Bayesian networks applied to the modeling of gene interactions

Francisco Sánchez Vega

Supervisors at CIS-JHU: ***Donald Geman and Laurent Younes***

Correspondant ENST/MVA: ***François Roueff***

Baltimore, August 2005

Abstract

Understanding the way in which genes interact among themselves to orchestrate the basic functions of life has become a main challenge of modern biology. The levels of expression of these genes provide a source of quantitative information for the study of these relations that has recently been made accessible to a wide scientific community through the development of microarray technologies. Within this context, Bayesian networks are a type of probabilistic graphical models that have been used to model gene interaction networks. Because of the very small sample regime inherent to microarray experiments, it remains unclear whether or not these models can actually provide a reliable description of the structures of statistical dependence among genes. In this report, we present a series of experiments and simulations, based on very simple networks and synthetic data, that attempt to provide some insight in this direction.

Acknowledgements

First of all, I want to thank professors Donald Geman and Laurent Younes, my two supervisors at The Johns Hopkins University, for all their help during the last months. From a scientific perspective, this piece of research would have never seen the light of day without their constant support and expert guidance. From a personal point of view, their trust, their patience and their kindness during this time have also earned them my most sincere expression of respect and admiration.

I also want to thank all the people at the Center for Imaging Science for their hospitality and for providing a perfect environment for my work. From the faculty members to the technical and administrative staff, and including fellow graduate students, they all welcomed me with a smile and they made me feel an important part of their team from the very first day. Muchas gracias.

Back in Europe, I want to thank professor François Roueff for accepting to be my internship tutor at Telecom Paris and for his helpfulness and availability at all the times that I needed to contact him.

This internship flags the end of a long period of studies that began in Madrid six years ago and that has allowed me to visit several countries and to meet extraordinary people all along the way. They are too many to be mentioned here, but they all share a space in my memory and my heart. Thank you for teaching me that nationalities are meaningless when it comes to real friendship.

Por último, quiero dar las gracias a mis padres, a mi hermano y, muy en especial, a mis abuelos, que no pudieron aguantar hasta el final, pero que siempre supieron que este día llegaría y que, con su cariño, su ternura y su amor contribuyeron a hacerlo posible.

Contents outline

After an introduction containing a brief description of the biological background and a discussion of previous works in the field, a first set of experiences studies the influence of several factors (such as the number of nodes, the choice of parameters for the conditional probability distributions and the presence of noise) over the required sample size to guarantee a certain level of reliability for the learning performance.

In a second section, several discrete procedures to explore the space of all possible directed acyclic graphs are considered and their performances are compared.

Our experiments conclude by considering the use of a model averaging approach based on the extraction of high-confidence features by bootstrapping from a given dataset, as proposed by Friedman et al. Once the experiments have been presented, some global conclusions are drawn and some ideas for future work are sketched.

Finally, two appendixes have been included to complement the contents of the main chapters:

1. A tutorial on Bayesian networks, that provides a reasonably comprehensive description of the basic concepts related to them.

2. A description of the Matlab implementation, that offers technical details about the algorithms used in the experiments and whose main goal is to enhace the reproducibility of all the simulations that are discussed in the text.

Contents

Chapter 1

Introduction

The contents of this report summarize the work that was carried out during my research internship at the Center for Imaging Science of The Johns Hopkins University, under the supervision of Professors Donald Geman and Laurent Younes.

This internship, which lasted for four months and began on April 18 2005, was the finishing line for my studies at the Double Degree in Telecommunications Engineering program from *Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid* (ETSIT-UPM) and *École Nationale Supérieure des Télécommunications de Paris* (ENST Paris). It also counted towards the completion of my Master of Research Degree in Applied Mathematics for Computer Vision and Artificial Learning from *École Normale Supérieure de Cachan*.

1.1 Educational context: The Johns Hopkins University

[The information provided in this section has been adapted from www.jhu.edu]

1.1.1 History, location and general information

The Johns Hopkins University was founded in Baltimore in 1876 as the first American university oriented toward graduate education and research.

The university is named for its initial benefactor, Baltimore merchant Johns Hopkins, whose $7 million bequest – the largest U.S. philanthropic gift to that time – was divided evenly to finance the establishment of both the university and The Johns Hopkins Hospital.

The university was born under the claim that American education needed a high intellectual superstructure to top off its system of broad public education. The initial idea was to follow the model of European institutions while taking advantage of all of the key elements that shaped up research in America. This lead to the establishment of a creative faculty that was given the freedom and support to pursue research, fellowships to attract the brightest students, graduate education emphasizing original work in laboratory and seminar, and scholarly publications.

Johns Hopkins also revolutionized the teaching and practice of medicine and medical research in the United States. With the opening of The Johns Hopkins Hospital in 1889, followed four years later by the School of Medicine, Hopkins ushered in a new era marked by rigorous entrance requirements for medical students, a vastly upgraded medical school curriculum with emphasis on the scientific method, the incorporation of bedside teaching and laboratory research as part of the instruction, and integration of the medical school with the hospital through joint appointments.

Teaching and research, the creation and dissemination of new knowledge, and innovative methods of patient care have since then been the hallmarks of Johns Hopkins. Today, the university enrolls 18,000

Figure 1.1: Location of Baltimore and aerial view of the city

full-time and part-time students on three major campuses in Baltimore, one in Washington, D.C., one in Montgomery County, Md., and facilities throughout the Baltimore-Washington area and in Nanjing (China) and Bologna (Italy).

In all, the university has eight academic divisions. The Zanvyl Krieger School of Arts and Sciences, the G.W.C. Whiting School of Engineering and the School of Professional Studies in Business and Education are based at the Homewood campus in northern Baltimore. The schools of Medicine, Public Health, and Nursing are in east Baltimore, sharing a campus with The Johns Hopkins Hospital. The Peabody Institute, founded in 1857 and a leading professional school of music, has been affiliated with Johns Hopkins since 1977. It is located on Mount Vernon Place in downtown Baltimore. The Paul H. Nitze School of Advanced International Studies, founded in 1943, has been a Hopkins division since 1950. It is located in Washington, D.C.

The Applied Physics Laboratory is a division of the university co-equal to the eight schools, but with a non-academic mission. APL, located between Baltimore and Washington, is noted for contributions to national security, space exploration and other civilian research and development. It has developed more than 100 biomedical devices, many in collaboration with the Johns Hopkins Medical Institutions.

Separately incorporated but closely affiliated is The Johns Hopkins Health System, formed in 1986 to coordinate a vertically integrated delivery system covering the full spectrum of patient care. Wholly owned subsidiaries include Johns Hopkins Medical Services Corporation (trading as Johns Hopkins Community Physicians), and three hospitals: The Johns Hopkins Hospital, Johns Hopkins Bayview Medical Center, and Howard County General Hospital, which have 1,510 licensed acute care beds and 369 skilled nursing and other special beds. During year 2001, 73,000 patients were discharged, and there were 1.5 million outpatient and emergency visits.

Homewood campus, which constitutes the headquarters for the university and is the campus were my internship has taken place, has nearly 4,000 full-time undergraduates and nearly 1,400 full-time graduate students in two schools, the Krieger School of Arts and Sciences and the Whiting School of Engineering.

In terms of financial impact, the university employs more than 25,000 people in full- time, part-time and temporary positions. It is one of Maryland's five largest private employers. The so-called "Johns Hopkins Institutions", which include the university and The Johns Hopkins Health System, together constitute the state's largest private employer. In fiscal year 1999, spending by the university, the Health System and their affiliates generated - directly and indirectly - an estimated $5 billion of income in Maryland, roughly one of every 33 dollars in the state's economy.

Johns Hopkins ranks first among U.S. universities in receipt of federal research and development funds. The School of Medicine ranks first among medical schools in receipt of extramural awards from the National Institutes of Health. The Bloomberg School of Public Health is first among all public health schools in research support from the federal government.

Figure 1.2: Aerial view of the Homewood campus in Baltimore.

When it comes to recognitions and awards, The Johns Hopkins University is considered the best university in the United States in the field of bioengineering (according to the ranking by US News & World Report), and one of the best in the field of medicine. The Johns Hopkins Hospital, has consistently been ranked as the best hospital in the United States during the last fifteen years (again, ranking by US News & World Report). Also, it is interesting to note the existence of many Nobel prize winners who have taught or carried out research at Hopkins, including recent laureates for medicine (Richard Axel in 2004, Paul Greengard in 2000), chemistry (Peter Agre in 2003), economics (Robert Mundell in 1999) or physics (Riccardo Giacconi in 2002).

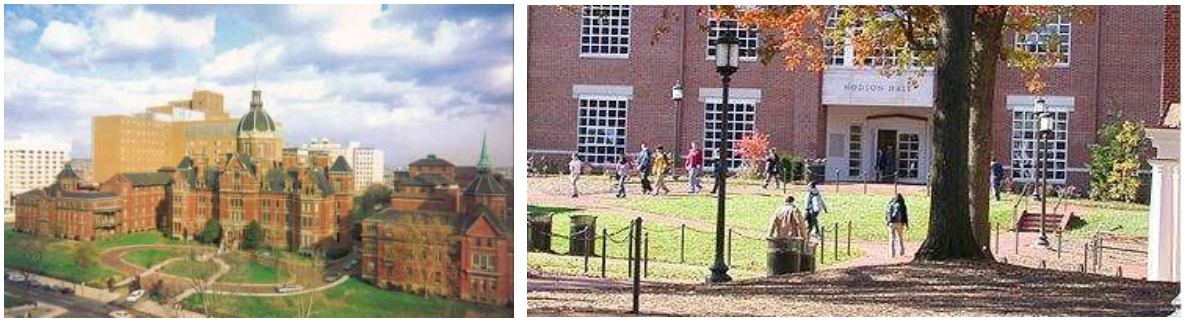

Figure 1.3: The Johns Hopkins Hospital (left) and Hodson Hall (right), one of the buildings for the Whiting School of Engineering)

1.1.2 The Whiting School of Engineering

With just under 1,300 full-time undergraduate students and 600 graduate students, the Whiting School of Engineering remains a relatively small school. It has nine academic departments, a number of them among the top in the nation as ranked by U.S. News & World Report. In the 2005 rankings, the undergraduate programs came in at 13th overall in the country while graduate programs ranked at 21st in the nation, with Biomedical Engineering at number one.

Johns Hopkins welcomed the first class of engineering students in 1913. At that time, civil, electrical, and mechanical engineering formed the triad of disciplines available to aspiring engineers.. The University awarded its first undergraduate engineering degrees in 1915.

DEPARTMENTS	FACTS AND FIGURES
• Applied Mathematics and Statistics • Biomedical Engineering • Chemical and Biomolecular Engineering • Civil Engineering • Computer Science • Electrical and Computer Engineering • Geography and Environmental Engineering • Materials Science and Engineering • Mechanical Engineering	• External Research Support: $58.1 million School Endowment: $62 million (FY03) • 1,283 Undergraduate students (25% female) 643 Graduate students (26% female) • 117 Academic & 34 Research Faculty, plus 30 associated research scientists and engineers. • Degrees Awarded in 2004: 253 B.S, 21 B.A., 151 M.S., 54 Ph.D.

Figure 1.4: The Whiting School of Engineering

In 1919, the Department of Engineering became the School of Engineering. When the School celebrated its 25th anniversary in 1937, more than 1,000 students had completed engineering degrees at Hopkins. As it had done during World War I, the School contributed to the war effort throughout World War II by developing technical short courses to prepare war industry personnel.

By 1946, the School had grown to six engineering departments and in 1961, it formally changed its name to the School of Engineering Sciences, a change that symbolized its commitment to "a unified scientific approach."

In 1966, the School merged with the Faculty of Philosophy, creating the School of Arts and Sciences. The image of engineering at Johns Hopkins quickly faded, and many clamored for the restoration of an engineering school. With the support of many, the G.W.C. Whiting School of Engineering – the University's first named division – was established in 1979. In the following years, the School prospered at all levels.

In 1992, four general areas of excellence were established for the Whiting School – biomedical engineering, environmental systems and engineering, materials science and engineering, and electronics and information technology. The theme became the rallying cry for the Whiting School's $50 million campaign to enhance endowment, faculty, and scholarships. Numerous collaborative efforts were undertaken that resulted in new centers, new faculty research initiatives, and a new minor in entrepreneurship and management, open to all Hopkins undergraduates. The campaign received a huge boost in 1995, when electrical engineering alumnus and entrepreneur Michael Bloomberg ('64) made what he described then as an "initial commitment" of $55 million to the Johns Hopkins Initiative, divided among all the academic divisions of the University. The Whiting School's portion of that gift, $15 million, was the largest

in the history of engineering at Johns Hopkins. The engineering school's growth in all areas resulted in a higher standard of education and engendered increased recognition at national and international levels.

With the Whiting School firmly ensconced in a higher tier of engineering institutions, several new graduate programs were introduced after 2001, including bioinformatics, homeland security, and technical entrepreneurship and innovation. In 2003 the School announced three new bioengineering concentrations for undergraduates namely biomolecular engineering, biomaterials engineering, and biomechanics. These last developments reflect the interest of the School for targeting the increasingly close ties among the fields of biology, chemistry, medicine and engineering.

1.1.3 The Center for Imaging Science

The Center for Imaging Science (CIS) belongs to the Whiting School of Engineering and was established in 1995 by the Army Research Office to bring together a diverse group of researchers whose work is highly interdisciplinary and rests on theoretical advances in mathematics and statistics, traditional signal and systems processing, and information theory.

Figure 1.5: Clark Hall, building of the Center for Imaging Science.

The director of the center is Michael I. Miller and the principal faculty includes Professors Donald Geman, John Goutsias, Carey Priebe, Jerry Prince, Trac Tran, Rene Vidal and Laurent Younes. All of them have appointments in the Departments of Biomedical Engineering, Electrical and Computer Engineering or Mathematical Sciences.

The research program at CIS is organized around three principal themes:

1. Representation and synthesis of complex shapes and scenes;
2. Computationally efficient shape detection and recognition;
3. Image formation and sensor modeling.

Due to the rapid development of imaging sensor technologies, investigators in the physical and biological sciences are now able to observe living systems and measure both their structural and functional behavior across many scales, from global, aggregate behavior to the microscopic scale of sub-cellular structure. Combining biomedical imaging science with computational modeling, researchers at CIS focus their efforts in tasks such as:

Figure 1.6: The members of the Center for Imaging Science.

- Developing systems that can interpret images of natural scenes, CT scans and other data obtained with bio-medical imaging devices, and aerial and satellite images acquired by remote sensing.
- Addressing automatic recognition and machine perception problems, such as the sematic understanding of shapes and objects appearing in images.
- Inferring, non-invasively, the structural and functional properties of complex biological systems and neural circuits. This includes, for instance, the study of the cohorts of neuropsychiatric illnesses such as schizophrenia, depression, epilepsy, dementia of Alzheimer type, and Parkinson's. It also includes the relatively recent challenge of learning gene interaction networks from microarray data, which introduces the subject of my internship.

1.2 Internship goals and motivation

When I first arrived at the Center for Imaging Science in April 2005, some research work in the field of microarray data analysis and interpretation had already been carried out at the department. An example of this can be found at the publications related to the so-called "Top Scoring Pairs" classifier ([21], [22]).

These works, however, dealt mainly with classification, that is, with the use of gene expression data for the identification of certain diseases and the discrimination of healthy vs. unhealthy samples from real patients. The most ambitious goal fueling my internship was to evolve from classification to modeling. We wanted to address the challenge of learning structures of statistical dependence among genes. These structures correspond to the concept of gene interaction networks that will be presented in the following section.

We decided to begin by looking at the state of the art in this field, and found that one of the most commonly accepted approaches to the problem was based on the use of graphical models and Bayesian

networks. We therefore chose to begin our research work by studying this approach, before deciding whether to accept it as the basis for future research and attempt to improve it or guide our efforts in a different direction.

Within this context, the basic initial goals for my internship were stated as follows:

- To study the theoretical framework of graphical models and focus on Bayesian networks, attempting to obtain a general understanding of the mathematical concepts associated with these tools, of their advantages and their limitations.
- To design experiments and run Matlab simulations with synthetic data and very simple networks in order to get a hands-on approach on the practical use of these models before choosing to apply them or not to possible future experiments involving real genetic data.

Why Bayesian networks? Apparently, the main advantages inherent to this approach can be summarized as follows:

- First of all, they provide a qualitative description of the relationships among genes in terms of conditional independence that offers good interpretability from a biological point of view.
- Secondly, they also provide a quantitative description of these relationships in terms of conditional probability distributions. This probabilistic nature is well adapted to the addressed problem because
  a. it is capable of handling noise and uncertainty inherent in both the biological processes and the microarray experiments and
  b. it makes the inference scheme robust since the confidence in the learned structures can be estimated objectively as a function of the observed data.

On the other hand, however, we found that this methodology also had some potentially important flaws whose degree of relevance needed to be assessed and taken into account:

- Bayesian networks are based on the use of directed acyclic graphs, but feedback is an essential feature of many biological systems. This might therefore be an inadequate oversimplification when dealing with real genetic networks.
- In many real experimental scenarios, hidden variables may need to be considered and the learning algorithms currently present in the literature are still at an early stage of development when it comes to addressing this task. This might imply a serious risk that would force researchers to sacrifice biological meaningfulness for greater computational tractability or vice versa.
- When learning Bayesian networks from data, only partially directed graphs associated to equivalence classes can be learned, and thus there is a substantial loss of information concerning edge directions and possible causal relations between genes.

Perhaps most importantly, the small samples inherent to microarray studies requires that interactions between hundreds of genes be learned from very reduced datasets, and it is yet unclear whether or not the posterior probabilities over network structures can be learned with sufficient robustness in these conditions. We are facing a challenge that is yet not fully understood and there are actually no real guarantees concerning its feasibility. Using a simple analogy, we may think of three coins. The first one of them is unbiased, the second is biased towards almost always getting heads and the third is unbiased towards almost always getting tails. Let us imagine that we throw the first coin and, depending on its result, we choose between the second or the third coins for a second toss, obtaining two binary values for each experiment. We may be sure of robustly learning the conditional probability distributions and the structures of statistical dependency for samples containing two binary values that are generated with these coins if we are given the results of several hundred experiments, but it remains unclear how far we

may go when trying to estimate the same properties for samples containing several thousand values if only the results for a few tens of experiments are available.

When designing the simulations to carry out, we decided to first experiment with synthetic data instead of using publicly available real datasets. This was mainly due to the lack of ground truths for most of these datasets, which makes it very difficult to assess the validity of any given results. For example, if an interaction between genes is found that is not supported by biological literature, it is not possible to decide (at least not without further expensive laboratory experiments) whether the algorithm has discovered a new, previously unknown interaction, or whether it is just a spurious edge. Furthermore, when consulted, certain geneticists tend to delve into the biological literature until they find arguments to substantiate many of these findings, but it is sometimes difficult to establish whether the finding is real or whether it is just a product of the circumstances. In order to avoid this kind of situation, we decided to generate our own synthetic datasets so that the success rate for the learned edges can be reliably calculated for all our simulations.

1.3 Scientific context and theoretical framework

Following the sequencing of the human genome, we find ourselves in the dawning of what has been called the 'post-genomic' era. Now that at least a draft outline of this genome is known, many researchers have focused their efforts in the challenge of understanding the way in which all this genetic material relates to an organism physiology and biological functions.

1.3.1 Functional genomics

Unfortunately, the millions of bases of DNA sequences that are already available do not tell us what genes do, how cells work, what goes wrong in disease, how we age or how to make good drugs. In this context, ***functional genomics*** arises as a field of knowledge aiming at the deconstruction of the genome to assign biological function to genes and gene interactions. The goal is not simply to provide a catalogue of all the genes and information about their function, but to understand how components work together and collaborate in the life cycle of cells and organisms.

In direct connection with this, biological and biomedical research is currently undergoing a profound transformation thanks to the massive increase in the amount of genetic information that is constantly being made available to the scientific community. New technologies open the way for new types of experiments and, as a result, observations, analysis and discoveries are being made in an unprecedented scale.

Geneticists are usually faced with multiple kinds of biological information including DNA sequences, gene maps and gene expression data, protein structure and protein interaction measurements, etc. In order to take full advantage of this large and rapidly increasing body of data, new automated techniques are required. The application of information science, computer science and biostatistics to the challenge of knowledge acquisition from genomic data is commonly known as ***bioinformatics*** and constitutes the general scientific framework of reference for my internship at The Johns Hopkins University.

A good example of the magnitude of the amount of newly available genetic information is provided by the so-called "DNA chips". Located at the confluence of several scientific disciplines such as biology, chemistry, robotics, florescence detection and photolithography, DNA microarray technologies are amongst the most powerful and versatile tools for the acquisition of genomic data. Today, biologists can use these elements to obtain near-comprehensive expression data for individual cells, tissues or organs. Currently available commercial tools provide the means to get systematic quantitative information of the levels of expression of hundreds of thousands of RNA measures using a single chip.

In practice, these measurement platforms permit a large number of exhaustive comparisons, including transcriptional activity across different tissues in the same organism, across neighboring cells of different types in the same tissues or across groups of patients with and without a particular disease. The central

hypothesis that we will adopt is that, with sufficient data, these measures can actually be used to provide insight into the underlying mechanisms of genetic regulatory networks. The ultimate goal for the current project will thus be to develop new bioinformatics techniques capable of discovering the "true" biological functional pathways in gene regulation, an accomplishment which would eventually lead to the discovery of many potentially interesting biomedical applications.

1.3.2 Biological background

Let us first begin by introducing some basic biological concepts.

All living organisms are made up of cells, which constitute their fundamental working units. Cells are made up of different molecules inside a membrane. In nature, it is possible to find both unicellular (bacteries, yeasts, etc.) and multicellular organisms (a human being is supposed to have the order of 6×10^{23} cells).

Figure 1.7, shows a typical decomposition of the chemical constituents of a bacterial cell. We will focus our interest in three key components: DNA, RNA and proteins.

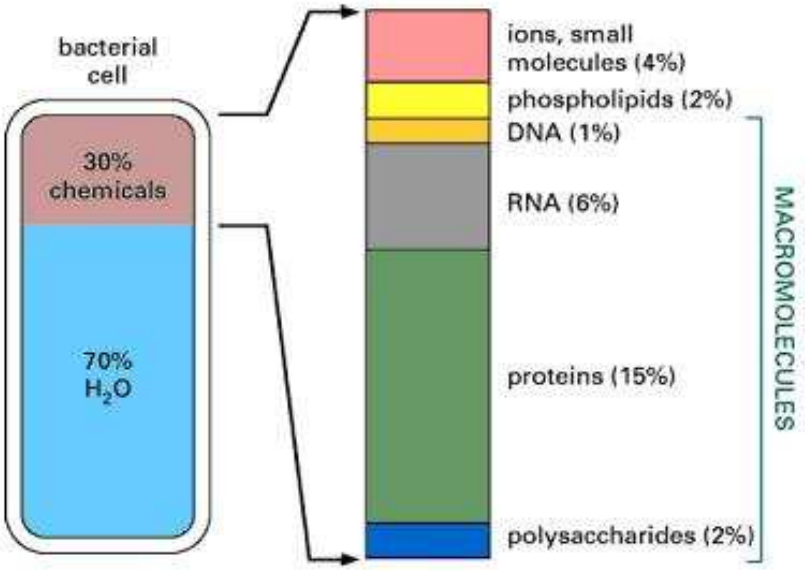

Figure 1.7: Chemical decomposition of a typical bacterial cell, taken from [3]

Other than water, ***proteins*** account for the highest percentage of weight in the cell. From a chemical point of view, proteins are chains of amino acids. There are 20 different amino acids that combine to form these chains. Proteins are usually related to several functions at the cellular level:

- Structural, like the collagen that joins bones and tissues.
- Catalytic, in the form of enzymes that take part in many biochemical reactions.
- Hormonal, since certain hormones have proteic nature (like insulin).
- Regulatory/environmental, since proteins control cellular interactions with the environment, including signal transduction, molecular recognition and molecular exchange through the cellular membrane.

For the synthesis and regulation of proteins, cells need a certain set of instructions, which is provided through the DNA.

Deoxyribonucleic Acid (DNA) is a molecule, present in all cells, that contains genetic information transmitted between generations. Each molecule of DNA may be viewed as a pair of chains of the nucleotides adenine (A), thymine (T), cytosine (C) and guanine (G). Moreover, each chain has a polarity, from 52(head) to 32(tail). The two strands join in opposing polarity through the coordinated force of multiple hydrogen bonds at each base-pairing, where A binds to T and C binds to G.

The term ***genome*** refers to the entire set of DNA molecules in an organism. Its overall function is to drive the generation of molecules, mostly proteins, that will regulate the metabolism of a cell and its response to the environment while providing structural integrity.

DNA is the same in almost every cell of the human body (with some exceptions, such as certain blood cells or the gametes, for example). This means that a liver cell and a brain cell have the same DNA content and code in their nucleus. What distinguishes these cells from one another is the portion of their DNA that is transcribed and translated into proteins.

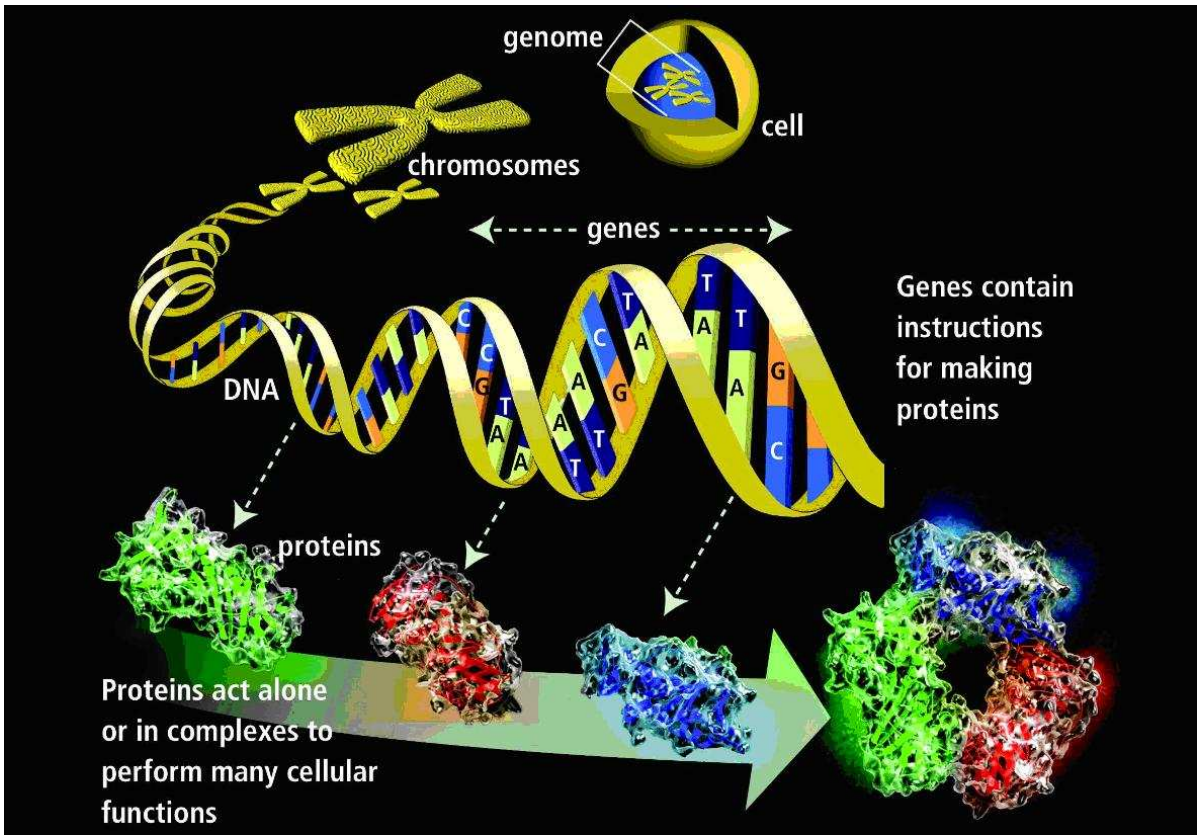

Figure 1.8: From chromosomes to proteins, taken from [4]

A ***gene*** is a continuous segment of DNA that encodes the necessary instructions for the synthesis of a particular protein. In order for the genome to direct or effect changes in the cytoplasm of the cell, a transcriptional program may be activated for the purpose of generating new proteins. DNA remains in the nucleus of the cell, but most proteins are needed in the cytoplasm and so, DNA is copied into a more transient molecule called RNA through a process called ***transcription***.

The ***Ribonucleic Acid (RNA)*** molecule is a sequence of base pairs that correspond to those in the DNA molecule using the complementary A-T, C-G, with the principal distinction being that the nucleotide uracil is substituted for the thymine nucleotide.

The RNA that codes for proteins is called ***messenger RNA***, and the part of the DNA that provides that code is called an ***open reading frame*** (ORF). When read in the standard 52 to 32 direction, the

portion of DNA before the ORF is considered upstream, and the portion following the ORF is considered downstream.

The specific determination of which genes to transcribe is determined by promoter regions, which are DNA sequences upstream of an ORF. Many proteins have been found containing parts that bind to these specific promoter regions, and thus activate or deactivate transcription of the downstream ORF. These proteins are called ***transcription factors***.

The cytoplasm is where the cellular machinery, in particular the ribosomal complex, acts to generate the protein on the basis of the mRNA code. A protein is built as a polymer or chain of amino acids, and the sequence of amino acids in a protein is determined by the mRNA template. The mRNA provides a degenerate coding in that it uses three nucleotides ($4^3 = 64$ combinations) to code for each of the twenty common naturally occurring amino acids that are joined together to form the polypeptide or protein molecule. Consequently, several trinucleotide sequences (also known as ***codons***) correspond to a single amino acid. There is no nucleotide between codons, and a few codons represent start (or initiation) and stop (or termination). The process of generating a protein or polypeptide from an mRNA molecule is known as ***translation***.

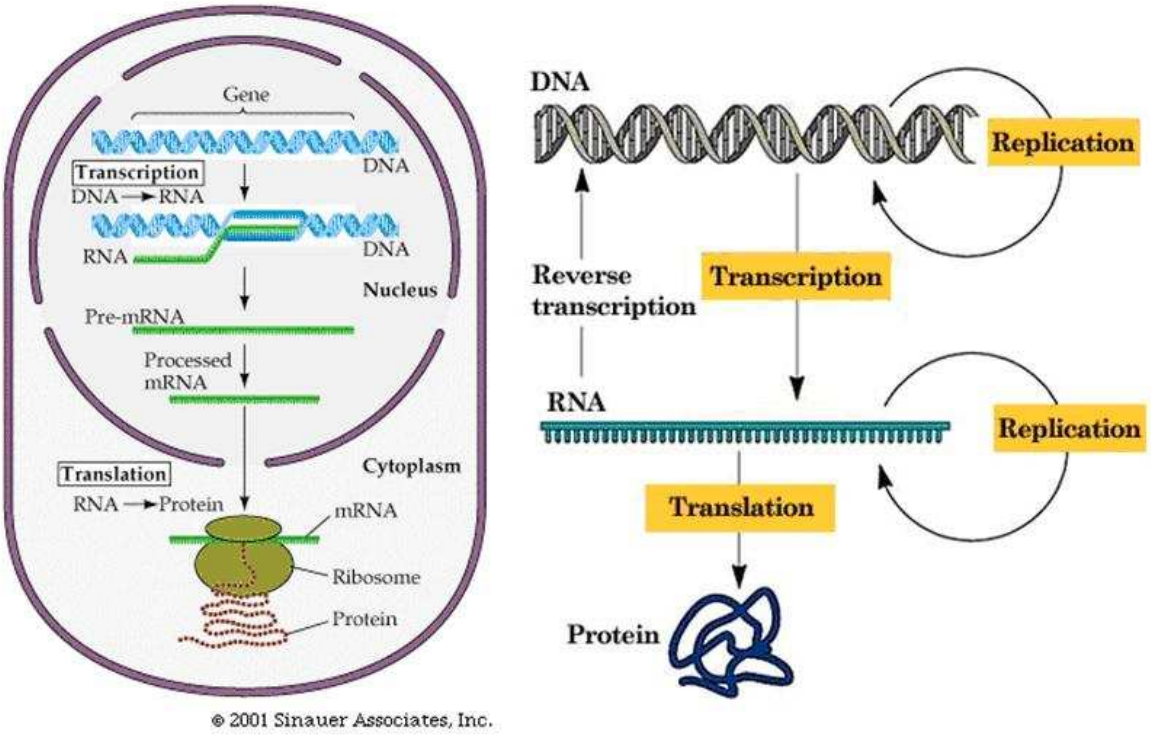

Figure 1.9: Left panel shows a schematic representation of the transcription and translation processes. Right Panel shows the central dogma of biology. (taken from [5])

Transcription and translation are the two main actors in the so-called ***central dogma*** of biology (figure 1.9).

The initiation of a transcription process (and the associated translation) can be caused either by external events or by a programmed event within the cell. Heat shock or stress, for example, can cause the initiation of a transcriptional program (and this fact can be sometimes used by genetic researchers in specifically designed experiments). There are also fully autonomous, internally programmed sequences of transcriptional expression. Finally, certain pathological internal derangements of the cell can lead to transcriptional activity, as is the case for certain self-repair or damage-detection programs.

1.3.3 Microarray gene expression data

A ***DNA microarray*** is a collection of microscopic DNA spots, known as probes, attached to a solid surface, such as glass, plastic or a silicon chip and forming an array because of their spatial disposition. Thousands of these DNA segments can be measured in a single microarray, providing scientists with the expression levels of large numbers of genes simultaneously. In practice, the terms DNA chip and GeneChip are also used to refer to microarrays, even though the latter is a genericized trademark of Affymetrix.

Microarrays can be fabricated using a variety of technologies, including printing with fine-pointed pins onto glass slides, photolithography using pre-made masks, photolithography using dynamic micromirror devices, ink-jet printing, or electrochemistry on microelectrode arrays.

The most common use of this technology is to quantify mRNAs that are transcribed from different genes and that encode different proteins, as described in the previous section. This collection of genes that are expressed or transcribed from genomic DNA is usually referred to as the ***expression profile*** or the '***transcriptome***'. DNA microarrays therefore measure concrete levels of mRNA abundance.

The main idea behind microarray technologies is the concept of ***hybridization***. Since DNA is a double stranded structure held together by complementary interactions (in which C always binds to G, and A to T), complementary strands favorably reanneal or "hybridize" to each other when rejoined after being separated. The usual approach makes use of this fact by fixing a set of probes (known sequence) to a surface and creating an interaction with a set of fluorescently tagged targets (unknown sequences). This results in a highly parallel search by each molecule for a matching partner with the eventual pairings of molecules on the surface determined by the rules of molecular recognition.

After hybridization, the fluorescently lit spots indicate the identity of the targets and the intensity of the fluorescence signal is in correlation to the quantitative amount of each target. In practice, it is not possible to measure concrete quantitative amounts of each target. Instead, the results show the relative comparison between the reference pool and the target pool. Typically, green is used to label the reference pool, representing the baseline level of expression and red is used to label the target sample in which the cells were treated with some condition of interest. After hybridizing the mixture of reference and target pools, a green signal is obtained for the cases where the considered condition reduces expression level of the gene under study and a red signal is observed for cases where the condition increases expression level.

Depending upon manufacturer specific protocols, the hybridization process on a microarray typically occurs over several hours. All unhybridized sample probes are then washed off and the microarray is lit under laser light and digitally scanned to record the brightness level at each grid location.

The use of microarrays for expression profiling was first published in 1995 in Science (Schena et al. [7]) and the first complete eukaryotic genome (*Saccharomyces cerevisiae*) on a microarray was published in 1997 also in Science (DeRisi et al. [8]). Nowadays, expression microarrays are already sufficiently well engineered and cost-effective to allow thousands of researchers to productively employ them to drive their investigations. However, there is still a long way ahead in terms of cost reduction, reproducibility and accuracy in order to reach the levels of reliability required for clinical practices. The lack of standardization also presents an interoperability problem that needs to be overcome, so that cooperation among different laboratories working with different platforms can be enhanced.

Finally, for the purposes of my internship, it is convenient to emphasize the inherent small sample regime associated to microarray experiments. This is, in fact, what makes most standard, well-known biostatistical techniques unsuitable for the analysis of their results. After all, there has been a long history of the development of biostatistical techniques to analyze large studies with large numbers of cases and with many variables, so we might think of directly applying all this knowledge to the elucidation of the relationships among genes expressed on a microarray chip. Whereas a high-quality clinical study usually involves thousands to tens of thousands of observed samples over which up to several tens or hundreds of variables (at most) are measured, typical genomic studies reverse those proportions and imply the need to deal with tens or hundreds of measurements made over thousands of variables. In practice, this leads to highly undetermined systems, which in turn accounts for a very high risk of overfitting in all learning

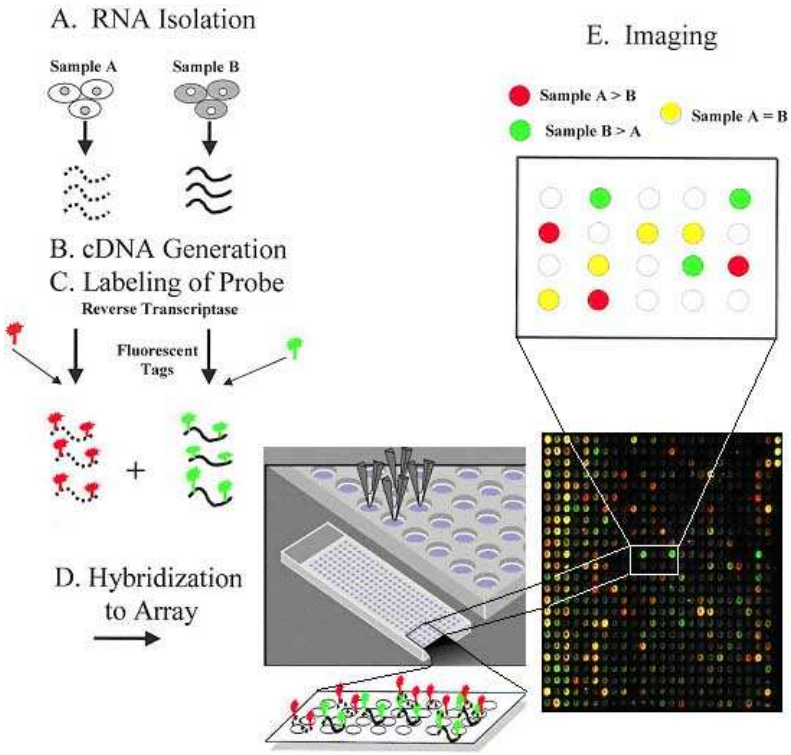

Figure 1.10: Phases of a microarray experiment, from [6]

procedures. This is why the use of techniques that can maximally inform of the relationships between variables of interest based on greatly reduced datasets acquires an enormous importance.

1.3.4 Gene regulatory networks

The concept of gene regulatory networks is based on the assumption that gene expression levels affect each other and that this, in turn, reflects interactions between cellular components.

In reality, an accurate global description of the cellular biochemistry needs to deal with three different levels corresponding to genes, proteins and metabolites, as depicted in figure 1.11. This fact is at the origin of three different types of networks:

a. Metabolic networks, that represent the chemical transformations between metabolites (that is, between the products of the cellular metabolism).

b. Protein networks, that represent protein-protein interactions, like those related to the co-formation of chemical complexes and protein modification by signaling enzymes.

c. Gene networks, that represent relationships that can be established between genes by observing how the expression level of each one affects the expression level of the others.

As pointed out by Brazhnik et al. [9], in any global biochemical networks, genes do not interact directly with other genes (as neither do the corresponding mRNAs). There is however an indirect interaction since gene induction or repression occurs through the action of specific proteins, which are themselves a consequence of the levels of expression of certain genes. Metabolites can also have an effect on gene expression levels and are obviously influenced by these same levels. According to this, it would be reasonable to adopt a three dimensional model of a global biochemical network in which

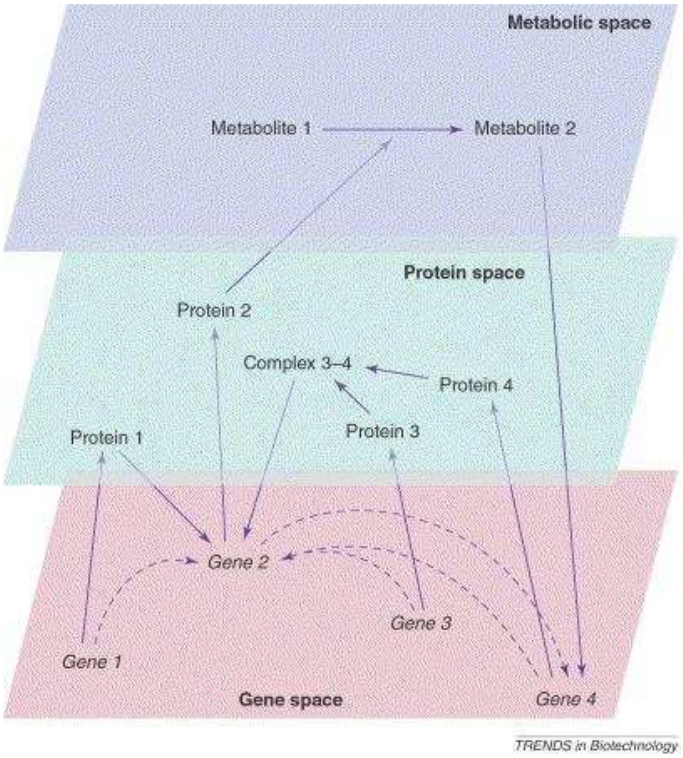

Figure 1.11: Global biochemical networks, from [6]

the three levels mentioned above would interact from different planes. In practice, however, methods to profile proteins and metabolites are currently being developed, but are not yet as widespread as methods to profile gene expression (see the comments concerning microarray technology above). As a result, it is a common approach to abstract the behavior of proteins and metabolites and just represent genes interacting among themselves in a so-called ***gene regulatory network***. This simplification of going from the global biochemical network to a gene network is akin to a projection of all interactions to the "gene plane" (figure 1.11).

Since the levels of gene expression can be quantitatively measured by observing the corresponding levels of mARN abundance in a microarray experiment, many researchers have focused their efforts in the challenge of using this information to learn valid mathematical models for this kind of networks. The goal is to find a reliable procedure capable of automatically reconstructing molecular pathways and inferring gene regulatory relations from data.

Some early approaches, such as Chen et al. [11], attempted to learn the mechanisms underlying interactions between genes by studying low-level dynamics. They wanted to obtain a mathematical description of the biophysical processes in terms of a system of coupled differential equations. In practice, however, this approach turned out to be useful only for small systems. Studies like the one carried out by Zak et al. [12] have actually proved that gene expression data is insufficient to establish a set of ordinary differential equations capable of describing a regulatory network of this type, and that richer data including other types of biological information is needed to ensure identifiability of all the necessary parameters.

As opposed to this very refined level of detail, certain researchers decided to work with the coarse-level scale of gene clustering. This is the case of the seminal paper by Eisen et al. in 1998 [10], and also D'haeseleer et al. in 2000 [13]. This strategy assumes that co-expression is indicative of co-regulation, and therefore attempts to identify genes that are involved in related biological processes by identifying genes with highly correlated levels of expression. In practice though, even if some genes are correctly

identified as being corregulated, the nature of their interactions usually remains unclear. Critics of this approach claim that it is only capable of grouping genes in monolithic blocks and that it cannot be used to infer the detailed form of the underlying regulatory interaction patterns.

As a trade-off between these two opposing trends, the approach of graphical models has arisen as the preferred methodology for many researchers on the field. Within this category, subsequent divisions can be made leading to different kinds of concrete graphical models: Bayesian networks, Gaussian graphical models, additive regulation models, qualitative causal models, state-space models, undirected dependency graphs and so on. The interested reader can refer to the references mentioned in [32] for more information related to each of them.

In this report, we will focus on the study of Bayesian networks, since they seem to be one of the mathematical tools that have yielded some of the most promising results in recent years.

The main theoretical framework for Bayesian networks was established and developed through a series of papers by several authors (Pearl [14], Heckerman [15], etc.). They were first applied to the problem of reverse engineering genetic networks from microarray expression data by Friedman et al. in 2000 [1]. Since then, other researchers who have published articles regarding this subject include Hartemink et al. [16] and Husmeier [17].

The main idea underlying the approach of Bayesian networks applied to gene interaction networks is to assume that the likelihood of a cell state, characterized by the levels of expression of the different genes that control cellular functions, can be specified by a joint probability distribution.

A Bayesian network over a set of random variables $X = \{X_1, ..., X_n\}$ is basically a representation of a joint probability distribution over X. When these variables are associated to gene expression levels, Bayesian networks seem to be well adapted to the modeling problem at hand, since they offer powerful mechanisms for tasks that range from basic inference to parameter and structure learning.

Assuming that a "true" Bayesian network for a given set of genes could be learned, it would be immediate to answer many biologically interesting questions, such as: "Are genes A and B conditionally independent given gene C?","What would be the effect over gene D of over expressing gene E","Which genes should be acted upon in order to modify the level of expression of gene E","Which genes are candidates to be connected on the same metabolic pathway?", etc...

Before going any further, let us introduce the mathematical foundations for the use and understanding of these probabilistic models.

1.3.5 Bayesian networks fundamentals

This section provides a quick overview of some key concepts concerning Bayesian networks that will be used in the rest of this report. For a more detailed description of the ideas presented here, the reader is invited to refer to the Bayesian networks tutorial in appendix A.

A ***Bayesian network*** over a set of random variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ is a representation of a joint probability distribution over $\mathbf{X}$. This representation has both a qualitative and a quantitative dimension (figure 1.12), and consists of three key elements:

1. A graphical structure G with the form of a Directed Acyclic Graph (DAG).
2. A family of conditional distributions for each node given its immediate parents in G.
3. A set of parameters θ for these distributions.

The graphical structure $G = (V, \varepsilon)$ consists of a set of nodes or vertices V, that represent the random variables X, and a set of directed edges or arcs, ε, that represent conditional dependence relations among the nodes. If there is a directed edge from node X_1 to node X_2, then X_1 is called the parent of X_2, and X_2 is called the child of X_1.

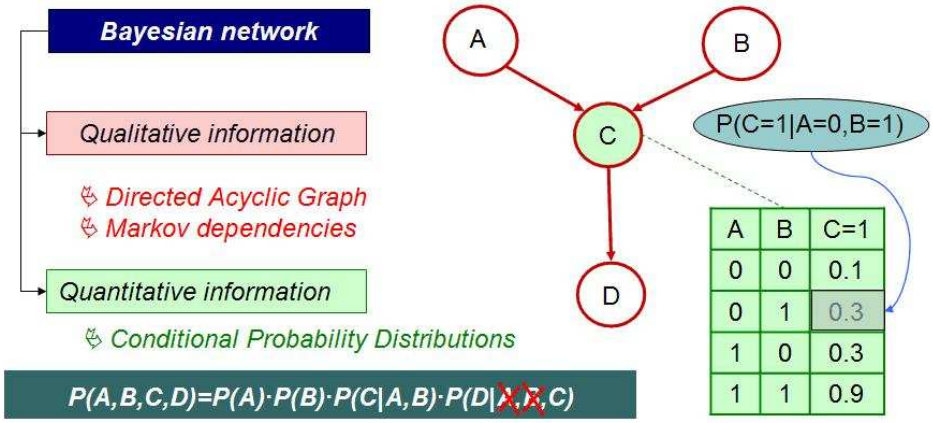

Figure 1.12: Basic concepts related to Bayesian networks used as tools for the representation of joint probability distributions

The central dogma of Bayesian networks is given by the so-called ***Markov Independencies***: each variable X_i is independent of its non-descendants given its parents in G. As a direct consequence of this, the joint distribution represented by a Bayesian network can be decomposed in the following product form:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | Pa_i^G) \tag{1.1}$$

where Pa_i^G is the set of parents of X_i in G. Usually, the conditional independencies represented by a Bayesian network go beyond the Markov independencies. The notion of ***d-separation***, a graph theoretic concept, makes it possible, for example, to construct a linear time algorithm capable of determining whether two subsets of nodes Y and Z are independent or not given a subset of so-called "evidence" nodes E (nodes whose value is fixed and known).

For the rest of this report, we will only work with nodes taking discrete binary values (0 and 1). In line with this, we will restrict the families of conditional probability distributions for all the Bayesian networks that we will be using in the simulations to be families of discrete multinomial distributions. As opposed to linear Gaussians, these distributions can capture non-linear dependence relations. Also, they provide certain desirable properties for the computation of the scoring function (such as conjugate priors, and a closed form for the integration of the likelihood) that will be discussed later on. Consequently, in what follows, we will focus on the problem of learning the network structure G and the network parameters θ.

Before going any further, it is important to note that some DAGs may represent exactly the same set of conditional independence statements, as illustrated in figure 1.13a. Graphs verifying this property are said to belong to the same ***equivalence class***.

This concept is very important because, when attempting to learn structure just from samples of a certain joint probability distribution, it is not possible to distinguish between equivalent graphs. If we define a ***v-structure*** as a combination of edges and nodes in which two parents share a certain child ($X \rightarrow Y \leftarrow Z$), then a theorem by Pearl and Verma (1990, [33]) states that two graphs are equivalent if and only if they have the same underlying undirected graph and the same v-structures. Each class of equivalence can therefore be represented by a Partially Directed Acyclic Graph (PDAG) where all edges except those corresponding to v-structures are represented by undirected links, implying that some members of the class may contain the edge with a certain orientation and some may contain it with the other (figure 1.13b). In practice, when learning Bayesian networks from microarray data, we will only attempt to learn the PDAG representing the appropiate equivalence class for a given dataset.

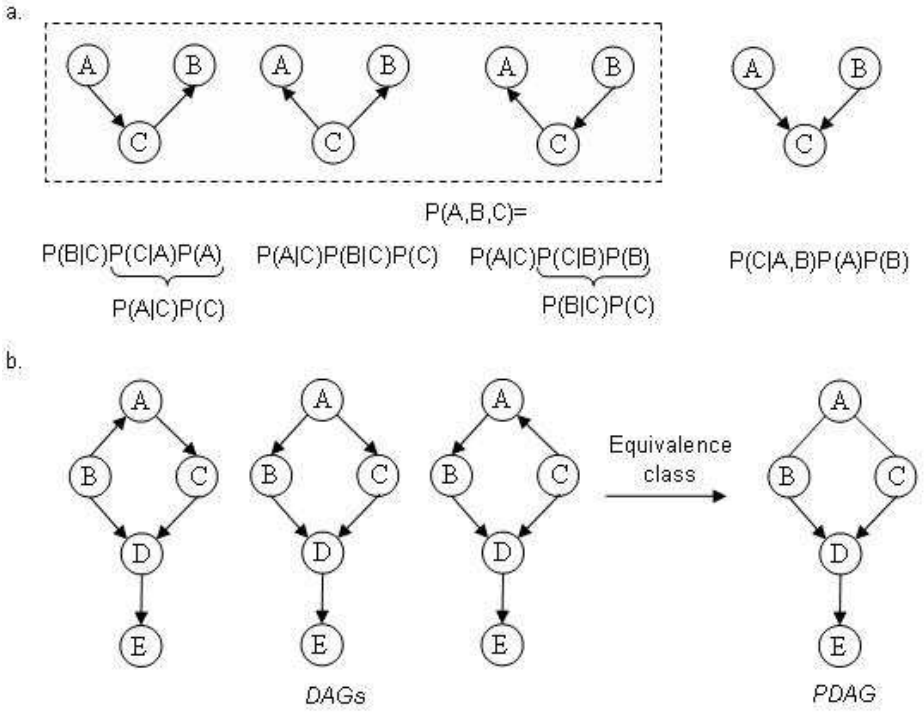

Figure 1.13: Bayesian network equivalence classes. (a) The first three structures from the left belong to the same equivalence class and share the same joint probability distribution, which is different from that of the structure on the right (adapted from Dirk Husmeier). (b) Example of three Bayesian networks belonging to the same class of equivalence and the associated PDAG that represents that class.

In order to learn structure, a common strategy consists in the use of a scored-based approach. The scoring function can be defined using the MAP principle.

The learning problem can be stated as follows:

- Consider a dataset with m samples $D = \{x(1), x(2), ..., x(m)\}$ (where each $x(i)$ represents a vector with values for the n variables under study $X = \{X_1, X_2, ..., X_n\}$).
- Assume that these samples have been drawn from some unknown generating network G^* with multinomial conditional probability distributions of parameters θ^*.
- Search for the simple model $B = (G, \theta)$ which is the most likely to have generated the data (a model whose underlying distribution will be close to the empirical distribution of the data).

For a given Bayesian network G, the probability of generating a certain dataset can be defined as $P(D|G)$. If we define P(G) as the a priori probability of network G, the probability of observing a certain dataset D for a network G expressed by P(D,G) can be obtained as:

$$P(D, G) = P(G|D)P(D) = P(D|G)P(G) \tag{1.2}$$

Using Bayes rule, the probability to be maximized within the framework of the learning problem can be rewritten as:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{1.3}$$

And the ultimate goal is to find the structure G^* that maximizes this expression:

$$G^* = \arg_G \max P(G|D) \tag{1.4}$$

In order to calculate the marginal likelihood for network structures, the conditional probability parameters θ need to be integrated out:

$$P(D|G) = \int P(D|G,\theta)P(\theta|G)d\theta \tag{1.5}$$

The Bayesian score for a given structure G and a dataset D can then be defined according to the following expression:

$$score_B(G:D) = \log P(D|G) + \log P(G) + c \tag{1.6}$$

Under certain regularity conditions, the integral is analytically tractable and has a closed form solution. This is the case when multinomial distributions are chosen for the parameters and when Dirichlet distributions are used for the parameter priors.

In these circumstances, the scoring function can be decomposed at a local level and the global score for a given network can be computed as the addition of the scores for each individual node, that we will call '*FamScore*'. This contribution depends only on the sufficient statistics of the associated variable X_k and its parents Pa_k over dataset D, which results in an expression of the following type::

$$score_B(G:D) = \sum_k FamScore_B(X_k, Pa_k : D) \tag{1.7}$$

Dirichlet distributions are desirable to formalize prior knowledge because they provide a very intuitive way to deal with such formalization.

Let $F_1, F_2, ...F_r$ be binary random variables. The associated Dirichlet density function with hyperparameters $a_1, a_2, ...a_r$, where $a_1, a_2, ..., a_r$ are integers ≥ 1, is

$$\rho(f_1, f_2, ..., f_{r-1}) = \frac{\Gamma(N)}{\prod_{k=1}^{r}\Gamma(a_k)} f_1^{a_1-1} f_2^{a_2-1} ... f_r^{a_r-1} \tag{1.8}$$

where $0 \leq f_k \leq 1$, $\sum_{k=1}^{r} f_k = 1$ and $N = \sum_{k=1}^{r} a_k$

- Random variables $F_1, F_2, ..., F_r$, that have this density function, are said to have a Dirichlet distribution.
- The Dirichlet density function is denoted $Dir(f_1, f_2, ...f_{r-1}; a_1, a_2, ..., a_r)$.

As a reminder, the Gamma function $\Gamma(x)$ is defined to be an extension of the factorial to real number arguments. If n is a natural number, then $\Gamma(n) = (n-1)!$. Otherwise, the Gamma function is defined as the following integral:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-x} dt \tag{1.9}$$

Figure 1.14 illustrates this definition with two easy to grasp examples.

Returning to the framework of Bayesian networks, the Dirichlet density function can be used to model prior knowledge and to define a scoring function:

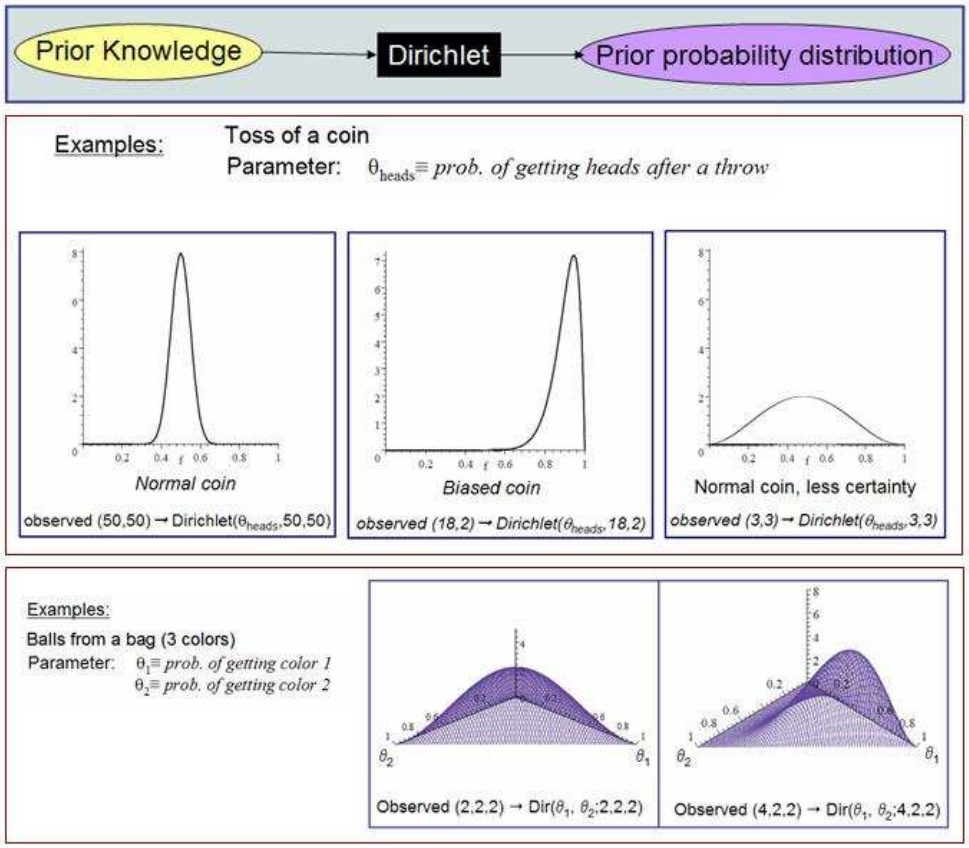

Figure 1.14: Dirichlet distributions used to formalize prior knowledge. The first example shows the use of Dirichlet distributions to formalize prior knowledge concerning the probability of getting heads for a certain coin after having observed different numbers of tosses and different results. For the normal coin (100 tosses, 50 heads and 50 tails) the distribution is centered on value 0.5. For the biased coin (20 tosses, 18 heads and 2 tails), the peak of the distribution is placed at a higher value. For the case of the normal coin and only 6 prior observations (3 heads and 3 tails), the function centered on 0.5 again but it is not so peaked as before. The second example provides a three dimensional scenario, where three parameters are considered that correspond to the probability of getting a certain color by extracting a ball from a bag containing balls of three different colors.

- The idea is to "materialize" this knowledge through the choice of a certain set of prior "imaginary counts" for each of the possible combinations of parent and node values (a_{ijk}). These counts can be used as the hyperparameters for a Dirichlet distribution modeling each of the discrete conditional probability distributions in the network.

$$P(\Theta_{ij}) = Dirichlet(\Theta_{ij1}, \Theta_{ij2}, ..., \Theta_{ij(n-1)}; a_{ij1}, a_{ij2}, ..., a_{ijn}) = \frac{\Gamma(a_{ij})}{\prod_{k=1}^{n} \Gamma(a_{ijk})} \prod_{k=1}^{n} \Theta_{ijk}^{a_{ijk}-1} \quad (1.10)$$

  where a_{ijk} is the number of prior samples for which node X_k takes value i (0 or 1) and the parents of node X_k take the j^{th} value of all their possible combinations. (see example below).

- Once the prior has been defined, the number of instances of each possible combination of parent and node values can be counted over the set of observed samples, giving a set of counts N_{ijk} (following the same notation used for the a_{ijk}). The posterior probability distribution is then also a Dirichlet distribution of parameters ($a_{ijk} + N_{ijk}$).

$$P(\Theta_{ij}|D) = Dirichlet(\Theta_{ij1}, \Theta_{ij2}, ..., \Theta_{ij(n-1)}; a_{ij1} + N_{ij1}, a_{ij2} + N_{ij2}, ..., a_{ijn} + N_{ijn}) \quad (1.11)$$

Figure 1.15 shows an example of application of this belief update procedure to the simple case of coin toss presented in figure 1.14.

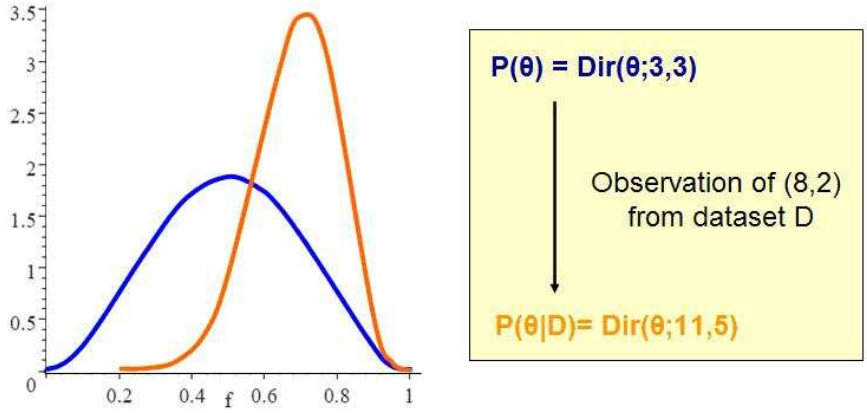

Figure 1.15: Belief update example. Dirichlet distributions are conjugate priors. Continuing with the example above for the toss of a coin, consider the case of having observed 3 heads and 3 tails, which gives a Dirichlet distribution of the form $Dir(\theta_H; 3, 3)$. If a new dataset containing 8 heads and 2 tails is observed, the posterior probability will also follow a Dirichlet distribution of the form $Dir(\theta_H; 3{+}8, 3{+}2)$.

- Under this circumstances, the parameters for the conditional probability distributions $\hat{\theta}$ that maximize $P(\theta|D)$ can be learned as:

$$\hat{\theta}_{ijk} = P(X_k = x_i | Pa_{X_k} = p_j, D) = \frac{a_{ijk} + N_{ijk}}{\sum_j (a_{ijk} + N_{ijk})} \tag{1.12}$$

- Furthermore, the local FamScore for each node X_i in the network can then be computed using a closed form formula for the integration above:

$$FamScore(X_k, Pa_{X_k} : D) = \log \prod_{j \in Val(Pa_{X_k})} \frac{\Gamma(a_{jk})}{\Gamma(a_{jk} + N_{jk})} \prod_{i \in Val(X_k)} \frac{\Gamma(a_{ijk} + N_{ijk})}{\Gamma(a_{ijk})} \tag{1.13}$$

  where the notation is the same as before and $a_{jk} = \sum_{i \in Val(X_k)} a_{ijk}$, $N_{jk} = \sum_{i \in Val(X_k)} N_{ijk}$.

When the Dirichlet parameter priors are properly chosen, this score is both class equivalent and consistent (meaning that, as the number of observed samples grows to infinity, structures belonging to the true generating class of equivalence will get the best score and the rest will score strictly lower).

In spite of its apparent conceptual complexity, once it has been properly defined the Dirichlet formalism offers an attractive ease of use and implementation.

In order to clarify these concepts and the notation used above, let us consider a very simple example of calculation of the Bayesian score within the scenario depicted in Figure 1.16 (and adapted from [30]).

Example

The objective in this example is to learn which of the two structures presented in figure 1.16 is better adapted to a given set of observed samples and a certain choice of priors (which are also shown in the same figure).

For structure A, the Dirichlet hyperparameters are obtained by simple counting

	Samples	
	X_1	X_2
1	0	0
2	0	1
3	0	0
4	1	1
5	0	0
6	1	0
7	0	0
8	1	1

	Priors	
	X_1	X_2
1	0	0
2	0	1
3	1	0
4	1	1

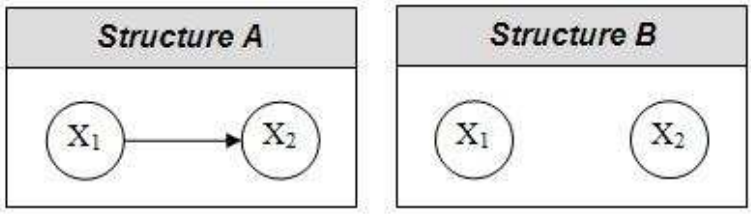

Figure 1.16: Example of score computation for a very simple learning problem.

1. over the priors (a):
   - a[value=0,parent=$\emptyset$,node=1]=$a_{0\emptyset 1}$=2
   - a[value=1,parent=$\emptyset$,node=1]=$a_{1\emptyset 1}$=2
   - a[value=0,parent value=0,node=2]=a_{001}=1
   - a[value=0,parent value=1,node=2]=a_{011}=1
   - a[value=1,parent value=0,node=2]=a_{101}=1
   - a[value=1,parent value=1,node=2]=a_{111}=1
2. and over the observed samples (N):
   - N[value=0,parent=$\emptyset$,node=1]=$N_{0\emptyset 1}$=5
   - N[value=1,parent=$\emptyset$,node=1]=$N_{1\emptyset 1}$=3
   - N[value=0,parent value=0,node=2]=N_{002}=4
   - N[value=0,parent value=1,node=2]=N_{012}=1
   - N[value=1,parent value=0,node=2]=N_{102}=1
   - N[value=1,parent value=1,node=2]=N_{112}=2

We will consider that the a priori probability P(G) for the two structures is the same, and so we will just ignore it for comparison purposes.

Substituting these values in in equation 1.13, the score for structure A is:

$$\begin{aligned}
score(G_1 : D) &= FamScore(X_1, (Pa_{X_1} = \emptyset) : D) + FamScore(X_2, (Pa_{X_2} = X_1) : D) \\
&= \log\left(\frac{\Gamma(a_{\emptyset 1})}{\Gamma(a_{\emptyset 1} + N_{\emptyset 1})} \frac{\Gamma(a_{0\emptyset 1} + N_{0\emptyset 1})}{\Gamma(a_{0\emptyset 1})} \frac{\Gamma(a_{1\emptyset 1} + N_{1\emptyset 1})}{\Gamma(a_{1\emptyset 1})}\right) \\
&+ \log\left(\frac{\Gamma(a_{02})}{\Gamma(a_{02} + N_{02})} \frac{\Gamma(a_{002} + N_{002})}{\Gamma(a_{002})} \frac{\Gamma(a_{102} + N_{102})}{\Gamma(a_{102})}\right) \\
&+ \log\left(\frac{\Gamma(a_{12})}{\Gamma(a_{12} + N_{12})} \frac{\Gamma(a_{012} + N_{012})}{\Gamma(a_{012})} \frac{\Gamma(a_{112} + N_{112})}{\Gamma(a_{112})}\right) \\
&= \log\left(\frac{\Gamma(4)}{\Gamma(4+8)} \frac{\Gamma(2+5)}{\Gamma(2)} \frac{\Gamma(2+3)}{\Gamma(2)}\right) \\
&+ \log\left(\frac{\Gamma(2)}{\Gamma(2+5)} \frac{\Gamma(1+4)}{\Gamma(1)} \frac{\Gamma(1+1)}{\Gamma(1)}\right) \\
&+ \log\left(\frac{\Gamma(2)}{\Gamma(2+3)} \frac{\Gamma(1+1)}{\Gamma(1)} \frac{\Gamma(1+2)}{\Gamma(1)}\right) \\
&= \log\left(7.2150 \cdot 10^{-6}\right) = -11.839
\end{aligned}$$

For structure B, the Dirichlet hyperparameters are also obtained by simple counting:

1. over the priors (a):
   - a[value=0,parent=$\emptyset$,node=1]=$a_{0\emptyset 1}$=2
   - a[value=1,parent=$\emptyset$,node=1]=$a_{1\emptyset 1}$=2
   - a[value=0,parent=$\emptyset$,node=2]=$a_{0\emptyset 2}$=2
   - a[value=1,parent=$\emptyset$,node=2]=$a_{1\emptyset 2}$=2
2. and over the observed samples (N):
   - N[value=0,parent=$\emptyset$,node=1]=$N_{0\emptyset 1}$=5
   - N[value=1,parent=$\emptyset$,node=1]=$N_{1\emptyset 1}$=3
   - N[value=0,parent=$\emptyset$,node=2]=$N_{0\emptyset 2}$=5
   - N[value=1,parent=$\emptyset$,node=2]=$N_{1\emptyset 2}$=3

Substituting these values in equation 1.13, the score for structure B is:

$$\begin{aligned}
score(G_1 : D) &= FamScore(X_1, (Pa_{X_1} = \emptyset) : D) + FamScore(X_2, (Pa_{X_2} = \emptyset) : D) \\
&= \log\left(\frac{\Gamma(a_{\emptyset 1})}{\Gamma(a_{\emptyset 1} + N_{\emptyset 1})} \frac{\Gamma(a_{0\emptyset 1} + N_{0\emptyset 1})}{\Gamma(a_{0\emptyset 1})} \frac{\Gamma(a_{1\emptyset 1} + N_{1\emptyset 1})}{\Gamma(a_{1\emptyset 1})}\right) \\
&+ \log\left(\frac{\Gamma(a_{\emptyset 2})}{\Gamma(a_{\emptyset 2} + N_{\emptyset 2})} \frac{\Gamma(a_{0\emptyset 2} + N_{0\emptyset 2})}{\Gamma(a_{0\emptyset 2})} \frac{\Gamma(a_{1\emptyset 2} + N_{1\emptyset 2})}{\Gamma(a_{1\emptyset 2})}\right) \\
&= \log\left(\frac{\Gamma(4)}{\Gamma(4+8)} \frac{\Gamma(2+5)}{\Gamma(2)} \frac{\Gamma(2+3)}{\Gamma(2)}\right) + \log\left(\frac{\Gamma(4)}{\Gamma(4+8)} \frac{\Gamma(2+5)}{\Gamma(2)} \frac{\Gamma(2+3)}{\Gamma(2)}\right) \\
&= \log\left(6.7465 \cdot 10^{-6}\right) = -11.906
\end{aligned}$$

Since structure A obtains a higher score, it should be preferred over structure B and therefore, it seems that, according to the given dataset and the choice of priors, the two variables are more likely to be dependent than independent.

Discrete search

In practice, once a suitable scoring function as the one above has been defined, it is necessary to explore the space of all possible DAGs in order to find the structure that best fits the data. Since the number of candidate networks increases super-exponentially with the number of nodes (the optimization problem has been proved to be NP-hard, [18]) direct search soon becomes intractable and other discrete search procedures must be taken into account. Both traditional approaches such as greedy hill-climbing or simulated annealing and new heuristic methods such as the Sparse Candidate algorithm proposed by Friedman et al. [19] can be used at this point.

Model averaging and subnetworks

Because of the small sample regime inherent to microarray expression data, finding the single top-scoring structure for a given dataset may be not enough to learn the true edges of the generating network. This is why different model averaging techniques have been proposed in the literature. The basic idea behind them is to learn several networks (either by accepting several top-scoring structures for a given dataset or by generating additional datasets through a process of bootstrap sampling) and to assign a level of confidence to each of the individual learned features (such as the presence of a certain edge between two nodes or the lack thereof). The simplest way to do this is to consider the percentage of learned networks that contain the feature under consideration. After that, subnetworks may be learned by keeping only those features with confidence over a certain threshold. These subnetworks need no longer follow the formalism of Bayesian networks (they may present directed cycles, for example), but some researchers suggest that the gain of robustness and accuracy associated to them might still render them valuable in terms of biological meaning.

1.3.6 Overview of previous works on the field

In order to bring this introductory chapter to an end, we will briefly review some of the main results obtained by other researchers using this approach.

Today, Nir Friedman continues to be at the head of a scientific team that includes his former student Dana Pe'er, while collaborating with other researchers like Daphne Koller. In Pe'er's PhD dissertation [2], a full description of their proposed methodology at the time is provided and supplemented with some concrete experimental results. This doctoral research project was adopted as the starting point of reference for my work at Johns Hopkins.

Friedman and Pe'er propose a Bayesian score-based approach like the one described in the previous section for learning structures, therefore using a conditional log-posterior, multinomial distributions and Dirichlet priors (a strategy which is also followed by Hatermink et al. [16] and Husmeier [17]). Implementing appropriate discrete search procedures, they explore the space of all possible directed acyclic graphs and are capable of finding the top-scoring structure for a given dataset. Additionally, they propose a model averaging approach that allows them to learn gene subnetworks based on high-confidence features extracted from the data. For this, the search over the space of candidate high-confidence features is enhanced through the use of a second scoring function based on statistical significance and the use of a seed-based approach.

Their whole strategy has been summarized in a recent article by Pe'er published in Science magazine [20]. Figure 1.17 shows some of the networks that they have been capable of learning using 300 full-genome profiles from the yeast *Saccharomyces cerevisiae*.

Their study also provides measures of statistical significance that were carried out using randomization procedures over the original dataset as well as simulations with synthetic data. These are shown in figure 1.18.

Finally, they provide a comparison of the performance of their Bayesian network approach and other concurrent strategies (fig. 1.19). For this, they established a "true" model using commonly accepted

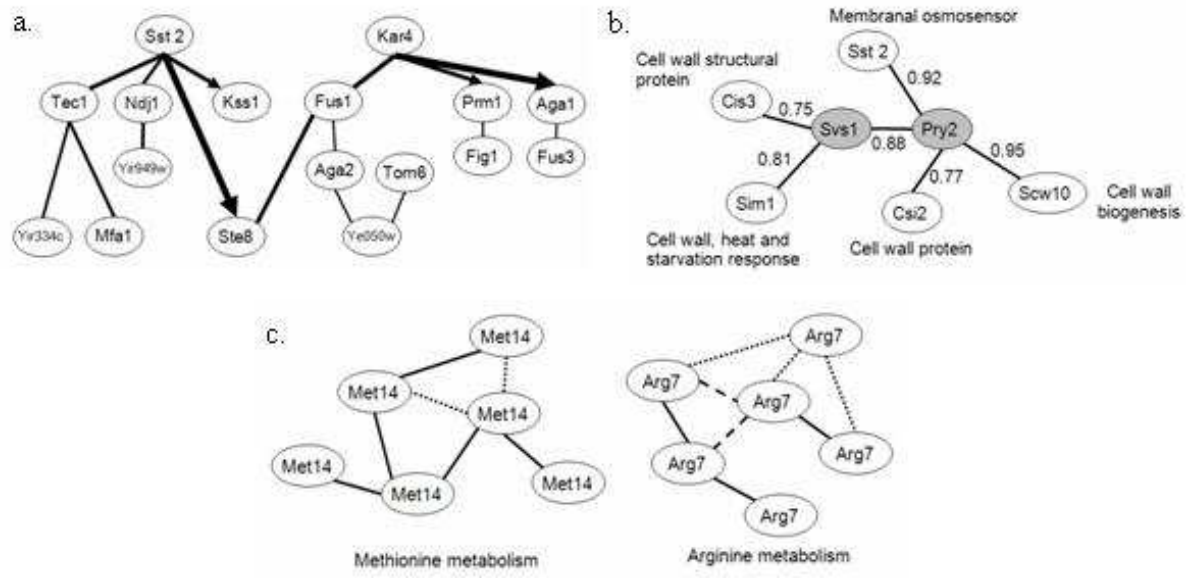

Figure 1.17: Samples of learned subnetworks from Pe'er's PhD dissertation [2]. Dataset: 300 samples, 100-fold bootstrap. (a) Gene subnetwork for mating response in Saccharomyces cerevisiae (learned over a subset of 565 genes using a seed approach). The widths of the arcs correspond to features confidence and they are oriented only when there is high confidence in their orientation. (b) Example of sub-network used for hypothesis generation. (subset of 947 genes, seed approach) Numerical values represent confidence levels. At the time of the study, the role of genes Svs1 and Pry2 was unclear. It was later discovered by biologists that they are related to cell wall processes, as could have been expected from a simple observation of the learned sub-network. (c) Results for amino-acid metabolism fragments using a 0.75 threshold for feature confidence. (subset of 947 genes, seed approach) Solid lines represent correct learned edges, dashed lines represent incorrect ones and dotted lines stand for missed edges w.r.t. commonly accepted biological relations.

gene interactions gathered from literature and compared the number of true positives as a function of false positives that were obtained using the different methods.

Their results seem to suggest a double-sided conclusion:

- On the one hand, it seems evident that the correspondence between the learned structures and the structures of reference is statistically significant; the accuracy is beyond anecdotal. Also, Bayesian networks seem to offer a better performance than their competitors in the comparison study.
- On the other hand, however, the final results and their practical implications are far from ideal. In order to guarantee an acceptable level of false positives, the number of true positives that can be learned must be kept very small. In other words, features with a very high confidence are rarely false, but the need to impose such high confidence levels prevents many true relations from being accepted.

As mentioned before, Hartemink et al. also used the same score-metric and the same basic underlying ideas for learning structure. In their paper [16] they present an example of a case where Bayesian networks were used to confirm a biological hypothesis (figure 1.20).

The work of Hartemink et al. is very interesting because they show an application of the Bayesian network approach for practical purposes considering a very simplified scenario. They sacrifice comprehensiveness in order to achieve robustness: they work with 52 samples and they study a set of only three genes, finally getting to determine the probabilistic independence or lack thereof between two of them.

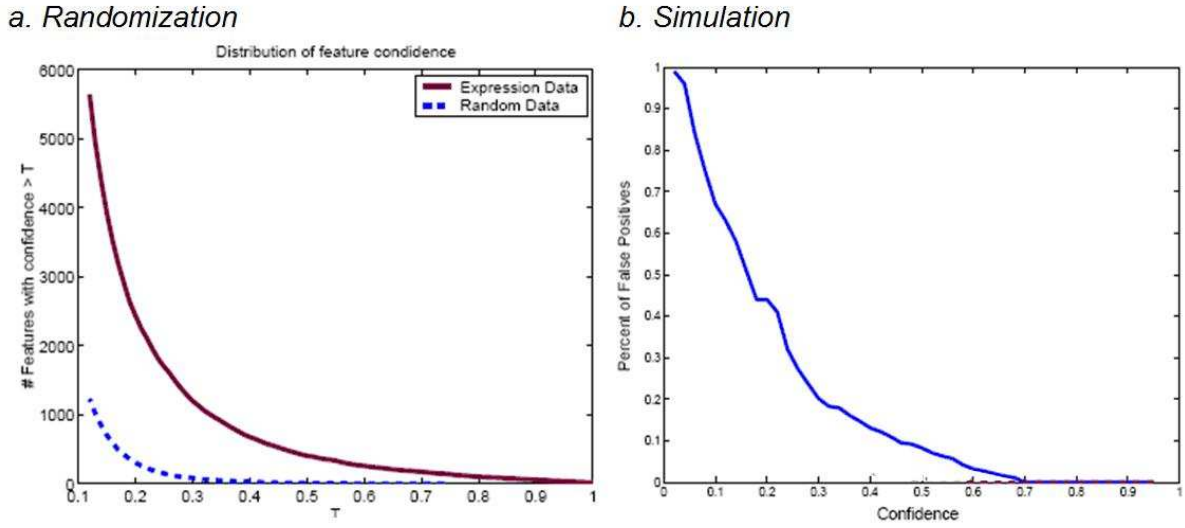

Figure 1.18: Randomization and simulation experiment results, from [2]. (a) A new dataset was generated by permutation of the observed values independently for each gene through the 300 full-genome original dataset (permutation of "experiments" or "columns" in the microarray). In such a new dataset, all genes are independent of each other and there are no "real dependecies". The graph shows the differences in the feature confidence distribution between random and real data. (b) A Bayesian network was learned from real data and then it was used to generate 300 synthetic samples. The graph shows the rate of false positives in this synthetic dataset as a function of the confidence threshold.

Within this context, their results appear to be sufficiently reliable and therefore they present a case of study where their theoretical approach provably leads to biologically sound conclusions.

Finally, Dirk Husmeier also seems to arrive at some conclusions that point in the same direction in a related article [17], where he carries out some simulations using synthetic and "synthetic realistic" data (that he generates using differential equations taken from the biological literature). His methodology though is actually somewhat different since he works with dynamic Bayesian networks and focuses on the analysis of time series for gene expression values.

To sum up, it seems that previous work carried out by different authors on the field of Bayesian networks applied to learning gene interaction networks from microarray data have been able to prove that some certain degree of learning can be accomplished and that this degree varies with several factors, such as the training size, the number of genes considered and the complexity of the target network. In practical applications, however, because of the small sample regime, the abundance of spurious edges makes it very difficult to extract true interactions and so biologists wanting to use these techniques are usually forced to either reduce the scope of their investigation to a limited subset of preselected genes or accept a compromise between the number of true edges that they attempt to infer and the price in terms of possible spurious edges that they may be willing to pay.

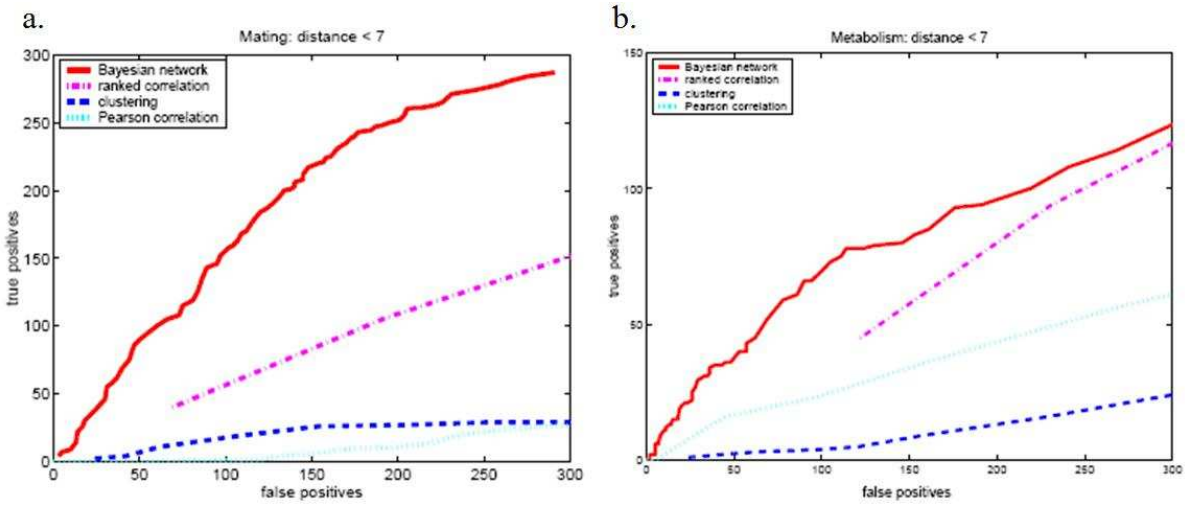

Figure 1.19: Performance comparison for Bayesian networks, clustering, correlation and ranked correlation, from [2]. A reference network is "hand constructed" from biological literature and used as ground truth. A false positive represents a pair of genes that are connected in the learned network, but unconnected in the reference network. A true positive is a pair of genes that are connected by a path of length at most 6 in both networks. (a) Results for mating network (learned over a subset of 565 genes using a seed approach). (b) Results for AA metabolism network (subset of 947 genes, seed approach)

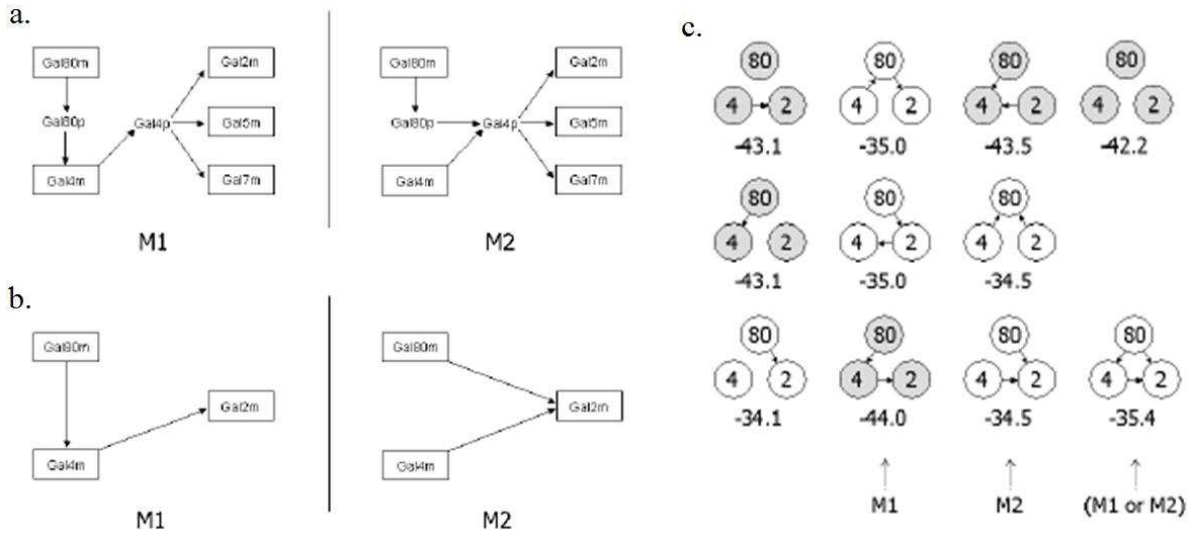

Figure 1.20: Case of study described by Hartemink et al. [16]. (a) Two competing models of representative Bayesian networks for describing a portion of the galactose system in yeast. M1 was originally accepted in biological literature, but it was later replaced by M2 following experimental discoveries. (b) Simplified version of the conditional independence assertions that distinguish models M1 and M2. (c) Bayesian scores for all model equivalence classes of the simplified three variable system. The models can be divided in two groups, where those who include a direct edge between Gal80 and Gal2 obtain a substantially better score. This lends support to the claim that Gal80 and Gal2 are not conditionally independent given Gal4, and therefore to model M2.

Chapter 2

Experiments and results

This chapter presents the simulations and experiments carried out during my internship and so it constitutes the core of this report.

All simulations were made using Matlab 7. Appendix B provides a description of the code that I wrote for their implementation, including:

- General coding conventions such as the format used to specify the networks and their parameters.
- A description of the Matlab algorithms that were written and used to generate the synthetic samples and the priors, to calculate the score, to automatically determine the class of equivalence of a given network, to check for the absence of directed cycles within a graph, etc.
- A description of a graphic interface for the case of 4 variables, that was created to facilitate both experimentation and result visualization for the different proposed simulation scenarios.

Readers are therefore invited to browse through this appendix in order to grasp the more technical details of the experiments, so that here we will mainly focus on the mathematical concepts under study.

As a general guideline, all the experiments described in this report share the following common mathematical choices:

- The observed samples are always considered to be binary, taking the value 0 or 1.
- The conditional probability distributions are always discrete multinomials (over binary variables). For each node X_k there is a parameter θ_{ijk} that represents the probability of variable k taking value i when its parents in the network take value j (with j being one of their possible $2^{numParents_k}$ combinations) .
- The sets of priors that have been used are the same for all the simulations:
  - A uniform prior has been considered for all possible structures, making them all a priori equally probable.
  - For the choice of parameter priors, a Dirichlet distribution has been used, with hyperparameters obtained by counting over a set of prior imaginary samples that included a representative for each possible vector of node values. The size of this prior was therefore always equal to $2^{numNodes}$. Figure 2.1 illustrates this point.
- Finally, in this context of multimomial distributions and Dirichlet priors for all the experiments, the scoring function used in all of the simulations is the one proposed by Friedman et al. (eq. 1.7 and 1.13).

3 variables	4 variables		5 variables			
000	0000	1000	00000	01000	10000	11000
001	0001	1001	00001	01001	10001	11001
010	0010	1010	00010	01010	10010	11010
011	0011	1011	00011	01011	10011	11011
100	0100	1100	00100	01100	10100	11100
101	0101	1101	00101	01101	10101	11101
110	0110	1110	00110	01110	10110	11110
111	0111	1111	00111	01111	10111	11111

Figure 2.1: Choice of prior datasets intended to formalize uniform prior beliefs. The hyperparameters a_{ijk} are obtained by counting over them (ex. $\{X_1 = 0, X_2 = 1, X_3 = 1\} \equiv$'011').

2.1 Score efficiency and network "learnability"

The first set of experiments was intended to study the number of samples that would be necessary to observe in order to learn a certain Bayesian network with a certain reliability, as well as the influence of several intrinsic characteristics of the target network, such as the number of nodes under consideration, the characteristics of the underlying conditional probability distributions, the presence of noise, etc.

The main goal of these simulations was to study the conditions that need to be imposed to a certain dataset for it to be considered informative enough to allow the learning of individual, global networks (since we will not consider any kind of model averaging strategies in this section) given a certain scoring function and a certain set of priors (both for the parameters and for the structure).

2.1.1 Influence of the number of nodes

We began by studying the effect that the number of nodes has over the required sample size for the observed dataset.

Experiment 1.

- We chose four different Bayesian networks, presented in figure 2.2. The first three structures show an increasing order of complexity built over the same initial pattern. The fourth structure is the same as the first one, but two unconnected nodes were introduced as "dummy" or "noisy" variables. We wanted to study the number of samples needed by the learning procedure to identify these variables and recover the true relations.
- We fixed a value for their conditional probability parameters θ, so that:
  - When a node X_k had a single parent X_j, $\theta_{10k} = P(X_k = 1|X_j = 0) = 0.82$ and $\theta_{11k} = P(X_k = 1|X_j = 1) = 0.18$.
  - When a node X_k had two parents X_{j1} and X_{j2}, then $\theta_{100k} = P(X_k = 1|X_{j1} = 0, X_{j2} = 0) = 0.15$, $\theta_{101k} = 0.3$, $\theta_{110k} = 0.7$ and $\theta_{111k} = 0.85$
- For each of the 4 networks under study and for each sample size, we generated 100 datasets of synthetic samples.
- For each of these datasets, we defined the reference score as the score of the true generating network and we performed a direct search over all the possible structures that could be built

using the given number of nodes for each case. If a structure not belonging to the equivalence class of the generating network received a score equal to or greater than the reference, we counted it as a failure and we went to the next dataset.

- Finally, we calculated the success rate (as 100 minus the number of failures) for each pair of structure and sample size. The results are shown in figure 2.3.

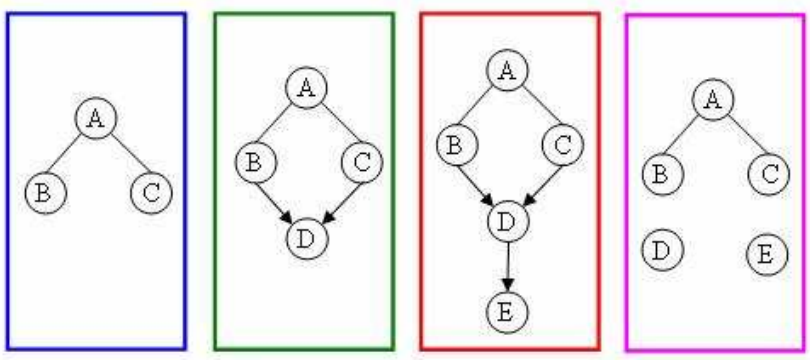

Figure 2.2: The four structures used in experiment 1.

The number of different structures that can be explored for n nodes or variables is $3^{(n(n-1)/2)}$. This implies 27, 729 and 59049 candidate models for the cases of 3,4, and 5 variables respectively. From this, we can directly eliminate all the models containing directed cycles (by using the algorithm to check for acyclicity described in appendix B), which leaves 25, 543 and 29281 candidates.

In practice, we can only make a distinction between equivalence classes, and so the spaces to explore can be reduced. From the literature, we found that the ratio of the number of equivalence classes to the number of DAGs calculated empirically seems to tend asymptotically to 0.267 [24]. Therefore, even though the number of total candidate graphs to explore can be somewhat reduced by taking these two aspects (lack of directed cycles and equivalence classes) into account, the exponential growth with n still makes it intractable to use direct search procedures for all but very small numbers of variables. This observation is at the origin of the use of discrete search procedures like those described in the next section.

If we look at figure 2.3, we see that for the first three networks there seems to be an initial phase during which the success rate experiences quick growth with respect to the number of samples. After it, this rate seems to become more stable and, even if it continues to grow in the long term, this growth is severely slowed down. This turning point in the behavior of the curve can be estimated at around 40-50 samples for the first network, at around 700-800 samples for the second and at around 1500-1600 for the third.

The fourth network under study also seems to present this tendency, although the transition is smoother and can be estimated at around 2500 samples. The results obtained for the third and the fourth networks, when compared to those of the first one, suggest that additional unconnected nodes need not be, in general, easier to learn than additional connected ones. In fact, it seems more difficult to spot the presence of new unconnected nodes in network four than it is to correctly learn the five-node connected structure of network three.

In any case, this first experiment clearly shows that the number of nodes has a direct effect over the necessary sample size to learn structure. As expected, networks containing a higher number of nodes imply more relations among them and are therefore more difficult to learn. Furthermore, these

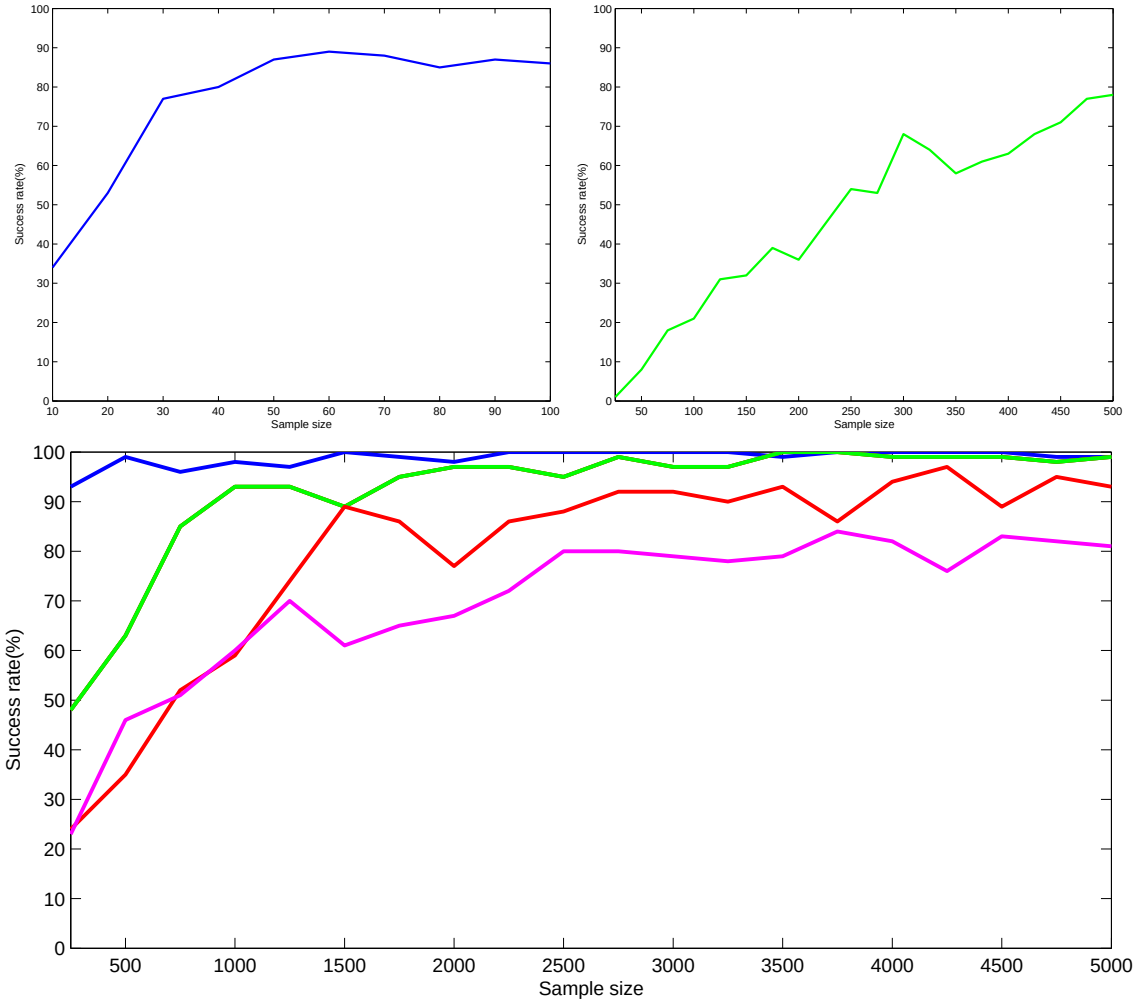

Figure 2.3: Results for experiment 1. Structures from figure 2.2 are considered, using the same color coding. Top panels show a close up of the results for the networks with 3 and 4 variables for small sample sizes. The graph shows the influence of the number of nodes over the Bayesian score performance and its evolution with the sample size.

simulations provide a valuable insight concerning the order of magnitude for sample sizes that achieve a certain success rate as a function of the number of nodes. For the networks in the example, it seems that we may reasonably attempt to learn certain structures containing 3 variables with 100 samples, 4 variables with 1000 samples and 5 variables with 3000 samples.

2.1.2 Influence of the choice of parameters for the conditional probability distributions

Having studied the effect of the number of nodes and the structure, we decided to also consider the influence that the values of the parameters for the conditional probability distributions of the target network have over the learning results. As mentioned at the beginning of the chapter, we always worked with discrete, multinomial distributions and binary variables.

Experiment 2.

- We decided to begin by studying the behavior of a single v-structure and we fixed a sample size of 200.
- For the conditional probability distributions of the common child, the following patterns were considered:

  1. OR: $\Theta_{100k} = P_{OFF}$, $\Theta_{101k} = P_{ON}$, $\Theta_{110k} = P_{ON}$, $\Theta_{111k} = P_{ON}$.
  2. AND: $\Theta_{100k} = P_{OFF}$, $\Theta_{101k} = P_{OFF}$, $\Theta_{110k} = P_{OFF}$, $\Theta_{111k} = P_{ON}$.
  3. XOR: $\Theta_{100k} = P_{OFF}$, $\Theta_{101k} = P_{ON}$, $\Theta_{110k} = P_{ON}$, $\Theta_{111k} = P_{OFF}$.
  4. INCREMENTAL: $\Theta_{100k} = P_{OFF}$, $\Theta_{101k} = P_{SEMI-OFF}$, $\Theta_{110k} = P_{SEMI-ON}$, $\Theta_{111k} = P_{ON}$.

  where $P_{ON} = 0.5 + \alpha$, $P_{OFF} = 0.5 - \beta$, $P_{SEMI-ON} = 0.5 + \alpha/2$ and $P_{SEMI-OFF} = 0.5 - \beta/2$
- In all cases, ten values of α and β where evenly sampled between 0.05 and 0.5 in 0.05 intervals.

Figure 2.4 shows the results that were obtained for the simulation. Actually, the figure for the cases OR and AND is the same because of symmetry considerations, and thus only three really different patterns are shown.

As expected, the success rate is zero or very low for values near total randomness (0.45 and 0.55). For the OR/AND and the incremental patterns, the highest success rates are obtained for the most deterministic values of the parameters (near 1 and 0). The results for the XOR pattern, however, seem to be less affected by the particular choice of the parameters since, except for the region near 0.5 for both of them, the variations in the success rate are not big in magnitude. It is interesting to point out that in this case, the maximum is not achieved for the most deterministic joint choices of parameters, but for the choices involving one of the parameters being almost deterministic while the other one is kept almost random (near 1 and 0.45 or near 0 and 0.55).

Having studied the case of a single v-structure, we decided to look at a slightly more complex structure involving 4 variables and we repeated the experiment with the second structure from figure 2.2. We now fixed a sample size of 500 samples and we defined the parameters for nodes having a single parent as: $\Theta_{i10} = 0.5 + \alpha$, $\Theta_{i11} = 0.5 - \beta$.

The new results obtained for the same patterns are shown in figure 2.5. It is interesting to point out how the inclusion of the new parent node with respect to the simple v-structure has a strong impact in the shape of the success rate values.

The most striking change is that now, a very low success rate is always obtained for values corresponding to deterministic or near deterministic parameters (~ 1 and ~ 0), as opposed to the previous scenario.

The reason for this is that, when parameters become totally or almost totally deterministic, nodes 2 and 3 tend to take always the same values, which in turn makes the value of node 4 a deterministic direct consequence of the value of node 1. In this situation, a model incorporating edges between nodes 2 and 3 and between nodes 1 and 4 gets a better score that the "true" generating model. Furthermore, since the two central nodes always take the same value, the whole distinction between the XOR, AND/OR and incremental patterns loses its meaning, and the new shapes for the obtained results are much more similar to one another than before.

The conclusion from this is that, even if we can characterize the behavior of a certain pattern of parameters for an isolated v-structure, once it is inserted in a full, more complex Bayesian network, new dependencies may arise that can dramatically modify the actual observed pattern. This may lead

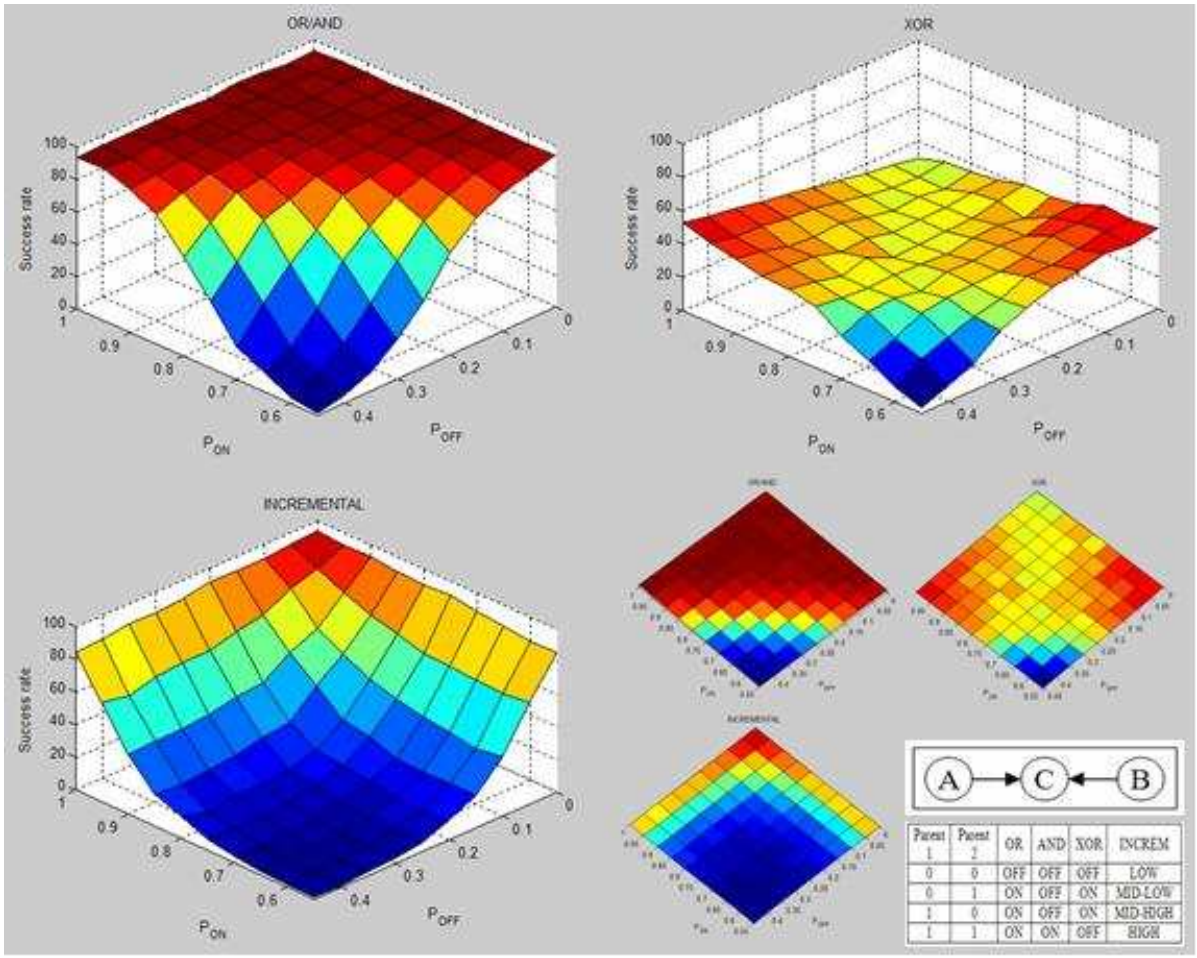

Figure 2.4: Results for experiment 2. Simple v-structure (bottom right). Figure shows the success rate as a function of the values for P_{ON} and P_{OFF} (which in turn measure the level of randomness for the conditional dependence distributions) for three different patterns (OR/AND, XOR and INCREMENTAL).

to situations in which the final learned model is not exactly the same as the one that generated the network, but it is capable of explaining the data equally well. This seems to be more so when very deterministic parameters are chosen. This might be considered a drawback of the Bayesian network approach, since in some situations like the ones we just discussed, these tools would prove themselves incapable of establishing the conditional independence of node D with respect to node A given nodes B and C, therefore jeopardizing the inference of some possible biologically meaningful relations.

2.1.3 Influence of the presence of noise

As we mentioned in the introduction, microarray measurements are still in a relatively early phase of technological development and, even though quick progress in the field is continuously being made, the simulation of noisy data seems an interesting aspect to be taken into account.

This is why we decided to run an experiment to have an idea of the effect of a certain noise probability over the performances of the scoring function under study.

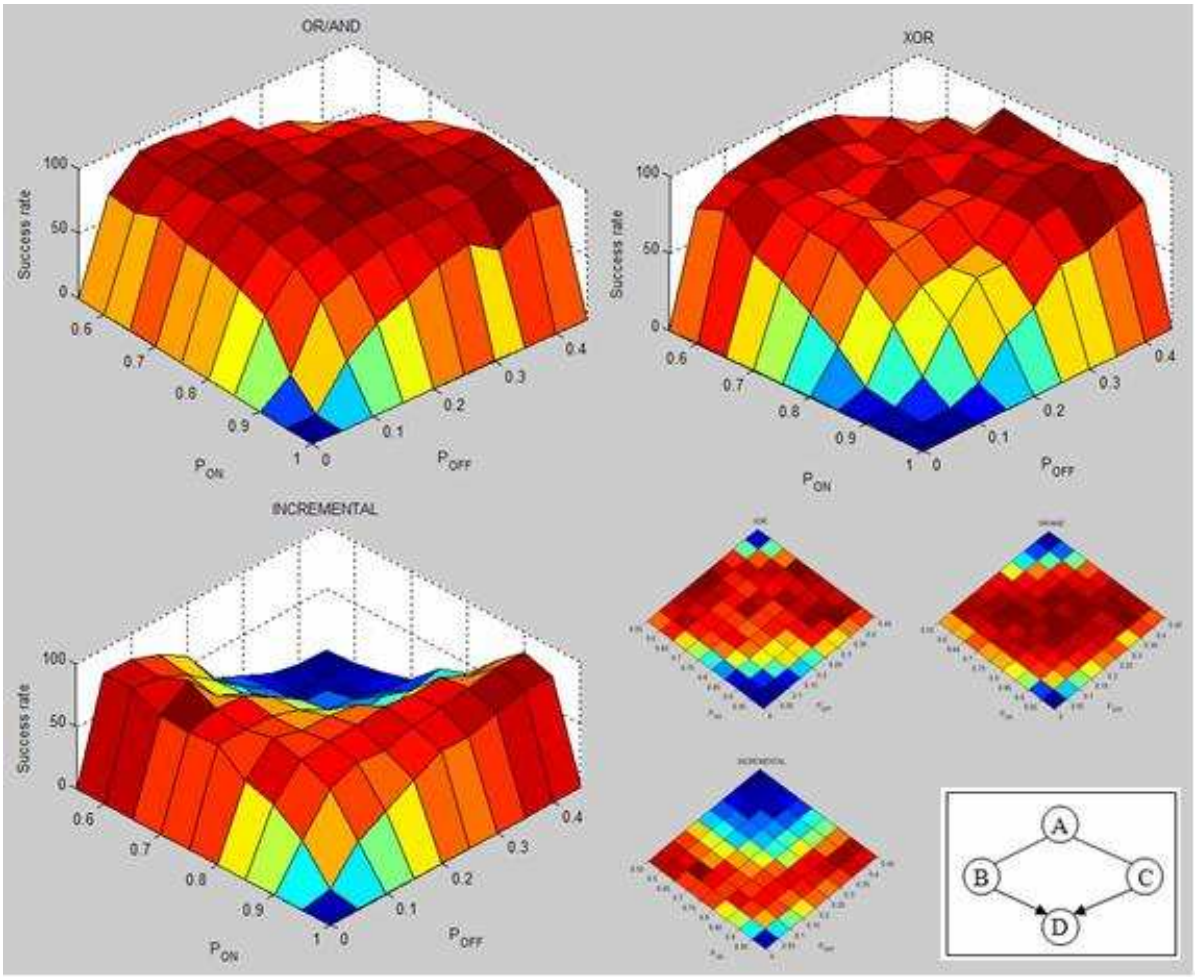

Figure 2.5: Results for experiment 2. Network containing 4 nodes (bottom right). As before, figure shows the success rate as a function of the values for P_{ON} and P_{OFF} (which in turn measure the level of randomness for the conditional dependence distributions) for three different patterns (OR/AND, XOR and INCREMENTAL). However, we observe that the addition of a single variable has a big effect over the shape of the performance surfaces.

Experiment 3.

- We chose to use structure two from figure 2.2 and the same values for the choice of parameters for the conditional probability distributions that were described in experiment 1.
- This experiment comprised two different simulations:
  1. First, we introduced noisy samples by randomly flipping individual values of a synthetic dataset with different probabilities (P_{FLIP}, a value of $P_{FLIP} = 0.01$ meaning that approximately one out of every one hundred samples is flipped). We studied the effect that this had on the success rate curves over a certain range of sample size values. Results are shown in figure 2.6 (top panel).
  2. After this, we considered the effect of introducing a fixed number of completely random samples (where all nodes take values 0 or 1 with a probability of 0.5) and studying the

influence that they had in the evolution of the success rate over a certain range of sample sizes. The idea was to see how many additional samples were needed to recover from the introduction of these "misleading" sample subsets (which were always the first samples of each considered dataset). Results are shown in figure 2.6 (bottom panel).

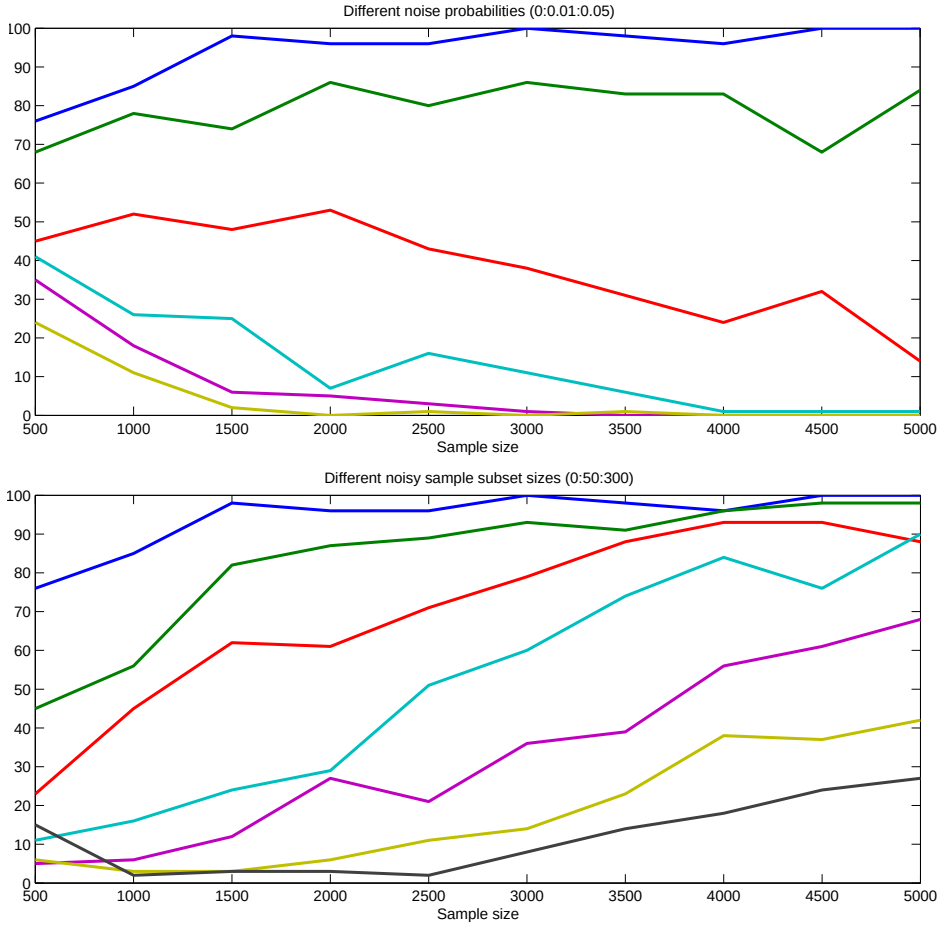

Figure 2.6: Results for experiment 3. Both panels show the evolution of the success rate with the sample size. In top panel, 6 different error probabilities are considered [different values of P_{FLIP}], ranging from 0 (blue line, top) to 0.05 (yellow line, bottom) in 0.01 intervals (curves with higher success rates correspond to sequentially lower error probabilities). In bottom panel 7 different sizes of an initial set of noisy (random) samples are considered, ranging from 0(blue line, top) to 300(grey line, bottom) in 50 sample intervals (again, the correspondence is sequential).

Interestingly, these two simulations may reflect two different practical scenarios:

- The first one, for example, might correspond to a situation in which the digital scanner for the microarray has a defect that makes it read a wrong value with a certain probability that affects all read values. In these circumstances, the results seem to show that getting more samples does not help solve the problem. On the contrary, not only does the success rate not improve with bigger sample sizes, but it decreases strongly (especially for values of $P_{FLIP} > 0.02$). It seems that when P_{FLIP} is very, very small, having more samples may help, but there is a certain threshold over

which, having more samples also implies having more errors, and the number of these surpasses the advantages of the bigger dataset.

- The second scenario, might correspond, for example, to a situation in which a punctual error has been made on the measuring process (imagine a defect of fabrication in one area of one of the single microarrays used in the experiment). However, the new measurements that are added after the initial error do not have this problem, and so the results show that, when new samples arrive, the score is capable of overcoming the difficulties associated to the noisy samples and the success rate grows with the global sample size.

In addition to this, the the first scenario could also be seen as an strategy to simulate the introduction of discretization noise. In practice, the levels of gene expression provided by microarrays measurements take continuous values that need usually be transformed into discrete quantities before being processed within the context of Bayesian networks (or at least they do when these are based on multinomial distributions). When raw data from the microarray is observed, there is not such a thing as a "expressed" (1) or "unexpressed" (0) genes, but only real numbers taking values within a certain range. A very straightforward approach would consist on finding the median of all values and consider that genes with higher values are expressed and genes with lower values are unexpressed. We might even prefer to consider three levels corresponding to genes that are underexpressed, overexpressed or "normally" expressed. Several approaches have been proposed in the related literature to address this discretization task, including statistical methods, clustering procedures and alike. In any case, it seems that the discipline is still at early stages of development and no real standards have yet been agreed upon. Simulations like the ones presented here stress the importance of making a right choice.

Because of all the above, and even though they are very simple, these experiments provide an interesting insight to the priorities that would need to be addressed by microarray technology experts if the Bayesian network formalism were to be adopted.

2.1.4 Further commentaries on the Bayesian score

We will finish this first section that deals with learning individual networks by presenting a final experiment on the behavior of the Bayesian score.

From a mathematical point of view, the Bayesian score has been proved to be consistent ([34]). This means that when the number of observed samples grows towards infinity, the real generating structure G* is guaranteed to maximize the score and all other structures that are not equivalent to G* will have a strictly lower score. Having this in mind, we decided to compare the scores achieved by the equivalence class of the real generating structure and the rest as a function of the number of observed samples for a very simple case.

Experiment 4

- Again, we chose to work with the second structure of figure 2.2 and the same parameters from experiment 1.
- Using this network, we generated a set of 100 datasets and we plotted the normalized average score obtained for all the 543 possible DAGs that can be constructed with four variables.
- We looked at the difference in these average scores for two datasets of 500 and 5000 samples.
- We also calculated the minimum and the maximum scores over the 100 datasets for each sample size.
- Finally, we calculated the score for sample sizes in the range of 500 to 5000, with intervals of 500 and we computed the variance.

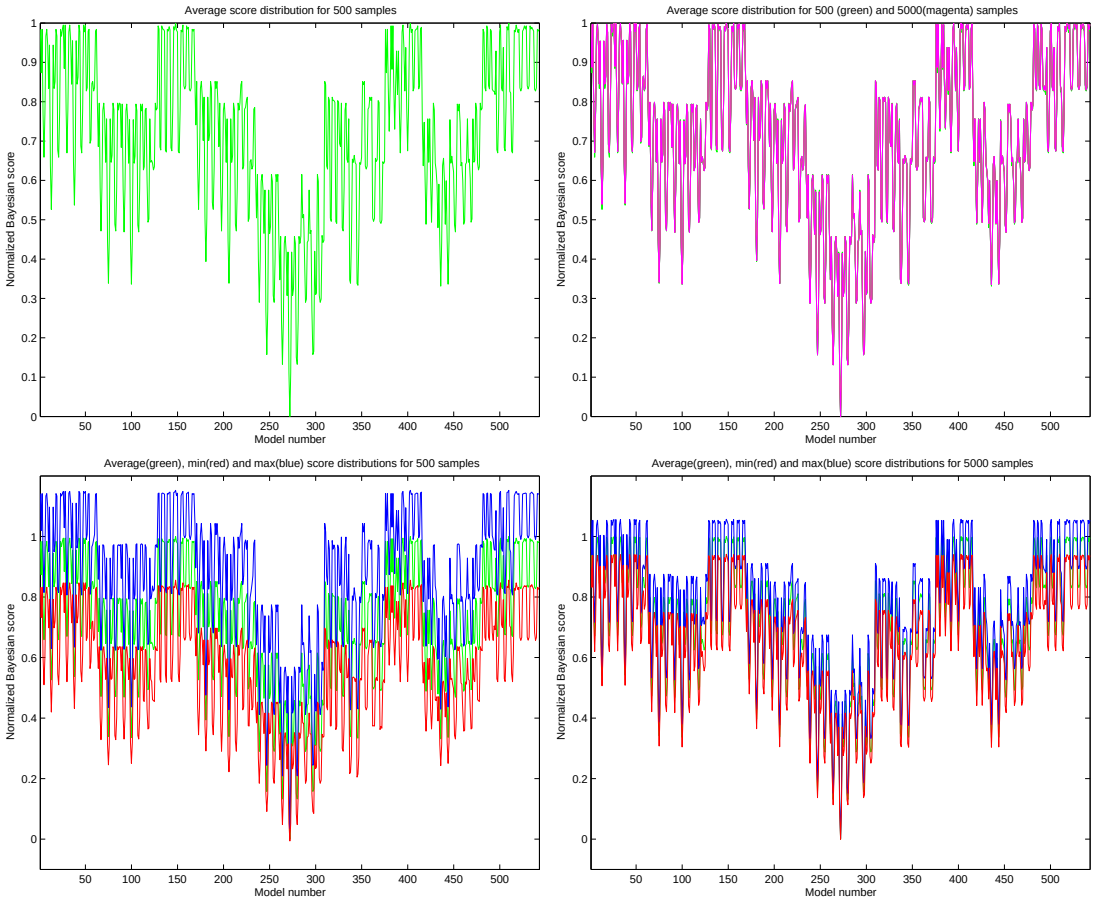

Figure 2.7: Results for experiment 4. Top panel shows the normalize score for all possible networks containing 4 nodes averaged over 100 datasets generated with a reference model (structure 2 from figure 2.2). Left panel shows the results for 500 samples (green) and right panel shows the results for 500 (green) and 5000 samples (magenta,superimposed and almost identical to 500). Bottom panel shows mean (green), min(red) and max(blue) score values (over 100 datasets) for the case of 500 (left) and 5000 (right) samples. We observe that the main difference between the results obtained for both sample sizes is the diminution of the standard deviation for the score.

Figure 2.7 (top) shows the results for the average scores for sample sizes of 500 (green) and 5000 (magenta). Since they are almost identical, the magenta curve is superimposed to most of the values of the green one on the right panel. It thus seems that the average scores are very similar for all models in these two situations.

However, if we represent the minimum score (green) and the maximum score (blue) together with the mean value (green) we observe a big difference between the 500 and the 5000 samples datasets (figure 2.7 bottom panels). This is further illustrated in figure 2.8, where the variance of the score is shown as a function of the sample size.

We might have thought that, for very small sample sizes such as the one of 500 samples, some models not belonging to the equivalence class of the true one could have an average score greater than the reference model, and that the effect of observing more samples allowed to progressively invert this disequilibrium. This would provide an intuitive explanation to the low success rates for small samples.

The results obtained, however, seem to point in a different direction, since they suggest that the fact of having a bigger dataset does not have a direct effect on the global average for the score. However, it helps to progressively reduce the range of variability for the obtained scores around the average, and this would be the explanation for the growth of the success rate with the sample size observed in the

previous experiments.

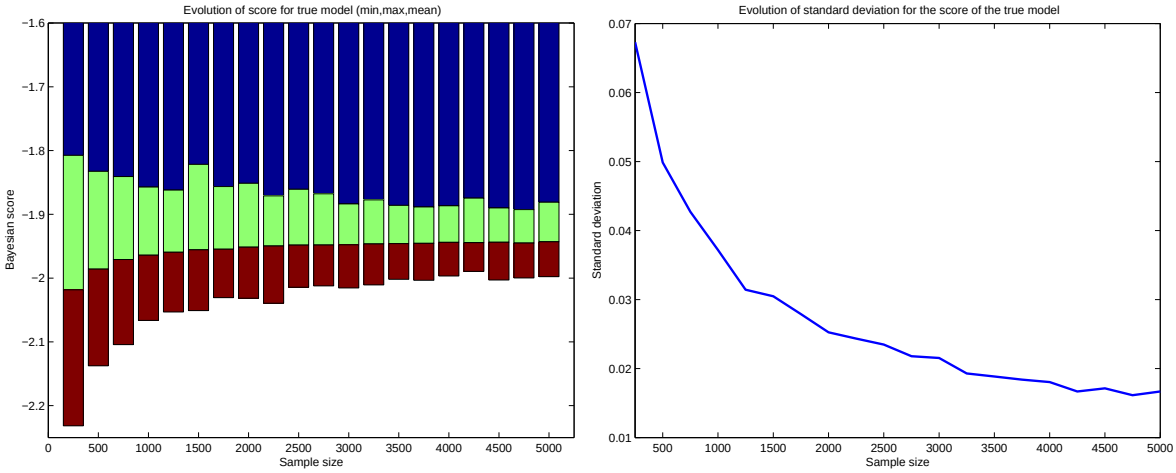

Figure 2.8: Results for experiment 4. Left panel shows the evolution of the mean(green),min(red) and max(blue) score for the true generating model between 500 and 5000 samples in 500 sample intervals. Right panel shows the evolution of the standard deviation. These results are consistents with the previous figure.

In any case, it seems evident that there must be a certain threshold on the sample size under which the average score distribution differs from the one that permits to correctly learn the good model (otherwise, we could just split a single dataset of 500 samples in 50 datasets of 10 samples and compute the average in order to learn the right model). This fact, together with its relation and implications for the bootstrap approach that will be presented later on are a possible subject of future research.

2.1.5 Partial conclusions

The main conclusions derived from the simulations discussed in this section, can be summarized as follows:

- Certain aspects such as the number of nodes of a Bayesian network seem to have an important influence in the degree of difficulty associated to its learning. As the number of nodes increases, the number of possible candidate networks that need to be taken into account grows exponentially and this has a double effect over the learning process:
  - On the one hand, there is a combinatorial explosion on the number of structures that need to be explored, and this makes the search for the good structure much more computationally expensive.
  - On the other hand, bigger samples sizes are required to learn the correct network, since having more nodes implies having more structures that closely resemble each other and that are capable of obtaining similar scores when the number of samples under observation is not big enough.
- Another factor that has an important influence over the success rate when attempting to learn a Bayesian network is the choice of parameters for the conditional probability distribution. As we have seen in experiment 2, both the ON-OFF pattern and the choice of actual values for the parameters (closer to deterministic relations or closer to total randomness) have a big impact in the learning process. Furthermore, when global network are considered, certain combinations of structure and parameters may become particularly difficult to learn, since they may give rise to situations where, for example, certain nodes almost always take the same values and this obscures

their pattern of interaction over their children (if two parents always take the same value, what was supposed to be an XOR may degenerate into a deterministic relation between parents and child).

- Our experiences involving the simulation of added noise show that Bayesian networks seem capable of recovering from the addition of a fixed number of noisy samples. However, when a fixed noise probability ($P_F LIP$) is considered and when it goes over a certain threshold, the arrival of new samples is incapable of improving the learning performance (since more samples also implies more errors).
- Finally, we have observed how, for the range of sample sizes considered in experiment 4, an increase on the sample size is beneficial to the learning problem not actually by providing a higher mean value to the score of the true model and a lower mean value for the rest (as we might have though by looking at the definition of score consistency, see appendix A), but by providing a progressive reduction of the standard deviation for all the individual scores that results in the true model getting the best values more and more often.

2.2 Discrete search methods

As we mentioned in the previous section, the number of possible structures that can be built with n variables is given by the expression $3^{(n(n-1)/2)}$. By discarding graphs that contain directed cycles and by working with equivalence classes, the number of total structures to be considered can be divided in practice by a certain factor, but the exponential growth remains there.

This is why the approach of direct search, which consists in exploring all the possible structures in order to select the one with the highest score, becomes intractable for situations involving more than just a few nodes. Since the field of discrete search is a relatively well-known one, some of its classical strategies have been proposed to overcome this difficulty. In this section, we will look at two of the most popular ones: greedy search and simulated annealing, together with the Sparse Candidate algorithm, which is a heuristic proposed by Friedman et al. [25] and intended to optimize the basic greedy search procedure.

2.2.1 General comparison

To begin with, we decided to test all the four general strategies over the same learning task, in order to get a basic idea of their general performances.

Experiment 5.

- We chose to work with four variables first and with five variables later and we selected the third and fourth structures from figure 2.2. The choice of parameters for the conditional probability distributions was also the same as in experiment 1.
- We generated synthetic datasets using these structures for each of the different sample sizes under study and we used the four approaches to learn the best network for each dataset.
- For the three new discrete search methods, if the learned network belonged to the equivalence class of the generating structure, we considered it a success and otherwise we counted a failure. For the direct search approach the results are the ones we obtained in experiment 1, where we counted a failure if there was a structure that obtained a score equal to or greater than that of the generating structure and did not belong to its same equivalence class.

Figures 2.9 and 2.10 show the results obtained for the experiment.

In the case of the network with four variables all search procedures seem to offer similar performances (which might be due to the fact of dealing with a very simple network).

In the case of the network with five variables, direct search, greedy search and sparse candidate seem to offer similar performances in terms of success rate, with direct search behaving slightly better for sample sizes over 2500. Simulated annealing, seems to get a lower success rate, but the fact that it was executed "on its own" must be taken into account. In practice, the execution on simulated annealing is frequently followed by an additional greedy search and their combined action usually equals or surpasses either of them when used isolated.

In terms of speed and computational complexity, greedy search and sparse candidate clearly outperform the other two approaches. When it comes to direct search and simulated annealing, it is interesting to observe the inversion of their positions in the ranking. In effect, whereas for 4 variables direct search is much faster than simulated annealing (with epsilon=0.001 and maxStalls=200), for 5 variables direct search becomes much lower. This reflect the exponential growth in complexity associated to this approach, which constitutes its main critical drawback. Even though simulated annealing begins as a relatively "slow option", its sensitivity to the number of considered variables is much lower than that of

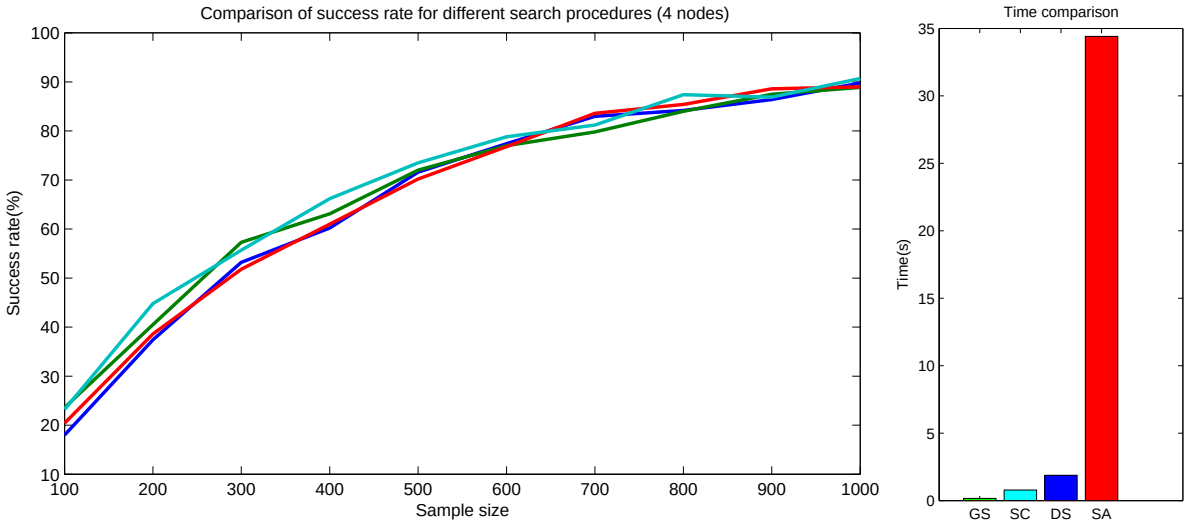

Figure 2.9: Results for the network containing 4 nodes. Left panel shows the evolution of the success rate with the sample size for direct search(blue), greedy search(green), simulated annealing(red) and sparse candidate(cyan). Right panel shows a comparison of the time that it takes a network from a given dataset of 1000 samples using the four procedures. Results for GS and SC were averaged over 1000 executions. Results for SA and DS were averaged over 500 executions. (Parameters for SA were epsilon=0.001 and maxStalls =200, for sparse candidate the parent candidate size was fixed to 2; simulated annealing was used "on its own", not followed by the execution greedy search).

the direct search procedure. The bigger the number of nodes, the poorer the direct search performance with respect to the others is clearly bound to become.

The version of the Sparse Candidate algorithm implemented in our code (see appendix B) is very similar in spirit to the basic greedy search approach, but it includes a refinement that consists in limiting the number of possible candidate parents for each node at each iteration. Using mutual information measurements, a subset of candidates is selected and then the greedy search is restricted to this subset. The gain in terms of complexity reduction cannot be observed for small numbers of nodes such as the ones considered for the experiments described in this report. In its original context, however, Friedman et al. proposed to use it in scenarios involving near 1000 variables. In this situations, selecting a subset of, say, ten or twenty candidate parents for each node can be highly beneficial, since it can speed up the search while maintaining the performance in terms of success rate. The critical point is, in any case, to guarantee that the set of candidates that are selected always contains the right ones. In our very simple scenarios, this actually seems to be the case.

2.2.2 Simulated annealing

Even though looking at figure 2.10. we may get the impression that simulated annealing, when used alone (without a final execution of greedy search), is not a very useful approach, since its performances seem to be worse than the rest and its time consumption is quite high, this strategy offers certain advantages that need to be taken into account.

The main one is the capability of customizing the trade-off in terms of precision and computational complexity that we want to achieve. In one extreme, direct search is guaranteed to find the top scoring structure, but the number of possible combinations that need to be explored soon becomes intractable. On the other end, greedy search is very fast and computationally inexpensive, but for certain applications we may need to dig deeper in search for the good structure.

Covering the space between the two, simulated annealing can be adjusted to specific application

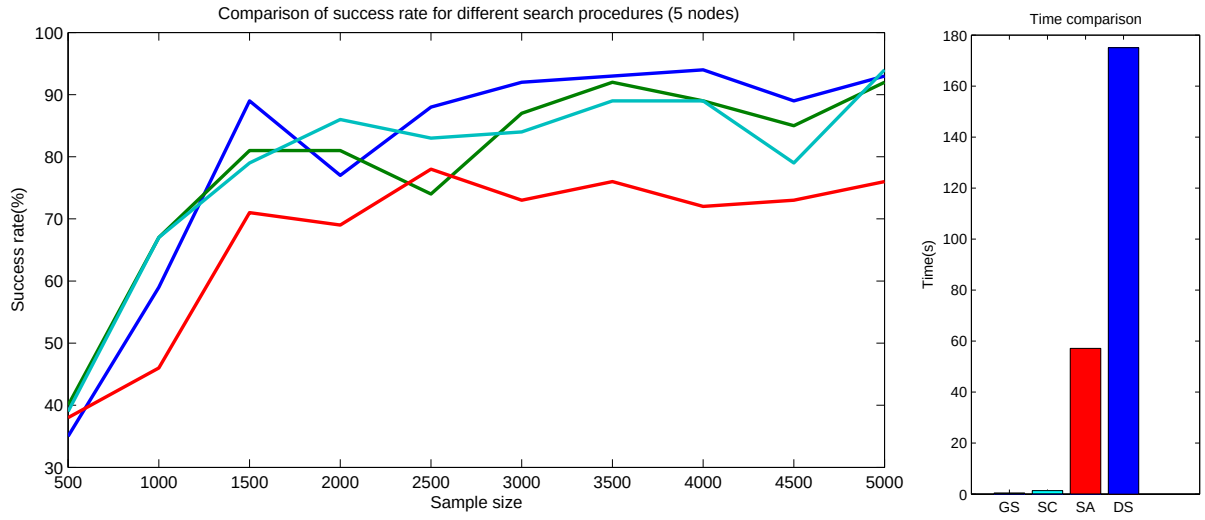

Figure 2.10: Results for the network containing 5 nodes. Same notation as in previous figure. In this case, results were averaged over 100 executions. The dataset used for time measures also contained 1000 samples. (Again, parameters for SA were epsilon=0.001 and maxStalls =200, for sparse candidate the parent candidate size was fixed to 2; simulated annealing was used "on its own", not followed by the execution greedy search).

requirements through two basic parameters: epsilon and maxStalls. Epsilon controls the diminution rate for the probability of accepting a change that does not constitute an improvement in terms of score (according to an expression of the type $(1-\epsilon)^{numiter}$. MaxStalls is the maximum number of consecutive times that the search procedure can be stuck (because the proposed change was not accepted) before it finally comes to a halt. For more details, see appendix B.

Experiment 6.

- We chose the same four variable structure and the same set of parameters of experiment 1.
- We generated 100 datasets of different sample sizes and we learned a network for each of them using simulated annealing:
  - On the first phase of the experiment, with fixed epsilon=0.01 and studied the evolution with maxStalls between 50 and 300 in 50 unit intervals.
  - On the second phase of the experiment, we fixed maxStalls=200 and varied the value of epsilon between $10^{-0.5}$ and 10^{-3} in intervals of -0.5 for $log_{10}epsilon$.

Figures 2.11 and 2.12 show the results for this experiment.

As expected, when a sufficiently low value of epsilon is used (in our case 0.001), simulated annealing offers very competitive success rates that are as high as the ones obtained with the other procedures. Also, when the value of epsilon is big, the search procedure is quite fast. The trade-off mentioned before is therefore evident in these circumstances.

Apparently, the variation of epsilon (at least within the ranges considered in this experiment) has a much stronger influence on the number of iterations than the number of maxStalls, and this implies a much stronger impact on the observed success rate. Evidently, the average time is also affected since more iterations imply more time for each search.

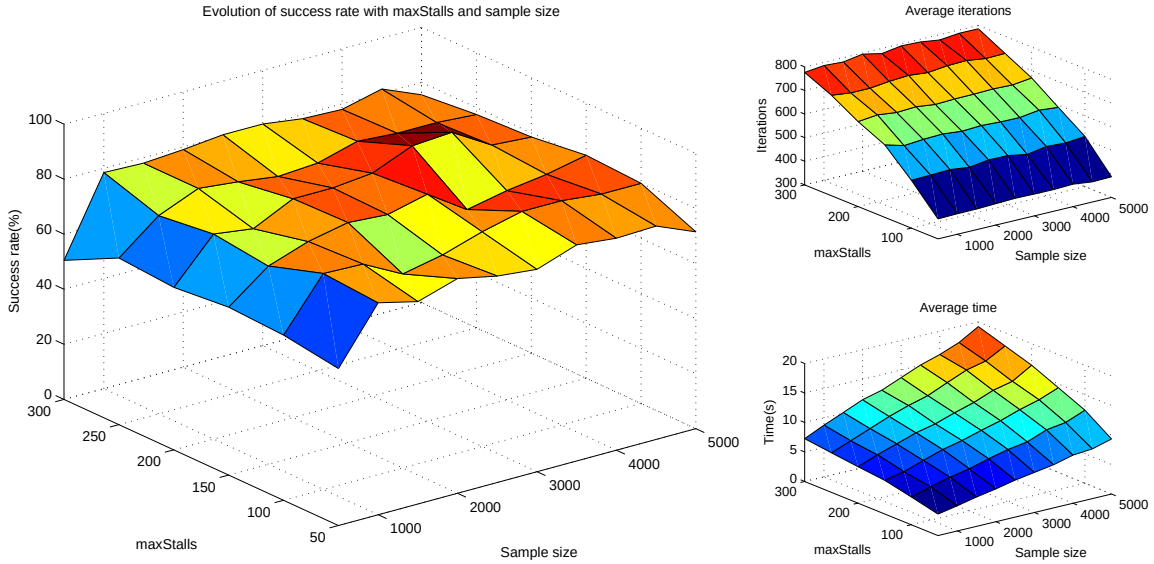

Figure 2.11: Simulated annealing: evolution of success rate (left), average number of iterations (top right) and average search time (bottom right) with variation of *maxStalls* and sample size.

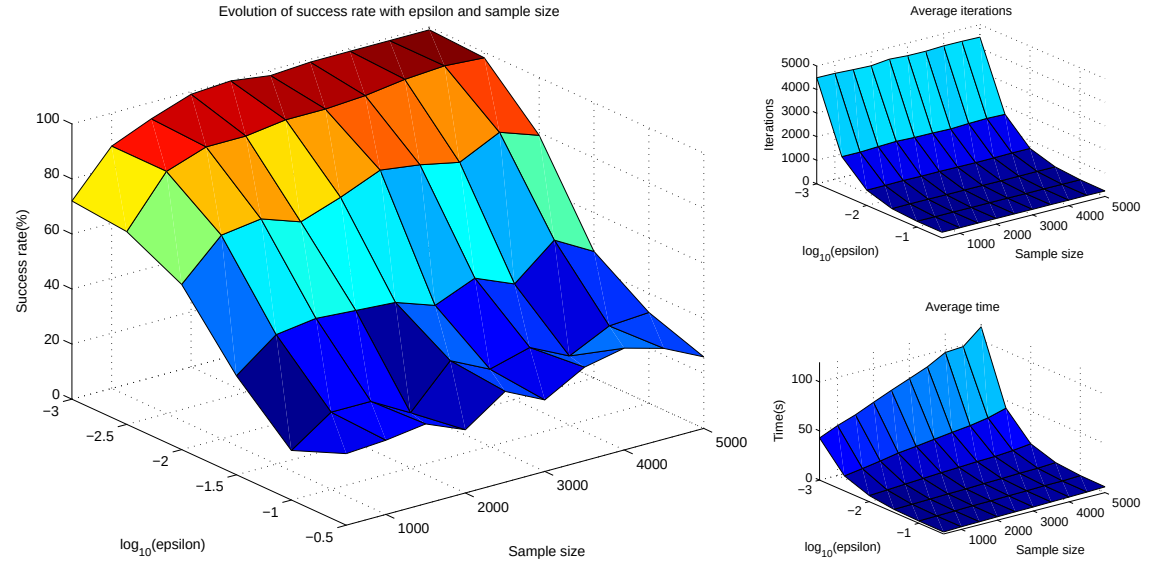

Figure 2.12: Simulated annealing: evolution of success rate (left), average number of iterations (top right) and average search time (bottom right) with variation of *maxStalls* and sample size.

2.2.3 Partial conclusions

After looking at the simulations carried out in this section, the following conclusions have been reached:

- In general terms, the field of discrete search is a relatively well-known one and different search methods are described in literature that could offer a good solution for the problem at hand. As we have seen from our experiments, the procedures considered seemed to perform reasonably well and seem to offer a reliable alternative to greedy search.
- Certainly, only very simple networks involving very few variables were considered, but we may hypothesize that the use of complexity reduction heuristics such as the Sparse Candidate algorithm and alike may offer the necessary functionalities to deal with the combinatory explosion of possible structures to explore while providing an acceptable level of degradation in the search performance with respect to the ideal direct search procedure.
- In my opinion, the current "accuracy bottleneck" associated to the learning problem is therefore closer to the conception of a good scoring function than it is to the development of efficient search procedures. When, as we saw in the previous section, certain true networks only obtain the highest score for half of the learning attempts for datasets generated by themselves, the fact of conducting a better search can do nothing to overcome this threshold.

2.3 Model averaging techniques

The results shown so far, as well as other results from literature, seem to suggest that it is not possible to learn complex networks from small samples just by searching for the single structure that maximizes the Bayesian score.

As we have seen, even for very simple networks containing only 4 or 5 variables, the fact of observing several thousand samples may only guarantee that the true generating structure will obtain the highest score in a limited percentage of attempts, which varies with the different factors described in the first section of this chapter.

It therefore seems that if we want to apply these techniques to real microarray data, with the subsequent small sample regime, other approaches need to be taken into account. One of the most popular choices, adopted by several researchers in the related literature (including Friedman et al., Hartemink et al. and Husmeier) is the use of model averaging techniques.

The basic idea behind this class of strategies is to learn more than one network from a given dataset and then try to extract features or relations that are shared by several of them. As before, the ultimate goal is to make the most of the information contained in the original data in order to obtain the highest possible degree of reliability for the learned structures.

For our experiments, we adopted the strategy of Friedman et al., who propose the use of non-parametric bootstrap for obtaining additional datasets from a single, original one. By learning the top scoring structure for each of these "bootstraped" datasets it is possible to obtain a set of candidate structures. A feature can then be defined as a concrete characteristic of one of these networks (such as the presence of an edge or the lack thereof) and it is possible to define the level of confidence in a feature as the percentage of candidate structures that contain it. By fixing a threshold for the minimum level of acceptable confidence, it is possible to select the most reliable features. This constitutes the basis for all the experiments described in this section.

For more details on the mathematical foundations of the model averaging approach, the interested reader is once again invited to refer to appendix A.

2.3.1 Choice of a threshold for the levels of confidence

We begun our study of model averaging techniques with a concrete experiment where we looked at the influence of choosing different values for the confidence threshold.

Experiment 7.

- We decided to work with the 5 node structure corresponding to the third network in figure 2.2. The parameters for the conditional probability distributions were the same as in experiment 1.
- We used this network and these parameters to create a single dataset for each of the sample sizes under study.
- For each of these sample sizes and for each value of the threshold α, the following steps were repeated 10 times and the results were averaged:
  1. 100 additional datasets were generated by bootstrap sampling from the original dataset.
  2. The top scoring Bayesian network was learned for each of these 100 datasets and its associated class of equivalence was determined.
  3. If an edge value (-1,0,1,2 - see below) was observed in more than $100 \times \alpha$ networks, it was considered a *positive*.
  4. If a positive edge value was the same as the edge value in the true generating network, it was considered a *true positive*. Otherwise, it was considered a *false positive*.

The four possible numerical values for each edge correspond to the criteria used for the Matlab implementation (presented in detail in appendix B). Basically, each network of n nodes is represented as a n-dimensional matrix M, where position M_{ij} corresponds to the edge that goes from node i to node j. Therefore, each edge can have four possible values:

- 1 if the edge is oriented from i to j
- -1 if the edge is oriented from j to i
- 0 if there is no edge
- 2 if the edge is undirected (remember that we are working with classes of equivalence)

As a consequence of this, the matrix representing each network is determined by the contents of its upper half and the elements on its diagonal are always equal to zero. This gives a total of $n(n-1)/2$ edge positions to learn for each structure containing n variables (10 in the case of the concrete network used in experiment 7, as shown in figure 2.13).

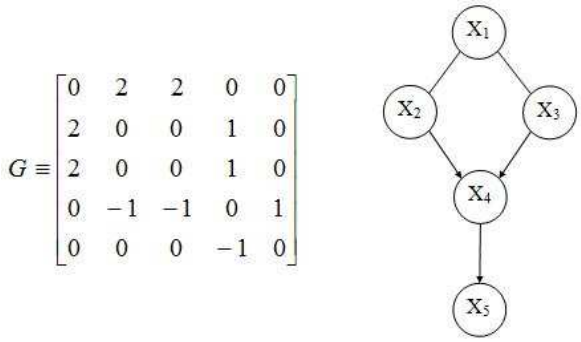

Figure 2.13: Example of graph - matrix equivalence for the structure used in experiment 7.

The results for experiment 7 are shown in figure 2.14.

As expected, the performances improve when the number of observed sample sizes increases. Also as expected, the figure shows the existence of a clear trade-off between the number of true positives that are found and the related number of false positives as a function of the confidence threshold. In effect, when high threshold are considered, the number of false positives goes to zero, but the number of true positives is also severely reduced. In these circumstances, most edges are neither correctly or incorrectly identified: their presence (or lack thereof) as well as their orientation in the graph remains uncertain. If lower levels of the threshold are used, more true edges are recovered but the risk exists of sporadically coming across erroneous relations.

A choice of $\alpha = 0.65$ seems to optimize this trade-off by providing an average of more than 8 correctly learned edges while keeping the average false positives under 0.1 (meaning, roughly, that less than one out of every one hundred learned edges is incorrect).

2.3.2 Evolution of performances with the sample size

Having observed the influence and the importance of the choice of the threshold for the confidence level, we may fix a value of $\alpha = 0.65$ and evaluate the evolution of the performance of the model averaging approach as a function of the sample size.

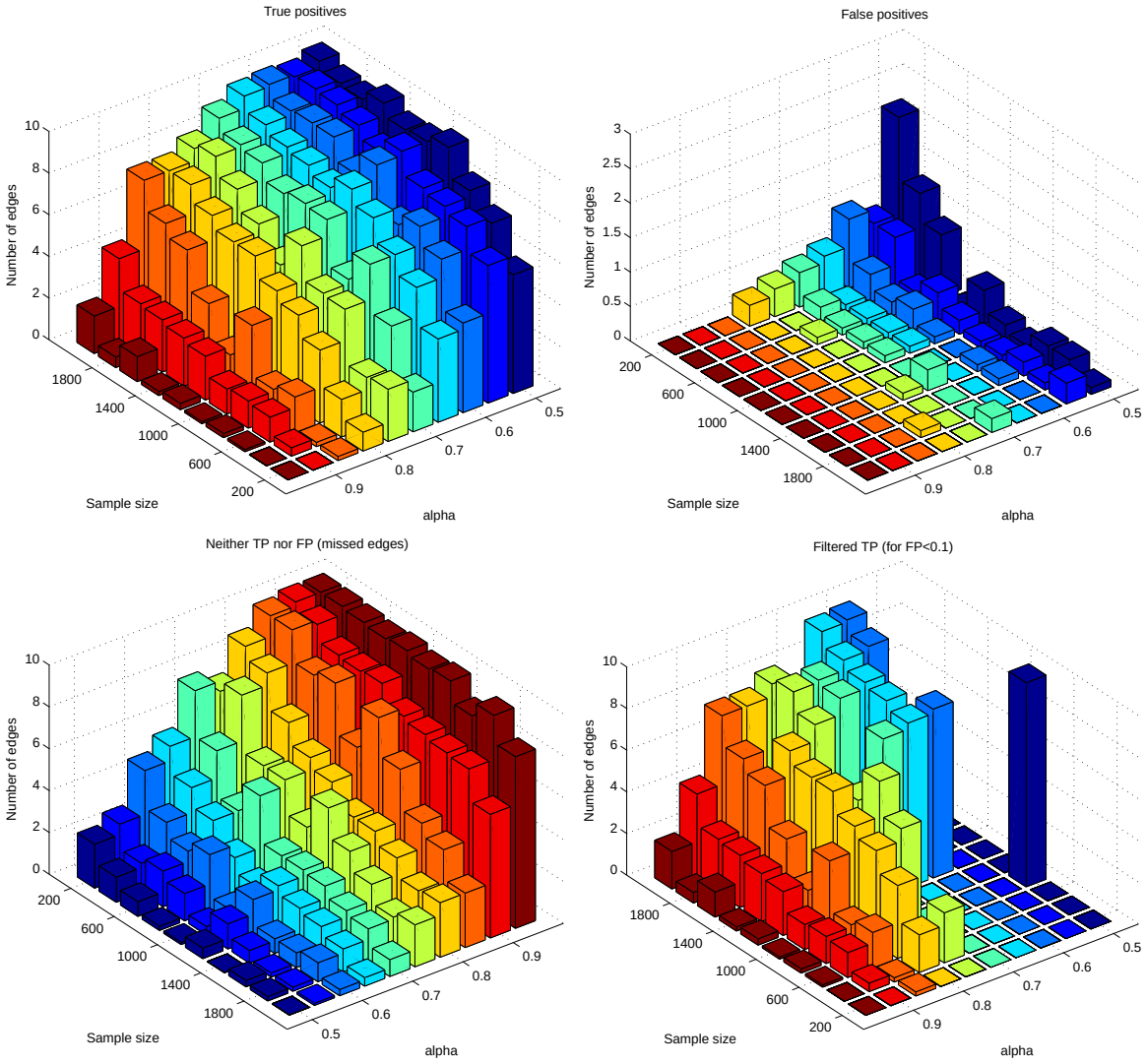

Figure 2.14: Influence of the threshold α for the model averaging approach.

Experiment 8.

- We considered the third structure in figure 2.2 (also shown in figure 2.16 top left), with the same parameters for the conditional probability distributions as in experiment 1 and we fixed a value of $\alpha = 0.65$.
- For each of the sample sizes under study the following steps were repeated 100 times:
  1. 100 additional datasets were generated by bootstrap sampling from the original dataset.
  2. The top scoring Bayesian network was learned for each of these 100 datasets and its associated class of equivalence was determined.
  3. As before, if an edge value (-1,0,1,2 - see figure 2.16) was observed in more than $100 \times \alpha$ networks, it was considered a *positive*. If a positive edge value was the same as the edge value in the true generating network, it was considered a *true positive* (also called a 'correct' edge). Otherwise, it was considered a *false positive* (or a 'wrong' edge). When

no value was observed in more than $100 \times \alpha$ networks for a given edge position, it was considered a 'missed' edge.

With this experiment, we wanted to study the evolution of the number of correct, wrong and missed edges as a function of the sample size. Figure 2.15. shows a graphical representation of the global average results.

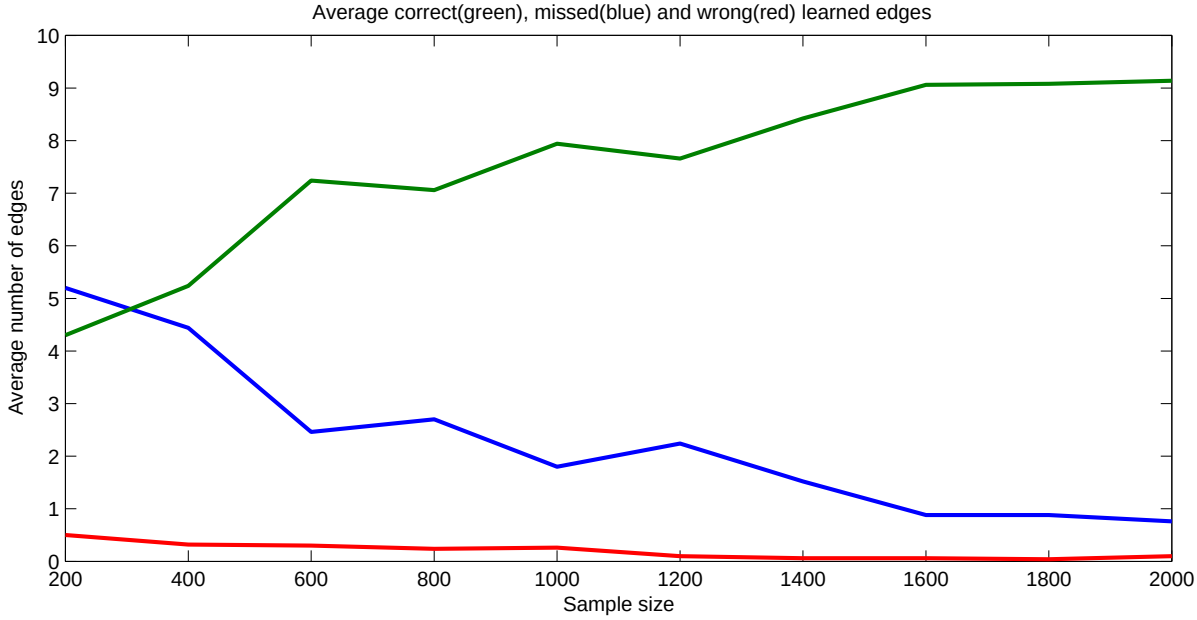

Figure 2.15: Results for experiment 8. The curves correspond to the average number of correct (green), missed (blue) and wrong (red) edges. The number of edge positions to be learned for the considered structure is 10.

The results seem reasonable in that, whereas at the beginning the average number of missed edges surpasses the number of correct ones, as the sample size increases the later take over while both the missed and the correct edges progressively decrease.

Furthermore, we decided to look at each of the individual edges in the network, and we observed the evolution of the percentage of times (over 100) that they were learned correctly, wrongly or that they were missed. The associated results are shown in figure 2.16.

When it comes to the wrong edges, the highest error rates seem to be associated to edge positions 4,5 and 10, which correspond to edges related to node D. This could be a direct consequence of this node being the center of the only v-structure in the network (sometimes, the learned model shares the same undirected graph of the generating network but the v-structures are misplaced).

In any case, it seems that these figures are coherent with the tendencies depicted in figure **??**, since they also show a clear growth of the rate of correctly identified edges when the size of the observed samples increases, as well as a diminution of the percentage of missed and wrongly inferred relations.

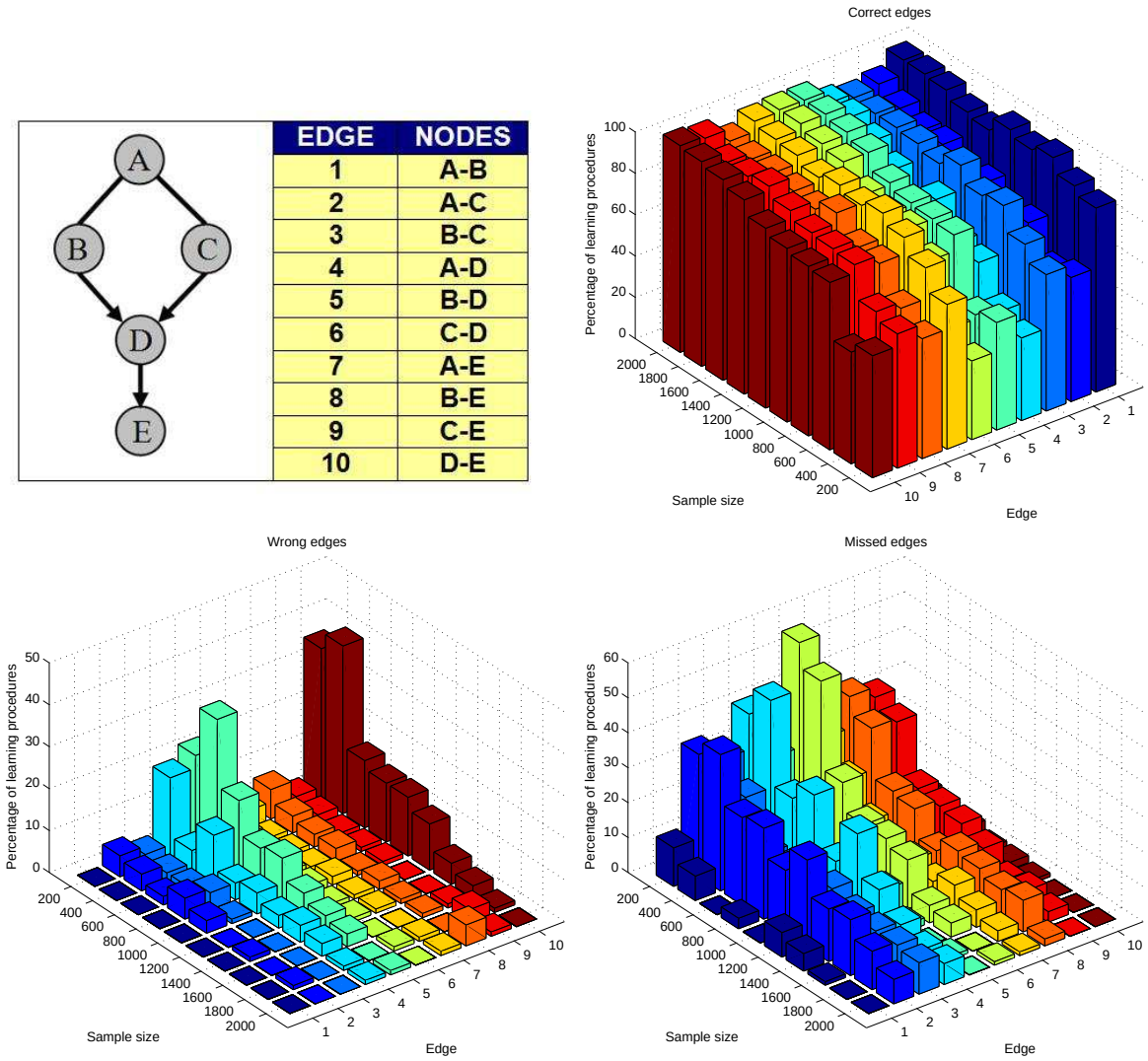

Figure 2.16: Results for experiment 8. Top left panel shows the correspondence between edge numbers on the three 3D figures and their position on the network. Results are shown for the number of learning attempts out of 100 (and each involving a model average over 100 bootstraped networks) in which the learned value for each edge was correct, wrong or missed.

2.3.3 Partial conclusions

To end this chapter on experiments and simulations, let us summarize the main conclusions for the model averaging section:

- First of all, it is interesting to observe that the model averaging approach seems to offer a real improvement over the single network learning strategy in that it makes it possible to simultaneously obtain high averages of true positives (close to ten, which is the maximum possible for the example considered) and low averages of false positives (actually zero) for relatively "affordable" sample sizes (even if they are still too large for microarray scenarios).
- When compared to the approaches described in previous sections, this approach is advantageous because, before, even for situations with high success rates, a wrong model could be learned that differed dramatically in terms of edges from the true one (even if it was learned less than 5% of the time). In most cases, previous strategies might have forced us to face a question of "all or nothing" (sometimes the proximity of two models in terms of their score was not directly correlated to their proximity in terms of individual edges; if the wrong model was learned, most edges could be simultaneously wasted). The model averaging approach, however, is much more "gradual" in this respect, the transition between final learned models as a function of the sample size seems to be smoother (at least when measured in terms of correct/wrong/missed edges as was the case for the experiments above).
- Finally, the bootstrap strategy proposed by Friedman et al. seems to be ingenious and interesting enough to begin with, but further refinements would be desirable for the future. The actual goal is to obtain the best possible sampling from the posterior distribution over the set of all possible DAGs in order to obtain a set of candidate structures over which the averaging can be more reliably made (for more details, see section dealing with subnetworks in Appendix A). In this context, the bootstrap approach is only a simplification due to computational complexity issues, that could therefore evolve into more efficient strategies. This could be the basis for possible future research.

Chapter 3

Conclusions and future work

Throughout the pages of this report, the different results obtained during my research internship have been presented. The main initial objective of this work, which involved designing and carrying out experiments and simulations with very simple networks in order to deepen our understanding of the Bayesian network approach, has consistently been addressed and some interesting conclusions have already been drawn at the end of each of the individual sections of the previous chapter.

Now that we have come to the end of our study, the question remains concerning the adequacy of Bayesian networks to tackle the problem of learning gene interaction networks from microarray expression data. Can we reasonably expect to be capable of learning the kind of biologically meaningful structural relations that could be at the basis of relevant follow-up studies?

Our simulations have shown that, even for the most simple scenarios involving networks that contain as few as 4 or 5 variables, large sample sizes may be required that vastly exceed the orders of magnitude currently available for microarray studies (a few tens or, at most, a couple of hundred observations). Certainly, in the most optimistic scenarios some very simple networks may be learned with a few hundred samples, but some combinations of factors (such as the randomness level of the conditional probability distributions or the presence of noise), while retaining just 4 variables, may always impose the need of thousands of samples just to guarantee that the Bayesian score will be capable of reliably selecting the "true" structures. If we were to consider larger sets of nodes, we would need to face additional errors due to the replacement of direct search procedures by discrete search approaches, such as those discussed in section 2.2 . This is why attempting to learn single global networks within this context seems to be a dead-end.

Model averaging techniques provide a ray of hope. As we have seen, by using certain bootstraping strategies we may reasonably attempt to make the most of the limited amount of information contained in a given dataset. In this condition, it seems that some high-confidence subnetworks can actually be reliably learned. Furthermore, the selection of a proper threshold for the confidence level of learned features makes it possible to choose between more conservative and more risky strategies, by controlling the trade-off between the rates of true and false positives that we may judge suitable or not within the framework of a particular experiment. Still, most of the learning tasks that were discussed in this report required the use of sample sizes that clearly exceeded the habitual empirical constraints in laboratories.

In my opinion, the key to the puzzle is to regard the task at hand as a complex challenge that calls for incrementally improved solutions in terms of refinement and complexity. Friedman et al. broke the ice with the very proposal of using Bayesian networks. At the beginning, the results may have seemed encouraging enough to attract the interest of other researchers, but the limitations inherent to the approach of attempting to learn single, global networks "all at once" must have soon become evident. Later on, the model averaging approach that they proposed, together with ideas like their Sparse Candidate algorithm or the use of dynamic Bayesian networks as described by Husmeier, contributed to a qualitative step forward.

Once again the current state of the art in this field, as read from recent papers, seems to have quantitatively exhausted these new ideas and new qualitative improvements are needed to open the trail for a further exploration of the subject. During my conversations with Professors Younes and Geman, we all seemed to agree that the next jump forward might be provided by the adoption of new hierarchical, or perhaps modular, strategies.

While it still remains somewhat unclear to us whether this should be based on a top-down or a bottom-up approach, we feel that the divide and conquer paradigm might offer substantial advantages with respect to the current single-level scenarios. Actually, the seed-based approach for the learning of subnetworks proposed by Friedman et al. seems to move along these guidelines, since it begins with a core of high confidence relations that is progressively expanded using an statistical significance score (an example of bottom-up approach). While encouraging and better than those obtained with other techniques, the results that they report in Pe'er's PhD dissertation show that the way ahead is still a long and winding one.

Another important idea for the future is related to the possibility of incorporating additional sources of biological information, such as protein and metabolite data, that could complement and enhance that already provided by current measurements of gene expression levels. Finding a reliable procedure for data fusion capable of combining these heterogeneous biological sources could have very interesting applications to the modeling task at hand and might lead to significant developments in the struggle against the small sample regime that curses today's microarray experiments.

Keeping this in mind, our interest hereafter is oriented towards the exploration of alternative theoretical frameworks within the domains of information theory and statistical learning that could be used to enrich, complement or eventually replace the Bayesian network paradigm. In line with this, we feel that maximum entropy models may provide a powerful, flexible body of knowledge to pursue our research in the near future and they shall therefore be at the origin of an extension to the work presented in these pages.

Bibliography

[1] N. Friedman, M. Lineal, I. Nachman, D. Pe'er. *Using Bayesian Networks to Analyze Expression Data.* Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, 2000.

[2] Dana Pe'er. *From Gene Expression to Molecular Pathways.* PhD. dissertation (under the supervision of Nir Friedman). Hebrew University, 2003.

[3] Jean-Philippe Vert *Notes du cours "Methodes à noyau en bioinformatique".* Master "Mathematiques, vision, apprentissage". ENS Cachan, France, 2005.

[4] U.S. Department of Energy *The Human Genome Project.*

[5] *Online microbiology courses.* Department of Microbiology and Molecular Genetics. College of Medicine, The University of Vermont.

[6] Konstantin V. Krutovskii and David B. Neale. *Forest genomics for conserving adaptive genetic diversity.* Forest Genetic Resources Working Papers. FAO, Rome, 2001.

[7] Schena, Mark, Shalon, Dari, Davis, Ronald W., Brown, Patrick O. *Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray.* Science 1995 270: 467-470

[8] DeRisi, Joseph L., Iyer, Vishwanath R., Brown, Patrick O. *Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale.* Science 1997 278: 680-686

[9] Paul Brazhnik, Alberto de la Fuente and Pedro Mendes. *Gene networks: how to put the function in genomics.* Trends in Biotechnology, Volume 20, Issue 11, 1 November 2002, Pages 467-472.

[10] Eisen, M.B., Spellman, P.T., Brown, P.O. and Botstein, D. (1998) *Cluster analusis and display of genomw-wide expression patterns.* Proc. National Academy of Sciences, USA, 95, 14863-14868.

[11] Chen, T., He, H.L. and Church, G.M. (1999) *Modeling gene expression with differential equations.* Pacific Symposium on Biocomputing, 4, 29-40.

[12] Zak, D.E., Doyle, F.J. and Schwaber, J.S. (2002) *Local identifiability: when can genetic networks be identified from microarray data?* Proceedings of the Third International Conference on Systems Biology, pp. 236-237.

[13] D'haeseleer,P., Liang, S. and Somogyi, R. (2000) *Genetic network inference: from co-expression clustering to reverse engineering.* Bioinformatics, 16, 707-726.

[14] Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, CA, USA.

[15] Heckerman, D. (1999) *A tutorial on learning with Bayesian Networks.* In Jordan, M.I. (ed.) *Learning in Graphical Models, Adaptive Computation and Machine Learning.* MIT Press, Cambridge, Massachussetts, pp. 301-354.

[16] Hatermink, A.J., Gifford, D.K., Jaakola, T.S. and Young, R.A. (2001) *Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks.* Pacific Symposium on Biocomputing., 6. 422-433.

[17] Husmeier, D. (2003) *Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks.* Bioinformatics 19: 2271-2282.

[18] Chickering, D.M. (1996) *Learning Bayesian networks is NP-complete.* In Fisher, D. and Lenz, H.J. (eds.), *Learning from Data: Artificial Intelligence and Statistics*, Vol. 5, Springer, New York, pp. 121-130.

[19] N. Friedman, I. Nachman, and D. Peer. *Learning Bayesian Network Structures from Massive Datasets: The Sparse Candidate Algorithm.* In Proc. UAI, 1999.

[20] Dana Pe'er (2005) *Bayesian Network Analysis of Signaling Networks: A Primer.* Science STKE 2005 (281), pl4.

[21] D. Geman, C. d'Avignon, D. Naiman, R. Winslow (2004) *Classifying gene expression profiles from pairwise mRNA comparisons*, Statist. Appl. in Genetics and Molecular Biology, 3, 2004.

[22] D. Geman, C. d'Avignon, D. Naiman, R. Winslow and A. Zeboulon(2004). *Gene expression comparisons for class prediction in cancer studies*, Proceedings 36'th Symposium on the Interface: Computing Science and Statistics, 2004.

[23] Kohane, I. S., Butte, A. J., and Kho, A. (2002) *Microarrays for an Integrative Genomics.* MIT Press.

[24] Chickering, D. M. (1995). *A transformational characterization of equivalent Bayesian network structures.* In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 87-98.

[25] N. Friedman, I. Nachman, and D. Pe'er.(1999) *Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm.* In Proc. UAI, 1999.

[26] M. I. Jordan (Ed.) Learning in graphical models. Cambridge MA: MIT Press, 1999

[27] Lauritzen, S., and Spiegelhalter, D. (1988). *Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems.* Journal of the Royal Statistical Society 50:157224.

[28] Dana Peer , Aviv Regev , Gal Elidan , and Nir Friedman (2001) *Inferring subnetworks from perturbed expression profiles.* Bioinformatics 17: 215-224S.

[29] Kevin Murphy (2001). *An introduction to graphical models.* [Online tutorial, http://www.cs.ubc.ca/ murphyk/Bayes/bnintro.html]

[30] Richard E. Neapolitan (2004) *Learning Bayesian networks.* Prentice Hall Series in Artificial Intelligence.

[31] Christian Borgelt, Rudolf Kruse (2002) *Graphical Models. Methods for Data Analysis and Mining.* John Wiley and Sons, LTD.

[32] J. Schfer and K. Strimmer (2005). *An empirical Bayes approach to inferring large-scale gene association networks.* Bioinformatics 21: 754-764.

[33] Verma, T. and Pearl, J. (1990). *Equivalence and synthesis of causal models.* In Proceedings of Sixth Con- ference on Uncertainty in Artificial Intelligence, pages 220-227.

[34] Nodelman, U., Shelton C.R., Koller D. *Learning Continuous Time Bayesian Networks.*

Appendix A

Bayesian networks tutorial

This appendix has been added to the report for reference purposes. It summarizes the initial phase of bibliographical research that took place at the beginning of my internship.

Most of the contents presented in the following sections have been adapted from sources including [2], [29], [30] and [31] and, as such, they do not constitute a piece of original research, but rather a work of synthesis and documentation.

Graphical models were born from the idea of modeling systems using graph theory. The term itself usually refers to a concrete type of probabilistic networks that use graphs to represent and manipulate joint probability distributions. Quoting Michael Jordan [26], they are "a marriage between probability theory and graph theory" and, as such, they are particularly well adapted to dealing with both uncertainty and complexity.

The origins of these mathematical artifacts are at the confluence of different disciplines, including but not limited to artificial intelligence (Pearl [14]) and statistics (Lauritzen & Spiegelhalter [27]).

According to the most common approaches, nodes represent attributes that are used to describe the underlying domain of interest and edges represent relations of conditional independence among the nodes. Usually a distinction is made between two different types of graphical models, depending on whether the edges are oriented (Bayesian Networks) or not (Markov Networks).

A.1 Representation

Before going into more details, let's consider a very simple example of a situation in which this type of model can be useful.

Let's imagine that we are interested in working with a certain set of random variables. Knowing their joint probability distribution can be very useful because it allows us to determine the probability of any event involving these variables.

Intuitively, if we work with binary variables, the joint probability distribution can be specified by a table with 2^n lines, where n is the number of random variables considered. Figure A.1 shows an example of this.

When the table above is known, the probability of any given expression that contains a combination of logical conditions over A, B and C can be very easily computed:

$$P(E) = \sum_{\text{rows matching E}} P(row) \tag{A.1}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

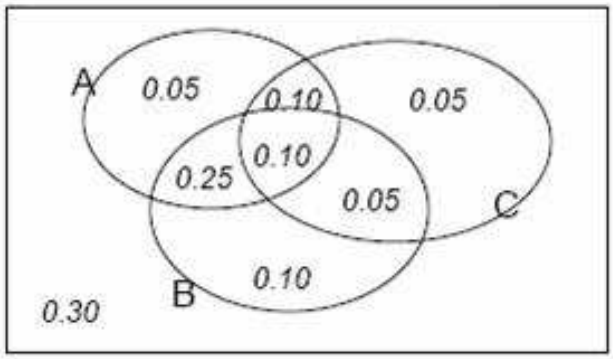

Figure A.1: Example of joint probability distribution, adapted from Andrew Moore's online tutorials (Carnegie Mellon University).

By applying Bayes' rule, it is also immediate to perform direct inference:

$$P(E_1|E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(row)}{\sum_{\text{rows matching } E_2} P(row)} \tag{A.2}$$

Unfortunately, working with this kind of table becomes intractable for more than just a small number of random variables (since the number of lines grows exponentially with n). In this context, it seems highly convenient to find a formalism capable of reducing the complexity of the chosen representation for a certain joint distribution and this is the idea behind Bayesian networks.

A Bayesian network over a set $X = \{X_1, ..., X_n\}$ is a representation of a ***joint probability distribution*** over X.

This representation consists of a ***directed acyclic graph*** (DAG) G whose vertices correspond to the random variables $X_1, ..., X_n$, and a parameterization which describes a ***conditional probability distribution*** for each variable given its immediate parents in G.

The information provided by a certain Bayesian network can therefore be regarded from a double perspective (figure A.2):

- **Qualitative information**: The graph G represents conditional independence properties of the distribution through the use of directed edges between the variables. This is formalized through so-called "Markov independencies", according to which each variable X_i is independent of its non-descendants given its parents in G.
- **Quantitative information**: The parameterization component of the network describes the conditional distributions for each random variable.

Thanks to the conditional independence properties determined by the structure of the graph, the conditional probabilities for each variable X_i depend only on its parents in G, denoted Pa_i^G ,and so the joint probabilities can be decomposed into the following product form:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa_i^G) \tag{A.3}$$

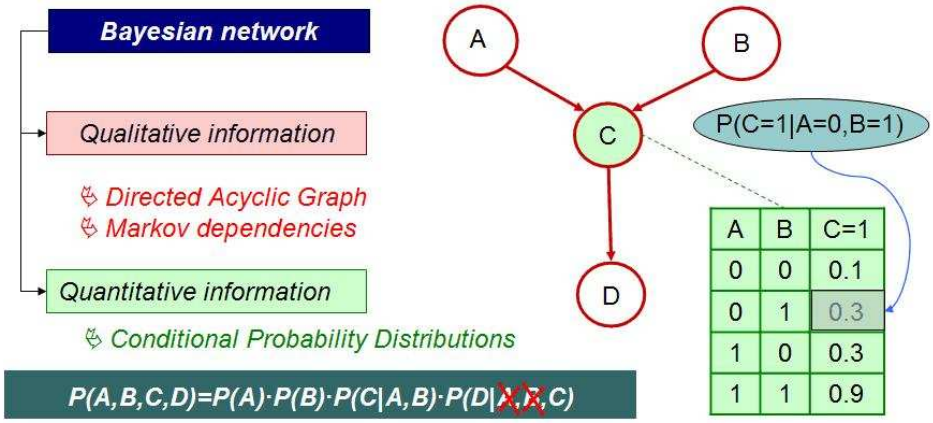

Figure A.2: Basic concepts related to Bayesian networks used as tools for the representation of joint probability distributions

Consequently, when working with Bayesian networks it is possible to establish an upper bound on the number of parameters used to specify a joint probability distribution as 2^k, where k is the maximum number of parents for each of the nodes on the graph; the new "tables" of conditional probabilities for each node need only list the value of $P(X_i|Pa_i^G)$ for each possible combination of parent values. Normally, this implies a remarkable reduction in complexity when compared to the previous bound 2^n, where n was the number of nodes itself (e.g. for the 'ICU Alarm' network, consisting of 37 variables, only 509 parameters need to be considered, instead of 2^{37}).

A.1.1 Independence and conditional independence: the Markov condition

As mentioned above, the nodes in a Bayesian Network represent attributes of the domain under study.

Recall that,

Definition 1. *Two events E and F are independent if one of the following holds:*

1. $P(E|F)$=$P(E)$*and* $P(E) \neq 0$,$P(F) \neq 0$
2. $P(E)$=*0 or* $P(F)$=*0*

The concept of independence is symmetric, and therefore $P(E|F) = \mathrm{P(E)}$ implies that $P(F|E)$=P(F). It is also immediate to observe that, if E and F are independent, then P(E,F)=P(E)P(F).

Definition 2. *Two events E and F are conditionally independent given G, with* $P(G) \neq 0$*, if one of the following holds:*

1. $P(E|F \cap G) = P(E|G)$ *and* $P(E|G) \neq 0$,$P(F|G) \neq 0$
2. $P(E|G) = 0$ *or* $P(F|G) = 0$

Using these concepts, it is possible to give a more rigorous definition of the Markov condition for Bayesian Networks.

Definition 3. *Let P be a joint probability distribution of the random variables in some set V and G a DAG with a set of edges E, G=(V,E). Graph G encodes the Markov assumption or Markov condition if each variable* $X_i \in V$ *is conditionally independent of its nondescendants given the set of all its parents* $(X \perp Non - descendants_X^G | Pa_X^G)$.

We may illustrate this idea by looking at a very intuitive example, based on a very simple network like the one in figure A.3a. We observe that, when no nodes are given, the probability of the child having a certain genetic disease depends on whether his father has it or not, and the probability of the father having the same disease depends on whether the grandfather has it or not. The three variables are therefore mutually dependent. However, if we know that the father actually has the disease (or that he does not have it), the fact of the grandfather having the disease or not no longer has an effect over the probability of the child having it, and the two variables become independent. This is why we say that the two nodes on the extremes of the network are conditionally independent given the central node.

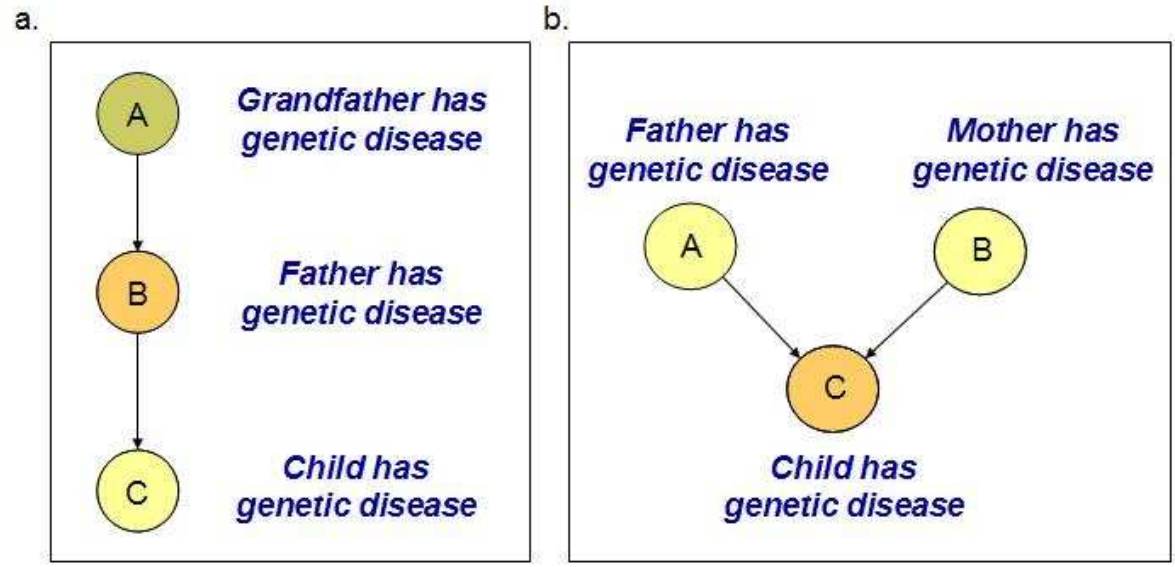

Figure A.3: Intuitive examples of conditional independence (a) and v-structure (b).

A.1.2 D-separation

The Markov condition offers an intuitive way of determining conditional independence relations for a given Bayesian Network.

As described in the previous section, this condition is useful to assert the following types of independence relations:

a. Two nodes that are not connected by a path in the network are independent.

b. Given the set of all its parents, a node is conditionally independent of the set of all its nondescendents.

However, the Markov condition is insufficient to determine the probabilistic independence (or the lack thereof) between two nodes when nodes other than their parents are given.

In line with the example above, we can think of a similar Bayesian Network like the one in figure A.3b. When no nodes are known, the probability of the father having a disease is independent of the mother having it or not, and vice versa. Nevertheless, if we know that the child has the disease and that the mother does not have it, it seems obvious that the father must have it. Consequently, given the value of the node "child has disease", the two parent nodes become dependent.

This type of probabilistic relation can be formalized through the concept of a v-structure.

Definition 4. *A v-structure is a subgraph of the form $X \rightarrow Y \leftarrow Z$, so that no edge exists between X and Z.*

V-structures are interesting because they reflect a situation in which, when given the value of the central node, the two previously independent parent nodes become dependent.

Intuitively, it may be useful to view dependence as a property that can "flow" between two nodes in a network, running through directed paths that connect them in the underlying DAG (figure A.4). In this metaphor, observation of an intermediate node in the path can have an ambivalent effect:

a. In a path of the pattern $X \rightarrow Y \rightarrow Z$, observation of Y "blocks" the flow of dependence, and therefore this kind of observation acts as a "wall".

b. In a path of the pattern $X \rightarrow Y \leftarrow Z$, observation of node Y "enables" the initially impossible flow of dependence between nodes X and Z, and therefore this kind of observation acts as a "bridge".

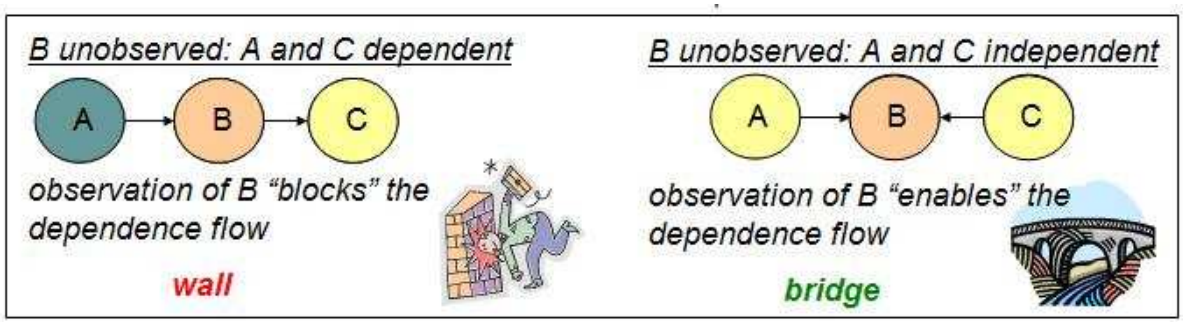

Figure A.4: Intuitive view of the "flows" of dependence.

This idea may be formalized through the following definitions:

Definition 5. *A graph G has a trail from X_1 to X_n, denoted $X_1 - ... - X_n$ if, for every i=1...n-1, G contains either $X_i \rightarrow X_{i+1}$ or $X_i \leftarrow X_{i+1}$.*

Definition 6. *Let G be a Bayesian network structure and $X_1 - ... - X_n$ be a trail in G. Let $E \in X$ be a subset of nodes. We say that there is an active trail between X_1 and X_n given evidence E if:*

- *Whenever we have a v-structure $(X_{i-1} \rightarrow X_i \leftarrow X_{i+1})$, X_i or one of its descendants are in E.*
- *No other node along the trail is in E.*

As we have seen before, intuitively this means that dependence can "flow" through every triplet $X_{i-1} - X_i - X_{i+1}$.

Definition 7. *Let X,Y,Z be three sets of nodes in G. We say that X and Y are d-separated given evidence Z, denoted d-$sep_G(X;Y|Z)$, if there is no active trail between any nodes $X_i \in X$ and $Y_i \in Y$ given evidence Z, for all $X_i \in X$ and $Y_i \in Y$.*

Again from an intuitive perspective, the concept of d-separation reflects a situation in which the only given (and therefore potentially "blocking") nodes are those at the center of each and every v-structure in the trail. This means that there are no "walls" and that there are as many "bridges" as necessary to guarantee the flow of dependence.

In practice, this concept is important because it provides a direct criterium to check for conditional independence relations between two sets of nodes in a Bayesian Network:

Theorem 1. *If a set of evidence variables E d-separates two sets of variables X and Z in a Bayesian network's graph, then X and Z are conditionally independent given E.*

The set of independence statements implied by a graph G is denoted as Ind(G) and includes all the statements of the form "X is independent of Y given Z". The notion of d-separation is at the base of certain graph algorithms that can be used to check the validity of any such conditional independence statement in linear-time.

A.1.3 Equivalence classes

It is useful to note that more than one graph may imply the same set of independence statements.

The most straightforward example is the case of simple graphs $X \rightarrow Y$ and $X \leftarrow Y$. It is immediate to observe that $Ing(G1) = Ind(G2) = \emptyset$.

In figure A.5a, a more elaborate example is shown, consisting of four different graphs. The three structures on the left correspond to the same joint probability distribution, whereas the structure on the right corresponds to a different one. The set of independence statements implied by the three structures on the left is also the same (A and B are dependent when C is unknown, and become independent when C is given) as opposed to the structure on the right (A and B are independent when C is unknown and become dependent when C is given). To express this fact, the first three graphs are said to be Markov equivalent.

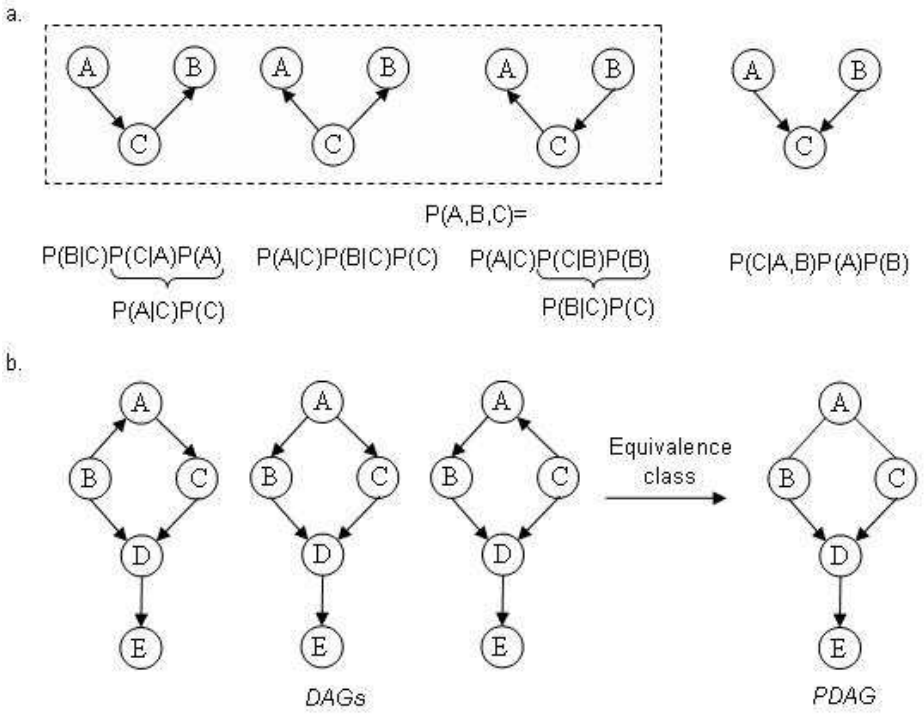

Figure A.5: Bayesian network equivalence classes. (a) The first three structures from the left belong to the same equivalence class and share the same joint probability distribution, which is different from that of the structure on the right (adapted from Dirk Husmeier). (b) Example of three Bayesian networks belonging to the same class of equivalence and the associated PDAG that represents that class.

Definition 8. *Two graphs G1 and G2 are Markov equivalent if they describe the same set of conditional independences: Ind(G1) = Ind(G2)*

Theorem 2. *(Verma and Pearl, 1990) Two Bayesian Network structures are equivalent if they have the same underlying undirected graph and they have the same v-structures.*

As a direct consequence of this theorem, each equivalence class can be uniquely represented by a partially directed graph in which the only oriented edges are those that correspond to v-structures (A.5b).

The concept of equivalence class is crucial when learning bayesian networks from data. In these situations, it is not possible to distinguish among equivalent graphs because the learning procedure is based solely on observations from a given joint probability distribution.

A.1.4 A brief commentary on Bayesian Networks and causality

Ever since Bayesian Networks are probabilistic models of dependencies between multiple measurements, they do not inherently capture causal relations. However, sometimes it may be interesting to model the actual mechanisms that generated the underlying structures of statistical dependency, and so different causal interpretations for Bayesian Networks have been proposed.

Definition 9. *(Pearl and Verma, 1991) A causal model of a set of variables U is a Directed Acyclic Graph in which*

1. *each node corresponds to one of the variables under analysis*
2. *links denote direct causal influences among them*

The model may be just qualitative or both qualitative and quantitative

One of the most commonly accepted approaches when it comes to causal models is the use of causal networks. In terms of representation, causal networks present many similarities to bayesian networks, since they also consist of a directed acyclic graph where each node represents a random variable along with a local probability model. The interpretation on the meaning of the edges is stricter for causal networks though, since for them the parents of a variable are its immediate causes.

Before going any further, it is important to make a distinction between the concept of observation and intervention. An observation is a passive measurement of the domain under study (e.g. a sample from a microarray measurement), whereas the term intervention refers to the act of setting the values of some variables using forces outside the causal model under study (e.g. gene knockout or over-expression).

A causal network models not only the distribution of the observations, but also the effects of interventions. If X causes Y , then manipulating the value of X affects the value of Y . On the other hand, if Y causes X, then manipulating X will not affect Y . Thus, although X $\rightarrow$ Y and X $\leftarrow$ Y are equivalent Bayesian networks (observing different values of X gives information on Y and viceversa), they are not equivalent causal networks.

Let's clarify this point with a simple example. Let us consider a certain interaction between two genes A and B. From simple observations, it is only possible to infer a certain level of correlation between their levels of expression, but it is not possible to decide on a particular orientation for the edge between the two nodes. Imagine now that it is possible to "artificially over-express" gene A through some kind of experimental procedure. In this scenario, we may compare the conditional probability of node B given A before and after the intervention on gene A:

- If there is a causal relation of the type A $\rightarrow$ B, then $P'(B|A) = P(B|A)$, since the new values that are measured for gene B will be modified following the intervention.
- If there is a causal relation of the type $A \leftarrow B$, then $P'(B|A) \neq P(B|A)$, since the new values for gene A (imposed externally) will no longer be determined by the values of gene B.

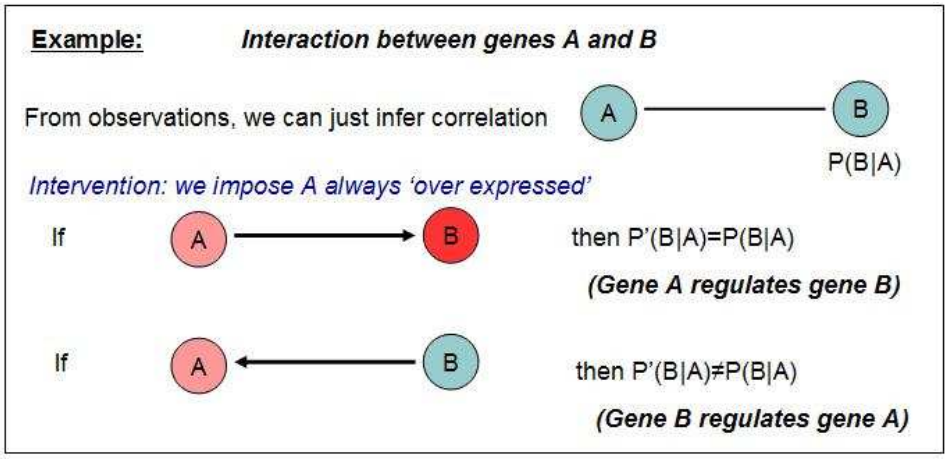

Figure A.6: Example of experiment capable of elucidating a very simple type of causal relation, as suggested in [2].

Experiments like this show that interventions are actually an important tool for inferring causality. These interventions may be costly to perform though, and so it may be useful to attempt to learn causality from observation alone. Much debate has taken place since long ago concerning this topic, and the interested reader can refer to [30] and to chapter 9 of [31] for further details on the subject.

In the context of Bayesian Networks applied to gene expression data, it may be interesting to point out the approach of Nir Friedman and Dana Pe'er at this respect. According to them, causal networks can be interpreted as Bayesian networks under two strong assumptions:

- First, it is necessary to accept that the (unknown) causal structure of the domain satisfies the Causal Markov Assumption: given the values of a variable's immediate causes, it is independent of its earlier causes. When the causal Markov assumption holds, the causal network satisfies the Markov independencies of the corresponding Bayesian network. Thus, this implies assuming that causal networks can provide a reasonable model of the domain.
- Second, it is necessary to assume that there are no latent or hidden variables that affect several of the observable variables. Unfortunately the second assumption does not hold in the main domain of interest studied here, and thus causal conclusions from the whole learning procedure must be treated with caution.

By accepting these two premises, Pe'er and Friedman essentially assume that one of the possible DAGs over the domain variables is the "true" causal network. As mentioned in the previous subsection, when working with observations alone, it is not possible to distinguish among structures belonging to the same equivalence class, but following their line of reasoning it should still be possible to learn a description of the class that contains the true model.

Once this class has been learned for a certain dataset (in the form of its associated PDAG), they suggest that some of the causal structures of the generating model could be extracted since the fact of observing a certain directed path in the PDAG implies the existence of the same causal link in all of the networks that could have generated that PDAG, and therefore this link must also be present in the "true" generating network.

In practice, though, and in the domain of interest for the present research project, the two assumptions described above are very questionable and so it is compulsory to carefully evaluate all possible causal relations that may be learned through the use of this approach. After all, it is necessary to keep in mind the fact that remaining within the framework of Bayesian Networks implies dealing with an approach that is not intended to infer causality and that, in the best of cases, could only provide a heuristic method for pointing out possible causal connections to be further investigated at a later stage.

A.2 Learning

In this section, we will face the problem of learning Bayesian Networks from a given set of observed samples.

In all cases, we will consider a given set of m samples $D = D_1, D_2, ..., D_m$, where each D_i is independently drawn and includes a set of n values $D_i = \{X_{1i}, ..., X_{ni}\}$ that correspond to the n nodes of a generating Bayesian network G* with an underlying distribution P*. The learning task may then imply determining the conditional probability parameters for each node (P*), determining the structure or topology of the model (G*), or attempting to determine both things at the same time.

We will begin by studying practical procedures for learning parameters that will afterwards constitute the basis for a scored based approach for learning structure.

As a general rule, we will always work with binary variables and we will consider multinomial, discrete conditional probability distributions that can be specified by tables.

A.2.1 Parameter estimation

Let's begin by assuming that the structure of the graph G is known. In practice, this will not normally be the case, but we need this assumption as a starting point since the procedures for parameter estimation will later become the conceptual base for their structure learning counterparts.

Our basic goal here will be to distinguish the best set of parameters, that is, the set of parameters that best fits the observed data. Two basic approaches can be discussed at this respect:

Maximum Likelihood Estimation

We define a likelihood function, $L(\theta : D)$, which measures the likelihood of the data.

$$L(\theta : D) = \prod_{m=1}^{M} P(x[m]|\theta) \tag{A.4}$$

where x[m] denotes a single set of n sample values for variables $X = \{X_1, X_2, ..., X_n\}$, a total of M samples is observed and θ denotes a given set of parameters.

The idea behind Maximum likelihood estimation is to choose a set of parameters $\hat{\theta}$ that maximize this likelihood:

$$\hat{\theta} = \max_{\theta} L(\theta : D) \tag{A.5}$$

The properties of Bayesian networks mentioned in the first section of the Appendix are particularly useful here, since they make it possible to decompose the global likelihood into local likelihood functions. The calculation is therefore greatly simplified and the search for the optimal parameters becomes tractable, since the local likelihood for each node can be optimized in an independent manner:

$$
\begin{aligned}
L(\theta : D) &= \prod_{m=1}^{M} P(x[m] : \theta) \\
&= \prod_{m=1}^{M} \prod_{i=1}^{n} (P(x_i[m]|pa_{X_i}[m] : \theta)) \\
&= \prod_{i=1}^{n} [\prod_{m=1}^{M} (P(x_i[m]|pa_{X_i}[m] : \theta))] \\
&= \prod_{i=1}^{n} L_i(\theta_{X_i|pa_{X_i}} : D) \qquad \text{(A.6)}
\end{aligned}
$$

where $L_i(\theta_{X_i|pa_{X_i}} : D) = \prod_{m=1}^{M} (P(x_i[m]|pa_{X_i}[m] : \theta)$ is the local likelihood function for X_i.

Under the assumption of discrete multinomial conditional probability distributions, the local likelihood function can be further simplified into a series of products that take into account the number of occurrences for all the possible combinations of parent and node values, as shown in the next paragraph.

Let's consider a possible value for each node ($x \in Val(X)$) and a possible combination of values for its parents ($u \in Val(Pa_X)$). We can then use the notation M[x,u] to refer to the number of instances in which the parents of a certain node take value u when this node takes value x. The marginal M[u] can then be obtained as $M[u] = \sum_x M[u, x]$. Using this notation, the likelihood function for a given node can be written as:

$$
L_i(\theta_{X|U} : D) = \prod_{u \in Val(Pa_{X_i})} \prod_{x \in Val(X)} \theta_{x|u}^{M[x,u]} \qquad \text{(A.7)}
$$

It can be proved that the choice of parameters maximizing this expression is:

$$
\hat{\theta}_{x|u} = \frac{M[x, u]}{M[u]} \qquad \text{(A.8)}
$$

In this context, M[x,u] and M[u] are called sufficient statistics since they summarize all the relevant information from the data that is needed in order to calculate the likelihood. This provides an easy to implement procedure for the estimation of the MLE parameters that relies in such a simple operation as the counting of observed samples verifying a certain condition.

Maximum A Posteriori Probability

The MLE approach can be considered a first step towards the objective of finding a suitable criterion for measuring the fitness of a Bayesian network to data. However, it shows a certain tendency to overfit the model to the particular data instance at hand. If, for example, no samples are observed verifying a certain property, MLE will assign a null probability to it, whereas we might have some previous knowledge according to which it might be reasonable to allow for a probability greater than zero. This overfitting tendency may be a critical drawback in small sample regimes, like the one we will be facing.

It may therefore be adequate to consider the Bayesian approach, which formulates the concept of prior belief in a principled manner. The idea is that now, in addition to the observed data D, we have some initial distribution $P(\theta)$ termed the prior, which is used to encode our beliefs regarding the domain prior to our observations. The flatness of this prior distribution reflects the level of certainty for our assumptions: a usual approach consists in using a uniform distribution which is mainly intended to provide all possible events with a non-zero probability. However, we may also consider heavily peaked distributions if we are certain about a specific belief concerning our domain.

The observation of a given dataset is then used to update the distribution $P(\theta)$, so that prior beliefs and experimental samples can be combined coherently. The resulting updated distribution is called the posterior, and is given by the expression:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{A.9}$$

The term P(D), termed the marginal likelihood, averages the probability of the data over all possible parameter assignments. Since it is a normalizing constant, which is independent of θ, we will usually ignore it in our calculations and we will just try to find the set of parameters maximizing the numerator: $\hat{\theta} = \max_\theta P(\theta|D) = \max_\theta P(\theta, D)$.

The question that arises at this point is how we can find a suitable procedure to transform a certain previous knowledge about the domain of study into a prior probability distribution.

First of all, let us define two highly desirable properties for the priors:

Definition 10. *A parameter prior $P(\theta)$ is said to satisfy global parameter independence if it decomposes into the following form:*

$$P(\theta) = \prod_{i=1}^{n} P(\theta_{X_i|Pa_i}) \tag{A.10}$$

Let X be a variable with parents U, we say that the prior $P(\theta_{X|U})$ has local parameter independence if $P(\theta_{X|U}) = \prod_{u \in U} P(\theta_{X|u})$

We say that the prior P(theta) satisfies parameter independence if it satisfies both global and local parameter independence.

Definition 11. *For a given class of likelihood functions $p(X|\theta)$, the class P of priors $p(\theta)$ is called a conjugate if the posterior $p(\theta|X) = p(X_1|\theta) \times p(\theta)/p(X)$ is of the same class .*

The choice of Dirichlet priors - a solution adopted by many researchers, including Friedman and Pe'er - satisfies these two desirable requirements.

Definition 12. *Let $F_1, F_2, ...F_r$ be binary random variables. The associated Dirichlet density function with hyperparameters $a_1, a_2, ...a_r$, where $a_1, a_2, ..., a_r$ are integers $\geq$ 1, is*

$$\rho(f_1, f_2, ..., f_{r-1}) = \frac{\Gamma(N)}{\prod_{k=1}^{r} \Gamma(a_k)} f_1^{a_1-1} f_2^{a_2-1} ... f_r^{a_r-1} \tag{A.11}$$

where $0 \leq f_k \leq 1$, $\sum_{k=1}^{r} f_k = 1$ and $N = \sum_{k=1}^{r} a_k$

- *Random variables $F_1, F_2, ..., F_r$, that have this density function, are said to have a Dirichlet distribution.*
- *The Dirichlet density function is denoted $Dir(f_1, f_2, ...f_{r-1}; a_1, a_2, ..., a_r)$.*

As a reminder, the Gamma function $\Gamma(x)$ is defined to be an extension of the factorial to real number arguments. If n is a natural number, then $\Gamma(n) = (n-1)!$. Otherwise, the Gamma function is defined as the following integral:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-x} dt \tag{A.12}$$

Figure A.7 illustrates this point with two easy to grasp examples.

This intuitive example can also be used to illustrate the concept of conjugate priors (figure A.8).

Returning to the framework of Bayesian networks, the Dirichlet density function can be used to model prior knowledge:

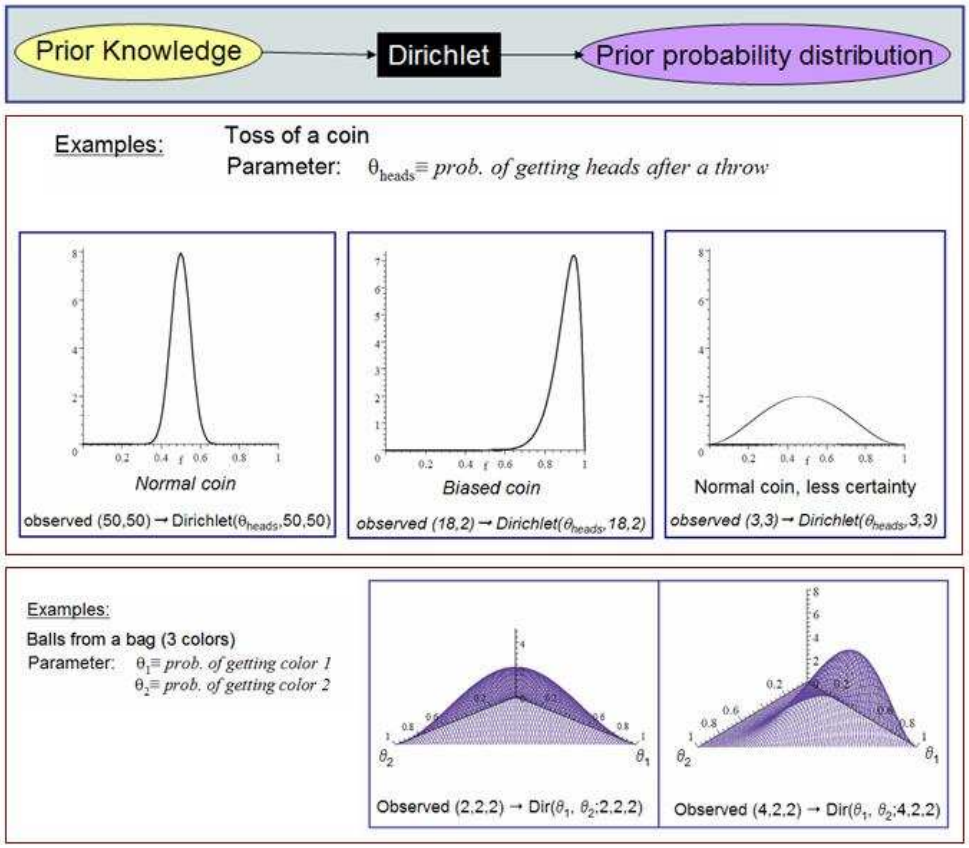

Figure A.7: Dirichlet distributions used to formalize prior knowledge. The first example shows the use of Dirichlet distributions to formalize prior knowledge concerning the probability of getting heads for a certain coin after having observed different numbers of tosses and different results. For the normal coin (100 tosses, 50 heads and 50 tails) the distribution is centered on value 0.5. For the biased coin (20 tosses, 18 heads and 2 tails), the peak of the distribution is placed at a higher value. For the case of the normal coin and only 6 prior observations (3 heads and 3 tails), the function centered on 0.5 again but it is not so peaked as before. The second example provides a three dimensional scenario, where three parameters are considered that correspond to the probability of getting a certain color by extracting a ball from a bag containing balls of three different colors.

- The idea is to "materialize" this knowledge through the choice of a certain set of prior "imaginary counts" for each of the possible combination of parent and node values (a_{ijk}). These counts can be used as the hyperparameters for a Dirichlet distribution modeling each of the discrete conditional probability distributions in the network.

$$P(\Theta_{ij}) = Dirichlet(\Theta_{ij1}, \Theta_{ij2}, ..., \Theta_{ij(n-1)}; a_{ij1}, a_{ij2}, ..., a_{ijn}) = \frac{\Gamma(a_{ij})}{\prod_{k=1}^{n} \Gamma(a_{ijk})} \prod_{k=1}^{n} \Theta_{ijk}^{a_{ijk}-1} \quad \text{(A.13)}$$

  where a_{ijk} is the number of prior samples for which node X_k takes value i (0 or 1) and the parents of node X_k take the j^{th} value of all their possible combinations. (see example in section 1.3.5).

- Once the prior has been defined, the number of instances of each possible combination of parent and node values can be counted over the set of observed samples, giving a set of counts N_{ijk}. The posterior probability distribution is then also a Dirichlet distribution of parameters $(a_{ijk} + N_{ijk})$.

$$P(\theta_{ij}|D) = Dirichlet(\theta_{ij1}, \theta_{ij2}, ..., \theta_{ij(n-1)}; a_{ij1} + N_{ij1}, a_{ij2} + N_{ij2}, ..., a_{ijn} + N_{ijn}) \quad \text{(A.14)}$$

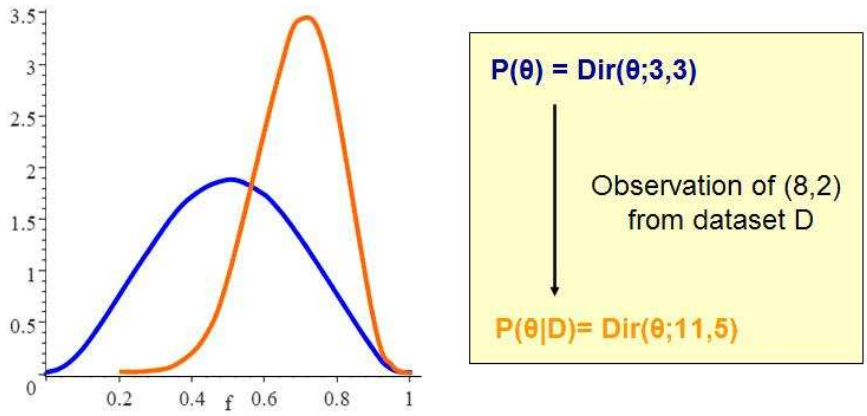

Figure A.8: Belief update example. Dirichlet distributions are conjugate priors. Continuing with the example above for the toss of a coin, consider the case of having observed 3 heads and 3 tails, which gives a Dirichlet distribution of the form $Dir(\theta_H; 3, 3)$. If a new dataset containing 8 heads and 2 tails is observed, the posterior probability will also follow a Dirichlet distribution of the form $Dir(\theta_H; 3{+}8, 3{+}2)$.

- Under this circumstances, the choice of values $\hat{\theta}$ that maximizes $P(\theta|D)$ is given by the expression:

$$\hat{\theta}_{ijk} = P(X_k = x_i | Pa_{X_k} = p_j, D) = \frac{a_{ijk} + N_{ijk}}{\sum_j (a_{ijk} + N_{ijk})} \tag{A.15}$$

In spite of its apparent conceptual complexity, once it has been properly defined the Dirichlet formalism offers an attractive ease of use and implementation.

A.2.2 Structure learning

After looking at two different approaches for learning parameters in a Bayesian network, we may take an interest in learning its topology or structure. So far, we had supposed this structure to be known but in practice this is not usually the case.

Learning structure for a Bayesian network implies learning the dependency relationships among the variables in the domain under study. When working with gene networks in the context of this internship, this point became of central interest.

The score-based approach that we will adopt to address this learning task is based on three elements:

a. A space of candidate models, namely the space of all possible directed acyclic graphs that can be built using a given set of variables. In practice though, we will work with PDAGs representing classes of equivalence, since the concrete DAGs cannot really be learned just from observations.

b. A scoring function, capable of measuring how well a certain structure fits a given dataset.

c. A search procedure capable of traversing the space of candidate structures within some reasonable limits in terms of time and computational resources.

The Bayesian score

A common choice for the scoring function is the so-called Bayesian score. Its conceptual basis constitutes an extension of the ideas presented in the previous section. The goal is to choose the structure that maximizes its own probability over the observed dataset, combined with a certain prior distribution over all possible structures.

The scoring function can be defined using the MAP principle.

The learning problem can be stated as follows:

- Consider a dataset with m samples $D = \{x(1), x(2), ..., x(m)\}$ (where each $x(i)$ represents a vector with values for the n variables under study $X = \{X_1, X_2, ..., X_n\}$).
- Assume that these samples have been drawn from some unknown generating network G^* with multinomial conditional probability distributions of parameters θ^*.
- Search for the simple model $B = (G, \theta)$ which is the most likely to have generated the data (a model whose underlying distribution will be close to the empirical distribution of the data).

For a given Bayesian network G, the probability of generating a certain dataset can be defined as $P(D|G)$. If we define P(G) as the a priori probability of network G, the probability of observing a certain dataset D for a network G expressed by P(D,G) can be obtained as:

$$P(D, G) = P(G|D)P(D) = P(D|G)P(G) \tag{A.16}$$

Using Bayes rule, the probability to be maximized within the framework of the learning problem can be rewritten as:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{A.17}$$

And the ultimate goal is to find the structure G^* that maximizes this expression:

$$G^* = \arg_G \max P(G|D) \tag{A.18}$$

In order to calculate the marginal likelihood for network structures, the conditional probability parameters θ need to be integrated out:

$$P(D|G) = \int P(D|G, \theta)P(\theta|G)d\theta \tag{A.19}$$

Taking this fact into account, these formulae can be rewritten into the expression for the Bayesian score that is presented in figure A.9.

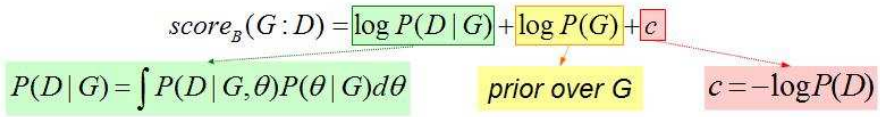

Figure A.9: Bayesian score and its intuitive decomposition.

One of the strong points of this score is the fact that it is well suited for small samples regimes. This is so thanks to two different aspects of the expression above:

1. The prior over all possible structures can be conveniently chosen in order to penalize overfitting topologies: if the prior probability is higher for simpler models, the P(G) term has the effect of penalizing complex ones.
2. The integration over all possible parameters provides a bias towards more simple structures, and this bias is weakened as the number of observed samples increases. Structures with many parameters are penalized, unless the probability of the true parameters is very peaked (which happens when the sample size is large). Thus the Bayesian score inherently takes care of the problem of overfitting a complex model to a small sample.

The computation of the integral above can be more or less easy to implement depending on certain properties of the chosen prior distribution, which will therefore determine subsequent properties of the resulting scoring function. Let us see different examples of this through the following definitions.

Definition 13. *A parameter prior verifies modularity if, for any graphs G_1 and G_2:*

$$\textit{Node } k \textit{ has the same set of parents in graphs } G_1 \textit{ and } G_2 \Rightarrow P(\theta_{X_k|Pa_k^{G_1}}|G_1) = P(\theta_{X_k|Pa_k^{G_2}}|G_2) \quad \text{(A.20)}$$

In practice, this means that the parameter prior depends only on the local structure of the graph, and so the integral expression can be re-written as:

$$P(D|G) = \prod_k \int_{\theta_{X_k|Pa_k}} \prod_m P(x_k[m]|pa_k[m], \theta_{X_k|Pa_k}) d\theta_{X_k|Pa_k} \quad \text{(A.21)}$$

As a result of this, when the parameter priors verify the modularity property the score can be decomposed into the local contributions of each node, that we will call '*FamScore*'. This contribution depends only on the sufficient statistics of the associated variable X_k and its parents Pa_k over dataset D, which results in an expression of the following type:

$$score_B(G : D) = \sum_i FamScore_B(X_k, Pa_k : D) \quad \text{(A.22)}$$

Another desirable feature for a scoring function is the fact that the score should reach its optimum on the true generating structure, that is, that given a sufficiently large number of samples, graph structures that exactly capture all the dependencies in the distribution will receive a higher score than all other graphs. This is formalized in the following definition.

Definition 14. *Let us consider a dataset of size M generated by some true model G*. A score is consistent if, as $M \to \infty$, the following properties hold with probability asymptotic to 1 (over all possible choices of the dataset D):*

- *The structure G* will maximize the score.*
- *All structures that are not equivalent to G* will have a strictly lower score.*

Finally, since it is not possible to distinguish between two different networks in the same equivalence class, we may take an interest in finding a scoring criterion that assigns the same score to structures belonging to the same equivalence class.

This property can be achieved using a BDe (Bayesian-Dirichlet equivalent) set of priors, which implies choosing a set of hyperparameters for the prior that does not bias the score between equivalent structures. In practice, a straightforward method to accomplish this property consists in to use a Bayesian network to generate the prior hyperparameters (which will obviously generate the same number of samples for all classes of equivalence):

$$a_{ijk} = M' \times P_a(X_k = x_i, Val(Pa_k) = p_j | G_c^h) \quad \text{(A.23)}$$

where a_{ijk} are the values of the hyperparameters, M' is the equivalent sample size and G_c^h is the user's prior network.

These three concepts are important because, assuming discrete, tabular conditional probability distributions and Dirichlet priors for the parameters, the Bayesian score is modular and can be computed using a close form formula:

$$FamScore(X_k, Pa_{X_k} : D) = \log \prod_{j \in Val(Pa_{X_k})} \frac{\Gamma(a_{jk})}{\Gamma(a_{jk} + N_{jk})} \prod_{i \in Val(X_k)} \frac{\Gamma(a_{ijk} + N_{ijk})}{\Gamma(a_{ijk})} \quad \text{(A.24)}$$

where the notation is the same as before and $a_{jk} = \sum_{i \in Val(X_k)} a_{ijk}$, $N_{jk} = \sum_{i \in Val(X_k)} N_{ijk}$.

Furthermore, the score is consistent ([34]) and, when so-called BDe priors (which stands for "Bayesian-Dirichlet likelihood equivalent") are used, the resulting score is also structure equivalent (see the works of David Heckerman for further details).

Search algorithm

Once an appropriate scoring function has been defined, the space of all possible directed acyclic graphs needs to be explored using graph transformations in order to find the structure that best fits the data (which will be the structure that maximizes the score).

The usual approach at this point is to consider simple operations resulting in so-called neighboring structures (namely the addition, removal or reversal of an edge). At each step of the learning process, only operations leading to "legal" structures (i.e. structures with no directed cycles) are permitted (figure A.10).

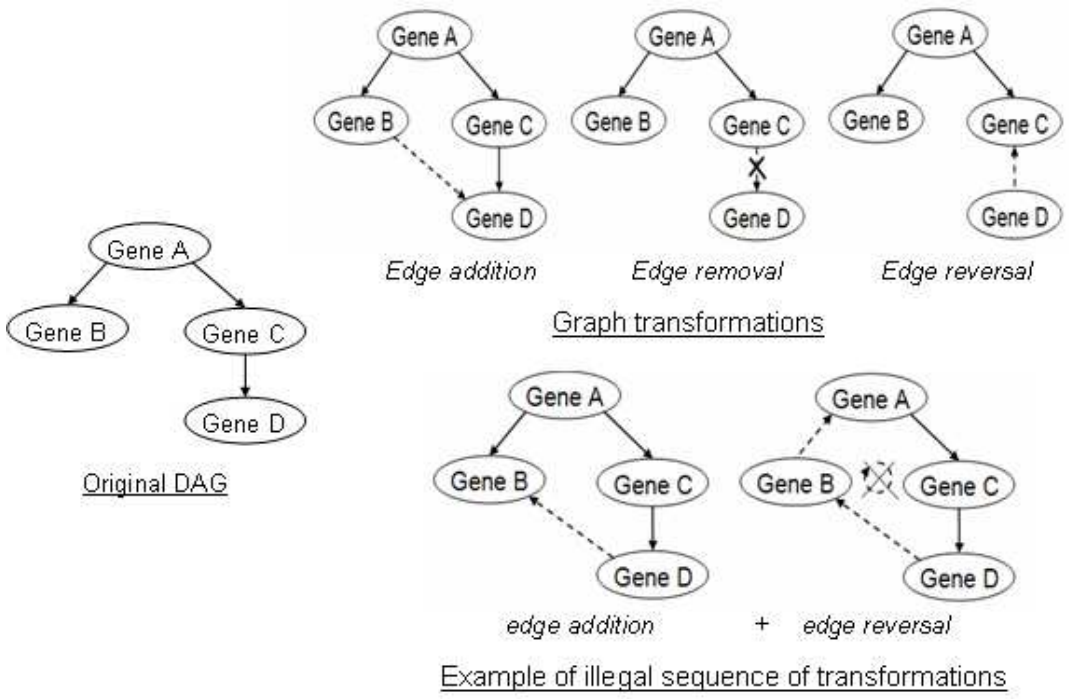

Figure A.10: Graph transformations used to explore the space of all possible DAGs.

The number of possible DAG's on n variables is superexponential in n. Apparently, there is no closed form formula for this but we may note that there are 25 DAGs on 3 nodes, 543 DAGs on 4 nodes, 29281 DAGs on 5 nodes and $O(10^8)$ DAGs on 10 nodes.

Due to this combinatorial explosion, direct search is only feasible for very small numbers of variables, and so other discrete search methods such as simulated annealing or greedy hill-climbing must be taken into account.

In practice, certain strategies can be implemented to reduce the computational resources needed for the search. This can be exemplified through the following two examples:

- Thanks to the decomposability property of the scoring function at a local level, when dealing with two models that differ only by a few arcs it is only necessary to recompute the score for the nodes whose family has been modified.

- Also, when dealing with scoring functions based on counting such as the one presented in the previous section, computed counts can be cached and reused in order to avoid unnecessary passes made over the observed data and thus reducing the computational load.

A.3 Subnetworks

The learning of subnetworks based on model averaging techniques seems to be a relatively popular strategy among researchers in the Bayesian networks community. However, as we will see, it usually implies a rupture with certain basic Bayesian network principles, such as the acyclicity constraint and so, the results obtained within this context must be examined with caution.

When dealing with small sample regimes, it is unclear whether or not the amount of information contained in a given dataset may be enough to guarantee that the posterior probability (and therefore the Bayesian score) computed for each candidate structure actually reflects its similarity to the true generating structure. Because of the risk of overfitting, many models may explain the data equally well and it may be difficult to distinguish the best among them.

Figure A.11. shows an example of this kind of situations. Since the difference in score for all the structures is very small, any of them might actually correspond to the true generating network that we attempt to learn. As a consequence, it seems that edges which vary from one structure to another cannot be easily learned.

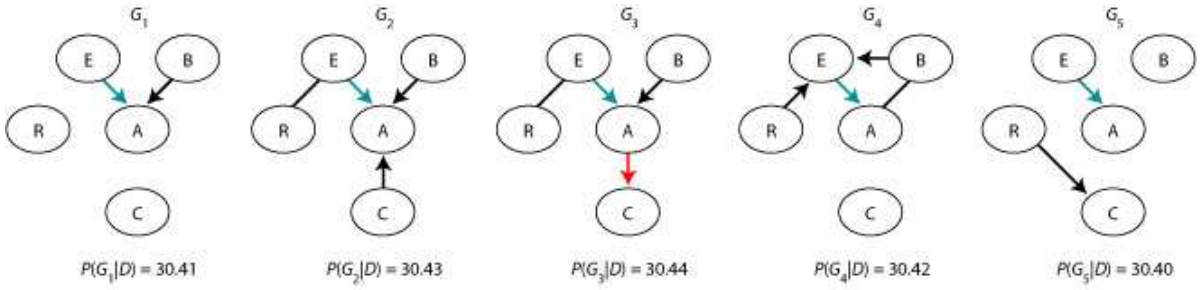

Figure A.11: Example of situation with several networks achieving similar high-scores, from Pe'er's Ph.D. dissertation [2].

Within this context, it seems reasonable to adopt some kind of model averaging technique. In effect, instead of querying single structures, we may search for common traits shared by many of them.

Definition 15. *A network feature is a property of a given structure, such as a particular oriented edge* $(X \rightarrow Y) \in G$ *or a conditional independence statement* $(d - sep_G(X; Y|Z))$.

Definition 16. *For each feature, an indicator function f(G) can be defined, having the value 1 when G satisfies the feature and value 0 otherwise. A level of confidence for the feature over a certain set of m networks can then be computed using the expression:*

$$conf(f) = \frac{1}{m} \sum_{i=1}^{m} f(G_i) \tag{A.25}$$

Actually, this definition of confidence is a simplification from a more rigorous approach that would begin by defining the posterior probability for each given feature as:

$$P(f(G)|D) = \sum_{G} f(G)P(G|D) \tag{A.26}$$

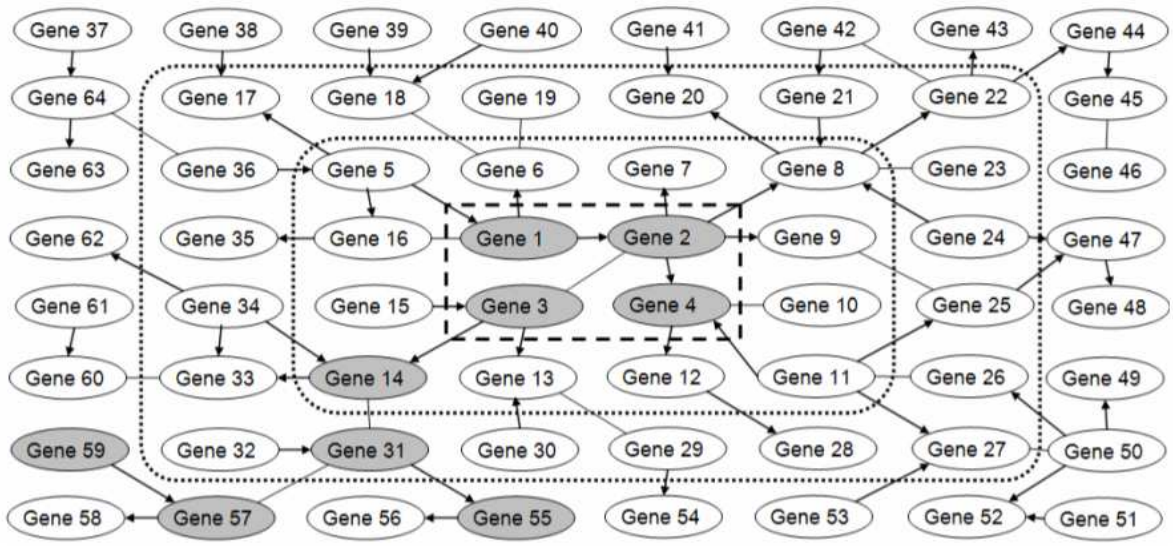

Figure A.12: Example of sub-network learning procedure. Figure depicts a set of high-confidence features (directed and undirected edges) over 64 genes. An initial seed is selected (genes 1-4) and then expanded using a statistical significance score and one of two possible search strategies: (a) radius-based direct search (dotted rectangles show restricted search spaces for radius one [genes 1 to 16] and two [genes 1 to 36] w.r.t. the seed) or (b) greedy hill-climbing (shadowed nodes are an example of a possible structure learned through the use of this strategy).

where G is the set of all possible structures containing the feature (for a given number of nodes). Ideally, all high-scoring networks belonging to G could be taken into account to estimate a good value for the confidence of each feature and those with the highest levels could be retained. In practice, however, the number of high-scoring structures to consider may be too large, and thus it is common to just sample networks from $P(G|D)$ using Markov Chain Monte Carlo (MCMC) approaches.

Since this sampling procedure remains computationally expensive, a further simplification (proposed by Friedman et al.) consists in using non-parametric bootstrap to generate "perturbed" versions of the original dataset and learn a single Bayesian network for each of them. Friedman et al. claim that the datasets obtained using this procedure should provide fairly reasonable descriptions of the real data structure.

Once these "bootstraped" datasets have been generated, they can be all considered equals for averaging purposes and the first expression for feature confidence shown above can be applied.

This sums up the approach adopted for the experiments that were carried out during my internship and that were described in the first part of this report.

Friedman et al., however, go one step beyond in their work and propose the use of a seed-based approach to explore the space of high-confidence features. It seems that, when more than just a few nodes are considered, the number of these features can still be excessively large and a criterium to select the truly relevant ones may be needed. The idea then is to begin with a seed that can either be pointed out by external biological knowledge our automatically selected following surprisingly high concentrations of high-confidence features. This seed is expanded using a new score based on statistical significance and through the use of a new discrete procedure, that can involve either greedy hill-climbing or radius limited direct search.

Figure A.12 offers an intuitive illustration of the whole strategy. For further details, the reader can refer to the original works of Friedman et al. [28] [2], [20].

In any case, it is important to note that these model averaging approaches, without further refine-

ments, no longer respect the Bayesian network formalism since the learned structures may contain, for example, "illegal" directed edges. This should be kept in mind when evaluating the results that they provide.

Appendix B

Matlab implementation

This second appendix provides a description of the technical details concerning the Matlab implementation for the different experiments and simulations presented in previous chapters.

As stated at the beginning of the second chapter, all the simulations were carried out using Matlab 7. When it comes to hardware, two different computers were used offering comparable performances:

- A laptop equipped with an AMD Athlon 64 3200+ processor and 1Gb RAM.
- A desktop equipped with an Intel Xeon 3 Ghz processor and 2Gb RAM.

It is interesting to point out that all the code was written from scratch using the toolboxes and functions included in Matlab and that no external toolboxes were used. This fact has a double sided implication:

- On the down side, some of the functions developed might contain hidden bugs or there might be certain situations (undiscovered as of yet) where the code fails to perform as it should. Even though I have thoroughly tested the code during these last three months, its robustness can never be the same as that of other publicly available implementations (like the BNT by Kevin Murphy, for example) which have already had several versions and have been used and debugged by hundreds or maybe thousands of people during much larger periods of time.
- On the bright side, the fact of having coded each and every function made the code much more transparent to us, we knew exactly what was happening at each simulation, and we could modify and customize all the possible choices of parameters, algorithms, benchmarks, datasets and alike. Also, this makes it possible to easily reuse the code developed and adapt it for possible future research, independently of whether this continues to deal with Bayesian networks or not.

B.1 Code basics

As a general convention for all the code, directed graphs are represented as squared matrixes whose dimension is equal to the number of nodes in the graph.

The value of position (i,j) of the graph matrix represents the kind of link (existence or absence of a directed or undirected edge and orientation) joining nodes i and j, and we can therefore consider three possible values for each position:

- M(i,j) == 0 $\Rightarrow$ nodes i and j are unconnected

- M(i,j) == 1 ⇒ directed edge from i to j
- M(i,j) == -1 ⇒ directed edge from j to i

This implies that we will always work with anti-symmetric matrixes where the elements on the main diagonal will always be zero. Some examples are shown in figure B.1.

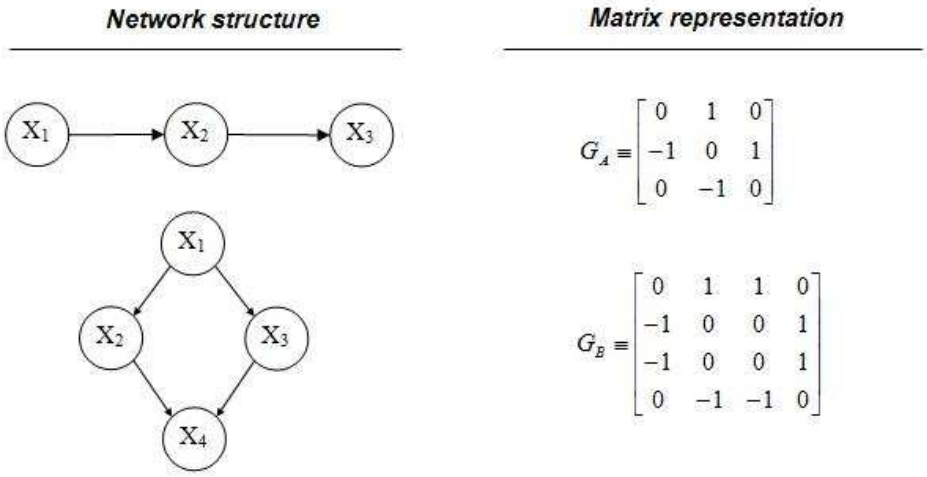

Figure B.1: Example of graph - matrix equivalences.

In the framework of bayesian Networks, however, we are interested in working with classes of equivalence represented by Partially Directed Acyclic Graphs (PDAGs). These graphs consist of both directed edges (which are represented using the criteria from above) and undirected edges (which are represented with a number 2 in the matrix).

The example in figure B.2 illustrates this point.

B.1.1 Synthetic data generation

The synthetic samples used in all the simulations have been created using a function that takes the following parameters as input:

- A graph structure
- A set of conditional probability distributions (multinomials), including:
  - A set of initial probability values for all root nodes, in the form of a vector V of size n [where V(i)= $P(X_i = 1)$]. Unused positions (corresponding to non-root nodes) contain value -1.
  - A matrix M of conditional probability values where each line i contains the probability of node i taking value 1 for the jth combination of its parents. Unused positions contain value -1.
- The number of samples to create (numsamples).

Some possible sample input parameters for graph C from figure B.2 can be:

$$\theta_0 = [0.5 - 1 - 1 - 1]$$

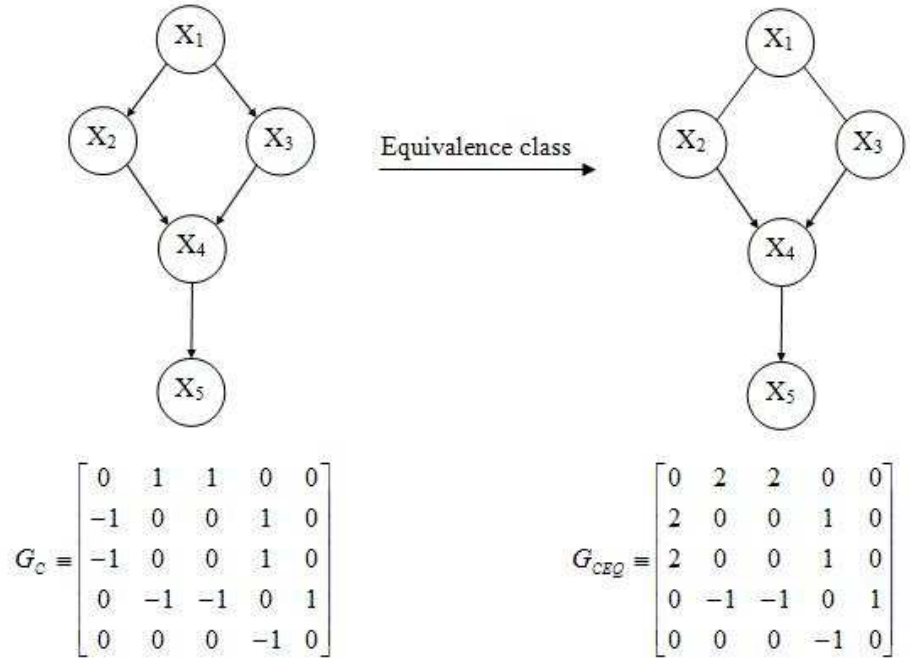

Figure B.2: Example of graph - matrix transformation for classes of equivalence.

$$\theta = \begin{pmatrix} -1 & -1 & -1 & -1 \\ 0.72 & 0.15 & -1 & -1 \\ 0.65 & 034 & -1 & -1 \\ 0.1 & 0.3 & 0.7 & 0.9 \\ 0.1 & 0.9 & -1 & -1 \end{pmatrix}$$

The output of the function is a matrix with *numsaples* lines and n columns, corresponding to the desired set of synthetic samples. As mentioned at the beginning of chapter two, only the binary case is considered and the two only possible values for the samples are 0 and 1.

The implementation of the function was not as simple as it might seem because nodes needed to be assigned values in a certain fixed order that is unknown a priori and depends on the generating structure under consideration. The solution was to go into a loop where a node is only assigned a value after having checked that all its parents have already been assigned values of their own.

The pseudocode for the algorithm is shown in figure B.3.

B.1.2 Priors generation

For all the simulations, uniform priors have been considered. A function that generates them for a given number of n nodes has been created (it basically returns a set of prior samples containing one sample for each possible combination of the n binary values).

The sets of priors provided by this function for the cases of 3,4 and 5 variables are shown in figure B.4.

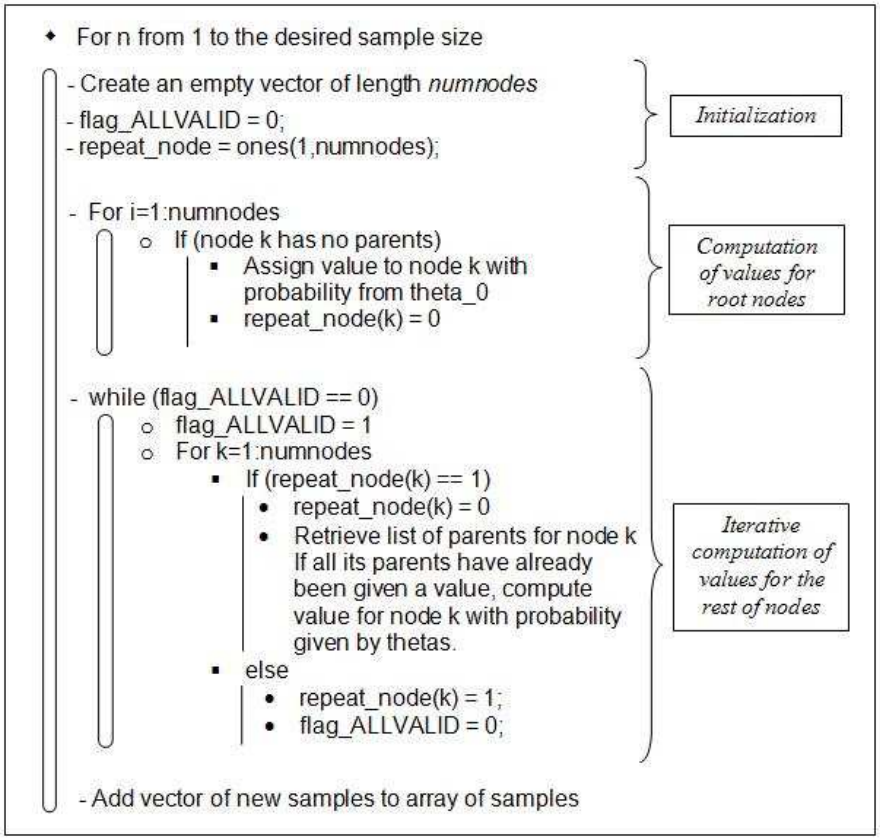

Figure B.3: Pseudo-code for synthetic data generation algorithm.

B.1.3 Bayesian score computation

The core of the score-based approach used for all simulations is a specific Matlab function that computes the Bayesian score for a certain graph structure given a set of samples and a set of priors.

The expression computed for each node is as follows:

$$FamScore(X_k, Pa_{X_k} : D) = \log \prod_{j \in Val(Pa_{X_k})} \frac{\Gamma(a_{jk})}{\Gamma(a_{jk} + N_{jk})} \prod_{i \in Val(X_k)} \frac{\Gamma(a_{ijk} + N_{ijk})}{\Gamma(a_{ijk})}$$

where a_{ijk} is the number of prior samples for which node X_k takes value i (0 or 1) and the parents of node X_k take the j^{th} value of all their possible combinations. The notation used for N_{ijk} is the same. (see example in section 1.3.5)

The Matlab function developed computes the counts for the all the possible combinations of values of each node and its parents both over the samples (N in the formula) and over the priors (a in the formula) and substitutes this counts in the expression above.

Since we work with integers, we use the factorial instead of the gamma function, and we decompose the log of the factorial into a sum of logs to avoid having to deal with excessively big numbers.

For large numbers of samples, Stirling's approximation can be used in order to gain efficiency in terms of computation time. However, since we are working within the framework of a small sample regime, this does not seem to be necessary in practice.

Also, when the number of nodes is not too big, as is the case for the simulations that we conceived, it is interesting to do a pre-treatment of the input samples consisting in the transformation of the original

3 variables	4 variables		5 variables			
000	0000	1000	00000	01000	10000	11000
001	0001	1001	00001	01001	10001	11001
010	0010	1010	00010	01010	10010	11010
011	0011	1011	00011	01011	10011	11011
100	0100	1100	00100	01100	10100	11100
101	0101	1101	00101	01101	10101	11101
110	0110	1110	00110	01110	10110	11110
111	0111	1111	00111	01111	10111	11111

Figure B.4: Choice of prior datasets intended to formalize uniform prior beliefs. The hyperparameters a_{ijk} are obtained by counting over them.

matrix of size $[numsamples] \times [numnodes]$ into another matrix of size $[2^{numnodes}] \times [numnodes+1]$ where we add a column containing the number of occurrences of each of the $2^n umnodes$ possible combinations of binary values of the observed samples. This implies significant savings in counting times for big numbers of samples (since we only need to count over $2^{numnodes}$ elements and then multiply by the number of instances instead of counting over the whole number of samples).

Figure B.5 shows an example of this pre-processing strategy.

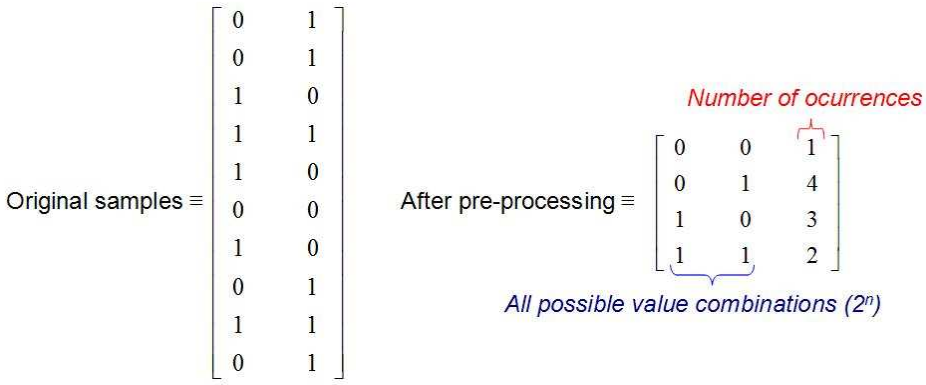

Figure B.5: Example of "sample translation".

B.2 Specific functions

In order to perform certain operations, some specific algorithms were conceived and implemented as Matlab functions. In this section, their pseudocode is presented.

B.2.1 Directed acyclicity test

The first of the specific algorithms that were created for the simulations was a function that takes the matrix for a certain graph as input and returns true if the graph contains directed cycles and false if it does not contain them.

The pseudocode for the function is presented in figure B.6.

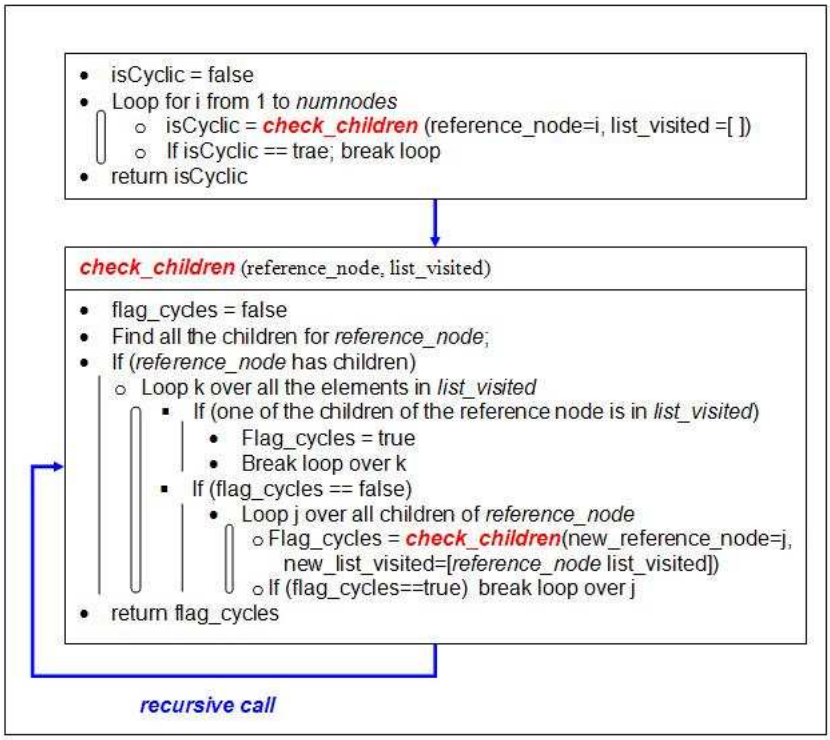

Figure B.6: Pseudocode for the directed acyclicity testing function.

B.2.2 Class of equivalence transformation

The second specific algorithm of interest developed within the framework of the internship was a very simple function that took a certain DAG matrix as input and returned the PDAG for its class of equivalence.

The basic idea here is to identify all the lines in the matrix containing a v-structure (containing two or more incomig edges, and therefore two or more -1 values) and keep them unchanged while turning all other directed edges into undirected ones. After this, the lines containing v-structures are revisited and the antysimmetric property of the matrix is used to force oriented values in mirror positions using them as a reference.

The pseudocode for this algorithm is shown in figure B.7.

This very simple algorithm is complex enough to deal with all the networks considered in the experiments presented in the report, since:

a. None of them contains any node with two incoming edges whose parents are linked with an edge (which should not be a v-structure, according to the definition, but which would be considered to be so by the transformation algorithm) and

b. None of them contains a node with more than three incoming edges (that is, several candidate v-structures sharing the same central node, which could confuse the current version of the transformation algorithm).

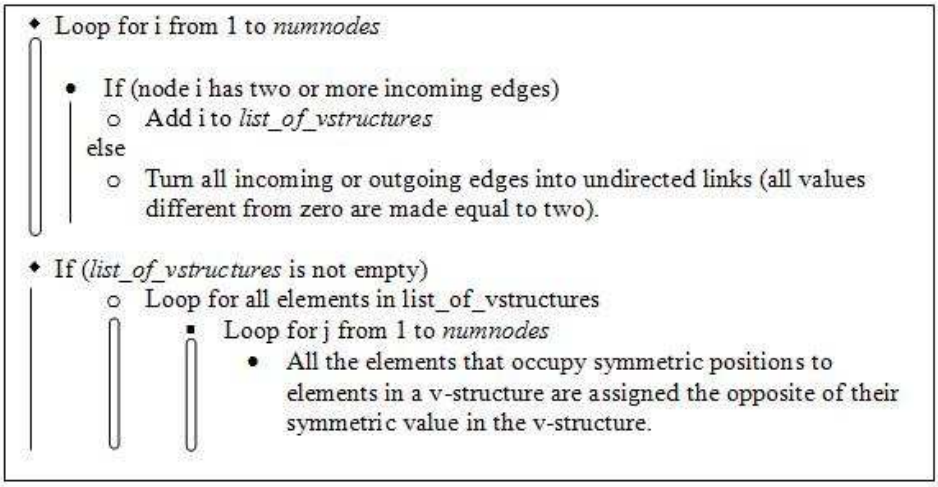

Figure B.7: Pseudocode for class of equivalence transformations.

B.3 Implementation of search procedures

As described in the second section of chapter two, several discrete search procedures have been implemented and used for the simulations. Here, we will provide a brief description of their technical implementation.

B.3.1 Direct search

In this approach, all possible combinations of "legal" (acyclic) graphs are taken into account.

Looking at the matrix representation of a structure, it is easy to observe that the total number of edge positions is n^2. Since the elements in the diagonal of the matrix are always zero, the remaining positions are n^2-n=n(n-1). Since the matrix is antisymmetric, the number of positions that can take different values for different structures is in fact reduced to n(n-1)/2. Finally, each of these positions may take 3 possible values (-1,0,1) and so the total number of structures that can be built with n nodes is $3^{n(n-1)/2}$.

The most straightforward way to perform this kind of search would be to use a set of n(n-1)/2 nested *for* statements allowing to explore all possible networks.

In practice, *for* loops in Matlab are expensive in terms of computational time, and so this approach was used only once to generate a list of models. With a single execution of this nest of *for* loops, all structures were visited and the function described in the previous section was used to check for directed cycles, so that all directed acyclic graphs were kept and the rest were discarded. Each of these remaining graphs was converted into a single vector of length n^2 (using Matlab (:)) and these vectors were stocked in an array. Using this trick, the set of all DAGs for a given number of nodes could be frequently revisited and retrieved for the different simulations using a single *for* loop.

B.3.2 Greedy hill-climbing

At each iteration, all positions in the upper half of the graph matrix are visited and, for each of them, the following transformations are taken into account (their effect being automatically reflected in the antisymmetric bottom half of the matrix):

- No modification

- Edge addition: if position equals 0, turn it into -1 and 1
- Edge inversion and removal: if position is not 0, multiply it by -1 and make it 0.

The new score resulting from each of these individual elementary transformations is computed and the single action over all positions that maximizes this score is selected.

At each iteration, only the action over a single edge position that maximizes the new score is performed. Therefore, a single iteration may imply a change in one of the n(n-1)/2 edges (but it is also possible that all positions remain unchanged).

The variation in score δ_s is computed at each iteration with respect to the previous one. Search stops as soon as δ_s becomes lower than or equal to zero.

B.3.3 Simulated annealing

In this case, a position in the upper half of the graph matrix is chosen at random (they could also be all visited at each iteration) and one of the elementary transformations described above is performed (one of the two possible edge additions for positions equal to zero and edge inversion or removal for the rest).

If the transformation improves the score, it is automatically accepted. Otherwise, it is accepted with probability P_A(iteration) = T(iteration) and rejected with P_R(iteration)=1-T(iteration).

T(iteration) decreases exponentially, according to the following expression:

$$T(iteration) = T_0(1-\epsilon)^{iteration} \tag{B.1}$$

This ϵ is the parameter mentioned in section 2.2. As a general rule for all the experiments described in this report, $T_0 = 1$.

The search stops when a maximum number of stall iterations is reached (given by the other parameter *maxStalls* discussed in section 2.2) and the structure with the highest score of all the structures visited until that moment is kept as the chosen model.

B.3.4 Sparse candidate algorithm

The sparse candidate algorithm is a heuristic proposed in [25] and [2], which mainly attempts to reduce the search space of candidate networks. For the small numbers of nodes considered in the simulations carried out in this internship (up to five variables), this is not very important, but it can make a big difference when larger numbers of nodes are taken into account (ex. cases of study with 800 genes considered by Friedman and Pe'er).

The basic idea behind the algorithm is to calculate at each step the mutual information between each node and each of the other nodes in the network. The top scoring nodes are then selected as the possible parent candidates for the node under study. One of the standard search procedures described above can then be used under the constraint of the parents for each node being restricted to the list of its eligible candidate parents.

According to [2], in practice, finding the nodes that maximize mutual information for a given node is equivalent to finding the nodes that maximize the score that results from each of them being added individually as a parent of the node under study, and so the sparse candidate algorithm can be easily implemented using the Matlab function that computes the bayesian score.

For the simulations presented in this report, a set of candidate parents was selected at the beginning of each iteration of the greedy search procedure. This procedure was modified to consider only those structures that respected the constraint of new parents belonging to the subset of eligible ones. Furthermore, two additional modifications were made to the basic greedy search approach presented above:

- The stop criteria that previously involved δ_s ¿0 was replaced by ($\delta_s >= 0$ & $stalls < 5$). The parameter *stalls* measured the number of consecutive iterations where the reported δ_s was equal to zero. As a consequence, the resulting algorithm was enabled to continue searching over sets of more than one structure obtaining the same score. This was sometimes beneficial because certain edge reversals, corresponding to undirected edges in the class of equivalence and which therefore apparently do not affect the score, may be critical for learning certain v-structures.

- As before, all positions are visited at each iteration and the three possible operations described before are considered for each of them. However, these positions are now visited in a random order, as opposed to the case of the basic greedy search where they were always visited following the same sequence or pattern. This, combined with the introduction of the stalls idea, makes it possible to dig deeper in the quest for high scoring networks.

B.4 Graphical interface

In order to provide an intuitive, user-friendly interface to some of the functions used for our experiments during the internship, a graphic menu for the case of 4 variables was programmed.

Figure B.8 shows a screen-shot of the graphic interface.

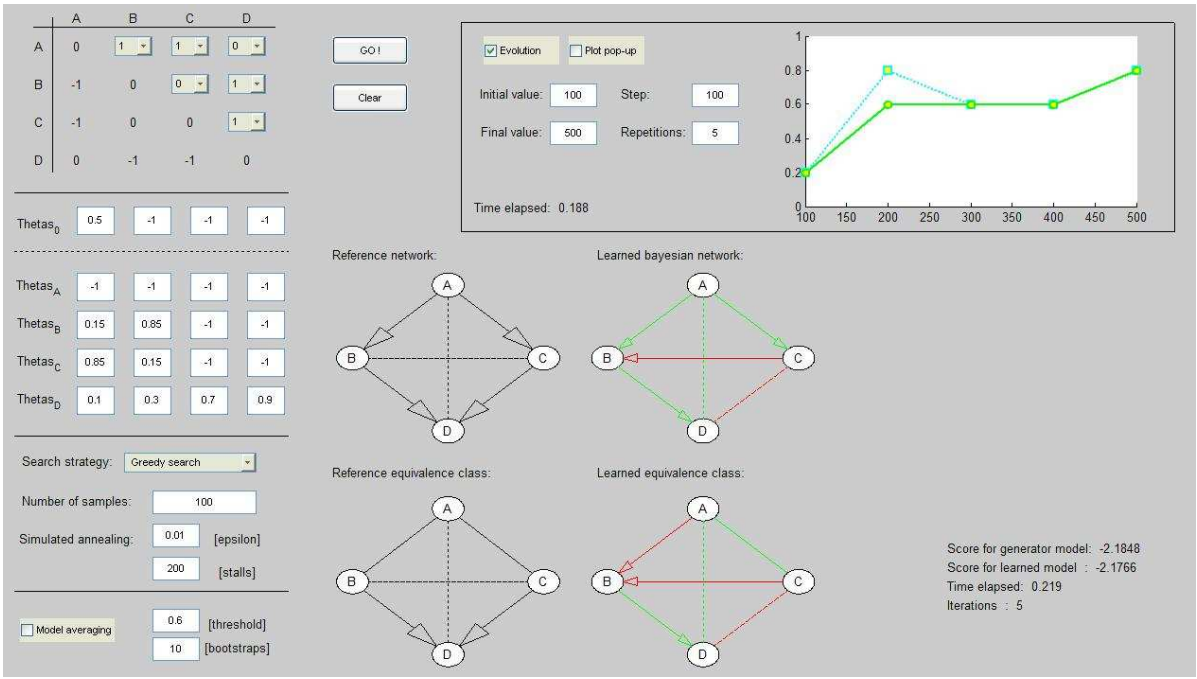

Figure B.8: Pseudocde for the directed acyclicity testing function.

The main functionalities of this interface include:

- Both network structures and values for the parameters of the conditional probability distributions can be easily customized, and all the discrete search procedures discussed in this apprendix can be used.

- When learning individual networks, the corresponding class of equivalence for the original and target networks are presented graphically.

- The model averaging approach has been implemented, and both the threshold for the level of confidence and the number of learned bootstrap networks can be chosen.
- For the graph representations for the networks learned using the model averaging approach, the color and thickness of an edge reflect the level of confidence of the feature.
  - Green lines are used for correctly accepted edges, red lines for wrongly accepted edges and blue lines for "missed edges" (meaning that they did not achieve the minimum required level of confidence to be accepted).
  - Thicker lines correspond to higher levels of confidence.
- Furthermore, it is possible to both learn a single network for a given sample size or to plot the evolution of the learning performances over a range of sample sizes by selecting an initial value, a final value and a sample size step. In these cases, the plot corresponds to the success rate for individual networks (with the dashed line representing the success rate measured over the underlying undirected graph or "eskeleton") and to the average number of correct, wrong and missed edges for the model averaging option.

In any case, the goal of this interface was not to become a complete scientific benchmark, but was just to offer an easy to grasp, pedagogically attractive tool to test the different concepts studied during my internship within a very simplified scenario.

The interface and all the necessary files for its execution can be downloaded from the following address:

http://www.cis.jhu.edu/∼sanchez/simulatorBN.zip