oti et al. [2006] prioritize interacting partners of known causing genes which are found inside a mapped loci lacking an identified causal gene.

As can be noticed from the previous paragraphs, methods also vary according to the type of information they use to predict disease genes. Most of the times, the underlying paradigm is that genes involved in a disease share common patterns at the functional level, which can be measured in several ways. Some procedures rely on functional annotation data, like GO for instance [Freudenberg and Propping, 2002, Turner et al., 2003, Franke et al., 2006]. However, it has been argued that such data were incomplete and biased towards better characterized genes, therefore poorly exploring unknown parts of the genome. Likewise, microarray expression data, though having a wider coverage, are prone to noise, which alters their ability to produce robust predictions. To counter this, other data sources supporting functional relatedness in a more robust way have been proposed such as protein-protein

interaction networks [Oti et al., 2006, Vanunu et al., 2010, Köhler et al., 2008, Wu et al., 2008] and co-expression measures [Oti et al., 2008, Ala et al., 2008]. Likewise, López-Bigas and Ouzounis [2004] and Adie et al. [2005] use sequence-based features to characterize disease genes. Nevertheless, using a single data source might still be unsufficient, which explains why mainstream research has been redirected towards integrative approaches [Aerts et al., 2006, Linghu et al., 2009, Hwang and Kuang, 2010]. For instance, the functional linkage network built by Linghu et al. [2009] incorporates multiple data sources by means of a naive Bayes classifier. Also, Aerts et al. [2006] generate several rankings from distinct heterogeneous data sources and eventually combines them into a single ranking using order statistics. Then, De Bie et al. [2007], followed by Yu et al. [2010] have developped a kernel-based approach allowing data fusion. Both use a one-class support vector machine (SVM) with multiple kernel learning to adjust the importance weights assigned to each data source in the learning process.

A major drawback of supervised approaches is that they cannot process diseases for which no causal gene has been discovered yet, which we refer to as "orphan" diseases. All methods that process diseases individually, trying to infer new causal genes from a training set of previously identified causal genes are termed as *local approaches* in the following. In contrast, we use the term *global approaches* to characterize the methods which try to learn causal genes simultaneously for a given set of diseases, and share disease gene information across diseases. In practice, this means that the different learning tasks associated to individual diseases are treated interdependently. To relate them, a widespread principle is to use a similarity measure between the different diseases. This time, the underlying paradigm goes a little further than before, stating that similar diseases are caused by similar molecular mechanisms and thus by functionally related genes. Freudenberg and Propping [2002] have pioneered this kind of methods to produce scores of association for disease/gene pairs, followed later by Ala et al. [2008] and Wu et al. [2008]. Köhler et al. [2008], Vanunu et al. [2010] and Hwang and Kuang [2010] implement a similar principle through label propagation on a graph. The two former methods use disease similarity to enforce some prior knowledge on a particular disease and then propagate labels on a PPI network. The latter have adapted a label propagation algorithm to deal with a heterogeneous network, whose nodes are diseases and genes.

At last, it is important to observe that from a machine learning point of view, most of the presented works relate to so-called *one-class* learning methods, in the sense that they only use a positive training set of genes

known to be related to some disease to build a scoring function, which is then applied on candidate genes to rank them. Since learning what a disease gene is from just a few examples characterized by many features is difficult from a statistical point of view, it could be tempting to go beyond this setting and try to learn instead what is *different* between a disease gene and a non-disease gene. While many powerful algorithms exist in machine learning to discriminate between two categories of objects, which could be disease and non-disease genes in our case, the reason why such methods have not been applied yet in the context of gene prioritization is the lack of negative examples. Indeed, it is usually not possible to determine that a gene is *not* involved in a particular disease, forcing prioritization algorithms to start from positive examples only. This issue was pointed out by López-Bigas and Ouzounis [2004] but not adressed. In this chapter, we put forward that although negative examples may not be available, the fact that we know in advance the candidate genes which have to be ranked allows us to exploit them as *unlabeled* examples during the learning process, and therefore to express the gene prioritization problem as a PU learning problem, instead of a one-class learning problem. The context of PU learning, also called *partially supervised classification* [Liu et al., 2002, Calvo et al., 2007], is the following: a "learner" is given a set of positive examples, defined as a set of data of interest (for instance, they might share some property or common features) and is asked to retrieve more of this kind of data from a set of unlabeled examples. Often, unlabeled data are easily available and numerous, whereas obtaining positive examples require human intervention or costly experiments, resulting in relatively small amounts. This information retrieval-like problem arises classically in areas such as text categorization and web page classification [Liu et al., 2002, Li and Liu, 2003, Liu et al., 2003, Yu et al., 2004a, Sriphaew et al., 2009, Pelckmans and Suykens, 2009, Elkan and Noto, 2008, Denis et al., 2005, Mordelet and Vert, 2010], and has recently been applied to the problem of predicting gene regulatory networks [Mordelet and Vert, 2008, 2010]. As far as we know, the only previous work which follows a similar viewpoint in the context of gene hunting is the work of Calvo et al. [2007]. However, their goal was different from ours since they aimed to retrieve disease genes in general, independently of which disease this might be. In our setting, we want to be able to relate a gene to the disease it is responsible for.

For a more extensive review of disease gene identificatin methods, we refer the reader to Tranchevent et al. [2010].

4.4 Methods

4.4.1 The gene prioritization problem

Let us first formally define the disease gene identification problem we aim to solve. We start from a list of N human genes $\mathcal{G} = \{G_1, \ldots, G_N\}$, which typically can be the full human genome or a subset of interest where disease genes are suspected. A multitude of data sources to characterize these genes are given, including for instance expression profiles, functional annotation, sequence properties, regulatory information, interactions, literature data, etc... We assume that for each data source, each gene $G \in \mathcal{G}$ is represented by a finite- or infinite-dimensional vector $\Phi(G)$, which defines an inner product $K(G, G') = \Phi(G)^\top \Phi(G')$ between any two genes G and G'. K is called a *kernel* in the machine learning community [Schölkopf and Smola, 2002]. Intuitively, $K(G, G')$ may be thought of as a measure of similarity between genes G and G' according to the representation defined by Φ. Since several representations are available, we assume that L feature vector mappings $\Phi_1, \ldots, \Phi_L$ are available, corresponding to L kernels for genes $K_1, K_2, \ldots, K_L$. Finally, we suppose given a collection of M disorders or disease phenotypes $\mathcal{D} = \{D_1, \ldots, D_M\}$. For each disorder D_i, the learner is given a set of genes $P_i \subset \mathcal{G}$, which contains known causal genes for this phenotype, and a set of candidate genes $U_i \subset \mathcal{G}$ where we want to find new disease genes for D_i. Typically U_i can be the complement set of P_i in $\mathcal{G}$ if no further information about the disease is available, or could be a smaller subset if a short list of candidate genes is given for the disease D_i. For each disease D_i, our goal is to retrieve more causal genes for D_i in U_i. In practice, we aim at ranking the elements of U_i from the most likely disease gene to the less likely, and assess the quality of a ranking by its capacity to rank the disease genes at or near the top of the list.

4.4.2 Gene prioritization for a single disease and a single data source

Let us first describe our gene prioritization approach ProDiGe for a single disease ($M = 1$) and a single data source ($L = 1$). In that case, we are given a single list of disease genes $P \subset \mathcal{G}$, and must rank the candidate genes in $U \subset \mathcal{G}$ using the kernel K. As explained in Section 4.3, most existing approaches define a scoring function $s : U \to \mathbb{R}$, using only positive examples in P, to quantify how similar a gene G in U is to the known disease genes in P. Here we propose to learn the scoring function $s(.)$ both from P and U, i.e., to formulate the problem as an instance of PU learning.

Intuitively, the motivation behind PU learning is to exploit the information provided by the distribution of unlabeled examples to improve the scoring function, as illustrated in Figure 4.4.2. Here we initially have a set of positive examples (genes known to be related to a given disease for instance) which are represented on the graph by blue crosses, and we want to retrieve more of them. One-class approaches usually try to estimate the support of the positive class distribution, which could be in that case delimited by the dashed line. Now suppose that we additionally observe a set of unlabeled examples, represented by U letters. Green Us are positive unlabeled and red ones are negative unlabeled but this information is not available. Then, we can have the feeling that the boundary should rather be set in the low density area as represented by the solid line, which better captures reality than the dashed line. Consequently, using the distribution of U in addition to the positive examples can help us better characterize the positive examples. This is particularly true in high dimension with few examples, where density estimation from a few positive examples is known to be particularly challenging.

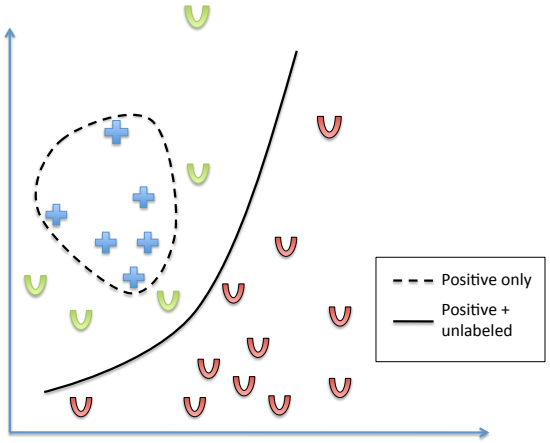


Figure 4.1: An intuitive example of how the unlabeled examples could be helpful.

In practice, a possible strategy to solve PU learning problems is to simply assign negative labels to elements in U, and train a binary classifier to discriminate P from U, allowing errors in the training labels. Assuming that the binary classifier assigns a score to each point during training (which is the case of, e.g., logistic regression or SVM), the score of an element in U is then just the scored assigned to it by the classifier after training. This

approach is easy to implement and it has been shown that building a classifier that discriminates the positive from the unlabeled set is a good proxy to building a classifier that discriminates the positive from the negative set. When the binary classifier used is a SVM, this approach leads to the biased SVM of Liu et al. [2002], which was recently combined with bagging to reach faster training time and equal performance [Mordelet and Vert, 2010]. In practice, the biased SVM over-weights positive examples during training to account for the fact that they represent high-confidence examples whereas the "negative" examples are to known to contain false negatives, namely, those we hope to discover. Here we use the variant of Mordelet and Vert [2010], which adds a bootstrap procedure to biased SVM. The additional bagging-like feature takes advantage of the contaminated nature of the unlabeled set, allowing to reach the same performances while increasing both speed and scalability to large datasets. The algorithm takes as input a positive and an unlabeled set of examples, and a parameter T specifying the number of bootstrap iterations. It discriminates the positive set from random subsamples of the unlabeled set and aggregates the successive classifiers into a single one (bootstrap aggregating). The output is a score function s such that for any example G, $s(G)$ reflects our confidence that G is a positive example. We then rank elements in U by decreasing score. For more details on the method, we refer the reader to Mordelet and Vert [2010].

4.4.3 Gene prioritization for a single disease and multiple data sources

When several data sources are available to characterize genes, e.g., gene expression profiles and sequence features, we extend our PU learning method to learn simultaneously from multiple heterogeneous sources of data through *kernel data fusion* [Lanckriet et al., 2004a]. Formally, each data source is encoded in a kernel function, resulting in $L \geq 1$ kernels $K_1, \ldots, K_L$. We investigate two strategies to fuse the L data sources:

- First, we simply define a new kernel which integrates the information contained in all kernels as the mean of the L kernels, i.e., we define:

$$K_{int} = \frac{1}{L} \sum_{i=1}^{L} K_i \,. \tag{4.1}$$

  In other words, the kernel similarity $K_{int}(G, G')$ between two genes is defined as the mean similarity between the two genes over the different data sources. This simple approach is widely used and often leads to

very good performance with SVM to learn classification models from heterogeneous information [Pavlidis et al., 2002, Yamanishi et al., 2004, Bleakley et al., 2007]. In our setting, we simply use the integrated kernel (4.1) each time a SVM is trained in the PU learning algorithm described in Section 4.4.2, to estimate a prioritization score from multiple data sources.

- Alternatively, we test a method for *multiple kernel learning (MKL)* proposed by Lanckriet et al. [2004b,a], which amounts to building a weighted convex combination of kernels of the form

$$K_{MKL} = \frac{1}{L} \sum_{i=1}^{L} \beta_i K_i \,, \tag{4.2}$$

where the non-negative weights β_i are automatically optimized during the learning phase of a SVM. By weighting differently the various information sources, the MKL formulation can potentially discard irrelevant sources or give more importance to gene characteristics with more predictive power. Again, combining MKL with our PU learning strategy described in Section 4.4.2 is straightforward: we simply use the MKL formulation of SVM instead of the classical SVM each time a SVM is trained.

4.4.4 Gene prioritization for multiple diseases and multiple data sources

When multiple diseases are considered, a first option is to treat the diseases independently from each other, and apply the gene prioritization strategy presented in Sections 4.4.2 and 4.4.3 to each disease in turn. However, it is known that disease genes share some common characteristics [López-Bigas and Ouzounis, 2004, Adie et al., 2005, Calvo et al., 2007], and that similar diseases are often caused by similar genes [Freudenberg and Propping, 2002, Ala et al., 2008, Wu et al., 2008, Köhler et al., 2008, Vanunu et al., 2010, Hwang and Kuang, 2010]. This suggests that, instead of treating each disease separately, it may be beneficial to consider them jointly and share information of known disease genes across diseases. By mutualizing information across diseases, one may in particular attempt to prioritize genes for orphan diseases, with no known causal gene. This is an important property since these diseases are obviously those for which predictions are the most needed.

We propose to jointly solve the gene prioritization problem for different diseases by formulating it as a *multitask* learning problem, and we adapt the multitask learning strategy of Evgeniou et al. [2005] to our PU learning framework. In this setting, instead of learning a scoring function over individual genes $G \in \mathcal{G}$ to rank them, we learn a scoring function over disease-gene pairs of the form $(D, G) \in \mathcal{D} \times \mathcal{G}$. Instead of starting from a set of positive examples $P \subset \mathcal{G}$ made of known disease genes for a particular disease, we start from a set of positive pairs $\big(D_{d(i)}, G_{g(i)}\big)_{i=1,\ldots,T} \subset \mathcal{D} \times \mathcal{G}$ obtained by combining the pairs where gene $G_{g(i)}$ is known to be a disease gene for disease $D_{d(i)}$. T is then the total number of known disease-gene pairs. Given the training set of disease-gene pairs, we then learn a scoring function s over $\mathcal{D} \times \mathcal{G}$ using our general PU learning algorithm described in Section 4.4.2, where the kernel function between two disease-gene pairs (D, G) and (D', G') is of the form:

$$K_{pair}\big((D, G), (D', G')\big) = K_{disease}(D, D') \times K_{gene}(G, G')\,. \tag{4.3}$$

In this equation, K_{gene} is a kernel between genes, typically equal to one of the kernels described in Sections 4.4.2 and 4.4.3 in the context of gene prioritization for a single disease. $K_{disease}$ is a kernel between diseases, which allows sharing of information across diseases, as in classical multitask learning with kernels [Evgeniou et al., 2005, Jacob and Vert, 2008a,b]. More precisely, we consider the following variants for K_{pair}, which give rise to various gene prioritization methods:

- The *Dirac kernel* is defined as

$$K_{Dirac}(D, D') = \begin{cases} 1 & \text{if } D = D', \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

  Plugging the Dirac kernel into (4.3), we see that the pairwise kernel between two disease-gene pairs for different diseases is 0. One can then show that there is no sharing of information across diseases, and that learning over pairs in this context boils down to treating each disease independently from the others [Evgeniou et al., 2005, Jacob and Vert, 2008a,b]. This is thus our baseline strategy, which treats each disease in turn, and does not provide a solution for orphan disease. We call this method ProDiGe1 below.

- The *multitask kernel* is defined by

$$K_{multitask}(D, D') = 1 + K_{Dirac}(D, D')\,. \tag{4.5}$$

This kernel, which was proposed by [Evgeniou et al., 2005], allows a basic sharing of information across diseases. In addition to the genes known to be causal for a disease of interest through the Dirac kernel, the addition of a constant in (4.5) allows all other known disease genes for other diseases to play the role of positive training examples, although to a lesser extent than the disease genes for the disease of interest. Here we do not use any specific knowledge about the different diseases and their similarity, and simply try to capture properties that may be shared by disease genes in general. This corresponds to a low information prior because a disease equally exploits knowledge about all other diseases. We call this variant ProDiGe2 below.

- The *phenotype* kernel is an attempt to capture phenotypic similarities between diseases to control the sharing of information across diseases. Indeed, many previous works have used as prior knowledge the fact that similar phenotypes are likely to be caused by similar genes [Freudenberg and Propping, 2002, Ala et al., 2008, Köhler et al., 2008, Lage et al., 2007, Wu et al., 2008, Vanunu et al., 2010, Hwang and Kuang, 2010]. This suggests that, instead of sharing information uniformly across diseases as the multitask kernel (4.5) does, it may be beneficial to do it in a more principled way. In particular, the more similar two diseases are, the more they should share information. In practice, this is obtained by defining a kernel $K_{phenotype}$ between diseases that measures their phenotypic similarity, and plugging it into the general pairwise kernel (4.3). Here we propose to use the phenotypic similarity measure for diseases based on text mining proposed by van Driel et al. [2006]. Since this measure is derived as a correlation measure, the matrix whose entries contain the pairwise similarity measures is eligible for kernel learning. We call the resulting gene prioritization method ProDiGe3 below.

- The *phenotype+Dirac* kernel. Finally, we propose a slight variant to the phenotype kernel by adding to it the Dirac kernel:

$$K_{P+D}(D, D') = K_{phenotype}(D, D') + K_{Dirac}(D, D')\,. \qquad (4.6)$$

The motivation for this kernel is that, since the description of disease phenotypes we use to build $K_{phenotype}$ is incomplete and does not fully characterize the disease, it may occur that two different diseases, with different disease genes, have similar or even identical phenotypic description. In this case, the addition of the Dirac kernel in (4.6) allows to still distinguish different diseases, and give more importance to the genes associated to the disease of interest than to the genes associated

to different diseases with similar phenotypes. We call ProDiGe4 the resulting gene prioritization method.

In summary, each of the four kernels for diseases presented above can be plugged into (4.3) to define a kernel for disease-gene pairs. Then, the PU learning strategy presented in Section 4.4.2 can be applied to learn a scoring function over $\mathcal{D} \times \mathcal{G}$. Finally, the ranking of candidate genes in U_i for a particular disease D_i is obtained by decreasing score $s(D_i, G)$ for $G \in U_i$. We thus obtain four variants summarized in Table 4.1. When heterogeneous

Table 4.1: Summary of ProDiGe variants. We propose four variants, which differ in the way they share information across diseases, as summarized in the third column of the table. The second column shows the kernel for diseases used by each variant to achieve the sharing of information. Apart from the choice of disease kernel, the four variants follow exactly the same procedure described in Section 4.4.4.

Name	Disease kernel	Sharing of disease gene information across diseases
ProDiGe1	K_{Dirac}	No.
ProDiGe2	$1 + K_{Dirac}$	Uniform sharing.
ProDiGe3	$K_{phenotype}$	Weighted by phenotypic similarity.
ProDiGe4	$K_{Dirac} + K_{phenotype}$	Weighted by phenotypic similarity and disease identity.

sources of information for genes are available, the two strategies proposed in Section 4.4.3 can be easily combined with each of the four ProDiGe variants, since each particular gene kernel translates into a particular disease-gene kernel through (4.3). In the experiments below, we only implement the MKL approach for ProDiGe1 to compare it to the mean kernel strategy. For other variants of ProDiGe, we restrict ourselves to the simplest strategy where the different information sources are fused through kernel averaging.

In practice, the PU learning strategy that underlies ProDiGe involves a SVM, which we ran using the libsvm implementation [Chang and Lin, 2001]. After observing in preliminary experiments that the regularization parameter C of the SVM did not dramatically affect the final performance, we set it constant to the default value $C = 1$ for all results shown below. The number of bootstrap iterations was set to 30.

4.5 Data

4.5.1 Gene features

The first type of data required by ProDiGe is the description of the set $\mathcal{G}$ of human genes. We used the dataset of De Bie et al. [2007], based on Ensembl v39 and which contains multiple data sources. We removed genes whose ID had a "retired" status in Ensembl v59, leaving us with 19540 genes. These genes are described by microarray expression profiles from Son et al. [2005] and Su et al. [2002] (MA1, MA2), expressed sequence tag data (EST), functional annotation (GO) , pathway membership (KEGG), protein-protein interactions from the Human Protein Reference Database (PPI), transcriptional motifs (MOTIF), protein domain activity from InterPro (IPR) and literature data (TEXT). For PPI data which consists in a graph of interactions, a diffusion kernel with parameter 1 was computed to obtain a kernel for genes [Kondor and Lafferty, 2002]. All other data sources provide a vectorial representation of a gene. The inner product between these vectors defines the kernel we create from each data source. All kernels are normalized to unit diagonal to ensure that kernel values are comparable between different data sources, using the formula:

$$\tilde{K}(G, G') \leftarrow \frac{K(G, G')}{\sqrt{K(G, G) \times K(G', G')}} \,. \tag{4.7}$$

4.5.2 Disease features

To define the phenotype kernel between diseases we borrow the phenotypic similarity measure of van Driel et al. [2006]. The measure they propose is obtained by automatic text mining. A disease is described in the OMIM database by a text record [McKusick, 2007]. In particular, its description contains terms from the Mesh (medical subject headings) vocabulary. van Driel et al. [2006] assess the similarity between two diseases by comparing the Mesh terms content of their respective record in OMIM. We downloaded the similarity matrix for 5080 diseases from the MimMiner webpage at `http://www.cmbi.ru.nl/MimMiner/MimMiner_Exp_AC_T_TXCS_basedonACMESH_filt_RW.mat.gz`.

4.5.3 Disease gene information

We collected disease-gene associations from the OMIM database [McKusick, 2007], downloaded on August 8th, 2010. We obtained 3222 disease-gene associations involving 2606 disorders and 2182 genes.

4.6 Results

4.6.1 Experimental setting

We assess the performance of various gene prioritization methods by leave-one-out cross-validation (LOOCV) on the dataset of known disease-gene association extracted from the OMIM database. Given the list of all disease-gene associations $(D_{d(i)}, G_{g(i)})_{i=1,\ldots,T}$ in OMIM, we remove each pair $(D_{d(i)}, G_{g(i)})$ in turn from the training set, train the scoring function from the $T-1$ remaining positive pairs, rank all genes G not associated to $D_{d(i)}$ in the training set by decreasing score $s(D_{d(i)}, G)$, and check how well $G_{g(i)}$ is ranked in the list. Note that in this setting, we implicitly assume that the candidate genes for a disease are all genes not known to be associated to the disease, i.e., $U_i = \mathcal{G} \backslash P_i$. In the LOOCV setting, each time a pair $(D_{d(i)}, G_{g(i)})$ is removed from the training set, the ranking is then performed on $U_{d(i)} \cup \{G_{g(i)}\}$. We monitor the success of the prioritization by the rank of $G_{g(i)}$ among candidate genes in $U_{d(i)}$. Since we are doing a LOOCV procedure, the rank of the left-out sample is directly related to the classical area under the Receiver Operating Characteristics curve (AUC), via the formula $rank = (|U| + 1) \times (1 - AUC)$. Therefore, an easy way to visualize the performance of a gene prioritization method is to plot the empirical cumulative distribution function (CDF) of the ranks obtained for all associations in the training set in the LOOCV procedure. For a given value of the rank k, the CDF at level k is defined as the proportion of associations $D_{d(i)}, G_{g(i)}$ for which gene $G_{g(i)}$ ranked among the top k in the prioritization list for disease $D_{d(i)}$, which can also be called the *recall* as a function of k.

We compare the proposed ProDiGe methods to two state-of-the-art gene prioritization methods. First we consider the 1-SVM L2-MKL from Yu et al. [2010], which extends and outperforms the Endeavour method [Yu et al., 2010], and which we denote MKL1class below. This method performs one-class SVM [Schölkopf et al., 2001] while optimizing the linear combination of gene kernels with a MKL approach in the same time. We downloaded a Matlab implementation of all functions from `http://homes.esat.kuleuven.be/~sistawww/bioi/syu/l2lssvm.html`. We used as input the same 9 kernels as for ProDiGe, and we set the regularization parameter of the algorithm $\nu = 0.5$, as done by Yu et al. [2010]. Second, we compare ProDiGe to the PRINCE method introduced by Vanunu et al. [2010], which is designed to share information across the diseases. Prior information consists in gene labels that are positive for genes known to be related to the query disease and zeros otherwise. PRINCE propagates these labels on a PPI network and pro-

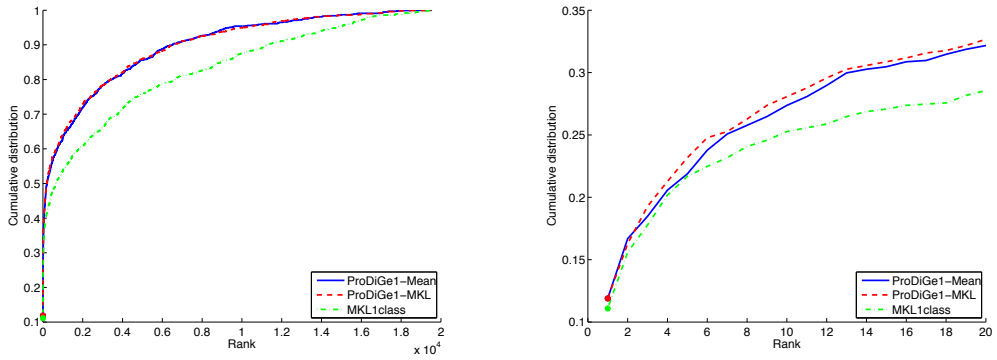
duces gene scores that vary smoothly over the network. We used the same PPI network for PRINCE as the one used by ProDiGe.

4.6.2 Gene prioritization without sharing of information across diseases

We first compare the ability to retrieve disease genes for the three local approaches which treat the diseases independently from each other, namely, the MKL1class method of Yu et al. [2010], and ProDiGe1 using either the mean kernel or the MKL strategy to integrate the 9 gene kernels. To be able to run a LOOCV procedure, these three algorithm were tested on the 285 diseases in our dataset having at least 2 known disease genes. Indeed, all three methods require at least one known disease disease gene for training, and for the purpose of LOOCV, we need in addition one known disease gene removed from the training set.

Figure 4.2 presents the CDF of the rank of the left-out positive gene, i.e., the number of genes that were ranked in the top k genes of the list as a function of k, for each method. Note that the rank is always between 1 (best prediction) and $19540-|P|$, where $|P|$ is the number of genes known to be associated to the disease of interest. The right panel zooms on the beginning of this curve which corresponds to the distribution of small values of the rank. We see clearly that both ProDiGe variants outperform MKL1class in the sense that they consistently recover the hidden positive gene at a better rank in the list. A Wilcoxon signed rank test confirms these visual conclusions at 5% level with P-values $6.1e^{-29}$ and $8.8e^{-28}$, respectively, for the average and MKL variants of ProDiGe. This illustrates the benefits of formulating the gene ranking problem as a PU learning problem, and not as a 1-class learning one, since apart from this formulation both MKL1Class and ProDiGe1 use very similar learning engines, based on SVM and MKL.

Both ProDiGe1 variants recover roughly one third of correct gene-disease associations in the top 20 genes among almost 19540, i.e., in the top 0.1%. However, we found no significant difference between the mean and MKL variants of ProDiGe in this setting (P-value=0.619). This means that in this case, assigning equal weights to all data sources works as well as trying to optimize these weights by MKL. Supported by this result and by the fact that MKL is much more time-consuming than a SVM with the mean kernel, we decided to restrict our experiments to the mean kernel in the following experiments with other variants of ProDiGe.



(A) (B)

Figure 4.2: Cumulative distribution function of the rank for local methods, in the LOOCV experiment. ProDiGe1-Mean and ProDiGe1-MKL refer to the ProDiGe1 variant combined with the mean kernel or the MKL strategy to fuse heterogeneous gene information. (A) Global curves, (B) Zoom on the beginning of the curves.

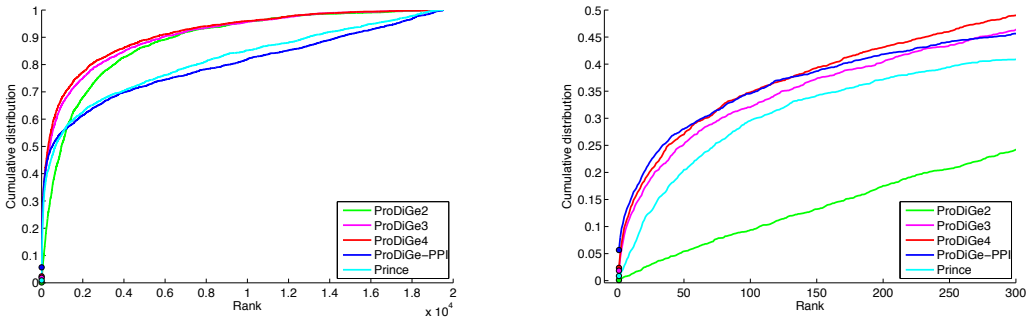
4.6.3 Gene prioritization with information sharing across diseases

In a second run of experiments, we assess the performance of methods which share information across diseases, namely, PRINCE and ProDiGe2,3,4. Since some of these methods require the phenotypic similarity between diseases, we limit ourselves to the 1873 diseases in the disease-gene association dataset which were also in the phenotypic similarity matrix from MimMiner. This corresponds to a total of 2544 associations between these diseases and 1698 genes.

Figure 4.3 shows the CDF curves for the four methods. Comparing areas under the global curve, i.e., the average rank of the left-out disease gene in LOOCV, the four methods can be ranked in the following order: ProDiGe4 (1682) > ProDiGe3 (1817) > ProDiGe2 (2246) > PRINCE (3065). The fact that ProDiGe3 and ProDiGe4 outperform ProDiGe2 confirms the benefits of exploiting prior knowledge we have about the disease phenotypes to weight the sharing of information across diseases, instead of following a generic strategy for multitask learning. The fact ProDiGe4 outperforms ProDiGe3 is not surprising and illustrates the fact that the diseases are not fully characterized by the phenotypic description we use. Zooming to the beginning of the curves (right picture), we see that the relative order between the methods is

conserved except for PRINCE which outperforms ProDiGe2 in that case. In fact, ProDiGe2 has a very low performance compared to all other methods for low ranks, confirming that the generic multitask strategy should not be pursued in practice if phenotypic information is available.

The fact that ProDiGe3 and ProDiGe4 outperform PRINCE for all rank values confirm the competitiveness of our approach. On the other hand, the comparison with PRINCE is not completely fair since ProDiGe exploits a variety of data sources about the genes, while PRINCE only uses a PPI network. In order to clarify whether the improvement of ProDiGe over PRINCE is due to a larger amount of data used, to the learning algorithm, or to both, we ran ProDiGe3 with only the kernel derived from the PPI network which we call ProDiGe-PPI in Figure 4.3. We see on the left picture that this variant is overall comparable to PRINCE (no significant difference between PRINCE and ProDiGe-PPI with a Wilcoxon paired signed rank test), confirming that the main benefit of ProDiGe over PRINCE comes from data integration. Interestingly though, at the beginning of the curve (right picture), ProDiGe-PPI is far above PRINCE, and even behaves comparably to the best method ProDiGe4. Since ProDiGe-PPI and PRINCE use exactly the same input data, this means that the better performance of ProDiGe-PPI for low ranks comes from the learning method based on PU learning with SVM, as opposed to label propagation over the PPI network.



(A) (B)

Figure 4.3: Cumulative distribution function of ranks in the LOOCV experiments, for global approaches which share disease gene information across diseases. ProDiGe2, 3, 4 refer to the three variants of ProDiGe which share information, while ProDiGe-PPI refers to ProDiGe3 trained only the PPI network data. (A) Global curve. (B) Zoom on the beginning of the curve.

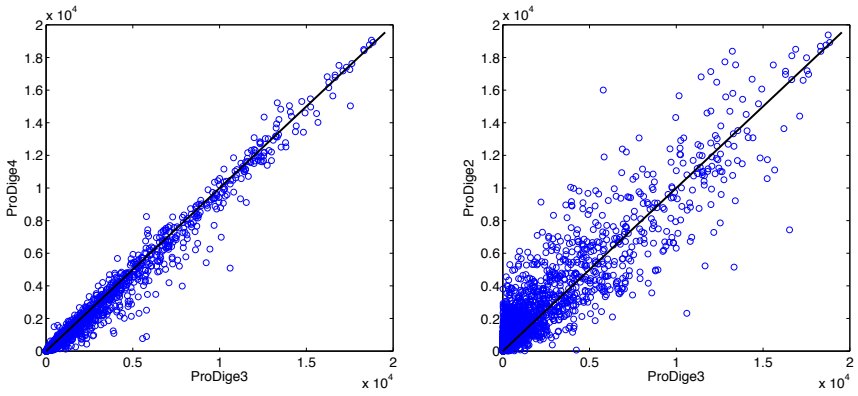


Figure 4.4: Comparison of rank measures between different variants of ProDiGe. Each point represent a disease-gene association. We plot the rank they obtain from the different methods when they are left out in the LOOCV procedure. Small rank are therefore better than large ranks.

To better visualize the differences between the different variants of ProDiGe, the scatter plots in Figure 4.4 compare directly the ranks obtained by the different variants for each of the 2544 left-out associations. Note that smaller ranks are better than large ones, since the goal is to be ranked as close as possible to the top of the list. On the left panel, we compare the Phenotype to the Phenotype+Dirac kernel. We see that many points are below the diagonal, meaning that adding a Dirac kernel to the Phenotype generally improves the performance as compared to using a Phenotype kernel alone. On the right panel, the Multitask kernel is compared to the Phenotype kernel. We see that the points are more concentrated above the diagonal, but with large variability on both sides of the diagonal. This indicates a clear advantage in favor of the Phenotype kernel compared to the Multitask kernel, although the differences are quite fluctuant.

4.6.4 Is sharing information across diseases beneficial?

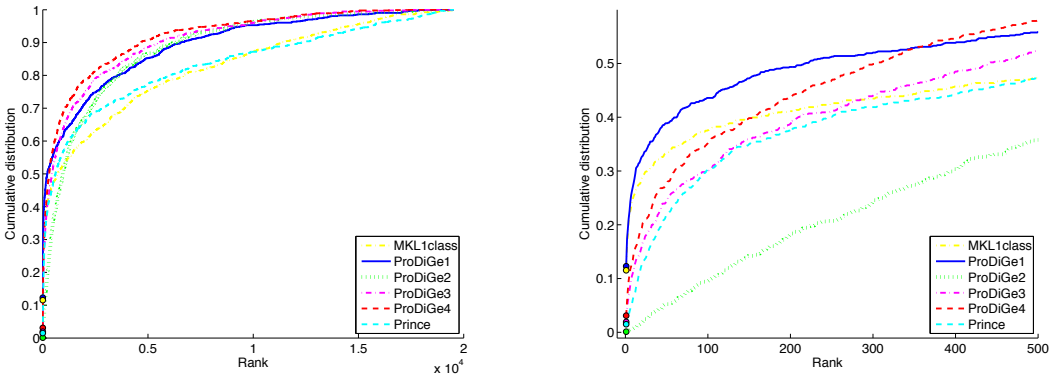
In order to check whether sharing information across diseases is beneficial, i.e., to compare the local approaches tested in Section 4.6.2 to the global approaches tested in Section 4.6.3, we restrict ourselves to diseases with phenotypic informations in the MimMiner similarity matrix and with at least two known associated genes in the OMIM database. This way, we are able to share information across diseases and at the same time, to run methods that do not share information because we ensure there is at least one train-

ing gene in LOOCV. This leaves us with 265 diseases, corresponding to 936 associations.

Figure 4.5 shows the CDF curves of the rank for the various methods on these data, including the two methods MKL1class and ProDiGe1 (with the mean kernel for data integration), which do not share information across diseases, and ProDiGe 2, 3, 4 and PRINCE, which do share information. Interestingly, we observe different retrieval behaviors on these curves, depending on the part of the curve we are interested in. On the one hand, if we look at the curves globally, ProDiGe 4 and 3 perform very well, having high area under the CDF curve, i.e., a low average rank (respectively 1529 and 1770). PRINCE and MKL1class have the worse average ranks (respectively 3220 and 3351). A systematic test of differences between the methods, using a Wilcoxon paired signed rank test over the ranks for each pair of methods, is summarized in Figure 4.6. In this picture, an arrow indicates that a method is significantly better than another at level 5%. This confirms that ProDiGe 4 is the best methods, significantly better than all other ones except ProDiGe 1. Three variants of ProDiGe are significantly better than PRINCE and MKL1Class.

On the other hand, in the context of gene prioritization, it is useful to focus on the beginning of the curve and not on the full CDF curves. Indeed, only the top of the list is likely to deserve any serious biological investigation. Therefore we present a zoom of the CDF curve in panel (B) of Figure 4.5. We see there that the local methods ProDiGe1 and MKL1class present a sharper increase at the beginning of the curve than the global methods, meaning that they yield higher concentration of disease genes in low rank values than other methods. Additionally, we observe that ProDiGe1 is in fact the best method when we focus on the proportion of disease genes correctly identified in up to the top 350 among 19540, i.e., in up to the top 1.8% of the list. These results are further confirmed by the quantitative values in Table 4.2, which show the recall (i.e., CDF value) as a function of the rank. ProDiGe 1, which does not share information across diseases, is the best when we only focus at the very top of the list (up to the top 1.8%), while ProDiGe 4, which shares information, is then the best method when we go deeper in the list.

At this point it is interesting to question what position in the list we are interested in. In classical applications where we start from a short list of, say, 100 candidates, then being in the top 5% of the list means that the correct gene is ranked in the top 5 among the 100 candidates, while the top 1% corresponds to the first of the list (see the last 3 columns of table 4.2). If we



(A) (B)

Figure 4.5: Cumulative distribution function of ranks for local and multitask approaches. (A) Global curve. (B) Zoom on the beginning of the curve.

only focus on the first gene of a short list of 100 candidates, then ProDiGe1 is the best method, with almost half of the genes (49.2%) found in the first position, followed by ProDiGe4 (43.4%) and MKL1class (41.1%). As soon as we accept to look further than the first place only, ProDiGe 4 is the best method, with 68.9% of disease genes in the top 5 of a list of 100 candidates, for example. On the other hand, if we consider a scenario where we start from no short list of candidates, and directly wish to predict disease genes among the 19540 human genes, then only the few top genes in the list are interesting (see the first 2 columns of table 4.2). In that case, the methods that do not share information are clearly preferable, with 27.8% (resp 25.3%) of genes correctly found in the top 10 among 19540 for ProDiGe 1 (resp. MKL1class).

In summary, sharing information is not beneficial if we are interested only in the very top of the list, typically the top 10 among 19.450 candidates. This setting is however very challenging, where even the best method ProDiGe1 only finds 12.3% of all disease genes. As soon as we are interested in more than the top 2% of the list, which is a reasonable level when we start from a short list of a few tens or hundreds of candidate genes, sharing information across diseases becomes interesting. In all cases, some variant of ProDiGe outperforms existing methods. In particular ProDiGe4, which shares information using phenotypic information across diseases while keeping different diseases distinct, is the best way to share information.

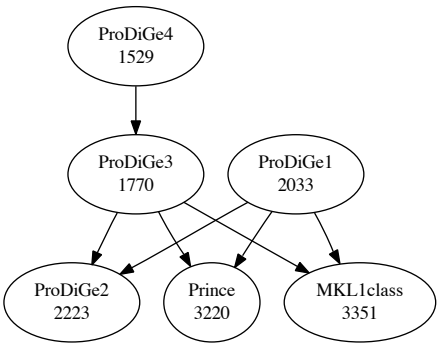


Figure 4.6: Wilcoxon paired signed rank tests for significant rank difference between all methods. ProDiGe1 and MKL1class are the only local approaches, which do not share information across diseases. The number in each ellipse is the average rank obtained by the method in the LOOCV procedure. An arrow indicates that a method is significantly better than another.

	top 1	top 10	top 1%	top 5%	top 10%
MKL1class	11.5	25.3	41.1	52.8	59.9
ProDiGe1	**12.3**	**27.8**	**49.2**	61.9	71.2
ProDiGe2	0.1	0.7	17.8	51.2	66.9
ProDiGe3	1.9	11.4	38.6	64.0	74.2
ProDiGe4	3.1	14.6	43.4	**68.9**	**78.4**
PRINCE	1.5	6.8	37.3	57.1	65.4

Table 4.2: Recall of different methods at different rank levels, for diseases with at least one known disease gene. The recall at rank level k is the percentage of disease genes that were correctly ranked in the top k candidate genes in the LOOCV procedure, where the number of candidate genes is near 19540. Top 1 and top 10 (first two columns) correspond respectively to the recall at the first and first ten genes among 19540, while top X% (last three columns) refer to the recall at the first X% genes among 19540.

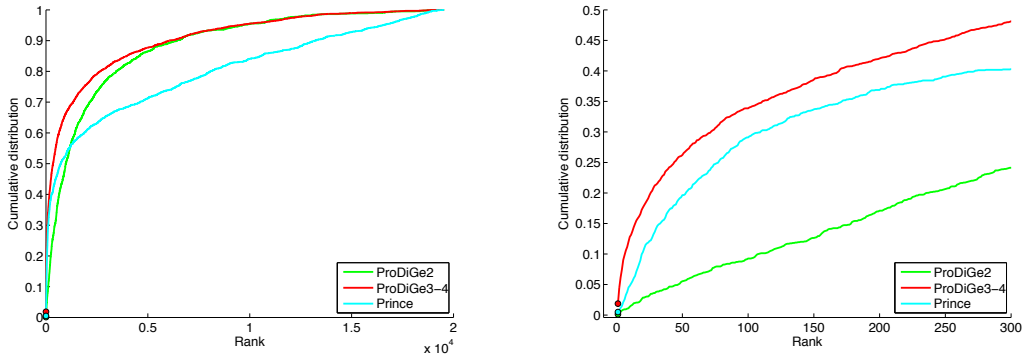
4.6.5 Predicting causal genes for orphan diseases

Finally, we investigate the capacity of the different gene prioritization methods to identify disease genes for orphan diseases, i.e., diseases with no known causative gene yet. ProDiGe1 and MKL1class, which treat diseases independently from each other and require known disease genes to find new ones, can not be used in this setting. Methods that share information across diseases, i.e., ProDiGe2, 3, 4 and PRINCE, can be tested in this context, since they may be able to discover causative genes for a given orphan diseases by learning from causative genes of other diseases. In fact, ProDiGe3 and ProDiGe4 boil down to the same method in this context, because the contribution of the Dirac kernel in (4.6) vanishes when no known disease gene for a disease of interest is available during training. We summarize them by the acronym ProDiGe3-4 below.

To simulate this setting, we start from the 1608 diseases with only one known disease gene in OMIM and phenotypic information in MimMiner, resulting in 1608 disease-gene associations involving 1182 genes. For each disease in turn, the associated gene is removed from the training set, a scoring function is learned from the associations involving other diseases, and the removed causal gene is ranked for the disease of interest. We compute the rank of the true disease gene, and repeat this operation for each disease in turn. Figure 4.7 and Table 4.3 show the performance of the different gobal methods in this setting. Interestingly, they are very similar to the results obtained in the multitask setting (Figure 4.3 and Table 4.2), both in relative order of the methods and in their absolute performance. Overall, ProDiGe3-4 performs best, retrieving the true causal gene in the top 10 genes of the list 13.1% of times, and in the top 5% of candidate genes 66.9% of times. This is only slightly worse than the performance reached for diseases with known disease genes (respectively 14.6% and 68.9%), highlighting the promising ability of global approaches to deorphanize diseases.

4.7 Discussion

We have introduced ProDiGe, a new set of methods for disease gene prioritization. ProDiGe integrates heterogeneous information about the genes in a unified PU learning strategy, and is able to share information across different diseases if wanted. We have proposed in particular two flavours for disease gene ranking: a local (ProDiGe1) and a global (ProDiGe4) approach. The local approach learns new causal genes for each disease separately, based



(A) (B)

Figure 4.7: Cumulative distribution function of ranks for prioritization of causal genes for orphan diseases. (A) Global curve. (B) Zoom on the beginning of the curve.

	top 1	top 10	top 1%	top 5%	top 10%
ProDiGe2	0.1	1.4	16.8	50.4	68.1
ProDiGe3-4	**1.9**	**13.1**	**42.7**	**66.9**	**76.1**
PRINCE	0.5	4.8	36.9	52.9	60.6

Table 4.3: Recall of different methods at different rank levels, for orphan diseases. In this case, since the disease has no known causal genes, only the causal genes of other diseases intervene in the learning, meaning that ProDiGe3 and 4 are equivalent approaches.

on already known causal genes for each disease, while the global approach additionally transfers information about known disease genes across different diseases, weighting information sharing by disease phenotypic similarity. We have demonstrated the efficiency of both variants on real data from the OMIM database where they outperform Endeavour and PRINCE, two state-of-the-art local and global gene prioritization methods.

A particularity of ProDiGe is the possibility to encode of prior knowledge on disease relatedness through the disease kernel. While a Dirac kernel prevents sharing of information across diseases, we tested different variants to share information including a generic multitask kernel and kernels taking into account the phenotypic similarity between diseases. We demonstrated the relevance of using the phenotypic similarity, compared to the generic mul-

titask kernel, and have enhanced it by the addition of a Dirac kernel. Given the influence of the disease kernel on the final performance of the method, we believe that there is still much room for improvement in the design of the prior, using the general ProDiGe framework. We note in particular that if other descriptors were available for phenotypes, one could also integrate these data and the prior they induce on task relatedness in the disease kernel.

A important question in practice is to choose an approach between the local (ProDiGe 1) and global (ProDiGe 4) ones. We have seen that the former has higher recall in the top 1 or 2% of the list, while the latter is better after. Hence a first criterion to chose among them is the rank level that we are ready to investigate. In addition, one could think that the local method ProDiGe1, which can not be used for orphan disease, is more generally handicapped compared to ProDiGe4 when the number of known disease genes is small, while it is in a better situation when many genes are already known. Indeed, if enough causal genes are known for a given disease, there is intuitively no need to borrow information from other diseases, which may mislead the prediction. This dependency of the relative performance of a local and a global approach on the number of training samples has previously been observed in other contexts [Jacob and Vert, 2008b] where a global approach was shown to bring tangible improvements over a local one when the number of positive examples was low. We have however checked for the presence of such an effect, and found that it could not be brought to light, as illustrated in Figure 4.7 which plots the mean and standard deviation of the rank of the left-out gene in LOOCV as a function of the number of known genes of the disease during training. We observe no trend indicating that the performance increases with the number of training genes, and no different behaviour between the local and multitask approaches, as long as at least one disease gene is known. This surprising finding, which is coherent with the observation there was no big difference in performance for orphan and non-orphan diseases, suggests that the number of known disease genes in not a relevant criterion to choose between the local and multitask version of ProDiGe. Instead, we suggest in practice to use the local version ProDiGe 1 if we are interested only in genes ranked in the very top of the candidate gene lists (below the top 1%), and ProDiGe 4 as soon as we can afford going deeper in the list.

Finally, except for the work of Calvo et al. [2007], the PU learning point of view on this long-studied gene prioritization problem is novel. Classical one-class approaches which learn a scoring function to rank candidate genes using known disease genes only are prone to over-fitting in large dimensions when

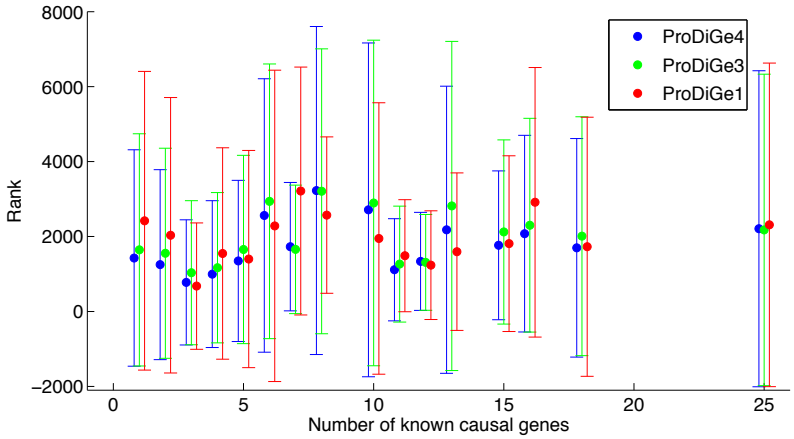


Figure 4.8: Effect of the number of related genes on the performance.

the training set if small, which results in poor performance. We observed that our PU learning strategy, augmented by a multitask point of view to share information across diseases, was useful to obtain better results in the disease gene identification task. In fact, learning from positive and unlabeled examples is a common situation in bioinformatics, and PU learning methods combined or not with multitask kernels have a good potential for solving many problems such as pathway completion, prioritization of cancer patients with a higher risk of relapse, or prediction of protein-protein or protein-ligand interactions.

Acknowledgments

We are very grateful to Yves Moreau, Léon-Charles Tranchevent and Shi Yu for providing the gene datasets and for the useful explanations and discussions that followed. We also acknowledge Roded Sharan and Oded Magger for making their Matlab implementation of PRINCE available to us.

Conclusion

In this thesis, we have highlighted the ubiquity of the problem of learning from positive and unlabeled examples in many research fields and in particular in biology. The main message is that using the unlabeled data and hence carrying out PU learning methods helps to reach better performances as compared to one-class methods which use positive examples only. Chapter 2 provided a comprehensive analysis of PU learning techniques. Besides, it presented a new PU learning algorithm which extends the state-of-the- art biasedSVM. Extensive experiments on a benchmark dataset have assessed its relevance to the PU learning problem and have provided a thorough analysis of its behavior in response to various parameters. The efficiency of PU learning was then demonstrated on two real problems. Chapter 3 dealt with the inference of regulatory networks while chapter 4 attacked the problem of identifying disease genes. In the former, treating unlabeled examples as negative examples, we were able to retrieve regulatory interactions with a high precision and recall. In the latter, the previously introduced algorithm was put into practice and was shown to outperform existing approaches for the prioritization of disease genes.

Concerning PU learning as a discipline, it is striking that the most successful techniques of the field generally amount to discriminating directly the positive set of examples against the unlabeled set. This approach might seem very simplistic at first sight and a natural idea, when no negative examples are available is indeed to try to identify some before applying a classical supervised method. There are many caveats to this approach. First, the identified set might not be exactly reliable resulting in a quick propagation of errors. Then, it might not be stable in the sense that a different training set would not yield the same set, which in turn would result in a highly unstable classifier. And last, another possibility is that the reliable set consists in negative examples that are too easily identified, therefore bringing little additional information. That probably explains why the many heuristic methods which rely on the identification of a set of reliable negative examples turn out

to be less accurate and very sensitive to the prior identification step. Overall, I have the feeling that the huge amounts of unlabeled data available in many situations are still under-exploited and that PU learning remains an immature area of machine learning research. As for our own contribution to this field, in chapter 2, we have introduced the baggingSVM. The main feature of this algorithm was the addition of a bagging-like procedure. Bagging a SVM classifier, and more generally a linear classifier, is known to be useless, not to say detrimental. However, our hope was that, due to the contaminated nature of the unlabeled set (which contains hidden positive examples), the increased variability of classifiers trained on bootstrapped subsamples would still enhance the performance of the biasedSVM. The experiments were not able to confirm this since the performance of both methods was found to be identical. However, the conclusion was that our algorithm presents clear advantages in terms of scalability and speed over the biasedSVM, while performing equivalently. Therefore, I believe that even though it did not have the expected effect, the heuristic justification we have brought is still valid to explain the good properties of the baggingSVM. An interesting practical consequence is that the PU learning setting allows an SVM on a very large number of points to be replaced by a reasonable number of SVMs on much smaller sets of points. However, this is solely supported by our intuition and empirical facts and it cruelly lacks proper theoretical grounds. Note that the bagging was justified in terms of classification error, whereas we would like to build a justification in terms of prioritization instead. Technically, the difficulty lies in the quantification the instability of the SVM score function and the establishment of a relationship to a theoretical measure of the prioritization performance (like the probability of correctly ranking a pair of examples, one of which has a positive label while the other is a negative example).

Chapter 3 introduced SIRENE, a new algorithm for the inference of regulatory networks. SIRENE decomposes the problem into a series of subproblems where the goal is to learn the regulated genes of a given TF. To do so, we proposed to discriminate the positive set of known targets against a subset of unlabeled examples. The inherent principle was that targets of the same TF must behave similarly. Of course, this a simple model of reality and one might argue for instance that inhibited targets obviously behave differently from activated targets. Therefore, if SIRENE is useful to discover the "big picture", the model needs further refinement to unravel more complex types of interactions. Likewise, people are often interested in discovering synergistic effects of a set of TFs and the nature of these effects. Another possible improvement direction is to extend SIRENE to make it able to pre-

dict targets for orphan TF. A straightforward way to do that is to resort to a multitask approach, very close to that we developped in chapter 4 for gene ranking. As it is, SIRENE does not exploit the data describing the TFs themselves. From these data, we could imagine building a similarity score (a kernel) between for TFs, such that similar TFs have similar targets. However, the TF kernel should be designed carefully so that it encodes a biologically meaningful prior on the information shared by two TFs. Incorporating prior knowledge is indeed a powerful idea but as we have shown in chapter 4, not all priors are actually efficient. For the disease identification problem though, we were able to improve performance by a slight modification of the disease kernel. This suggests that there might still be room for improvement in this direction. Furthermore, a current trend in the field of disease gene ranking is to try and find complexes of proteins or pathways involved the development of a disease rather than isolated genes [Lage et al., 2007, Vanunu et al., 2010]. This is an issue we are currently not able not adress with ProDiGe.

At last remark I would make is that the work of this thesis was largely motivated by the fact that it could help guide future experimental studies. Indeed, one could question the relevance of this work (and of many others), since it remains very focused on methodology and has not yet been subjected to any experimental validation. In a general perspective, the work of statisticians and machine learners might seem far too disconnected from reality and one could cast serious doubts on the practical utility of this type of research. In my opinion though, a first requirement before submitting any result to a biologist for further validation is to ensure that this is a high-confidence result. That is why I am convinced that the work of bionformaticians is useful, that it is worth spending time and efforts on methodology, yet keeping in mind the finality of this research.

Bibliography

E. A. Adie, R. R. Adams, K. L. Evans, D. J. Porteous, and B. S. Pickard. Speeding disease gene discovery by sequence based candidate prioritization. *BMC Bioinformatics*, 6:55, 2005. doi: 10.1186/1471-2105-6-55. URL http://dx.doi.org/10.1186/1471-2105-6-55.

S. Aerts, D. Lambrechts, S. Maity, P. Van Loo, B. Coessens, F. De Smet, L.-C. Tranchevent, B. De Moor, P. Marynen, B. Hassan, P. Carmeliet, and Y. Moreau. Gene prioritization through genomic data fusion. *Nat. Biotechnol.*, 24(5):537–544, May 2006. doi: 10.1038/nbt1203. URL http://dx.doi.org/10.1038/nbt1203.

T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for identifying Boolean networks and related biological networks based on matrix multiplication and fingerprint function. *J. Comput. Biol.*, 7(3-4):331–343, 2000. doi: 10.1089/106652700750050817. URL http://dx.doi.org/10.1089/106652700750050817.

U. Ala, R.M. Piro, E. Grassi, C. Damasco, L. Silengo, M. Oti, P. Provero, and F. Di Cunto. Prediction of human disease genes by human-mouse conserved coexpression analysis. *PLoS Comput Biol*, 4(3):e1000043, Mar 2008. doi: 10.1371/journal.pcbi.1000043. URL http://dx.doi.org/10.1371/journal.pcbi.1000043.

S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. A basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

R. Amato, A. Ciaramella, N. Deniskina, C. Del Mondo, D. di Bernardo, C. Donalek, G. Longo, G. Mangano, G. Miele, G. Raiconi, A. Staiano, and R. Tagliaferri. A multi-step approach to time series analysis and gene expression clustering. *Bioinformatics*, 22(5):589–596, Mar 2006. doi: 10.1093/bioinformatics/btk026. URL http://dx.doi.org/10.1093/bioinformatics/btk026.

B. Aranda, P. Achuthan, Y. Alam-Faruque, I. Armean, A. Bridge, C. Derow, M. Feuermann, A. T. Ghanbarian, S. Kerrien, J. Khadake, J. Kerssemakers, C. Leroy, M. Menden, M. Michaut, L. Montecchi-Palazzi, S. N. Neuhauser, S. Orchard, V. Perreau, B. Roechert, K. van Eijk, and H. Hermjakob. The intact molecular interaction database in 2010. *Nucleic Acids Res*, 38(Database issue):D525–D531, Jan 2010. doi: 10.1093/nar/gkp878. URL `http://dx.doi.org/10.1093/nar/gkp878`.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68: 337 – 404, 1950.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000. doi: 10.1038/75556. URL `http://dx.doi.org/10.1038/75556`.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 6, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1015330.1015424.

G.D. Bader, D. Betel, and C.W.V. Hogue. Bind: the biomolecular interaction network database. *Nucleic Acids Res*, 31(1):248–250, Jan 2003.

M. Bansal, G. Della Gatta, and D. Bernardo. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, 22(7):815–822, Apr 2006. doi: 10.1093/bioinformatics/btl003. URL `http://dx.doi.org/10.1093/bioinformatics/btl003`.

Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego di Bernardo. How to infer gene networks from expression profiles. *Mol Syst Biol*, 3:78, 2007. doi: 10.1038/msb4100120. URL `http://dx.doi.org/10.1038/msb4100120`.

P. L. Bartlett and A. Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. *J. Mach. Learn. Res.*, 8:775–790, 207. URL `http://jmlr.csail.mit.edu/papers/v8/bartlett07a.html`.

M. J. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. L. Wild. A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics*, 21(3):349–356, Feb 2005. doi: 10.1093/bioinformatics/bti014. URL `http://dx.doi.org/10.1093/bioinformatics/bti014`.

A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl. 1):i38–i46, Jun 2005. doi: 10.1093/bioinformatics/bti1016. URL `http://dx.doi.org/10.1093/bioinformatics/bti1016`.

D. Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. Eastwood, A. P. Wojtovich, S. J. Elliott, S. E. Schaus, and J. J. Collins. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat. Biotechnol.*, 23(3):377–383, Mar 2005. doi: 10.1038/nbt1075. URL `http://dx.doi.org/10.1038/nbt1075`.

K. Bleakley, G. Biau, and J.-P. Vert. Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(13):i57–i65, Jul 2007. doi: 10.1093/bioinformatics/btm204. URL `http://dx.doi.org/10.1093/bioinformatics/btm204`.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0. doi: http://doi.acm.org/10.1145/279943.279962. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.9164`.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM Press. URL `http://www.clopinet.com/isabelle/Papers/colt92.ps.Z`.

L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`.

A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac Symp Biocomput*, pages 418–429, 2000.

A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proc. Natl. Acad. Sci. USA*, 97(22):12182–12186, Oct 2000. doi: 10.1073/pnas.220392197. URL http://dx.doi.org/10.1073/pnas.220392197.

B. Calvo, N. López-Bigas, S.J. Furney, P. Larrañaga, and J.A. Lozano. A partially supervised classification approach to dominant and recessive human disease gene prediction. *Comput Methods Programs Biomed*, 85 (3):229–237, Mar 2007. doi: 10.1016/j.cmpb.2006.12.003. URL http://dx.doi.org/10.1016/j.cmpb.2006.12.003.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL http://www.kyb.tuebingen.mpg.de/ssl-book.

K.-C. Chen, T.-Y. Wang, H.-H. Tseng, C.-Y. F. Huang, and C.-Y. Kao. A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae. *Bioinformatics*, 21(12): 2883–2890, Jun 2005. doi: 10.1093/bioinformatics/bti415. URL http://dx.doi.org/10.1093/bioinformatics/bti415.

T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pac. Symp. Biocomput.*, pages 29–40, 1999.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20: 273–297, 1995. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1022627411411. 10.1023/A:1022627411411.

T. De Bie, L.-C. Tranchevent, L. M. M. van Oeffelen, and Y. Moreau. Kernel-based data fusion for gene prioritization. *Bioinformatics*, 23(13):i125–i132, Jul 2007. doi: 10.1093/bioinformatics/btm187. URL http://dx.doi.org/10.1093/bioinformatics/btm187.

S.E. Decatur. Pac learning with constant-partition classification noise and applications to decision tree induction. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 83–91, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. URL http://portal.acm.org/citation.cfm?id=645526.657273.

F. Denis. Pac learning from positive statistical queries. In *Proceedings of the 9th International Conference on Algorithmic Learning Theory*, ALT '98, pages 112–126, London, UK, 1998. Springer-Verlag. ISBN 3-540-65013-X. URL http://portal.acm.org/citation.cfm?id=647716.757188.

F. Denis, R. Gilleron, and M. Tommasi. Text classification from positive and unlabeled examples. In *Proc. of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.5291.

F. Denis, R. Gilleron, A. Laurent, and M. Tommasi. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.7101.

F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, 348(1):70–83, 2005. doi: http://dx.doi.org/10.1016/j.tcs.2005.09.007.

D. di Bernardo, M.J. Thompson, T.S. Gardner, S.E. Chobot, E.L. Eastwood, A.P. Wojtovich, S.J. Elliott, S.E. Schaus, and J.J. Collins. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat Biotechnol*, 23(3):377–383, Mar 2005. doi: 10.1038/nbt1075. URL http://dx.doi.org/10.1038/nbt1075.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, Dec 1998. URL http://www.pnas.org/cgi/reprint/95/25/14863.pdf.

C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, New York, NY, USA, 2008. ACM. doi: 10.1145/1401890.1401920. URL http://dx.doi.org/10.1145/1401890.1401920.

T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005. URL http://jmlr.csail.mit.edu/papers/volume6/evgeniou05a.

J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol.*, 5(1):e8, Jan 2007. doi: 10.1371/journal.pbio.0050008. URL http://dx.doi.org/10.1371/journal.pbio.0050008.

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 Res*, 36(Database issue):D866–D870, Jan 2008. doi: 10.1093/nar/gkm815. URL http://dx.doi.org/10.1093/nar/gkm815.

L. Franke, H. van Bakel, L. Fokkens, E.D D de Jong, M. Egmont-Petersen, and C. Wijmenga. Reconstruction of a functional human gene network, with an application for prioritizing positional candidate genes. *Am J Hum Genet*, 78(6):1011–1025, Jun 2006. doi: 10.1086/504300. URL http://dx.doi.org/10.1086/504300.

J. Freudenberg and P. Propping. A similarity-based method for genome-wide prediction of disease-relevant human genes. *Bioinformatics*, 18 Suppl 2:S110–S115, 2002.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *J. Comput. Biol.*, 7(3-4):601–620, 2000. doi: 10.1089/106652700750050961. URL http://dx.doi.org/10.1089/106652700750050961.

T. S. Gardner, D. Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, Jul 2003. doi: 10.1126/science.1081900. URL http://dx.doi.org/10.1126/science.1081900.

C. Giallourakis, C. Henson, M. Reich, X. Xie, and V. K. Mootha. Disease gene discovery through integrative genomics. *Annu. Rev. Genomics Hum. Genet.*, 6:381–406, 2005. doi: 10.1146/annurev.genom.6.080604.162234. URL http://dx.doi.org/10.1146/annurev.genom.6.080604.162234.

P. Glenisson, B. Coessens, S. Van Vooren, J. Mathys, Y. Moreau, and B. De Moor. Txtgate: profiling gene groups with text-based information. *Genome Biol*, 5(6):R43, 2004. doi: 10.1186/gb-2004-5-6-r43. URL http://dx.doi.org/10.1186/gb-2004-5-6-r43.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2001.

T. Hwang and R. Kuang. A heterogeneous label propagation algorithm for disease gene discovery. In *SDM*, pages 583–594, 2010.

L. Jacob and J.-P. Vert. Efficient peptide-MHC-I binding prediction for alleles with few known binders. *Bioinformatics*, 24(3):358–366, Feb 2008a. doi: 10.1093/bioinformatics/btm611. URL `http://dx.doi.org/10.1093/bioinformatics/btm611`.

L. Jacob and J.-P. Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008b. doi: 10.1093/bioinformatics/btn409. URL `http://bioinformatics.oxfordjournals.org/cgi/reprint/btn409`.

L.J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering. String 8–a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Res*, 37(Database issue):D412–D416, Jan 2009. doi: 10.1093/nar/gkn760. URL `http://dx.doi.org/10.1093/nar/gkn760`.

T. Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2.

Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, Nashville, Tennessee, USA, 1997. Morgan Kaufmann Publishers Inc.

M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic Acids Res.*, 32(Database issue):D277–80, Jan 2004. doi: 10.1093/nar/gkh063. URL `http://dx.doi.org/10.1093/nar/gkh063`.

M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Journal of the ACM*, pages 392–401, 1993. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.669`.

S. Köhler, S. Bauer, D. Horn, and P.N. Robinson. Walking the interactome for prioritization of candidate disease genes. *Am J Hum Genet*, 82(4):949–958, Apr 2008. doi: 10.1016/j.ajhg.2008.02.013. URL `http://dx.doi.org/10.1016/j.ajhg.2008.02.013`.

R. I. Kondor and J. Lafferty. Diffusion Kernels on Graphs and Other Discrete Input. In *ICML 2002*, 2002. URL http://www-2.cs.cmu.edu/~imk/diffusion-kernels.ps.gz.

J. M. Kwon and A. M. Goate. The candidate gene approach. *Alcohol Res Health*, 24(3):164–168, 2000.

K. Lage, E.O. Karlberg, Z.M. StÃÿrling, P.I. Olason, A.G. Pedersen, O. Rigina, A.M. Hinsby, Z. TÃijmer, F. Pociot, N. Tommerup, Y. Moreau, and S. Brunak. A human phenome-interactome network of protein complexes implicated in genetic disorders. *Nat Biotechnol*, 25(3):309–316, Mar 2007. doi: 10.1038/nbt1295. URL http://dx.doi.org/10.1038/nbt1295.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004a. doi: 10.1093/bioinformatics/bth294. URL http://bioinformatics.oupjournals.org/cgi/content/abstract/20/16/2626.

G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *J. Mach. Learn. Res.*, 5:27–72, 2004b. URL http://www.jmlr.org/papers/v5/lanckriet04a.html.

G.R.G. Lanckriet, N. Cristianini, M.I. Jordan, and W.S. Noble. Kernel-based integration of genomic data using semidefinite programming. In B. SchÃČÂűlkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 231–259. MIT Press, 2004c.

R.D. Leclerc. Survival of the sparsest: robust gene networks are parsimonious. *Mol Syst Biol*, 4:213, 2008. doi: 10.1038/msb.2008.52. URL http://dx.doi.org/10.1038/msb.2008.52.

W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003*, pages 448–455. AAAI Press, 2003.

F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.1122.

X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI'03: Proceedings of the 18th international joint conference*

on Artificial intelligence, pages 587–592, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

K-C. Liang and X. Wang. Gene regulatory network reconstruction using conditional mutual information. *EURASIP J Bioinform Syst Biol*, page 253894, 2008. doi: 10.1155/2008/253894. URL `http://dx.doi.org/10.1155/2008/253894`.

B. Linghu, E.S. Snitkin, Z. Hu, Y. Xia, and C. Delisi. Genome-wide prioritization of disease genes and identification of disease-disease associations from an integrated human functional linkage network. *Genome Biol*, 10 (9):R91, 2009. doi: 10.1186/gb-2009-10-9-r91. URL `http://dx.doi.org/10.1186/gb-2009-10-9-r91`.

B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 387–394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.9010`.

B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *Intl. Conf. on Data Mining*, pages 179–186, 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.5405`.

N. López-Bigas and C. A. Ouzounis. Genome-wide identification of genes likely to be involved in human genetic disease. *Nucleic Acids Res.*, 32(10): 3108–3114, 2004. doi: 10.1093/nar/gkh605. URL `http://dx.doi.org/10.1093/nar/gkh605`.

Z. Ma, S. Gong, H. Richard, D. L. Tucker, T. Conway, and J. W. Foster. GadE (YhiE) activates glutamate decarboxylase-dependent acid resistance in Escherichia coli K-12. *Mol. Microbiol.*, 49(5):1309–1320, Sep 2003.

L. M. Manevitz and M Yousef. One-class SVMs for document classification. *J. Mach. Learn. Res.*, 2:139–154, 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.6083`.

D. Marbach, C. Mattiussi, and D. Floreano. Replaying the evolutionary tape: biomimetic reverse engineering of gene networks. *Ann N Y Acad Sci*, 1158:234–245, Mar 2009. doi: 10.1111/j.1749-6632.2008.03944.x. URL `http://dx.doi.org/10.1111/j.1749-6632.2008.03944.x`.

A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1:S7, 2006. doi: 10.1186/1471-2105-7-S1-S7. URL http://dx.doi.org/10.1186/1471-2105-7-S1-S7.

V.A. McKusick. Mendelian inheritance in man and its online version, omim. *Am J Hum Genet*, 80(4):588–604, Apr 2007. doi: 10.1086/514346. URL http://dx.doi.org/10.1086/514346.

G.R. Mishra, M. Suresh, K. Kumaran, N. Kannabiran, S. Suresh, P. Bala, K. Shivakumar, N. Anuradha, R. Reddy, T.M. Raghavan, S. Menon, G. Hanumanthu, M. Gupta, S. Upendran, S. Gupta, M. Mahesh, B. Jacob, P. Mathew, P. Chatterjee, K.S. Arun, S. Sharma, K.N. Chandrika, N. Deshpande, K. Palvankar, R. Raghavnath, R. Krishnakanth, H. Karathia, B. Rekha, R. Nayak, G. Vishnupriya, H.G.M. Kumar, M. Nagini, G.S.S. Kumar, R. Jose, P. Deepthi, S.S. Mohan, GandhiT.K.B., H.C. Harsha, K.S. Deshpande, M. Sarker, T.S.K. Prasad, and A. Pandey. Human protein reference database–2006 update. *Nucleic Acids Res*, 34 (Database issue):D411–D414, Jan 2006. doi: 10.1093/nar/gkj141. URL http://dx.doi.org/10.1093/nar/gkj141.

F. Mordelet and J.-P. Vert. Sirene: Supervised inference of regulatory networks. *Bioinformatics*, 24(16):i76–i82, 2008. doi: 10.1093/bioinformatics/btn273. URL http://dx.doi.org/10.1093/bioinformatics/btn273.

F. Mordelet and J-P. Vert. A bagging SVM to learn from positive and unlabeled examples. Technical Report 00523336, HAL, October 2010. URL http://hal.archives-ouvertes.fr/hal-00523336.

K. Nigam, A.K. Mccallum, S. Thrun, and T.M. Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2/3):103–134, 2000. URL http://citeseer.ist.psu.edu/nigam99text.html.

M. Oti, B. Snel, M. A. Huynen, and H. G. Brunner. Predicting disease genes using protein-protein interactions. *J Med Genet*, 43(8):691–698, Aug 2006. doi: 10.1136/jmg.2006.041376. URL http://dx.doi.org/10.1136/jmg.2006.041376.

M. Oti, J. van Reeuwijk, M.A. Huynen, and H.G. Brunner. Conserved co-expression for candidate disease gene prioritization. *BMC Bioinformatics*,

9:208, 2008. doi: 10.1186/1471-2105-9-208. URL http://dx.doi.org/10.1186/1471-2105-9-208.

P. Pavlidis, J. Weston, J. Cai, and W.S. Noble. Learning gene functional classifications from multiple data types. *J. Comput. Biol.*, 9(2):401–411, 2002. doi: 10.1089/10665270252935539.

K. Pelckmans and J.A.K. Suykens. Transductively learning from positive examples only. In *Proc. of the European Symposium on Artificial Neural Networks (ESANN 2009)*, 2009. URL ftp://ftp.esat.kuleuven.be/pub/SISTA/kpelckma/esann09_ssl.pdf.

C. Perez-Iratxeta, P. Bork, and M. A. Andrade. Association of genes to genetically inherited diseases using data mining. *Nat. Genet.*, 31(3):316–319, Jul 2002. doi: 10.1038/ng895. URL http://dx.doi.org/10.1038/ng895.

J.J. Rice, Y. Tu, and G. Stolovitzky. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics*, 21(6):765–773, Mar 2005. doi: 10.1093/bioinformatics/bti064. URL http://dx.doi.org/10.1093/bioinformatics/bti064.

H. Salgado, S. Gama-Castro, M. Peralta-Gil, E. Díaz-Peredo, F. Sánchez-Solano, A. Santos-Zavaleta, I. Martínez-Flores, V. Jiménez-Jacinto, C. Bonavides-Martínez, J. Segura-Salazar, A. Martínez-Antonio, and J. Collado-Vides. RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Res.*, 34(Database issue):D394–D397, Jan 2006. doi: 10.1093/nar/gkj156. URL http://dx.doi.org/10.1093/nar/gkj156.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002. URL http://www.learning-with-kernels.org.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.*, 13:1443–1471, 2001.

B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, The MIT Press, Cambridge, Massachussetts, 2004.

C. Scott and G. Blanchard. Novelty detection: Unlabeled data definitely help. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*

2009, volume 5, pages 464–471, Clearwater Beach, Florida, 2009. JMLR: W&CP 5. URL http://jmlr.csail.mit.edu/proceedings/papers/v5/scott09a.html.

A. R. Shah, C. S. Oehmen, and B.-J. Webb-Robertson. SVM-HUSTLE–an iterative semi-supervised machine learning approach for pairwise protein remote homology detection. *Bioinformatics*, 24(6):783–790, Mar 2008. doi: 10.1093/bioinformatics/btn028. URL http://dx.doi.org/10.1093/bioinformatics/btn028.

P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13(11):2498–2504, Nov 2003. doi: 10.1101/gr.1239303. URL http://dx.doi.org/10.1101/gr.1239303.

L.M. Silver. *Mouse Genetics: Concepts and Applications*. Oxford University Press, 1995. URL http://www.informatics.jax.org/silver/.

C.G. Son, S. Bilke, S. Davis, B.T. Greer, J.S. Wei, C.C. Whiteford, Q-R. Chen, N. Cenacchi, and J. Khan. Database of mrna gene expression profiles of multiple human organs. *Genome Res.*, 15(3):443–450, Mar 2005. doi: 10.1101/gr.3124505. URL http://dx.doi.org/10.1101/gr.3124505.

K. Sriphaew, H. Takamura, and M. Okumura. Cool blog classification from positive and unlabeled examples. *Advances in Knowledge Discovery and Data Mining*, pages 62–73, 2009. URL http://dx.doi.org/10.1007/978-3-642-01307-2_9.

C. Stark, B-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Res*, 34(Database issue):D535–D539, Jan 2006. doi: 10.1093/nar/gkj109. URL http://dx.doi.org/10.1093/nar/gkj109.

I. Steinwart. Sparseness of Support Vector Machines. *J. Mach. Learn. Res.*, 4:1071–1105, 2003.

G. Stempfel. *Robustesse des séparateurs linéaires au bruit de classification*. PhD thesis, Universit‘e de Provence, 2009.

G. Stempfel and L. Ralaivola. Learning svms from sloppily labeled data. In Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas, editors, *Artificial Neural Networks âĂŞ ICANN 2009*, volume 5768 of

Lecture Notes in Computer Science, pages 884–893. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-04274-4_91.

A.I. Su, M.P. Cooke, K.A. Ching, Y. Hakak, J.R. Walker, T. Wiltshire, A.P. Orth, R.Q. Vega, L.M. Sapinoso, A. Moqrich, A. Patapoutian, G.M. Hampton, P.G. Schultz, and J.B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *Proc. Natl. Acad. Sci. U. S. A.*, 99(7): 4465–4470, Apr 2002. doi: 10.1073/pnas.012025199. URL http://dx.doi.org/10.1073/pnas.012025199.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nat. Genet.*, 1999. doi: doi:10.1038/10343. URL http://dx.doi.org/10.1038/10343.

J. Tegner, M. K. S. Yeung, J. Hasty, and J. J. Collins. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc. Natl. Acad. Sci. USA*, 100(10):5944–5949, May 2003. doi: 10.1073/pnas.0933416100. URL http://dx.doi.org/10.1073/pnas.0933416100.

N. Tiffin, J. F. Kelso, A. R. Powell, H. Pan, V. B. Bajic, and W. A. Hide. Integration of text- and data-mining using ontologies successfully selects disease gene candidates. *Nucleic Acids Res.*, 33(5):1544–1552, 2005. doi: 10.1093/nar/gki296. URL http://dx.doi.org/10.1093/nar/gki296.

LÃl'on-Charles Tranchevent, Francisco Bonachela Capdevila, Daniela Nitsch, Bart De Moor, Patrick De Causmaecker, and Yves Moreau. A guide to web tools to prioritize candidate genes. *Briefings in Bioinformatics*, 11, 2010. doi: 10.1093/bib/bbq007. URL http://bib.oxfordjournals.org/content/early/2010/03/21/bib.bbq007.abstract.

D. L. Tucker, N. Tucker, and T. Conway. Gene expression profiling of the ph response in escherichia coli. *J Bacteriol.*, 184(23):6551–6558, Dec 2002.

E. Turlin, M. Perrotte-piquemal, A. Danchin, and F. Biville. Regulation of the early steps of 3-phenylpropionate catabolism in Escherichia coli. *J. Mol. Microbiol. Biotechnol.*, 3(1):127–133, Jan 2001.

F. S. Turner, D. R. Clutterbuck, and C. A. M. Semple. Pocus: mining genomic sequence annotation to predict disease genes. *Genome Biol.*, 4 (11):R75, 2003. doi: 10.1186/gb-2003-4-11-r75. URL http://dx.doi.org/10.1186/gb-2003-4-11-r75.

M. van Driel, K. Cuelenaere, P.P.C.W. Kemmeren, J.A.M. Leunissen, and H.G. Brunner. A new web-based data mining tool for the identification of candidate genes for human genetic disorders. *Eur J Hum Genet*, 11 (1):57–63, Jan 2003. doi: 10.1038/sj.ejhg.5200918. URL `http://dx.doi.org/10.1038/sj.ejhg.5200918`.

M.A. van Driel, J. Bruggeman, G. Vriend, H.G. Brunner, and J.A.M. Leunissen. A text-mining analysis of the human phenome. *Eur J Hum Genet*, 14(5):535–542, May 2006. doi: 10.1038/sj.ejhg.5201585. URL `http://dx.doi.org/10.1038/sj.ejhg.5201585`.

O. Vanunu, O. Magger, E. Ruppin, T. Shlomi, and R. Sharan. Associating genes and protein complexes with disease via network propagation. *PLoS Comput Biol*, 6(1):e1000641, Jan 2010. doi: 10.1371/journal.pcbi.1000641. URL `http://dx.doi.org/10.1371/journal.pcbi.1000641`.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

R. Vert and J.-P. Vert. Consistency and convergence rates of one-class SVMs and related algorithms. *J. Mach. Learn. Res.*, 7:817–854, 2006. URL `http://jmlr.csail.mit.edu/papers/v7/vert06a.html`.

S.R. Waterman and P.L.C. Small. Transcriptional expression of escherichia coli glutamate-dependent acid resistance genes gada and gadbc in an hns rpos mutant. *J. Bacteriol.*, 185(15):4644–4647, Aug 2003.

X. Wu, R. Jiang, M.Q. Zhang, and S. Li. Network-based global inference of human disease genes. *Mol Syst Biol*, 4:189, 2008. doi: 10.1038/msb.2008.27. URL `http://dx.doi.org/10.1038/msb.2008.27`.

I. Xenarios, L. SalwÃ■nski, J. Duan, X, P. Higney, S-M Kim, and D. Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res*, 30(1):303–305, Jan 2002.

K. Xia, D. Dong, and J-D.J. Han. Intnetdb v1.0: an integrated protein-protein interaction network database generated by a probabilistic model. *BMC Bioinformatics*, 7:508, 2006. doi: 10.1186/1471-2105-7-508. URL `http://dx.doi.org/10.1186/1471-2105-7-508`.

Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20: i363–i370, 2004. URL `http://bioinformatics.oupjournals.org/cgi/reprint/19/suppl_1/i323`.

H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web page classification without negative examples. *IEEE Trans. Knowl. Data Eng.*, 16(1):70–81, 2004a. doi: 10.1109/TKDE.2004.1264823. URL `http://dx.doi.org/10.1109/TKDE.2004.1264823`.

J. Yu, V.A. Smith, P.P. Wang, A.J. Hartemink, and E.D. Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, Dec 2004b. doi: 10.1093/bioinformatics/bth448. URL `http://dx.doi.org/10.1093/bioinformatics/bth448`.

S. Yu, T. Falck, A. Daemen, L-C Tranchevent, Y.Ak. Suykens, B. De Moor, and Y. Moreau. L2-norm multiple kernel learning and its application to biomedical data fusion. *BMC Bioinformatics*, 11:309, 2010. doi: 10.1186/1471-2105-11-309. URL `http://dx.doi.org/10.1186/1471-2105-11-309`.

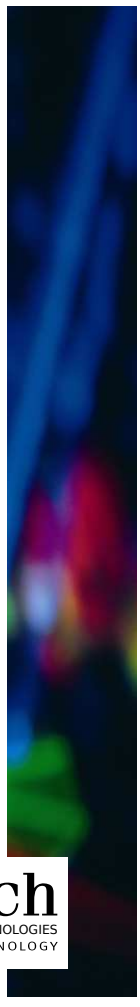
K. Zhang, I.W. Tsang, and J.T. Kwok. Maximum margin clustering made practical. 20(4):583–596, 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2010620.

Méthodes d'apprentissage statistique à partir d'exemples positifs et indéterminés en biologie

Résumé : En biologie, il arrive que les techniques de laboratoire soient inadaptées à la complexité du problème traité, pour des problèmes de temps et de coût. En parallèle, les biotechnologies permettent à présent de produire des données caractérisant une cellule à l'échelle du génome et sont porteuses d'espoir pour la compréhension de problèmes biologiques complexes, tels que l'identification d'interactions de régulation transcriptionnelle ou encore de gènes responsables du développement d'une maladie génétique. Dans les deux cas, le but est de prioriser une longue liste de gènes à tester expérimentalement pour dégager les candidats les plus prometteurs. Pour cela, on dispose d'un petit nombre d'exemples positifs (des interactions ou gènes de maladie identifiés) ainsi que de nombreux exemples indéterminés. Ce type de problème a été étudié en soi dans le domaine de l'apprentissage statistique. L'objet de cette thèse est l'étude de ces méthodes et leur application à des problèmes biologiques.
Mots clés : Apprentissage statistique, Exemples positifs et indéterminés, Réseaux de régulation transcriptionnelle, Gènes de maladies

Learning from positive and unlabeled examples in biology

Abstract: In biology laboratory techniques may be unsuitable to the complexity of the problem at hand due to cost issues. In parallel, biotechnologies now allow to generate high-throughput data which characterize a cell at the genome-scale, raising great expectations as for tackling complex biological problems such as transcriptional regulatory interaction inference and the identification of genes whose disruption triggers some genetically inherited disease. In both cases the goal is to prioritize a list of thousands of genes to determine the best candidates for further experimental study. In both cases positive examples exists in small amounts (confirmed interactions or identified disease genes) while there are plenty of unlabeled examples. The problem of learning from positive and unlabeled examples has been studied in itself in machine learning. The subject of this thesis is the study of such methods and their application to biological problems.
Keywords: Machine learning, Positive and unlabeled examples, Transcriptional regulatory networks, Disease genes



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY