UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MARIANA RECAMONDE MENDOZA

A Study on Machine Learning Methods for the Reverse Engineering of Genetic Regulatory Networks

Trabalho Individual I
TI-1364

Prof^a^. Dra. Ana L. C. Bazzan
Advisor

Porto Alegre, June 2011

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS 5

LIST OF FIGURES 7

LIST OF TABLES 9

ABSTRACT 11

RESUMO 13

1 INTRODUCTION 15

2 BIOLOGICAL ASPECTS OF GENE REGULATORY NETWORKS . . . 19
2.1 The DNA structure 19
2.2 Central Dogma of Molecular Biology 20
2.2.1 DNA Replication 21
2.2.2 Transcription 23
2.2.3 Translation 24
2.3 Regulation of Gene Expression 25
2.4 Genetic Regulatory Networks 25
2.4.1 Macro-characteristics 26

3 MEASURING GENE EXPRESSION LEVELS 27
3.1 Microarray 27
3.2 Real-Time PCR 28
3.3 High-Throughput Sequencing 29
3.3.1 Roche/454 FLX Pyrosequencer 30
3.3.2 Illumina/Solexa 30
3.4 Discussion 33

4 CLUSTERING ALGORITHMS 35
4.1 Hierarchical Clustering 36
4.1.1 Agglomerative Hierarchical Clustering 36
4.1.2 Advantages and Disadvantages 38
4.1.3 Application to Gene Expression Data 38
4.2 K-means 39
4.2.1 Advantages and Disadvantages 40
4.2.2 Application to Gene Expression Data 41
4.3 Self-Organizing Maps 41

4.3.1 Advantages and Disadvantages 43
4.3.2 Application to Gene Expression Data 43
4.4 Tools 44
4.5 Discussion 45

5 BAYESIAN NETWORKS 47
5.1 Bayesian Networks Structure 47
5.2 Learning Bayesian Networks 49
5.3 Markov Chain Monte Carlo: a Sampling Method 51
5.3.1 MCMC Convergence Assessment 53
5.4 Application to Gene Expression Data 54
5.5 Advantages and Disadvantages 57
5.6 Tools 58
5.7 Discussion 58

6 BOOLEAN NETWORKS 61
6.1 Random Boolean Networks 61
6.2 Learning Random Boolean Networks 63
6.2.1 Consistency Problem 64
6.2.2 Best-Fit Extension Problem 64
6.3 Application to Gene Expression Data 64
6.4 Advantages and Disadvantages 67
6.5 Tools 68
6.6 Discussion 68

7 NEURAL NETWORKS 71
7.1 Artificial Neural Networks 71
7.2 Learning Artificial Neural Networks 73
7.2.1 Paradigms 74
7.2.2 Error-Correction Learning 76
7.3 Application to Gene Expression Data 78
7.4 Advantages and Disadvantages 79
7.5 Tools 80
7.6 Discussion 80

8 CONCLUSION 83

REFERENCES 85

LIST OF ABBREVIATIONS AND ACRONYMS

DNA	*Deoxyribonucleic acid*
RNA	*Ribonucleic acid*
mRNA	*Messenger RNA*
rRNA	*Ribosomal RNA*
tRNA	*Transfer RNA*
ncRNA	*Non-coding RNA*
siRNA	*Small interfering RNA*
miRNA	*Micro RNA*
GRN	*Gene Regulatory Network*
PCR	*Polymerase Chain Reaction*
SOM	*Self-Organizing Maps*
BN	*Bayesian Network*
DBN	*Dynamic Bayesian Network*
DAG	*Directed Acyclic Graph*
MCMC	*Markov Chain Monte Carlo*
$(MC)^3$	*Metropolis Coupled Markov Chain Monte Carlo*
RBN	*Random Boolean Network*
PBN	*Probabilistic Boolean Network*
GA	*Genetic Algorithm*
ANN	*Artificial Neural Network*
RNN	*Recurrent Neural Network*
BP	*Backpropagation*
PSO	*Particle Swarm Optimization*
BPTT	*Backpropagation Through Time*

LIST OF FIGURES

Figure 2.1: Nucleotides present in DNA and RNA 19
Figure 2.2: The DNA structure 20
Figure 2.3: Central dogma of molecular biology 21
Figure 2.4: DNA replication 22
Figure 2.5: Transcription 23
Figure 2.6: Structure of transfer RNA 24
Figure 2.7: Common motifs in GRN 26

Figure 3.1: The microarray experiment 28
Figure 3.2: High-throughput sequencing methods 29
Figure 3.3: The Roche/454 FLX Pyrosequencer 31
Figure 3.4: The Illumina/Solexa Genome analyser 32

Figure 4.1: Graphical representation of hierarchical clustering 36
Figure 4.2: Difference between agglomerative and divisive hierarchical clustering algorithms 37
Figure 4.3: Inter-cluster linkage metrics for hierarchical clustering 37
Figure 4.4: Example of the application of K-means algorithm 40
Figure 4.5: The self-organizing map structure 43

Figure 5.1: Example of simple Bayesian network structure 48
Figure 5.2: Bayesian inference 50
Figure 5.3: MCMC convergence test 53
Figure 5.4: Bayesian Hierarchical Model 56
Figure 5.5: Difference between static BNs and dynamic BNs 57

Figure 6.1: Example of a Boolean network with simple structure 61
Figure 6.2: State transition graph for the example Boolean network 63

Figure 7.1: Elements of the neuronal model 72
Figure 7.2: Multilayer artificial neural network structure 73
Figure 7.3: Learning paradigms 75
Figure 7.4: Backpropagation Through Time 79

LIST OF TABLES

Table 5.1: Number of nodes vs. number of networks 50

Table 6.1: Boolean functions for the example Boolean network 62
Table 6.2: State transition table for the example Boolean network 63

ABSTRACT

With the recent and outstanding advances in gene expression measuring technologies, a huge volume of data has been produced, introducing a new interesting research area in the field of bioinformatics: the reconstruction and analysis of gene regulatory networks from the data themselves. By constructing a graph model for the genetic interactions comprising organisms, scientists are able to test hypothesis *in silico* and make new predictions about an organism's response to a particular stimulus or environmental change. Computer science has played an inportant role in this process, providing the tools and algorithms for the analysis of such data. This work aims to outline some widely used machine learning methods in this context and identify their respective weaknesses and strengths, as well as review their main applications to the inference of gene regulatory networks from gene expression data.

Keywords: Artificial Intelligence, Machine Learning, Reverse Engineering, Gene Regulatory Networks, Gene Expression.

Um Estudo sobre Métodos de Aprendizagem de Máquina para a Engenharia Reversa de Redes Regulatórias Genéticas

RESUMO

Com o recente e notável avanço nas técnicas para medição de expressão gênica, um grande volume de dados está sendo continuamente produzido, introduzindo uma interessante linha de pesquisa no ramo da bioinformática: a reconstrução e análise de redes regulatórias genéticas a partir dos dados propriamente ditos. Através da construção de um modelo de grafo, cientistas são capazes de testar hipóteses *in silico* e realizar novas predições a respeito da resposta de um organismo a diferentes estímulos e altera ções ambientais. A ciência da computação vem desempenhando um papel importante neste processo, fornecendo ferramentas e algoritmos para a análise dos dados biológicos. Este trabalho visa destacar algoritmos de aprendizagem de máquina amplamente utilizados neste contexto e identificar os pontos fracos e fortes associados a cada um deles, assim como revisar as principais aplicações destes métodos à inferência de redes regulatórias genéticas a partir de dados de expressão gênica.

Palavras-chave: Inteligência Artificial, Aprendizagem de Máquina, Engenharia Reversa, Redes Regulatórias Genéticas, Expressão Gênica.

1 INTRODUCTION

Since the discovery of DNA structure, scientists have dedicated a great effort to better understand how the information comprised in a DNA molecule is interpreted and propagated so that the phenotypic variability observed in nature may be possible. In a general way, all organisms cells carry the same set of genes; what distinguishes their unique properties and function is the subset of expressed genes. Gene expression is the term used to denote the process by which functional gene products are synthesized based in the genetic information coded in a gene. These products are often proteins, which have a prime relevance for life maintenance on account of almost every cellular function be protein-dependent. Therefore, regulation of gene expression, or gene regulation, is essential for life versatility and adaptability (SWAIN; MANDEL; DUBITZKY, 2010).

An important goal emerged on genomic research and, more recently, in bioinformatics, is to understand the nature and control of cellular function and the reasons why cellular systems fail in disease. Despite all the knowledge about genes, RNA transcripts, proteins and metabolites as individual entities, very little is known about how these elements are integrated and interact in a biological system, as well as how to manipulate their functioning (MARBACH, 2009). In contrast to the former reductionist approach, which performs a gene-by-gene analysis in order to reveal how an organism works and how its components relate to each other, scientists are attempting to accomplish this task by investigating the behavior of genes in a holistic way. What prompted this shift was the awareness that genes activity is not isolated or independent of each other (SHMULEVICH et al., 2002). Indeed, nowadays it is known that genes compose intricated networks through which they work in concert to promote life sustainability. Therefore, the discovery of such networks, named gene regulatory networks, based in experimental data is a logical step towards a better understanding about the biological role played by each gene. In literature, this problem is commonly referred to as reverse engineering of gene regulatory networks.

Gene regulatory networks (GRN) are graph models that reflect the mechanisms and dynamics of gene expression and regulation by mapping the physical or influence interactions between genes of a particular organism. The vertices of this graph usually represent genes, while the edges describe the regulatory relationships between genes (FOGELBERG; PALADE, 2009). Once modeled, these networks explain how genes are over and under expressed in response to perturbation signals and environmental changes. The main benefit of constructing a global regulation model is to provide insight about how to control or optimise parts of the system while taking into account the consequences for the system as a whole (HECKER et al., 2009). This genome-wide approach is relevant since it renders significant information for pharmaceutical industry and medical treatments development. If one knows, for instance, the exact target gene over which a particular pathogenic agent acts in its host, infections could be treated more effectively and with reduced side effects. In plants, this knowledge could be applied for the development of cures less environmental and health aggressive than pesticides. Additionally, the resulting model could be used for in silico experiments in order to test hypotheses and make predictions about the behavior of the biological system under different experimental conditions.

The expressive and continuous progress in techniques for gene expression profiling is another important contributing factor for the latter interest in inferring gene regulatory networks from the data themselves. The current available technologies allow the measurement of the expression of thousands of genes at once, providing valuable information to create a comprehensive picture of cellular function. The expression level is often given in terms of mRNA concentration in the cell, which acts as a proxy for the transcription activity, since this can not be directly measured. The basic assumption is that one can infer causality of transcription regulation from changes in the available gene expression profiles (HACHE; LEHRACH; HERWIG, 2009). However, this approach has two important shortcomings. First, the process of regulation is in fact much more complex and involves other biological entities than just transcripts. Therefore, the constructed model represents a simplification of the real biological process. Yet it is a valuable source of knowledge towards a better understanding of gene expression and regulation. Second, the prevailing technologies produce such a massive amount of data that its analysis is no longer achievable through manual efforts.

According to HECKER et al. (2009), the use of novel computational methods to learn large-scale models by an integrative analysis of the available biological data is both essential and challenging. The developed methodologies must be both statistically sound and computationally tractable for analyzing such data sets and inferring biological interactions from them (FRIEDMAN et al., 2000). A wide range of mathematical and computational methods have been already applied for the purpose of reverse engineering gene regulatory networks. The proposed modeling frameworks vary from abstract boolean descriptions to detailed differential equations, where every representation has its drawbacks and benefits. Due to the typically limited knowledge about the biological system under study, machine learning methods have been favored in the inference of gene regulatory networks because they are able to extract information based solely on the biological data, without relying on any prior knowledge that one has about the system that originated it.

The present work presents a broad review about the main machine learning techniques used in the process of inference of gene regulatory networks. This study will concentrate, mostly, in making a survey about the learning algorithm embedded in each method, the source and characteristics of the input biological data and, finally, the quality of the inference result. The main goal is to identify the advantages and limitations inherent to each computational method and possibilities of improvement in order to achieve more precise and accurate results in the future.

This work is organized as follows. In Chapter 2 some of the main concepts involved in gene expression and regulation processes, essentials for a comprehensive understanding about the purpose of the current work, are explained. In Chapter 3 a review about some of the well-known biological experiments and technologies for gene expression measuring is provided. Next, the main machine learning methods already applied to the problem of reverse engineering gene regulatory networks are introduced and discussed in dedicated chapters. Chapter 4 addresses clustering algorithms, providing a detailed description of algorithms such as hierarchical clustering, K-means and self-organizing maps. Chapter

5 presents Bayesian networks and a commonly applied algorithm for Bayesian learning, the Markov Chain Monte Carlo. Chapter 6 covers the modeling and learning of gene regulatory networks by means of random Boolean networks. Chapter 7 deals with neural networks and describes the most well-known error-correction learning algorithms for training these networks. Finally, the main conclusions raised by the literature review presented in this work are discussed in Chapter 8. This discussion is complemented with a brief comment on promising directions for future research.

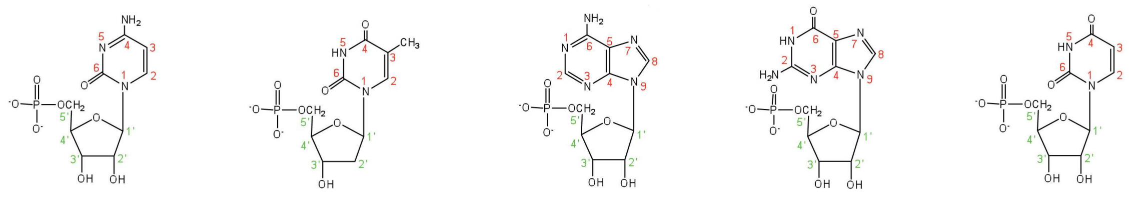
2 BIOLOGICAL ASPECTS OF GENE REGULATORY NETWORKS

In this chapter some of the main biological concepts involved in gene expression and regulation will be briefly presented. The goal is to provide reader with a background on key cellular entities and events for life sustainability and variability. The discussed topics are all closely related to the major subject of this work, gene regulatory networks, and are therefore very relevant for a better understanding on the role of such networks in organisms and the benefits of being able to infer their structure from experimental data.

2.1 The DNA structure

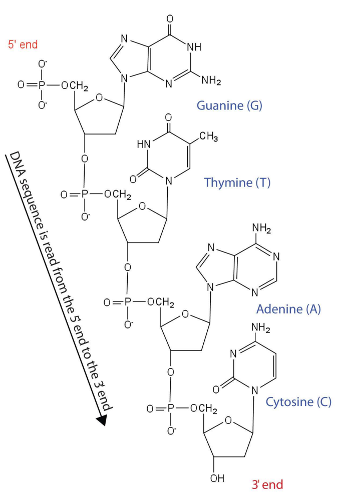
Every single organism in nature is composed of a genome, which carries all biological instructions for constructing and maintaining life. More specifically, these instructions are codified in small portions of the DNA (deoxyribonucleic acid), a polymeric molecule made up of chains of monomeric subunits called nucleotides (BROWN, 2002). The backbone of each DNA nucleotide is composed of three components, as shown in Figure 2.2(a): a deoxyribose sugar, which is a pentose, a phosphate group attached to the 5'-carbon of the pentose, and a nitrogenous base attached to the 1'-carbon of the pentose. There are four distinct nitrogenous bases: cytosine, thymine, adenine and guanine. Cytosine and thymine are double-ring compounds classified as purines, while adenine and guanine are single-ring compounds known as pyrimidines. For the sake of simplicity, nucleotides are often represented by their base's first letter: C, T, A, G, respectively.

Although the existence of DNA has been discovered in 1869 by Johann Friedrich Miescher, a Swiss biochemist, the notion of DNA as a genetic material and the concept of gene were only introduced in 1944, by Avery, MacLeod and McCarty (ZAHA et al.,

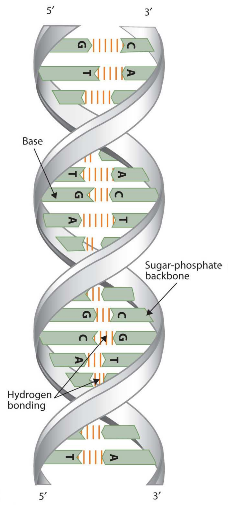


(a) Cytosine (b) Thimine (c) Adenine (d) Guanine (e) Uracile

Figure 2.1: **Nucleotides present in DNA and RNA.** Structure of DNA's monomeric subunits (a) cytosine, (b) thymine, (c) adenine and (d) guanine. The (e) uracil nucleotide is only found in RNA, as a substitute for thymine. Reproduced from (BALL; HILL; SCOTT, 2011).



(a) Primary structure



(b) Secondary structure

Figure 2.2: **The DNA structure.** (a) DNA is made up of nucleotides, each of which is composed of a pentose, a phosphate group and a nitrogenous base. The nucleotide is identified by the base it contains: adenine (A), guanine (G), cytosine (C) or thymine (T). (b) The complementary property between pairs A – T and C – G provides a double-helix structure to the DNA molecule. Reproduced from (BALL; HILL; SCOTT, 2011).

2003). Later, in 1953, James Watson and Francis Crick elucidated the three-dimensional structure of DNA proposing the double-helix model, depicted in Figure 2.2(b).

In the Watson-Crick model, the two individual DNA strands are wrapped around each other in a helix shape, with the sugar-phosphate backbone in the outside, exposed to the aqueous environment, and the nitrogenous bases in the internal portion. Pairs of bases of opposite strands form bonds between each other according to a restrict rule: A only pairs with T, while C only pairs with G. This process is referred to as complementary base pairing and is a crucial feature for cellular events that will be further introduced. Another important characteristic of the double-helix model is the antiparallel alignment of DNA strands: one strand has direction $3' \rightarrow 5'$ and the other is disposed in direction $5' \rightarrow 3'$.

The stability of the double-helix model is guaranteed by two main features. First, the hydrophobic portion of the DNA molecule, i.e. the nitrogenous bases, is protected from the aqueous environment by the sugar-phosphate backbone, which has hydrophilic properties. Second, the chemical structure of nitrogenous bases, which comprises a ceto and amino groups, allows the formation of hydrogen bonds between the pair of bases. A pair C – G has three hydrogen bonds, while the pair A – T has two hydrogen bonds. Therefore, DNA molecules with more C – G pairs are more stable as they require a higher temperature to disassociate. The principle of complementary pairing between purine and pyrimidine bases also provides DNA molecule with constant long dimensions and perfect fitting between both strands.

2.2 Central Dogma of Molecular Biology

All cells within an organism carry a copy of its genome, made up of one or more long DNA molecules. In more complex organisms, i.e. eukaryotics, as animals and plants, the genome is inside the cell's nucleus and is organized in the form of chromosomes. These

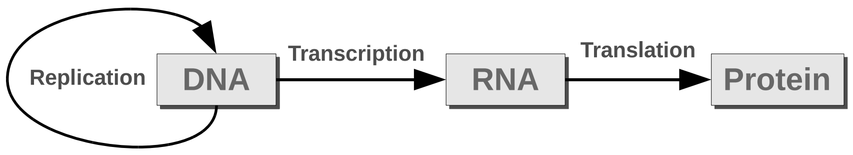


Figure 2.3: **Central dogma of molecular biology.** Three main processes are responsible for the perpetuation and interpretation of genetic information encoded in DNA: (1) replication, in which new copies of DNA are made; (2) transcription, in which RNA is produced from a segment of DNA; and (3) translation, in which the information in RNA is translated into a protein sequence.

cells are usually diploids, which means that they carry two instances of each chromosome, one from each parent. In contrast, prokaryotic organisms, such as virus and bacteria, lack a nuclear membrane. The nuclear region consists of circular DNA, known as nucleoid, and contains only one set of chromosomes.

Regardless the cellular complexity, all organisms life depends on the cell's ability to save, transfer and translate the genetic instructions encoded in the DNA, which define the structure and function of all livings things. More specifically, a DNA strand is composed of thousands of functional portions, called genes. Each gene is composed by a coding region, which carries the necessary and sufficient information for the production of two other key classes of polymers through the process of gene expression – RNA and proteins, and a regulatory region, responsible for the control of gene expression and thus protein biosynthesis.

RNA (ribonucleic acid) is a polymer chemically and structurally similar to DNA, differing from the latter in two main aspects: RNA is composed of a ribose sugar and contains the nitrogenous base uracil (U) instead of thymine. Some viruses use RNA rather than DNA as their genetic material, and all organisms use messenger RNA (mRNA) to carry the genetic information that directs the synthesis of proteins.

Proteins are the main functional components of organisms. They are composed of special monomers called amino acids, which are bonded together by peptide bonds, and play a crucial role for the development and survival of organisms. In brief, proteins are on duty of functions as: catalyse chemical reactions, as enzymes; defend organism, as antibodies; activate or deactivate a specific set of genes, as transcription factors; and provide structural support, as fibrous proteins such as actin, collagen and elastin.

The mechanisms by which the genetic material is perpetuated through generations and interpreted to allow the synthesis of these vital molecules are the basis of the well-known Central Dogma of Molecular Biology and will be discussed in what follows.

2.2.1 DNA Replication

For an organism to grow and reproduce, the cellular ability to divide and duplicate is essential in order to increase cellular complexity and pass on the organism phenotype. These processes, however, require DNA to be duplicated inside the cell so that it can be split between two daughter cells, which will be identical to the parent cell. The mechanism of DNA copy is known as replication.

When Watson and Crick suggested the double-helix model for DNA structure, in 1953, they made one of the most famous statements in molecular biology: "*It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material*" (BROWN, 2002). In fact, years later it was discovered that the structure of DNA as two long nucleotide strands con-

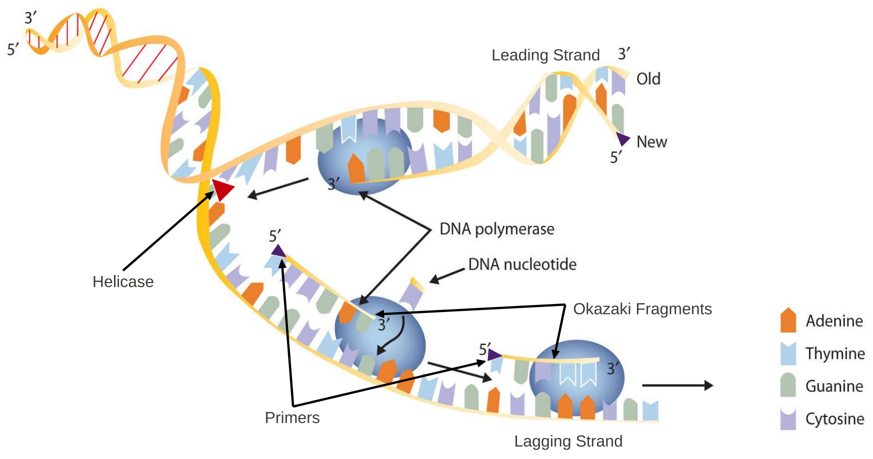


Figure 2.4: **DNA replication.** DNA is replicated through a semiconservative process: the parental DNA chain is separated such that each one of the DNA strands work as a template for the synthesis of the new DNA. The synthesis is based in the complementary pairing between nitrogenous base. Reproduced from (BALL; HILL; SCOTT, 2011).

nected by the principle of complementary pairing between nitrogenous bases is indeed the key of the replication process. Figure 2.4 depicts in a simplified form how replication of DNA occurs.

Replication starts at particular points of DNA, known as origins, which are targeted by helicases enzimes responsible for breaking up the hydrogen bonds between bases and unwinding a short segment of DNA. Once initiated, two replication forks can emerge from the origin and progress in opposite directions along the DNA: replication is therefore bidirectional in most genomes (BROWN, 2002). With the two strands of DNA separated, each individual strand act as a template for the synthesis of a complementary DNA chain. Special proteins known as DNA polymerases synthesize the new DNA by adding complementary free nucleotides that match the sequence in the template strand, as stated by the previously mentioned rule: A only pairs with T, while C only pairs with G. The exact point where complementary pairing begins is determined by special strands of nucleic acid called primers, which provide a short double-stranded region with a $3'$ end onto which the enzyme can add new nucleotides.

However, DNA polymerases enzimes are only able to synthesize DNA in the $5' \rightarrow 3'$ direction, which means that only one strand of the parental double-helix, the one in $3' \rightarrow 5'$ direction called the leading strand, can be copied in a continuous manner. The replication of the lagging strand, which has $5' \rightarrow 3'$ direction, is carried in a discontinuous fashion, resulting in a series of short segments known as Okazaki fragments (BROWN, 2002). These fragments are processed by a specific type of DNA Polymerase to remove primer sequences and add new deoxyribonucleotides to fill the gaps. Finally, Okazaki fragments are merged by the DNA ligase enzime. Note that replication requires only one primer to initiate complementary strand synthesis on the leading polynucleotide, and one primer for every segment of discontinuous DNA synthesized on the lagging strand. Since one strand of the new DNA comes from the parent cell, replication is widely referred to as a semi-conservative process.

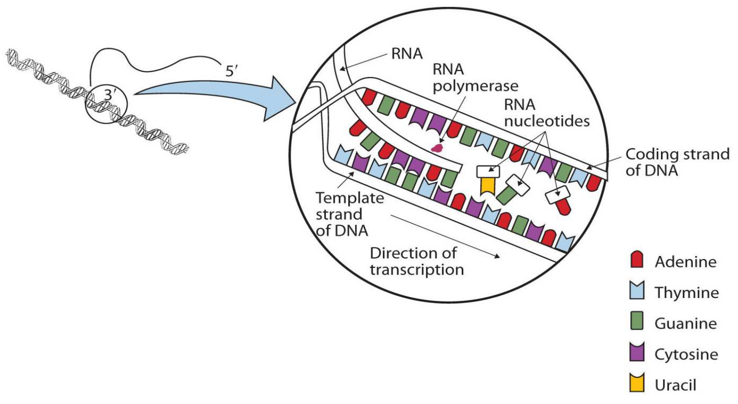


Figure 2.5: **Transcription.** The first step towards gene expression is given by the transcription process, in which RNA molecules are synthesized based on information contained in the nucleotides sequence of a double-helix DNA chain. This is not a regular process in the sense that the specific DNA sequences to be transcribed and the amount of RNA produced are regulated by special proteins and are varying parameters. Reproduced from (BALL; HILL; SCOTT, 2011).

2.2.2 Transcription

Genetic information encoded on DNA is only useful to direct the growth and functioning of an organism once it is expressed, e.g. when the protein it codes for is produced. The first step towards gene expression is given by the transcription process. Transcription is the mechanism by which RNA molecules are synthesized based on information contained in a double-helix DNA molecule, as shown in Figure 2.5.

Transcription starts and finishes at specific points of a gene's regulatory region, called promoter and terminator respectively. When a promoter is recognized by a RNA Polymerase, the class of protein in charge of transcription, the two strands of the double-helix DNA unwind at specific sites along the DNA molecule, just as happens in replication, by breaking the hydrogen bonds between nitrogenous bases. Once the DNA strands are separated, ribonucleotides are added one after another to the growing $3'$ end of the RNA transcript following the base-pairing rules and the nucleotides sequence present in the template, which is given by the $3' \to 5'$ strand of DNA. The resulting RNA is antiparallel and complementary to the template strand and is identical to the corresponding coding strand of the DNA (the strand in $5' \to 3'$ direction in the parental DNA molecule), except that uracil replaces thymine bases.

The RNA synthesis stops when a terminator is identified by the RNA Polymerase. At this point, the RNA transcript is released and the RNA Polymerase is responsible for wrapping the parental DNA chains around each other in the helix shape. The produced RNA transcript is referred to as pre-RNAs or primary transcript and in eukaryotes species, mainly, it needs to be posteriorly processed to constitute a biologically active RNA. The RNA processing often involves modification (insertion/deletion) of some nitrogenous bases, changes in the chemical structure and splicing, in which non-coding regions (introns) are eliminated and coding regions (exons) are joined, forming the mature RNA.

There are many different types of RNA, all of them produced and post-processed as described above. However, three types are essential for protein synthesis and therefore are more discussed in literature: messenger RNA (mRNA), ribosomal RNA (rRNA), and transfer RNA (tRNA). mRNA carries information about protein sequence to the ribosomes, the protein factories in the cell, and corresponds to 5% of the total RNA. Also, mRNA exists for a relatively short time: it is continuously being degraded and resynthesized. The ribosomes are composed of rRNA, the most abundant type of RNA, which makes up to 80% of the total RNA found in an eukaryotic cytoplasm. Finally, tRNA corresponds to around 15% of the total RNA and is responsible for transferring specific amino acids to the ribosomal site of protein synthesis, which are attached to the growing polypeptide chain.

Until 1990s, other classes of RNA apart from mRNA, tRNA and rRNA were essentialy unknown. Nonetheless, in recent years an enormous number of non-protein-coding RNAs (ncRNAs) have been found to play an important role on post-transcriptional gene regulation (LI et al., 2010). Small interfering RNAs (siRNAs) and microRNAs (miRNAs) are the most common types of ncRNAs. SiRNAs typically induce endonucleolytic cleavage of mRNA with near-perfect complementarity (ALI NAHVI; GREEN, 2011). miRNAs, in turn, interfere in gene expression through both translational repression and mRNA destabilization mechanisms. According to LI et al. (2010), many ncRNAs show abnormal expression patterns in cancerous tissues.

2.2.3 Translation

The final result of gene expression is the proteome, the collection of functioning proteins synthesized by a living cell (BROWN, 2002). The instruction for building a protein is carried in the nucleotides sequence of the mature mRNA. Each three consecutive nucleotides, called codon, code a specific amino acid. The amino acids are the monomers that compose a protein through the formation of a polypeptide chain.

As previously mentioned, the translation of a codon, which involves the decoding of its genetic code into an amino acid, occurs at the ribosomes, made up of rRNA. Before the initiation of translation, amino acids need to be covalently bonded to the correct transfer RNA (tRNA), e.g. the one carrying the complementary sequence of the codon by which the amino acid is produced. The bond occurs between the carboxyl group of the amino acid and the $3'$ OH of the tRNA and is intermediated by an enzyme known as aminoacyl-tRNA synthetase, just as shown in Figure 2.6.

Next, the tRNA carrying the complementary nucleotides to the codon to be translated (the anticodon), binds to the ribosome close to the growing extremity of the polypeptide chain and interacts with the mRNA through base pairing of codon and anticodon. Once the new amino acid is incorporated in the chain through a peptide bond with the last

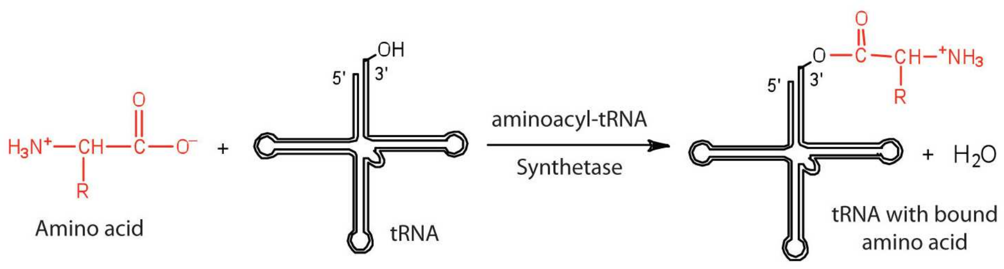


Figure 2.6: **Structure of transfer RNA.** Before translation can takes place, the amino acid must be attached to its unique tRNA. This crucial process requires an enzyme known as aminoacyl-tRNA synthetase. Reproduced from (BALL; HILL; SCOTT, 2011).

aggregated amino acid, positioned at the end of the chain, the ribosome moves right in order to allow the next tRNA, bringing a new amino acid, to correctly attach to the mRNA at its complementary position. This process is continuously repeated until a special codon, called stop codon, is reached. At this point, the polypeptide chain is released from the ribosome into the cell cytoplasm.

In prokaryotes, transcription and translation are coupled: the translation begins while the mRNA is still being synthesized, and both processes happen at the cytoplasm. In contrast, in eukaryotes, transcription and translation are spatially and temporally separated: transcription occurs in a membrane-bound nucleus and translation takes place in the cytoplasm.

2.3 Regulation of Gene Expression

In Section 2.2 the steps composing the pathway by which expression of the genome specifies the content of the proteome were presented. According to BROWN (2002), this biochemical signature is not entirely constant: even the simplest unicellular organisms are able to alter their proteomes to cope with changes in the environment. Regulation of gene expression, or simply gene regulation, is the process by which cells regulate the exact moment and rate with which the information encoded in genes is turned into gene products.

In procaryotes, the control of the rate of transcriptional initiation is the predominant site for gene regulation. In contrast, gene regulation in eukaryotes is much more complex and may happen in a wide variety of ways, as a result of different molecules' activity, from proteins to ncDNA. In what follows, some of the possible regulation mechanisms in eukaryotes are enumerated (BROWN, 2002).

- **Gene acessibility:** the physical structure of chromatin (complex of DNA and proteins located inside the nucleus) can controls access of RNA polymerase and transcription factors to the promoter region, avoiding the activation of transcription;
- **Transcription initiation:** initiation of transcription process is influenced by activators, repressors and other control systems like ncRNA;
- **mRNA processing:** mechanisms such as changes in nucleotides sequence, splicing and mRNA degradation affects protein synthesis;
- **Translation initiation:** the ability of ribosomes to recognize and initiate synthesis from the correct start codon can affect the expression of a gene product;
- **Post-translational modification:** the chemical modification (glycosylation, acetylation, fatty acylation,...) of a protein after its translation may be a control point of protein synthesis;
- **Small RNAs:** small RNA-mediated control can be exerted either at the level of the translatability of the mRNA, the stability of the mRNA or via changes in chromatin structure.

2.4 Genetic Regulatory Networks

As previously discussed, biological systems are complex organisms composed of innumerous entities, such as DNA, RNA and proteins, which interact in order to produce the specific features of each organism. The manner by which these components are interconnected and relate to each other, allowing an harmonic work crucial for cellular sustainability, defines gene regulatory networks (GRN).

Genetic regulatory networks are high-level conceptual representations of the mutual influence between genes in an organism. Their main goal is to capture the dependencies between the molecular entities, representing the physical interactions gene–gene (influence interactions) and indirect gene regulation via protein, metabolites and ncRNA (BANSAL et al., 2007; HECKER et al., 2009). The usual graphical formalism is a direct graph, in which nodes denote genes or other molecular entity, and a connection from node A to node B suggests that A exerts regulation over B, either of activating or inhibiting nature.

The main purpose of exploring GRN is to understand the relationship between genes within a cell and how they respond to intra and extracellular stimulus. In BANSAL et al. (2007), some specific practical utilities of gene regulatory networks are outlined: 1) allows the identification of functional modules[1], (2) helps in the prediction of network response to external perturbation, as well as of genes directly affected by the perturbation and (3) efficient methodology for identifying real physical interactions through integration with additional information from experimental data.

2.4.1 Macro-characteristics

According to FOGELBERG; PALADE (2009), GRN are not just random directed graph. Instead, they carry important macro-characteristics, which will be briefly described in the sequence.

- **Connectivity:** GRN are scale-free networks. The probability distribution for the node's degree follows a power-law, meaning that most of the genes are regulated by only a few others, while some nodes are known as *hubs* and have influence over the expression of many others.

- **Modules:** genes are organized into modules, which define groups of genes co-regulated or equi-regulated, and functionally linked by their phenotypic effects.

- **Motifs:** GRN are composed of subgraphs called motifs, which are much more frequent in GRN's structure than in a randomly generated graph. Common motifs are auto-regulation, feed-forward triangle, cascade and convergence, depicted in Figure 2.7.

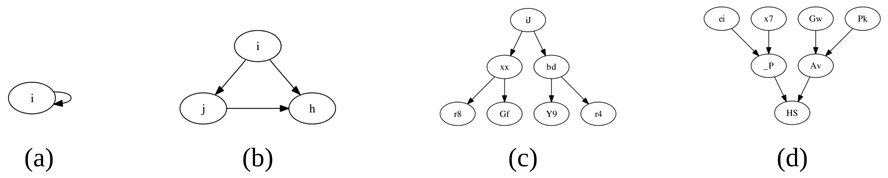


Figure 2.7: **Common motifs in GRN.** (a) auto-regulation, (b) feed-forward triangle, (c) cascade and (d) convergence. Reproduced from (FOGELBERG; PALADE, 2009).

[1]In GRN context, a functional module is a subset of genes that regulate each other with multiple interactions but have few regulatory relations to other genes outside the subset.

3 MEASURING GENE EXPRESSION LEVELS

According to (FOGELBERG; PALADE, 2009), there are four types of biological data available to address the problem of inference of GRN: expression data, perturbation data, phylogenetic data and chemical and gene location data. Gene expression data is nowadays the most widely used as input for reverse engineering algorithms, as the other types are currently not available in sufficient quantity to be incorporated in reverse engineering analysis. Expression data measures how active a gene $i \in N$ is in a given moment or under a given experimental condition. By active, the reader may understand that the functional product, e.g. protein or RNA, coded by the gene is being produced. This chapter will provide a brief review about the main techniques applied nowadays in the measurement of gene expression levels.

3.1 Microarray

The measurement of gene expression using microarrays, also known as DNA chips, is one of the more successful techniques among the many methods developed. It allows the identification and quantification of the mRNA transcripts present in the cells by assuming the following procedure: given that genes are expressed by transcription and translation of their genetic information into mRNA, which will be later used to synthesize proteins, if one is able to find out which and how much mRNA is around, one should also be able to discover which genes and with which intensity they are being expressed (SÁNCHEZ; VILLA, 2008).

A microarray consists of a solid surface on which strands of polynucleotide, called probes, are attached or synthesized by a machine in fixed positions. There are two different types of microarray according to the way probes are placed on the slide. On Spotted or cDNA microarrays, the probes are synthesized apart and printed mechanically on the slide. In contrast, oligonucleotide chips, whose main representatives are Genechip and Affymetrix (manufacturers), have the probes directly synthesized on the surface. In the latter, the synthesis process allows the creation of only small fragments so that a gene is not represented by one probe but by a set of them (a probe set).

Once probes are ready, mRNA is extracted from the subject cells and labeled with a fluorescent dye. Afterwards, the labelled transcripts, called targets, are deposited over the array and left inside a hybridization chamber for some hours. If a labelled target is complementary to one of the gene sequences placed on the probes, it will bind by hybridization to the corresponding spot. Finally, the array is washed in order to eliminate those targets which have not hybridized.

The expression level of genes is measured by illuminating the microarray with a laser light that causes the labeled molecules to emit fluorescent light proportionally to the quantity of hybridized mRNA it contains. This way, active genes will produce more mRNA, which will attach to the DNA on the microarray producing brighter areas. Spots that are not bright indicate that their genes are not active. The emitted light is captured by a special scanner yielding an image that consists in a grid of shined spots, each one corresponding to a probe. Finally, this image is processed and transformed into numbers, consisting the basis of the gene expression analysis.

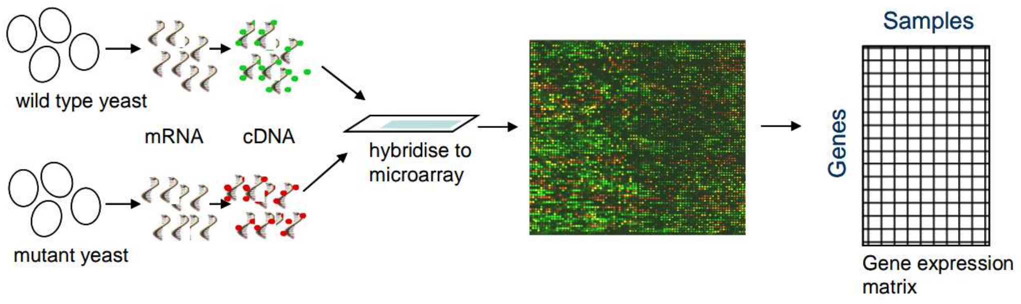


Figure 3.1: **The microarray experiment.** A comparison between gene expression of species with different features (in this case, two yeast cultures, a mutant and a wild one) may be held by extracting mRNA from both samples, labeling it with different fluorescent dyes and letting it hybridize to the microarray. The fluorescence of each spot on the microarray reflects the relative mRNA concentrations, which are later scanned and the resulting intensity is stored as a gene expression matrix. Reproduced from supplementary material provided by SCHLITT; BRAZMA (2007) .

3.2 Real-Time PCR

Polymerase chain reaction (PCR) is a method that allows exponential amplification of short DNA sequences (usually 100-500 pair bases) within a long DNA molecule in a very fast fashion. The process is performed in vitro and generates an incredible amount of DNA for further analysis. PCR use primers, which bind to the two strands of the DNA by complementarity and define the portion of DNA to be copied. The DNA, the primers and other components of reactions are mixed and placed in a Thermocycler, which raises and lowers the temperature of the block in discrete, pre-programmed steps, allowing the duplication of DNA molecule. After the copy is finished, the same primers can be used again, not only to make another copy of the input DNA strand, but also of the short copy made in the first round of synthesis, leading to exponential amplification.

After several rounds of amplification, the PCR product is analysed on an agarose gel and compared with a standard or reference genetic material, providing a qualitative tool for detecting the presence or absence of a particular DNA. Notwithstanding its wide applicability in disease diagnosis, for instance, the need to measure mRNA to analyse differences in gene expression between samples has driven the creation of a traditional quantitative reverse transcriptase PCR (RT-PCR) method. In RT-PCR, the PCR method is extended using reverse transcriptase to convert mRNA into complementary DNA (cDNA), which is then amplified by PCR and analysed in an agarose gel by electrophoresis. Although this method has already been used to measure the levels of a particular mRNA under different experimental conditions, it does not provide a quantitative measure for gene expression at all due to the extra reverse transcriptase step: products are seen after the exponential phase of amplification, which lowers the sensitivity.

Real-time PCR is therefore, an improvement of regular PCR and allows the quantitative estimation of the amount of a given sequence present in a sample. Products are measured after each cycle by detecting a fluorescent light from labelled PCR products, rather than at the end of the run, and the samples are analysed during the exponential phase, where differences in quantity of products are maximized. This method has thus many advantages over conventional PCR: increased speed due to reduced cycle number, lack of post-PCR gel electrophoresis detection of products and higher sensitivity of the fluorescent dyes used for labeling samples. Real-Time PCR has been extensively applied to quantitatively determine levels of gene expression. Data analysis in both traditional and real-time PCR data require normalization to known standards to determine relative or absolute quantity of starting target gene expression.

3.3 High-Throughput Sequencing

The increasing demand for gene expression data has driven the development of high-throughput sequencing technologies. The goal of these new sequencing methods is to parallelize the sequencing process, such that a larger amount of data is provided in a shorter space of time and the overall cost of DNA sequencing is reduced, encouraging more researches in the field. Gene expression level is measured by quantifying the number of sequence reads for each mRNA/gene. A wide range of applications depends on the availability of sequencing data, such as genome sequencing, metagenomic, epigenetics, discovery discovery of non-coding RNAs and protein-binding sites (MACLEAN; JONES; STUDHOLME, 2009), just to give a picture of the relevance in efficiently provide such nature of data.

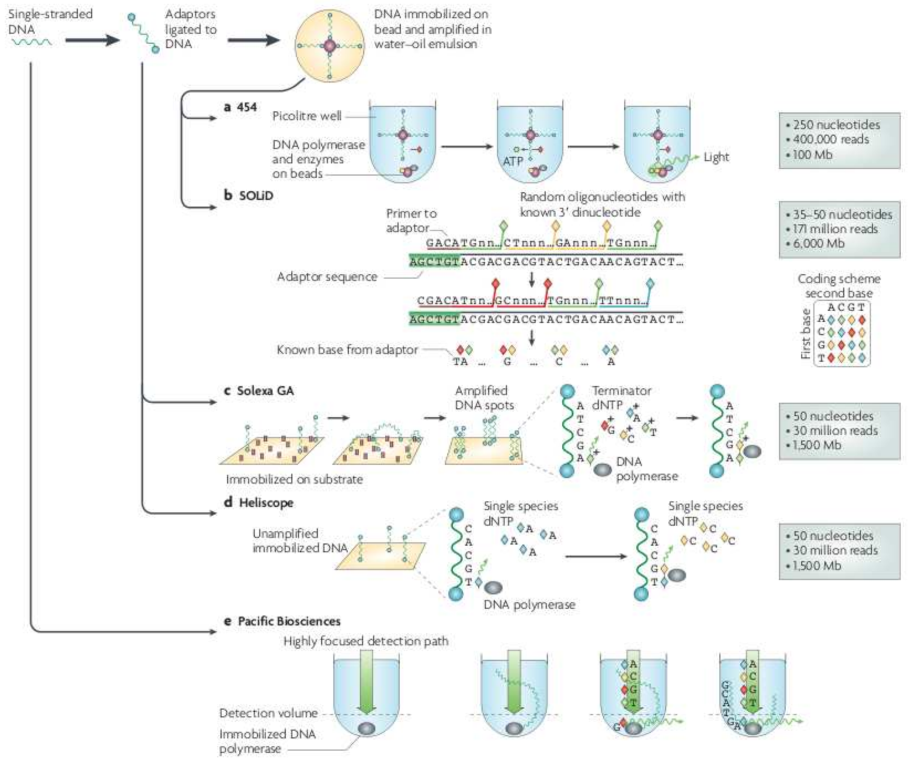


Figure 3.2: **High-throughput sequencing methods.** Reproduced from MACLEAN; JONES; STUDHOLME (2009).

Figure 3.2, reproduced from (MACLEAN; JONES; STUDHOLME, 2009), shows a general picture of the main new-generation high-throughput sequencing technologies. Features such as the read length, the number of reads and the total amount of sequence generated in a typical run are highlighted and compared. In what follows more details are given about two of the widespread used high-throughput sequencing methods, Roche/454 and Illumina/Solexa.

3.3.1 Roche/454 FLX Pyrosequencer

The first high-throughput sequencer to achieve commercial introduction was the Roche/454 FLX Pyrosequencer, in 2004 (MARDIS, 2008). This method is based on the "sequencing by synthesis" principle of pyrosequencing procedure. In pyrosequencing, a complementary strand to the (single) DNA strand to be sequenced is enzymatically synthesized. The activity of the DNA polymerase is detected with another chemiluminescent enzyme, such that each incorporation of a nucleotide by DNA polymerase results in the release of pyrophosphate, which initiates a series of downstream reactions that ultimately produce light by the chemiluminescent enzyme. The amount of light produced is proportional to the number of nucleotides incorporated and the chemiluminescent signals allows the determination of the sequence of the template.

The Roche/454 FLX method requires DNA to be amplified before sequencing procedure, which is done through PCR. As explained in MARDIS (2008), the library fragments are mixed with a population of agarose beads whose surfaces carry oligonucleotides complementary to the 454-specific adapter sequences on the fragment library, so each bead is associated with a single fragment. The agarose beads are placed in individual mixtures of water-oil and PCR reactants, and after the process of thermal cycling about one million copies of each DNA fragment on the surface of each bead is produced. These amplified single molecules are then sequenced en masse. More details of Roche/454 FLX sequencer procedure are shown in Figure 3.3.

After approximately 8 hour of processing, the Roche/454 FLX Pyrosequencer provides an average read length of 250 nucleotides, which are processed by an analysis software and filtered to remove poor-quality sequences. The resulting reads yield 100 Mb of quality data on average. The major drawback of Roche/454 FLX Pyrosequencer is that it cannot properly interpret long stretches (more than 6 bases) of the same nucleotide (homopolymer run), so these areas are prone to base insertion and deletion errors during base calling. By contrast, because each incorporation step is nucleotide specific, substitution errors are rarely encountered in Roche/454 sequence reads.

3.3.2 Illumina/Solexa

Illumina released the Solexa Genome analyser in early 2007 (MACLEAN; JONES; STUDHOLME, 2009). As the Roche/454 FLX Pyrosequencer, the method begins by ligating oligonucleotide adaptors to the DNA and immobilizing the ligation products onto agarose beads. The beads are placed into a water-oil emulsion and DNA is amplified by PCR. After amplification stage, the sequencing process starts by adding all four nucleotides simultaneously to the flow cell channels, along with DNA polymerase, for incorporation into the oligo-primed cluster fragments. The nucleotides carry a base-unique fluorescent label and the $3'$-OH group is chemically blocked such that each incorporation is an unique event. An imaging step follows each base incorporation step, and after that the $3'$-OH blocking group is chemically removed to prepare each strand for the next incorporation by DNA polymerase. This series of steps continues for a specific number of cycles determined by user, which permits discrete read lengths of 25-35 bases. A base-calling algorithm assigns sequences and associated quality values to each read. Finally, a quality checking pipeline evaluates the Illumina data from each run, removing poor-quality sequences. Illumina Solexa Genome analyser provides about 30 million reads with an average length of 50 nucleotides. After quality filtering, the resulting data set has around 1.5Gb. The overall process of Solexa Genome analyser is depicted in Figure 3.4.

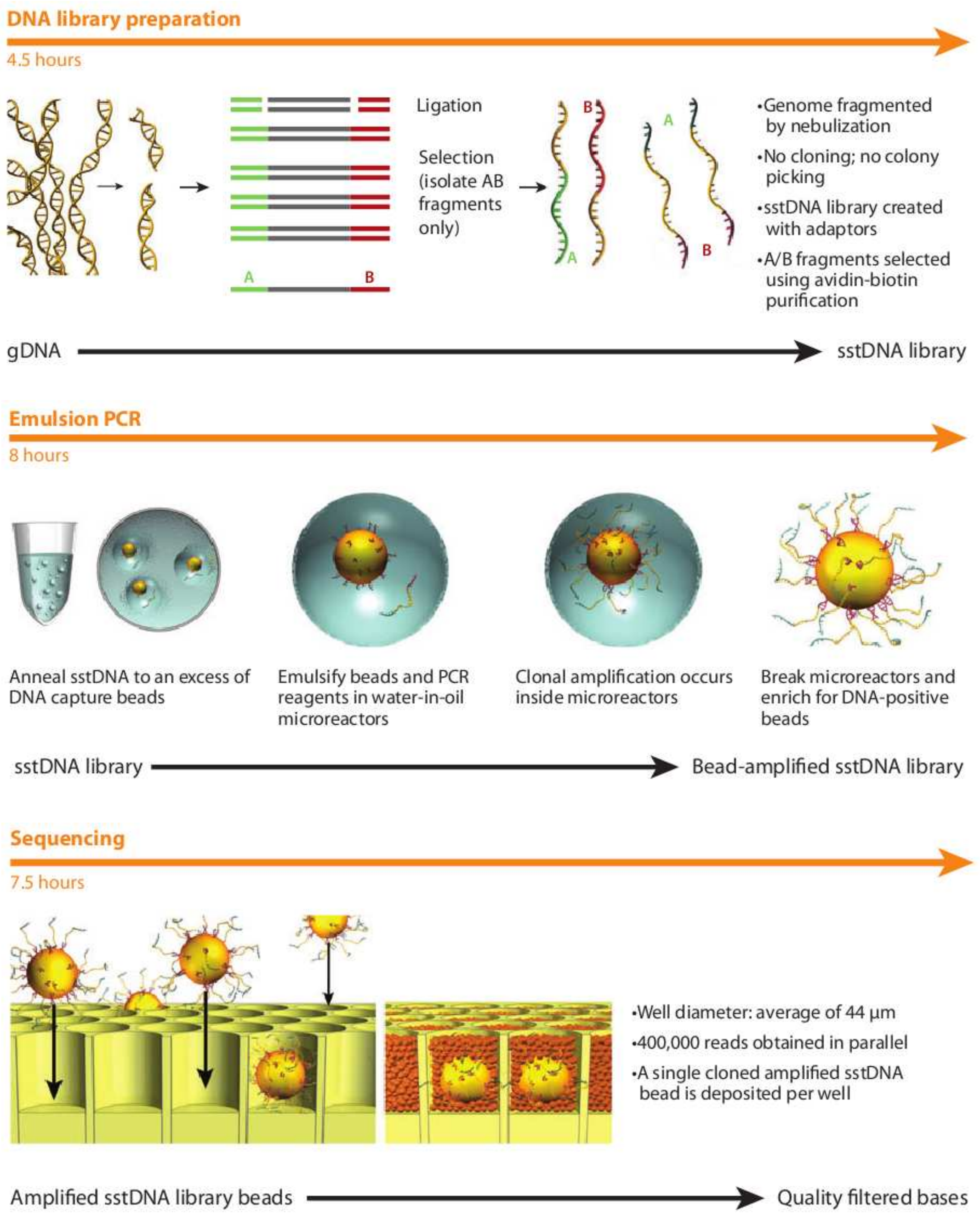
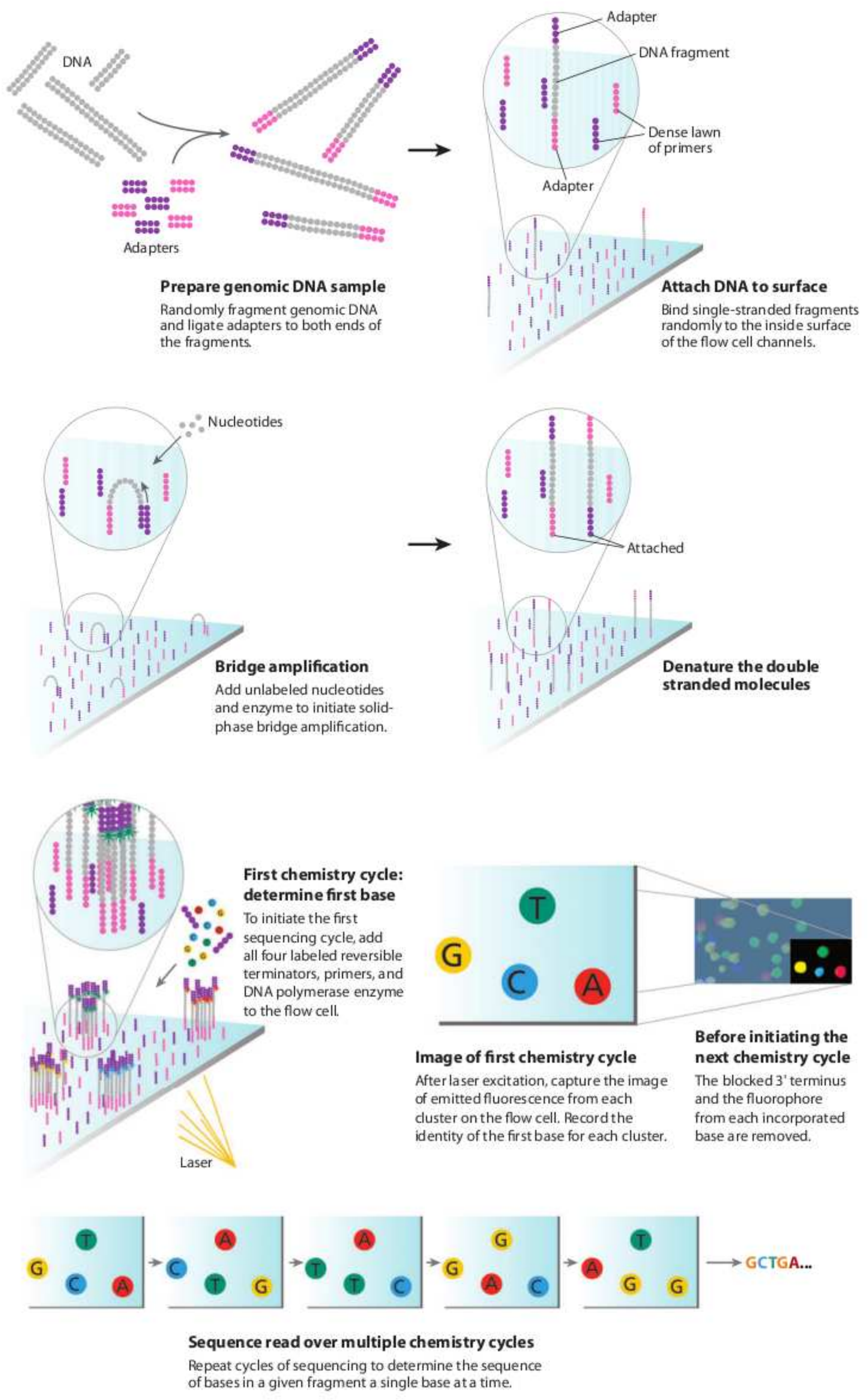

DNA library preparation
4.5 hours
Ligation
Selection (isolate AB fragments only)
A
B
•Genome fragmented by nebulization
•No cloning; no colony picking
•sstDNA library created with adaptors
•A/B fragments selected using avidin-biotin purification
gDNA
sstDNA library
Emulsion PCR
8 hours
Anneal sstDNA to an excess of DNA capture beads
Emulsify beads and PCR reagents in water-in-oil microreactors
Clonal amplification occurs inside microreactors
Break microreactors and enrich for DNA-positive beads
sstDNA library
Bead-amplified sstDNA library
Sequencing
7.5 hours
•Well diameter: average of 44 µm
•400,000 reads obtained in parallel
•A single cloned amplified sstDNA bead is deposited per well
Amplified sstDNA library beads
Quality filtered bases

Figure 3.3: **The Roche/454 FLX Pyrosequencer.** Reproduced from (MARDIS, 2008).



DNA
Adapters
Adapter
DNA fragment
Dense lawn of primers
Adapter
Prepare genomic DNA sample
Randomly fragment genomic DNA and ligate adapters to both ends of the fragments.
Attach DNA to surface
Bind single-stranded fragments randomly to the inside surface of the flow cell channels.
Nucleotides
Attached
Bridge amplification
Add unlabeled nucleotides and enzyme to initiate solid-phase bridge amplification.
Denature the double stranded molecules
First chemistry cycle: determine first base
To initiate the first sequencing cycle, add all four labeled reversible terminators, primers, and DNA polymerase enzyme to the flow cell.
Laser
T
G
C
A
Image of first chemistry cycle
After laser excitation, capture the image of emitted fluorescence from each cluster on the flow cell. Record the identity of the first base for each cluster.
Before initiating the next chemistry cycle
The blocked 3' terminus and the fluorophore from each incorporated base are removed.
GCTGA...
Sequence read over multiple chemistry cycles
Repeat cycles of sequencing to determine the sequence of bases in a given fragment a single base at a time.

Figure 3.4: **The Illumina/Solexa Genome analyser.** Reproduced from (MARDIS, 2008).

3.4 Discussion

Gene expression measuring technologies are in continuous progress. Nonetheless, some platforms have been consolidated as reference approaches for this specific purpose. Between those, three were highlighted in this chapter: microarray, real time PCR and high-throughput sequencing. In the latter, two methods stand out due to the wide application in recent years, the Roche/454 FLX Pyrosequencer and the Illumina/Solexa Genome analyser. Despite the outstanding advances in experiments technology, data availability is still an obstacle to be overcome in the process of inference of gene regulatory networks. The volume of generated data is huge, but it usually contains a relatively small number of experimental conditions and methods when compared with the large number of observed genes. This is known as the *dimensionality problem* and is one of the main issues faced by researchers nowadays.

Another important problem concerning biological data is that the gene expression data provided by the discussed technologies quantify the concentration of mRNA and ignores information about possible interventions and environmental changes after the transcription phase. Chemical and structural modifications of mRNA, as well as the blocking of translation by miRNAs, are common events. Therefore, a network reconstruction based solely on this type of data may result in a not fully veridical model. A natural course to solve this limitation and enrich the reconstructed network is the feeding of other types of data, such as protein concentration measurements, to the reverse engineering algorithm. However, those are still not available in sufficient amount to be embedded in the procses (HACHE; LEHRACH; HERWIG, 2009).

4 CLUSTERING ALGORITHMS

When one wants to uncover the gene regulatory networks underlying a certain organism, the goal is to somehow discover which genes are highly correlated in terms of their expression. In other words, one aims to find out what genes respond together to a given signal or perturbation, e.g. a virus infection, by being induced to synthesized the products they encode, or, putting it more simply, to express themselves. Computationally, an intuitive and commonly applied approach to study correlation between genes is clustering algorithms.

Clustering is the process of organizing data into groups of similar objects based only on information found in the data that describes the objects and their relationships (BERKHIN, 2002; TAN; STEINBACH; KUMAR, 2005). Each group is referred to as a cluster and is composed of objects that are similar between themselves and dissimilar to objects of other groups. From a machine learning perspective, clustering is an unsupervised learning technique that creates a data concept by partitioning a domain based on observed similarities. According to BERKHIN (2002), condensing data in clusters causes the loss of certain fine details, but achieves simplification, which is a benefit when dealing with a large mass of information.

A wide range of fields have profited from clustering analysis, as outlined by TAN; STEINBACH; KUMAR (2005). In biology, for instance, clustering was applied to automatically find a taxonomy classification of all living things. In business, it is useful for creating groups of potential and current customers for additional analysis and marketing activities. In the process of information retrieval, clustering algorithms can be used to group search results into a small number of clusters, each of which captures a particular aspect of the query. Finally, in medicine, cluster analysis has helped detecting patterns in the spatial or temporal distribution of a disease.

In the gene expression context, the purpose of using clustering algorithms is to group either genes or samples that share common characteristics. In the gene-based clustering, the specific goal is to identify groups of genes that have similar expression patterns over a set of experiments (EISEN et al., 1998). On the other hand, the sample-based clustering regards the partition of samples into homogeneous groups based on phenotypic features (JIANG; TANG; ZHANG, 2004). This process is usually based on a small subset of genes, called *informative genes*, whose expression levels are strongly correlated with the class distinction. The use of *informative genes* aims to reduce dimensionality.

According to FRIEDMAN et al. (2000), such analysis by clustering algorithms has proven to be useful in discovering genes that are co-regulated, since co-expressed genes have a high probability of being functionally related. In what follows, a survey of some widely known clustering techniques already applied for data mining gene expression profiles is provided.

4.1 Hierarchical Clustering

Hierarchical clustering consists of a technique for building a cluster hierarchy, usually represented as a *dendogram*. Such approach allows exploring data on different levels of granularity: clusters are obtained by pruning the tree at some level (BERKHIN, 2002). The number of clusters is therefore controlled by the level of the hierarchy of tree in which splitting is performed. For two dimensional data, clusters may be also represented as nested clusters. Both graphical representations are shown in Figure 4.1 for an hypothetical example of four data points.

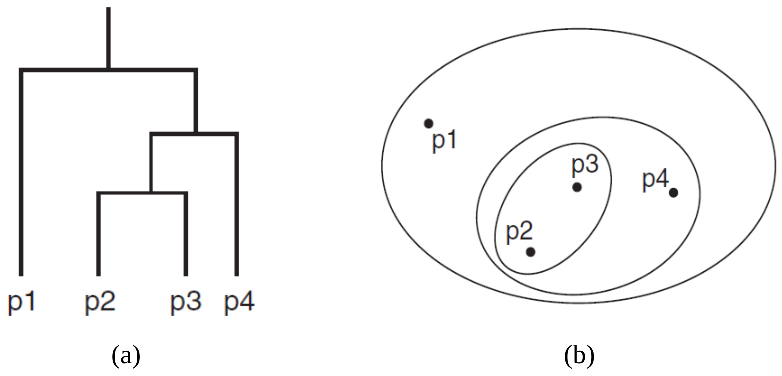


Figure 4.1: **Graphical representation of hierarchical clustering.** Example of four hypothetical data points represented as a (a) dendogram and as (b) nested clusters.

This class of clustering algorithms include two distinct categories: agglomerative and divisive. The agglomerative category applies a bottom-up strategy: given a set of N data points to be clustered, each data point is initialized as a singleton cluster and then clusters are recursively merged according to common features analysis. In contrast, divisive clustering has a top-down approach. The algorithm starts from one global cluster, comprising all data points, and sequentially splits data into smaller cluster until a stopping criteria is achieved, which is usually a specific number of clusters. The methodology difference between this two categories is depicted in Figure 4.2. As the agglomerative approach is by far the most common (TAN; STEINBACH; KUMAR, 2005), it will be the focus of this Section.

4.1.1 Agglomerative Hierarchical Clustering

Algorithm 4.1 describes the basic mechanism of the agglomerative strategy in hierarchical clustering. The key operation is the computation of the distance between two clusters, saved as means of a similarity matrix, in which all the decision about merging

Algorithm 4.1: Basic algorithm for agglomerative hierarchical clustering.

```
Assign each data point to a cluster
Compute the similarity matrix
repeat
    Merge the two closest clusters
    Update the similarity matrix to reflect the new distance between the new cluster and
    the original ones
until Only one cluster remains
```

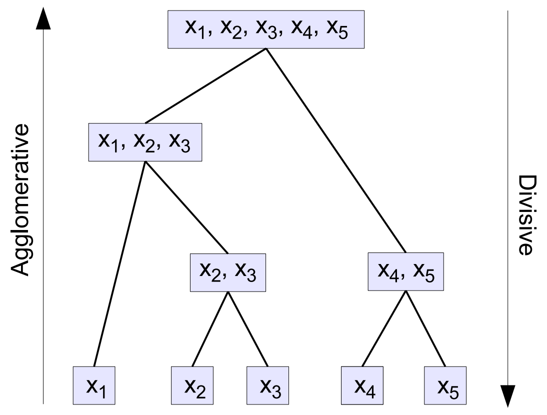


Figure 4.2: **Difference between agglomerative and divisive hierarchical clustering algorithms.** The agglomerative category has a bottom-up approach, while the divisive hierarchical clustering performs a top-down strategy.

will be based. The concept of distance or similarity is very particular of the problem being tackled and depends directly on the features embedded on data. For applications such as clustering cities or objects based on their colors, one can easily imagine the use of coordinates or RGB value, respectively, to define which instances are more similar to each other. However, the concept of distance is not that clear for all types of applications and may represent by itself a challenging step in the clustering analysis.

The step of joining clusters may be performed in different ways according to the perspective by which clusters are seen or represented. Also, it significantly affects hierarchical algorithms, since it reflects the particular concept of closeness and connectivity (BERKHIN, 2002). For a graph-based approach, there are three major inter-cluster linkage metrics: single link, average link, and complete link. Single link metric defines cluster proximity as the proximity between the closest two points that are in different clusters, or using graph terms, the shortest edge between two nodes in different subsets of nodes (TAN; STEINBACH; KUMAR, 2005). In contrast, complete link metric calculates the largest pairwise distance between points of distinct clusters. Finally, the average link defines cluster proximity to be the average pairwise proximities, or average length of edges, of all pairs of points from different clusters. Figure 4.3 shows a graphical representation of the mentioned link methods.

Another cluster closeness definition, based on geometric methods, is commonly applied when a prototype-based view is used. In this case, each cluster is represented by a centroid, and therefore, the natural choice of most similar clusters are the ones with nearest centroids. An alternative technique, mentioned by TAN; STEINBACH; KUMAR (2005), is the Ward's method: it also assumes that a cluster is represented by its centroid, but it measures the proximity between two clusters in terms of the increase in the sum of

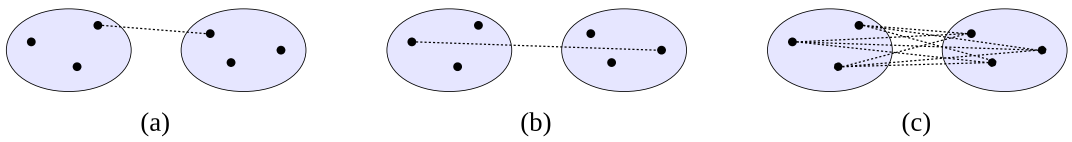


Figure 4.3: **Inter-cluster linkage metrics for hierarchical clustering.** Graphical examples of the different graph-based methods for merging clusters: (a) single link, (b) complete link and (c) average link.

the squared error that results from merging the two clusters.

Regarding complexity aspects, the agglomerative hierarchical clustering algorithm as described in Algorithm 4.1 has space complexity $O(m^2)$ and time complexity $O(m^2 logm)$, according to TAN; STEINBACH; KUMAR (2005).

4.1.2 Advantages and Disadvantages

The main advantage of hierarchical clustering algorithms is the flexibility regarding the level of granularity: it specifies clusterings at all granularities, simultaneously. This feature brings with it the benefit of freeing the user from the task of setting up the appropriate number of clusters for the data in question. This is a very important benefit when dealing with data for which there is not much information about and, thus, not enough support to define the number of clusters that best represents it. Also, hierarchical clustering allows the application of different concepts of similarity between data points, presenting, therefore a wide applicability.

Concerning the disadvantages of the method, hierarchical clustering has two important shortcomings. First, the decision making is very deterministic and based on a greedy strategy: clusters are merged or split towards the local optimal choice for that specific stage, which may not necessarily lead to the global optimum. Second, individual features of data points become less relevant as the clustering process progresses.

4.1.3 Application to Gene Expression Data

In (EISEN et al., 1998), the application of an agglomerative hierarchical clustering-based algorithm and visualization package for clustering gene expression data is described. The implemented software was used to analyse gene expression data collected in microarray experiments for two distinct cases: a single time course gene expression data of a canonical model of growth response in human cells and an aggregation of gene expression data of yeast *Saccharomyces cerevisiae*.

Cluster merging was performed based on the pairwise average-link strategy, computing similarities between data points by a form of correlation coefficient. Let G_i equal the (log-transformed) primary data for gene G in condition i. Also, consider $G_{\textit{offset}}$ to be the estimated mean of the observations. For any two genes X and Y observed over a series of N conditions, the similarity score is defined as follows:

$$S(X,Y) = \frac{1}{N} \sum_{i=1\ldots N} \left(\frac{X_i - X_{\textit{offset}}}{\phi_X} \right) \left(\frac{Y_i - Y_{\textit{offset}}}{\phi_Y} \right) \tag{4.1}$$

where

$$\phi_G = \sqrt{\sum_{i=1\ldots N} \frac{(G_i - G_{\textit{offset}})^2}{N}}. \tag{4.2}$$

According to the authors, when $G_{\textit{offset}}$ is set to the mean of observations on G, then ϕ_G becomes the standard deviation of G, and $S(X,Y)$ is exactly equal to the Pearson correlation coefficient of the observations of X and Y. In (EISEN et al., 1998), $G_{\textit{offset}}$ was set to 0, corresponding to a fluorescence ratio of 1.0.

After the application of the algorithm, the final results are represented in two different ways. The first refers to the dendogram, characteristic of hierarchical clustering: genes are assembled into a tree, where items are joined by very short branches if they are very similar to each other, and by increasingly longer branches as their similarity decreases. The second is a table representation using naturalistic color scale rather than numbers to designate expression levels. This alternative encoding preserves all the quantitative information, but allows the human brain to analyse and understand the results more easily. The twofold representation allows the observation of complex data sets in a natural way: first a scan and survey is made in the large-scale features context and then the analysis is focused on the interesting details.

EISEN et al. (1998) have observed in their work the presence of large contiguous patches of color representing groups of genes that share similar expression patterns over multiple conditions. This is more expressive for larger data sets, like the gene expression profile from yeast *Saccharomyces cerevisiae*, in which authors identified a strong tendency for these genes to share common roles in cellular processes. In the human data set, relationships among the functions of genes in clusters are obscured somewhat by the less complete functional annotation of human gene sequences. Nonetheless, when the clusters composition is examined, they are often found to contain genes known to share a common role in the cell. Through their experiments and analysis, authors also concluded that the noise inherent to single observations does not contribute significantly when genes are compared across even a relatively small number of non-identical conditions. Therefore, when designing experiments, authors judge more valuable to sample a wide variety of conditions than to make repeated observations on identical conditions.

In (PEROU et al., 2000) hierarchical clustering has been applied to investigate gene expression patterns in human breast tumours. Authors highlight the idea that phenotypic diversity might be accompanied by a corresponding variation in gene expression patterns, which in theory can be captured clustering data generated by microarray experiments. Samples from 42 different individuals, comprising 8.102 human genes, were collected using complementary DNA microarray. From these, 20 have been sampled twice: before and after a 16-week chemotherapy treatment. PEROU et al. (2000) focused their work in the clustering of a subset of 1.753 genes, whose transcripts varied in abundance by at least fourfold from their median abundance in this sample set. Additionally, a sample-based clustering is performed by grouping samples on the basis of similarity in their patterns of expression using hierarchical clustering as well.

The results exposed in (PEROU et al., 2000) shed some light on similarities and differences among the tumours and, most important, carry relevant biological information for the study of regulatory interactions among genes. For instance, authors identified sets of co-expressed genes for which variation in mRNA levels could be related to specific features of physiological variation, and, therefore, could provide valuable views of activities of specific regulatory systems. PEROU et al. (2000) infer that these portraits are a faithful representation of the tumour itself, and not only of the particular tumour sample, as the distinctive expression pattern of a tumour was recognized in distinct independent samples.

4.2 K-means

The K-means algorithm is a typical partition-based clustering method and by far the most popular clustering tool used in scientific and industrial applications (BERKHIN, 2002). The basic algorithm of the method is described in Algorithm 4.2. The algorithm begins with k initial centroids, in which k is a user-specified parameter and the number of desired clusters. This initialization can be performed either at random or based on some heuristic. The goal is to partition a data set into k disjoint subjects, assigning each data point to a centroid by optimizing an objective function, which is given by the sum of

Algorithm 4.2: Basic algorithm for K-means.

```
Select $k$ points as initial centroids
repeat
    Form $k$ clusters by assigning each point to its closest centroid
    Recompute the centroid of each cluster
until Centroids do not change
```

discrepancies between a point and its centroid expressed through and appropriate distance, as shown in Equation 4.3 (BERKHIN, 2002). When dealing with numerical attributes in an Euclidean space, the proximity measure is often quantified by means of Euclidean distance, although others like Manhattan distance may also be applied.

$$E = \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - c_j||^2 \tag{4.3}$$

The data points assigned to a centroid are form a cluster. The centroid of each cluster is then updated based on the points delegated to it. These steps are continuously repeated until points do not change between clusters or, equivalently, until centroids remain the same (TAN; STEINBACH; KUMAR, 2005).

An example of the application of K-means to the partition of a sample data into three clusters is depicted in Figure 4.4 (TAN; STEINBACH; KUMAR, 2005). Each subfigure shows the centroids at the start of the iteration, indicated by the "$+$" symbol, and the assignment of the points to those centroids. All points belonging to the same cluster have the same marker shape.

The basic K-means algorithm has time complexity $O(Ikmn)$, where m is the number of points, n the number of attributes and I the number of iterations necessary for partitioning data. As I is often small, the algorithm becomes linear in m, the number of points, and is efficient when k is significantly less than m. The space complexity is very modest, $O((m + k)n)$, since only data points and centroids need to be stored.

4.2.1 Advantages and Disadvantages

The K-means algorithm is relatively simple and fast, eespecially when the number of clusters is substantially smaller than the number of data points. Also, it is based on the firm foundation of analysis of variances. However, some issues may be faced. First, and most important, the number of clusters k is usually not priorly known and, therefore, is

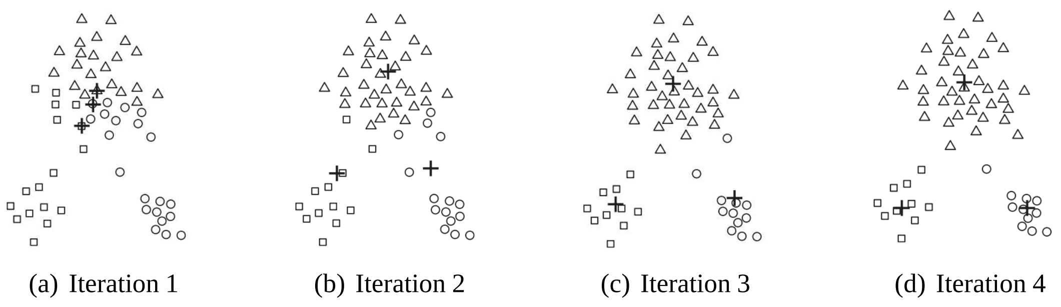

(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 4.4: **Example of the application of K-means algorithm.** Partitioning a sample data into three clusters using K-means algorithm.

by itself a parameter to be optimized. Second, the method is very sensitive to centroids initialization: different initial partitions can result in different final clusters. Third, the presence of noise may cause the algorithm to run poorly: K-means forces each data point into to a cluster and thus it is not robust in handling outliers. Forth, and finally, K-means algorithm does not work well with non-globular clusters.

The shortcomings regarding the decision about the optimal number of clusters and the initialization of centroids may be partially tackled by performing multiple runs of the algorithm and comparing results. However, for a large data set, with thousands of data points, this extensive parameter fine-tuning process may not be practical. Another possible approach is to combine hierarchical clustering with K-means: hierarchical clustering is applied to find the optimal k value and the centroids of the returned clusters are used as the initial configuration for K-means algorithm. Again, this is only practical for small samples and if k is relatively small compared to the sample size.

4.2.2 Application to Gene Expression Data

In (BAGIROV; MARDANEH, 2006), authors work in an optimization for the global K-means algorithm proposed by (LIKAS; VLASSISB; VERBEEKB, 2003), focusing in the application to gene expression data. While K-means has the problem of converging only to a local minima, which can significantly be different from global solutions as the number os clusters increase, global K-means constitutes a deterministic global optimization method that does not depend on any initial parameter values and employs the k-means algorithm as a local search procedure (LIKAS; VLASSISB; VERBEEKB, 2003). However, clustering algorithms based on global optimization techniques are usually not applicable to relatively large data sets (BAGIROV; MARDANEH, 2006).

BAGIROV; MARDANEH (2006) proposed a new version of the global K-means algorithm, computing a start point for the k-th clusters by minimizing a so-called auxiliary cluster function. This function has the benefit of reducing the number of parameters to be optimized and maintaining all variables in a continuous domain, allowing its application to large gene expression data sets. The clustering process is then performed in an incremental way: the number of clusters is successively increased. At the first iteration, the centroid of the set A and its corresponding cluster function are computed. The k-partition at the k-th iterations is performed applying K-means to the $k-1$ cluster centers from the previous iteration, iteratively, until the minimization of the cluster function reaches a given tolerance value ϵ. Authors have validated their method with six different gene expression data sets, comparing it afterwards with the global K-means algorithm, among others. They conclude that their algorithm outperforms global K-means algorithm as the number of clusters increases, but with the cost of requiring more computational efforts than the latter.

4.3 Self-Organizing Maps

Self-Organizing Maps (SOM) are neural network based clustering algorithms introduced by KOHONEN (1982) as an unsupervised and competitive learning method. The network consists of a set of neurons placed in the nodes of a lattice, competing between themselves to respond to an input pattern. The number of competing neurons, which typically varies from a few dozen up to several thousand, determines the accuracy and generalization capability of the SOM. The input patterns are introduced in the input layer, whose nodes are fully connected to those in the competitive layer. According to (HAYKIN,

Algorithm 4.3: Basic algorithm for self-organizing maps.

```
Define neighborhood function $h$
Initialize neurons with random weight values $W$
**repeat**
    Present an input vector $x_r$ at random
    Calculate winning node: $j^* = argmin_{\forall_j} ||x_r - w_j||$
    Update weight vectors of winning node and its neighbors such that
    $w_j(t+1) = w_j(t) + \eta(t) \times h_{j^*j}(t) \times ||x_r - w_j||$
    Adjust $\eta(t)$ and $\sigma(t)$ factors
**until** $W$ do not significantly changes
```

1998), the SOM algorithm forms a "*topographic map of the input patterns in which the spatial locations (i.e. coordinates) of the neurons in the lattice are indicative of intrinsic statistical features contained in the input patterns*".

The basic algorithm for evolving a SOM comprises four steps: initialization, competition and synaptic adaptation. In initialization, the synaptic weights of neurons are randomly initialized. Then, for each input pattern, the network's neurons compute their respective values of a discriminant function. The neuron with the largest value is declared the winner. This is the competition process. Finally, the winning neuron is moved towards the input pattern by updating its synaptic weight according to a pre-defined rule.

Another possible and widely used SOM algorithm, which is a variation of the basic algorithm described above, includes a cooperation step just before the synaptic weights update. In this version, the winning neuron defines a spatial location of a topological neighborhood of excited neurons and all the neurons in this neighborhood have their synaptic weights updated to a given rate. This mechanism is portrayed in Algorithm 4.3, in which t denotes time, $\eta(t)$ is the learning rate and $h_{j^*j}(t)$ stands for the neighborhood kernel, centered in the winning node (j^*), as shown in Equation 4.4. Note that both $\eta(t)$ and $\sigma(t)$ decrease monotonically with time (VESANTO; ALHONIEMI, 2000). This aims to allow further topological variability at a first moment, and limit it after some cycles of execution.

$$h_{j^*j}(t) = exp\left(-\frac{||w_{j^*} - w_j||}{2\sigma^2(t)}\right) \tag{4.4}$$

The computational complexity of SOM algorithm scales linearly with the number of samples and grows as $O(n^2)$ with the number n of neurons in the competitive layer (VESANTO; ALHONIEMI, 2000). This method is usually efficient in dealing with large data sets, but its performance may decrease when the number of samples is exorbitantly large. Improvements have already been suggested in order to make possible its application to extremely large data sets. In (KOHONEN et al., 2000), for instance, authors have successfully applied SOM to the self organization of a massive collection of documents: 6.840.568 patent abstracts were organized onto a 1.002.240-neurons map.

For clustering purpose, clusters are extracted from a SOM by identifying a group of neurons nearly mapped. When the training is complete, SOM forms a semantic map where similar samples are mapped next to each other and dissimilar samples are mapped apart. This proximity is usually represented as an unified distance matrix (U-Matrix), which stores the Euclidean distances between the weight vectors of neighboring neurons. When such distances are depicted in a gray scale image, light colors denote the closely distributed neurons, while darker colors indicate the most dissimilar neurons. Thus, groups

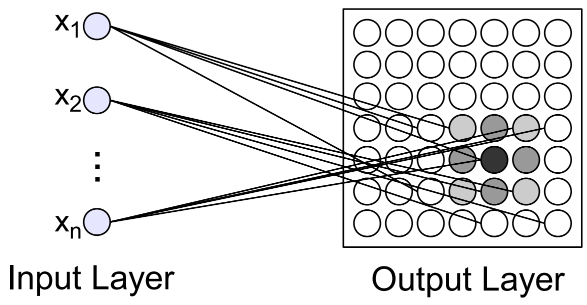


Figure 4.5: **The self-organizing map structure.** A multidimensional data is mapped into a two-dimensional space through a competitive learning process and the adaptation of winning neuron and its respective neighbors weights. The output layer is also known as competitive layer.

of light colors can be considered as a clusters, and the dark parts as the boundary regions.

4.3.1 Advantages and Disadvantages

The main benefit of SOM as a clustering algorithm is the reduction of computational cost, since it clusters a set of prototypes, represented by the neurons, rather than directly the data samples. Also, it is less sensitive to noise as prototypes are local averages of the data. Regarding its output, SOM provides an intuitively appealing map of a high-dimensional data set in two or three dimensional space, mapping similar samples near to each other (JIANG; TANG; ZHANG, 2004). However, SOM has an important drawback of requiring users to input the number of clusters and the grid structure of the neuron map. As these parameters are preserved through the training process, improperly-specified parameters will prevent the recovering of the natural cluster structure.

4.3.2 Application to Gene Expression Data

TÖRÖNEN et al. (1999) have applied a modified SOM algorithm, known as tree-structured SOM, to 6400 genes comprised in a published data of yeast gene expression during a diauxic shift, the shift from anaerobic fermentation of glucose to aerobic respiration of ethanol. The tree-structured SOM consists of several SOM arranged hierarchically in a pyramid-like fashion in several layers. The number of neurons at a higher level is four times the number of the previous level. The training is repeated layer by layer using knowledge about the winning neuron of previous level l-1 to compute the current winning neuron. As search is restricted to a small set of neurons linked to the previous winning neuron its neighbors, computational complexity is significantly reduced.

The SOM used by TÖRÖNEN et al. (1999) is a 16×16 map, containing thus 256 neurons, whose dimension was chosen based on test experiments. The number of genes in individual neurons varied between 10 and 49. After the training process, SOM is further modified using Sammon's mapping algorithm, where spatial distance correlates with the difference in average expression profile and the circle size with the number of genes within the neuron. Four clusters were selected for gene content analysis: two with an increasing pattern, one with decreasing pattern and one with no change in its expression. Authors analysis show that SOM rapidly and reliably clusters the gene expression data set into groups that show similar gene expression profiles. According to the authors, the concept of similarity comprises participation in a common pathway or regulation by a common regulatory element in a promoter region, or both. Therefore, clustering gene expression profiles using SOM approach is also efficient for discovering functionally related genes.

In MILONE et al. (2011), a pipeline for biological data integration and discovery of a priori unknown relationships between gene expressions and metabolite accumulations is discussed. According to the authors, as metabolites are the final products of gene actions, they are potentially relevant to elucidate gene functions and networks, especially when integrated with transcriptomic data. In this study, authors are particularly interested in investigating the occurrence of introgressed portions of a wild tomato (*Solanum pennelli*) species genetic material in certain chromosomes segments of tomato *Solanum lycopersicum*.

MILONE et al. (2011) explain each of the pipeline steps, from data understanding and pre-processing to visualization and biological assessment of results, and compare the performance of three clustering methods – hierarchical clustering, K-means and self-organizing maps – for the data mining step according to pre-defined measures. Authors observe that while hierarchical clustering comprises the vast majority of the patterns in the same dendogram's branch, K-means results in several nodes with very few patterns and few nodes with many patterns, and self-organizing maps present a more uniform distribution of patterns between nodes. Therefore, they assert that SOM is a more indicated method for gene expression data clustering as its balance distribution allows a more confident analysis in both neuron and neighborhood contexts, given that nearby nodes have gradual changes and form clusters with biological meaning. Also, SOM technique presented outstanding performance in terms of the quality measures defined by authors and provides better visualization capabilities when compared to K-means and hierarchical clustering.

4.4 Tools

Michael Eisen have developed a set of computational tools for clustering analysis and visualization of results of microarray experiments. The Cluster application performs a wide variety of types of cluster analysis and other processing on large microarray datasets. It currently includes hierarchical clustering (as described in EISEN et al. (1998)), self-organizing maps (SOMs), k-means clustering and principal component analysis. TreeView graphically browse results of clustering and other analyses from Cluster. It supports tree-based and image based browsing of hierarchical trees, as well as it allows multiple output formats for generation of images for publications. Both Cluster and TreeView applications are free for academic use and available online[1].

The software applied in the investigation held by (MILONE et al., 2011), named *meSOM, may also be downloaded[2] and applied in academic research. *omeSOM is a tool designed to give support to the data mining task of metabolic and transcriptional datasets derived from different databases. It implementes a neural model for biological data clustering and visualization. The software is focused on the easy identification of groups including different molecular entities, independently of the number of clusters formed. It provides a user-friendly interface and offers several visualization tools easy to understand by non-expert users (MILONE et al., 2010).

[1]`http://rana.lbl.gov/EisenSoftware.htm`
[2]`http://www.sourcesinc.sourceforge.net/omesom`

4.5 Discussion

This chapter presented some widely used methods for clustering gene expression data: hierarchical clustering, K-means and self-organizing maps. Examples of specific applications involving these techniques were given and a common agreement among authors conclusions is that clustering is not only efficient for grouping together genes with similar features and expression patterns, but also that resulting clusters are biologically consistent, e.g. genes clustered together are in fact involved in the same regulatory system or activity.

However, clustering algorithms don't completely solve the problem of discovering gene regulatory networks: even though genes are clustered according to their inherent patterns, the exact biological relationship or dependence among them is not inferred. One common approach, therefore, is to combine clustering algorithms to more sophisticated techniques, using it as a pre-processing method to reduce dimensionality of the inference problem. Following this direction, once clustering is applied and a co-expression network is generated, this information may be used to perform a fine study about genes which are highly correlated, aiming to uncover more detailed information about the regulatory interactions ruling their expression level. In this particular formulation, methods discussed in the following chapters are efficient and common alternatives for inferring the detailed structure of the regulatory network underlying genes in a specific cluster.

5 BAYESIAN NETWORKS

As discussed in Chapter 4, clustering algorithms are an useful coarse-grained approach for discovering genes that are co-regulated within an organism. However, a more ambitious goal emerging in the last years aims to reveal the full and exact network structure of regulation mechanism, rising the need for robust methods. The main requirement is the ability to cope with noise, which are inherent to experimental data. In this context, Bayesian networks have performed an important role in the analysis of gene expression data.

A Bayesian network (BN) is a graphical model that encodes probabilistic relationships among variables of interest (HECKERMAN, 1995) and it has been first proposed as a GRN reverse engineering method by FRIEDMAN et al. (2000). Its probabilistic nature is an advantage when dealing with missing and noisy data. Furthermore, it allows the learning of causal relationships between interacting entities and, therefore, are useful to gain understanding about a problem domain and to predict the consequences of intervention.

5.1 Bayesian Networks Structure

BNs are graphical models for representing the relationships between multiple interacting entities by means of joint probability distributions (HECKERMAN, 1995). The probabilities encoded by a BN are said to be Bayesian when the network is learned from prior knowledge alone, and physical when it is learnt from data. The probabilistic nature of BNs brings four distinct advantages, as highlighted by HECKERMAN (1995): 1) can handle incomplete data sets; 2) allows the learning of causal relationships, which in turn is useful for domain understanding and prediction making; 3) is a powerful and easy tool for combining prior knowledge and data; 4) avoids data overfitting.

Formally, BN models are defined by a directed acyclic graph (DAG) G, whose nodes represent the random variables $X = X_1, \ldots, X_n$ in our domain and edges denote, intuitively, the direct influence of one node on another by means of conditional dependence relations. Along with the graphical structure, the model is characterized by a family of conditional probability distributions F and their parameters q, which together specify a unique joint distribution for each variable in the set of interest. In graph G, when a directed edge exists from node A to node B, A is called the parent of B and B is said child of A.

Due to the acyclic property of BNs, the joint distribution of vertices in graph G may be decomposed in simpler conditional independence assumptions, economizing on the number of parameters. This decomposition follows the so-called Markov assumption: "*Each variable X_i is conditionally independent of its non-descendants, given its parents in G*".

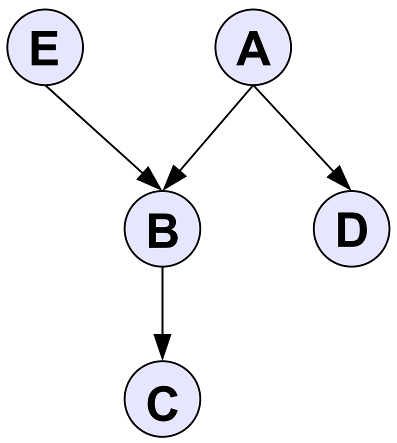


Figure 5.1: **Example of simple Bayesian network structure.** Several conditional independence statements are implied by this simple model: $Ind(A;E)$, $Ind(B;D|A;E)$, $Ind(C;A,D,E|B)$, $Ind(D;B,C,E|A)$, $Ind(E;A,D)$. The joint distribution has the product form: $P(A,B,C,D,E) = P(A)P(B|A,E)P(C|B)P(D|A)P(E)$. Reproduced from (FRIEDMAN et al., 2000).

According to FRIEDMAN et al. (2000), by applying the chain rule of probabilities and properties of conditional independences, any joint distribution that satisfies the Markov assumption can be decomposed into the product form:

$$P(X_1,\ldots,X_N) = \prod_{i=1}^{N} P(X_i|Pa^G(X_i)) \tag{5.1}$$

where $Pa^G(X_i)$ is the set of parents of X_i in graph G. The terms in the product of Equation 5.1 correspond to the conditional probability distributions F and their respective parameters q. To specify the complete joint distribution it is still necessary to determine a representation for the family F of conditional distributions. This choice will depend on the type of variable we are dealing with, being most commonly used a Gaussian distribution for continuous variables and a multinomial distribution for discrete variables.

Figure 5.1, reproduced from (FRIEDMAN et al., 2000), shows a simple example of Bayesian network structure. Let $Ind(G)$ be the set of independence statements in the form X is independent of Y given Z, i.e. $Ind(X;Y|Z)$. The following conditional independence statements derive from the relationships depicted in the graph: $Ind(A;E)$, $I(B;D|A;E)$, $Ind(C;A,D,E|B)$, $Ind(D;B,C,E|A)$, $Ind(E;A,D)$. Applying Equation 5.1 to the graph in this figure, we obtain the joint distribution in the product form: $P(A,B,C,D,E) = P(A)P(B|A,E)P(C|B)P(D|A)P(E)$.

It is possible to have more than one graph implying the exactly same set of independences $Ind(G)$. For instance, consider two graphs over variables X and Y: $X \rightarrow Y$ and $X \leftarrow Y$. They both imply the set of independences $Ind(G = \emptyset)$, and, therefore, they are said to be equivalent (FRIEDMAN et al., 2000). In a general way, two graphs, G and G', are equivalent if $Ind(G) = Ind(G')$.

Equivalent classes of network structures may be uniquely represented by a partially directed graph, in which an undirected edge $X - Y$ denotes that some members of the class contain the edge $X \rightarrow Y$, while others contain the edge $X \leftarrow Y$ instead. According to (PEARL; VERMA, 1991), two directed acyclic graphs are equivalent if and only if they have the same underlying undirected graph and the same v-structures (i.e., converging directed edges into the same node, such that $a \rightarrow b \leftarrow c$, and no edge exists between a and c).

Although Bayesian networks are based on DAGs, it is important to stress that not all directed edges in a Bayesian network can be interpreted causally. The DAG on which the Bayesian network model is based just asserts a set of independence assumptions among the domain variables (WERHLI, 2007).

5.2 Learning Bayesian Networks

The process of learning Bayesian networks aims to devise a Bayesian network model M from a given set of training data D such that M is the model that better explains the data and its embedded dependences. More precisely, we search for an equivalence class of networks that best matches D (FRIEDMAN et al., 2000). Although there are other approaches to learn a Bayesian Network besides the Bayesian learning (see, for instance, HECKERMAN (1995)), this is the most mentioned one in literature and therefore it will be the focus of the present section.

BN learning is performed in two distinct stages. First, we learn the network structure, e.g. how the entities are connected by edges. Defining $\mathbb{M}$ as the space of all possible models, the first goal is to find a model $M^* \in \mathbb{M}$ that is the most supported by the data D:

$$M^* = argmax_M P(M|D) \tag{5.2}$$

The second step consists of learning the parameters sets associated with the edges in model M^*, and whether the relationships between these entities are of activating or inhibitory nature, as well as its respective intensity. Having the best structure M^* and the data set D, we search for the best parameters set q:

$$q = argmax_q P(q|M^*, D) \tag{5.3}$$

A common approach to the problem of finding the best model given the data D is to introduce a statistically motivated scoring function that evaluates each network with respect to the training data and use it to search for the optimal network. In FRIEDMAN et al. (2000) authors evaluate the posterior probability of a model given the data:

$$\begin{aligned} S(M:D) &= logP(M|D) \\ &= logP(D|M) + logP(M) + C \end{aligned} \tag{5.4}$$

where C is a constant independent of M and

$$P(D|M) = \int P(D|q, M)P(q|M)dq \tag{5.5}$$

is the marginal likelihood, which averages the probability of the data over all possible parameter assignments to M. According to the authors, the particular choice of priors $P(M)$ and $P(q|M)$ determines the exact Bayesian score to be applied. A theoretical and mathematical review of the two most widely used Bayesian scores may be found in WERHLI (2007): the Bayesian Dirichlet likelihood equivalent scoring metric (BDe score) and the continuous Bayesian Gaussian likelihood equivalent scoring metric (BGe). While the first score is used for discrete variables, which are associated to multinomial distributions, the latter is used for continuous domain, where variables are associated to Gaussian distributions.

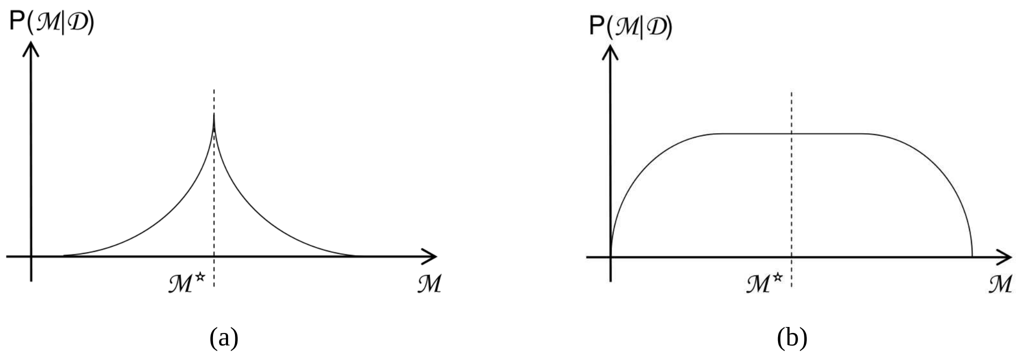


Figure 5.2: **Bayesian inference.** The vertical axis denotes the posterior probability $P(M|D)$ and the horizontal axis represents the network structures M. (a) For a large and informative data set, the best structure M^* is well defined in the posterior probability distribution. (b) The same does not occur for a small and less informative data set, whose posterior probability distribution is very diffuse. Reproduced from (WERHLI, 2007).

The integral in Equation 5.5 is analytically tractable when the data is complete and the prior $P(q|M)$ satisfies certain regularity conditions discussed in GEIGER; HECKERMAN (1994) and HECKERMAN (1995). Additionally, in this case, the posterior score has several properties, as highlighted by FRIEDMAN et al. (2000). First, the score is structure equivalent, i.e., if G and G' are equivalent graphs they are guaranteed to have the same posterior score. Second, the score is decomposable: it can be written as the sum

$$S(M : D) = \sum_i ScoreContribution(X_i, Pa^G(X_i) : D), \tag{5.6}$$

where the contribution of every variable X_i to the total network score depends only on the values of X_i and $Pa^G(X_i)$ in the training instances. Finally, the local contributions for each variable can be computed using a closed form equation.

However, the computation of $P(D|M)$ is not enough for the assessment of score $S(M : D)$, and hence for the identification of the optimal model M^*, due to the high similarity between networks in the same equivalence class. Prior knowledge $P(M)$ may be useful when deciding which model is the most proper one in the given equivalence class. Nevertheless, finding the best model M^* by direct computation of score S is still an impractical approach for two main reasons: the number of structures increases rapidly with the number of nodes and the posterior $P(M|D)$ is usually diffuse and is not adequately represented by a single network at the mode when the data set is small and slightly informative.

The problem of learning the structure M^* that maximizes the score $S(M : D)$ is known to be a NP-hard problem (CHICKERING, 1996). Several high-scoring networks may be found. Thus, heuristic search and sampling methods are often applied in this context. According to FRIEDMAN et al. (2000), the use of local search, by which one

Table 5.1: **Number of nodes vs. number of networks.** The number of networks grows super-exponentially with the number of nodes. Reproduced from WERHLI (2007).

Number of nodes	2	4	6	8	10
Number of topologies	3	543	3.7×10^6	7.8×10^{11}	4.2×10^{18}

arc is changed at each move and the gains provided by the modification are evaluated, does not necessarily find a global maximum but it does perform well in practice. As examples, authors cite beam-search, (stochastic) hill-climbing and simulated annealing. However, a much more usual solution in recent years is to resort to sampling methods in order to obtain a representative sample of high scoring network structures.

5.3 Markov Chain Monte Carlo: a Sampling Method

In this section we present a sampling method commonly combined to Bayesian networks in the problem of learning networks structure. Markov Chain Monte Carlo (MCMC) was proposed by METROPOLIS et al. (1953) and HASTINGS (1970), and applied to the context of inferring gene regulatory networks using the BN formalism by MADIGAN; YORK (1995).

Since data sets D are usually sparse, and this includes the special case of postgenomic data, the posterior probability $P(M|D)$ over structures is diffuse and not properly represented by a single optimum model M^*. Therefore, an appropriate solution consists of sampling networks from the posterior probability:

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)} = \frac{P(D|M)P(M)}{\sum_{M'} P(D|M')P(M')} \tag{5.7}$$

A direct approach to sample from $P(M|D)$ is impossible though, as the denominator in Equation 5.7 is a sum over the whole model space and is intractable (see Table5.1). METROPOLIS et al. (1953) and HASTINGS (1970) proposed, therefore, to create a Markov Chain, in the following form:

$$P_{n+1}(M_i) = \sum_k T(M_i|M_k)P_n(M_k) \tag{5.8}$$

where M_k is the current structure, M_i the new proposed model and T represents the Markov transition matrix, which is a matrix of transition probabilities. Under the condition of ergodicity[1] the distribution $P_n(M_k)$ converges to a stationary distribution $P_\infty(M_k)$, such that:

$$P_\infty(M_i) = \sum_k T(M_i|M_k)P_\infty(M_k), \tag{5.9}$$

In Equation 5.9, the posterior distribution in step $n + 1$ is equal to the distribution in step n, following the Markov assumption. As the transition matrix T completely determines the stationary distribution in Equation 5.9, it needs to be designed so that the posterior probability equals the stationary distribution of the Markov chain, i.e. $P(M|D) = P_\infty(M)$. A sufficient condition for this to be true is given by the equation of detailed balance:

$$\frac{T(M_i|M_k)}{T(M_k|M_i)} = \frac{P(M_i|D)}{P(M_k|D)} = \frac{P(D|M_i)P(M_i)}{P(D|M_k)P(M_k)} \tag{5.10}$$

The transition to a new structure, $M_k \to M_i$, is performed in two steps: first a new structure is proposed with proposal probability $Q(M_i|M_k)$. The action of proposing new

[1] An ergodic Markov chain is aperiodic and irreducible: all states are reachable from all other states and the probability that the next state is the same as the current state is non-zero.

Algorithm 5.1: Metropolis-Hastings algorithm.

```
Start with an initial structure $M_0$
for $i = 1$ to $I$ do
    Obtain a new DAG structure $M_i$ from the proposal distribution $Q(M_i|M_{i-1})$.
    Accept the new model with probability $A(M_i|M_{i-1})$ given by Eq. 5.11, otherwise
    leave the model unchanged.
end for
Allow the Markov chain to reach stationarity by discarding some initial samples,
$M_i \dots M_{\frac{I}{2}}$ for instance. This is burn-in period.
Compute the expectation values from the MCMC sample $M_{\frac{I}{2}+1} \dots M_I$:
$\langle f \rangle = \sum_M f(M)P(M|D) \approx \frac{2}{I} \sum_{i=\frac{I}{2+1}}^{I} f(M_i)$
```

Algorithm 5.2: Metropolis algorithm.

```
Start with an initial structure $M_0$
for $i = 1$ to $I$ do
    Obtain a new DAG structure $M_i$ from the proposal distribution $Q(M_i|M_{i-1})$.
    If the new model is not a DAG, reject it and go back to previous step.
    Accept the new model with probability $A(M_i|M_{i-1})$ given by Eq. 5.12, otherwise
    leave the model unchanged.
end for
Allow the Markov chain to reach stationarity by discarding some initial samples,
$M_i \dots M_{\frac{I}{2}}$ for instance. This is burn-in period.
Compute the expectation values from the MCMC sample $M_{\frac{I}{2}+1} \dots M_I$:
$\langle f \rangle = \sum_M f(M)P(M|D) \approx \frac{2}{I} \sum_{i=\frac{I}{2+1}}^{I} f(M_i)$
```

structures means to choose, at each iteration, one of the basic operations of adding, deleting or reversing an edge. Some of this actions may occasionally lead to invalid networks, e.g. cyclic networks, which must be dismissed. Finally, the proposed structure is accepted with acceptance probability $A(M_i|M_k)$, referred to as Metropolis-Hastings criteria, and defined according to the following equation:

$$\frac{A(M_i|M_k)}{A(M_k|M_i)} = min\left\{\frac{P(D|M_i)P(M_i)Q(M_k|M_i)}{P(D|M_k)P(M_k)Q(M_i|M_k)}, 1\right\} \tag{5.11}$$

Accept a new model M_i according to Equation 5.11 guarantees that the Markov chain will converge for the desired posterior distribution (WERHLI, 2007). The term $\frac{Q(M_k|M_i)}{Q(M_i|M_k)}$ is called Hastings factor and must be computed when an asymmetry exists between the networks. This asymmetry results from the different neighborhood sizes, which are defined by the set of all valid DAGs that might be reached from the current model with a single edge modification. When the neighborhoods have the same size, the Hastings factor equals to 1.

In order to avoid the computation of the Hastings factor, which is not trivial, it is possible to modify the step of proposing new structures such that all non DAGs models are rejected based in the prior knowledge that they can not been accepted as they are invalid BN structures. In this case, the acceptance probability is given by:

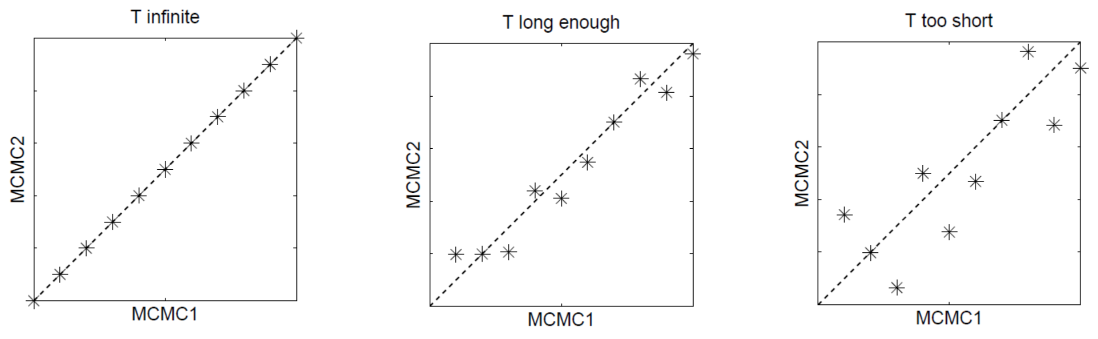
$$\frac{A(M_i|M_k)}{A(M_k|M_i)} = min\left\{\frac{P(D|M_i)P(M_i)}{P(D|M_k)P(M_k)}, 1\right\} \tag{5.12}$$

Algorithms 5.1 and 5.2 summarize both presented solutions for MCMC application: the Metropolis-Hastings algorithm and the Metropolis algorithm, respectively. In short, these algorithms start from an initial state and probabilistically transitions through a solution space, accepting transitions to better state with high probability and transitions to worse states with lower probability. According to WERHLI (2007), the possibility to avoid the finding of all neighbors in Hastings algorithm comes at a price: many more structures will be rejected, decreasing the acceptance rate and slowing down the convergence of the algorithm.

5.3.1 MCMC Convergence Assessment

According to WERHLI (2007), the MCMC approximation is exact in the limit of an infinitely long Markov chain if the condition of detailed balance (Eq. 5.10) is satisfied and if the Markov chain is ergodic. The convergence of MCMC is the guarantee that the resulting sample has been drawn from the correct and expected distribution. In WERHLI (2007), the MCMC convergence was assessed based in a simple heuristic approach, described in what follows.

The application of the MCMC algorithm results in a square matrix $n \times n$ of posterior probabilities, in which n denotes the number of nodes in the network. The technique used in WERHLI (2007) consists in running two MCMC simulations with different initial configuration, obtaining matrices P^1 and P^2, whose elements are denoted by p^1_{ij} and p^2_{ij}, respectively, and represent the probability of existing an edge between nodes X_i and X_j. After simulations are finished, both matrices are plotted against each other, e.g. p^1_{ij} vs. p^2_{ij}, generating a scatter plot. Examples of scatter plots for distinct performances of MCMC are shown in Figure 5.3. One can observe that the longer the simulation time, the better the convergence of the algorithm.



(a) (b) (c)

Figure 5.3: **MCMC convergence test.** The marginal posterior probabilities of the edges are plotted for two different simulation initializations. (a) For an infinite time, all the posterior probabilities of the edges are the same for both simulations. However, as the simulation length becomes shorter, the convergence starts to fail. Panels (b) and (c) exemplify the convergence test for a not long enough and a too short simulations, respectively. Reproduced from (WERHLI, 2007).

Algorithm 5.3: Bootstrap method applied by FRIEDMAN et al. (2000).

```
for $i = 1 \ldots m$ do
    Re-sample with replacement $N$ instances from $D$. Denote by $D_i$ new data set.
    Apply the learning procedure on $D_i$ to induce a network structure $G_i$.
end for
for each feature $f$ of interest do
    Compute $conf(f) = \frac{1}{m}\sum_{i=1}^{m} f(G_i)$, where $f(G)$ is 1 if $f$ is a feature in $G$ and 0
    otherwise.
end for
```

The convergence test proposed in (WERHLI, 2007) is necessary but not sufficient to guarantee algorithm convergence: the two simulations can reach the same meta-stable equilibrium, which might be different from the true equilibrium. Theoretically, the Markov chain will converges regardless the chosen proposal distribution Q and the initialization values. However, in practice it is known that some issue, such as extreme initialization values, may slow down the convergence of MCMC.

5.4 Application to Gene Expression Data

The pioneer work combining BN to the inference of gene regulatory networks was published by FRIEDMAN et al. (2000). Authors proposed to model the system as a joint distribution over a collection of random variables that describe system states. As the scaling of MCMC for large domains was still not clear, FRIEDMAN et al. (2000) opted for applying a nonparametric bootstrap method to estimate the confidence of features of Bayesian Networks learned from expression profiles. The bootstrap method (EFRON; TIBSHIRANI, 1993) is a very general re-sampling procedure for estimating the distributions of statistics based on independent observations. The main idea behind the bootstrap is to generate perturbed versions of the original data set and apply the learning algorithm over them. This way, all collected networks are fairly reasonable models of the data and reflect the effect of small perturbations to the data on the learning process.

The bootstrap algorithm is applied by FRIEDMAN et al. (2000) as shown in Algorithm 5.3. Authors limit the search space by focusing the attention of the search procedure on relevant regions of the search space. First, a relatively small number of candidate parents is identified for each gene based on simple local statistics (such as correlation). Later, the search is restricted to networks in which only the candidate parents of a variable can be its parents, resulting in a much smaller search space and, consequentially, in a faster search for a good structure. The proposed algorithm was applied to a data set with 800 genes and 76 gene expression measurements of the mRNA levels of 6177 *S. cerevisiae* open reading frames, in which authors managed to extract many biologically plausible conclusions through this analysis.

PE'ER et al. (2001) extended the framework proposed in (FRIEDMAN et al., 2000) to better handle perturbations and to identify significant subnetworks of interacting genes, which are shown to be related to known biological pathways. The bootstrap analysis is applied in the extraction of potential models, just like in PE'ER et al. (2001). In the sequence, authors extract from these high-scoring networks new statistically confident features (mediator, activator and inhibitor), which are further used in the identification of subnetworks of strong statistical significance along with features previously defined by FRIEDMAN et al. (2000). The idea of searching for subnetworks comes from the inherent limitation to BN's learning procedures of examining relations between two or three genes at a time. This way, authors not only broaden their viewpoint but also gain confidence about features that are not significant when isolated.

The results achieved by PE'ER et al. (2001) show that meaningful biological information can be extracted even from pairwise relations. Many uncovered genetic links were already identified by previous works. Yet, new biological insight has also been provided with strong evidence from literature. However, the full power of the proposed approach comes from the exploration of subnetworks: a comparison with a clustering approach shows that the reconstruction by means of BN provides a much richer context for regulatory and functional analysis.

In (IMOTO et al., 2003), authors explored a question already raised by PE'ER et al. (2001): how to include prior knowledge in the inference process in order to improve quality of analysis and the number of novel interactions discovered? According to IMOTO et al. (2003), the main issue faced in the reverse engineering of GRN is the limited amount of independent experimental conditions and the inherent noise in measurements, suggesting that the inclusion of prior knowledge might result in a higher reconstruction accuracy.

Following this direction, authors proposed a learning scheme using a Bayesian framework, in which a new networks' evaluation criteria is derived such that networks are selected based on microarray data and biological knowledge. In this approach, gene expression data is systematically integrated with biological knowledge from other types of postgenomic data and from literature, with automatic tuning of the balance between the both sources of data, aiming a more accurate reconstruction. This integration is performed via a prior distribution over network structures, which takes the form of a Gibbs distribution. The prior knowledge is encoded as an energy function and an inverse temperature hyperparameter determines the weight that is assigned to it. The hyperparameters are inferred together with the network structure by maximizing the joint posterior distribution with a heuristic greedy optimization algorithm. As prior knowledge, the authors extracted protein-DNA interactions from the Yeast Proteome Database. However, this framework has also been successfully applied to a wide variety of distinct sources of biological prior knowledge: transcription factor binding motifs in promoter sequences (TAMADA et al., 2003), protein-protein interactions (NARIAI et al., 2004), evolutionary information (TAMADA et al., 2005), and pathways from the KEGG database (IMOTO et al., 2006).

The integration of expression data with multiple sources of prior knowledge was investigated in (WERHLI; HUSMEIER, 2007). Each source was expressed in terms of an energy function and a prior distribution was later obtained in the form of a Gibbs distribution. The hyperparameters associated with the different sources of prior knowledge, which measure the influence of the respective prior relative to the data, are sampled from the posterior distribution with MCMC. Experiments with two independent sources of transcription factor binding locations from immunoprecipitation experiments with microarray gene expression data from the yeast cell cycle, and with the integration of KEGG pathways with cytometry experiments for determining protein interactions related to the Raf signaling pathway, have stressed the efficiency of this method to generate more accurate models.

The works published by IMOTO et al. (2006) and WERHLI; HUSMEIER (2007), as well as other papers in the GRN context, have the common feature of assuming that a molecular biological system may be successfully represented by a single regulatory net-

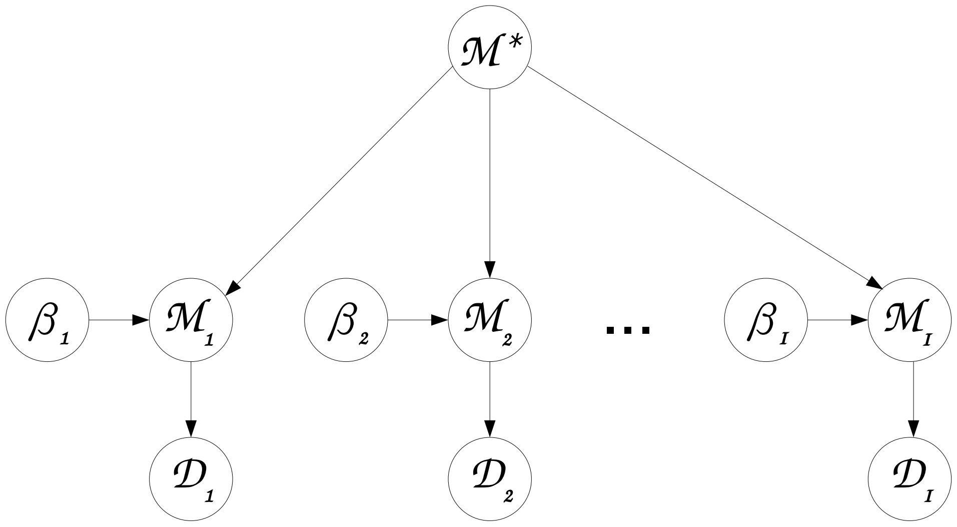


Figure 5.4: **Bayesian Hierarchical Model.** The nodes represent: data sets $(D_1, \ldots, D_n)$ obtained in different experimental conditions; hyperparameters $(\beta_1, \ldots, \beta_n)$ and network structures $(M_1, \ldots, M_n)$. The hypernetwork M_G acts as a coupling among the networks $M_1, \ldots, M_n$, inciting them to be similar.

work, when what is in fact inferred in the reconstruction process is its active parts, which may vary according to available experimental conditions. In (WERHLI; HUSMEIER, 2008), a method to reconstruct the regulatory structure of a network considering that its active parts can differ under different experimental conditions is proposed. The coupling scheme, depicted in Figure 5.4, is a compromise between learning networks from different subsets separately and learning networks from a monolithic fusion of individual data sets. While in the first approach no information is shared between the data sets, the latter does not provide a mechanism for discovering differences between the networks associated with the different experimental conditions.

The Bayesian hierarchical model proposed by WERHLI; HUSMEIER (2008) have achieved very good results in terms of reconstruction accuracy. However, the probabilistic model, when sampled with MCMC algorithm, have not always properly converged. The difficulty in crossing the valleys increases the chances of the algorithm to be trapped in a local optima, causing some high posterior probability samples to go unexplored in the analysis. MENDOZA; WERHLI (2010) proposed to tackle this problem by applying a parallel sampling scheme, known as Metropolis Coupled Markov Chain Monte Carlo $(MC)^3$. $(MC)^3$ involves the parallel execution of multiple Markov chains, some of which are heated by raising its posterior probability by a factor γ, and a state swap proposal between chains in predetermined intervals. Heated chains tend to have a higher acceptance probability because they perceive the landscape flatter than unheated chains, thus, they can more promptly cross valleys. An unheated chain, also known as cold chain, is able to jump a deep valley in a single step when swapping state with a heated chain. The acceptance probability for a cold chain is given by Equation 5.12, while a heated chain accepts new states with probability defined as:

$$A(M_i|M_k) = min\left\{\left(\frac{P(D|M_i)P(M_i)}{P(D|M_k)P(M_k)}\right)^{\gamma}, 1\right\} \tag{5.13}$$

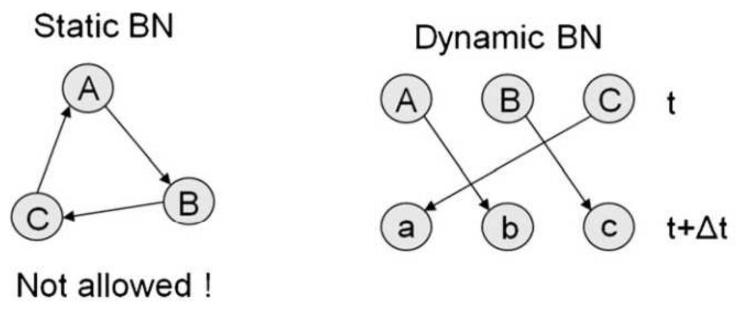
The results obtained in (MENDOZA; WERHLI, 2010) were satisfactory and a convergence improvement was observed, which occurred more frequently and in less simulation steps. However, authors stress that the tuning of the parameters involved in $(MC)^3$ algorithm is by itself a complex phase, as they are data-dependent and have direct effect over convergence properties.

5.5 Advantages and Disadvantages

Comparing to the clustering approach, introduced in Chapter 4, Bayesian networks have the clear advantage of providing a mechanism to model the problem domain as a whole by constructing a joint probability distribution over different combinations of the domain variables. Unlike clustering, BNs results in a graphical model, which not only group together genes involved in the same biological pathway, but also explicitly denotes the dependence relations and the conditional independences among genes and their expression levels.

Its probabilistic nature makes it robust to deal with uncertainty and, consequentially, with noisy data. Furthermore, as they are models of the problem domain probability distribution, they can be easily used for in silico predictions, computing the predictive distribution on the outcomes of possible actions. Following this direction, once modeled, BNs may also be applied in decision making process, assisting in the choice of actions that maximize the expected utility or score. Finally, a relevant advantage of Bayesian modeling is the possibility of aggregating expert domain knowledge within the training data whenever it is available. This prior knowledge is extremely helpful in accelerating and improving the learning process.

Nevertheless, BNs suffer from an important shortcoming: its underlying graphical model, the DAG, can not contain cycles, limiting its application to steady-state data. As feedback loops are known to be present in real biological networks, this is a significant drawback. Moreover, when applying standard MCMC methods it is necessary to check the acyclicity of proposed structures; this checking of acyclicity is one of the bottlenecks of MCMC simulations (WERHLI, 2007). Dynamic Bayesian networks (DBNs) overcome these limitations. DBNs are an extension of BNs able to infer interactions from time-series data (BANSAL et al., 2007). The nodes are split in input and output nodes, in order to unfold the basic DAG representation and avoid the formation of cycles. An example, reproduced from (HECKER et al., 2009), is shown in Figure 5.5. For further details about DBNs and its applications to gene regulatory networks inference, see (HUSMEIER, 2003).



(a) (b)

Figure 5.5: **Difference between static BNs and dynamic BNs.** Cycles in (a) static BNs are unfold in (b) dynamic BNs and represented as a temporal sequence. Reproduced from (HECKER et al., 2009).

5.6 Tools

The most well-known software for reverse engineering of GRN based on the Bayesian formalism is Banjo[2], developed in the Department of Computer Science at Duke University under the direction of Alexander J. Hartemink (YU et al., 2004). Banjo implements both static and dynamic Bayesian networks, allowing, hence, its application to steady-state and time-series data. Heuristic approaches are applied to search the network space in order to identify the network structure that best represents the relations hidden in the input biological data. The search algorithm in Banjo consists of a set of individual core components: proposing a new network (or networks), handled by a "proposer" component; checking the proposed network(s) for cycles, handled by a "cycle checker" component; computing the score(s) of the proposed network(s), handled by an "evaluator" component; and deciding whether to accept a proposed network, handled by a "decider" component. Networks are evaluated according to the BDe metric, and the output network, e.g. the one with maximum score, is presented to user as a signed directed graph. The software is available for download under a non-commercial license agreement.

5.7 Discussion

Bayesian networks is, perhaps, one of the most applied computational techniques to the problem of reconstructing regulatory networks from postgenomic data. Most of its popularity is due to the probabilistic formalism, which makes it specially suitable for coping with the inherent noise in data and the large uncertainty about the biological system structure. In fact, the probabilistic nature of Bayesian networks is a promising tool for uncovering implicit relations among genes, providing a more detailed picture about regulatory pathways underlying organisms. Genetic regulatory networks are believed to be sparse and Bayesian networks are especially suited for learning in such sparse domains (FRIEDMAN et al., 2000). Moreover, whenever biological knowledge is available, the method allows its integration as a prior distribution over networks structure, speeding up the learning process and increasing the significance of the achieved results.

Since the direct computation of the best network structure is impractical, as previously explained in this chapter, a sampling method or heuristic search is needed, which can be time demanding and, specially in the latter, sensitive to local optima. Therefore, current research on the application of BNs to the inference of GRN focus mainly on improvements in the learning algorithm, aiming to reduce computational time and reach more accurate results. The improvements vary from small but high impact changes in the classical algorithms, such as the inclusion of a new edge reversal move in MCMC to enhance convergence (GRZEGORCZYK; HUSMEIER, 2008), to parallel computation (MENDOZA; WERHLI, 2010) and the application of distinct optimization algorithms, such as evolutionary algorithms (AULIAC; FROUIN; BUC, 2008).

More efficient ways to deal with the dimensionality problem is also a point of interest in current researchs. The available experiments are able to measure the expression of a massive number of genes, but provide only a small number of samples, which makes difficult the estimation of model's parameters. As computational methods such as Bayesian networks usually need a good volume of data to correctly reconstruct models, this is an important point of improvement. The inclusion of a priori knowledge is one of the directions to deal with this issue. Its integration to the learning algorithm is not

[2]`http://www.cs.duke.edu/~amink/software/banjo/download`

completely elucidated, but it is known that it helps to improve convergence and results accuracy. However, as biological prior knowledge is as limited as the gene expression data, researchers have suggested alternative ways of increasing the volume of the training data set, like, for instance, learning from a combination of biological data sets (WERHLI; HUSMEIER, 2008).

6 BOOLEAN NETWORKS

One class of models that has received lot of attention in GRN reverse engineering process is Boolean networks. While Bayesian networks, presented in Chapter 5, is a typical continuous and stochastic modeling framework, Boolean networks is the most common discrete and deterministic approach for regulatory networks reconstruction. Its inherent simplicity, which at first sight might seen a shortcoming of the method, is in fact an appealing property, emphasizing generic network behavior rather than quantitative biochemical details (LÄHDESMÄKI; SHMULEVICH; YLI-HARJA, 2003). Therefore, in the current chapter the Boolean network formalism and its application to model the dynamics underlying gene expression data are introduced. The chapter begins by a formal definition of Boolean networks and some well-known learning approaches. In the sequence, relevant works regarding the application of this formalism for the inference of GRNs, are well as available software, are outlined. The current chapter ends with a discussion on the positive and negative features of Boolean networks.

6.1 Random Boolean Networks

A random Boolean network (RBN) is a directed graph $G(V, F)$ defined by a set of nodes $V = \{x_1, x_2, \ldots, x_N\}$, which in GRNs context represent genes, and a set of Boolean functions $F = \{f_1, f_2, \ldots, f_N\}$. Each node x_i, $i = 1, \ldots, N$, is a Boolean device that stands for the state of variable i: it can assume values 0/1, true/false, on/off, etc. In GRN context, $x_i = 1$ denotes that gene i is expressed, while $x_i = 0$ means that it is not expressed (LÄHDESMÄKI; SHMULEVICH; YLI-HARJA, 2003). An example of a simple Boolean network structure is depicted in Figure 6.1, where the double line node represents an expressed gene (state 1), while the single dashed line nodes denote not

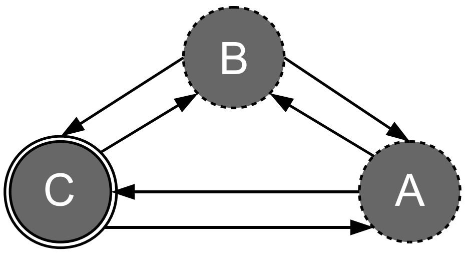


Figure 6.1: **Example of a Boolean network with simple structure.** The picture shows an example of $N = 3$ interacting genes, with $K = 2$, modeled as Boolean devices. The double line node represents an expressed gene (state 1), while the single dashed line nodes denote not expressed ones (state 0)

Table 6.1: **Boolean functions for the example Boolean network.** The Boolean functions for nodes A, B, and C, depicted in the example network of Figure 6.1, are specified in this table. Nodes A and B are controlled by function OR, while node C is controlled by function NAND.

(OR)			(OR)			(NAND)		
B	C	A	A	C	B	A	B	C
0	0	0	0	0	0	0	0	1
0	1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

expressed ones (state 0).

Each node has its value determined by a Boolean function $f_i \in F$, which represents the rules of regulatory interactions between nodes, and K_i specific inputs, denoting its regulatory factors (also called *in-degree*). A function f_i specifies, for each possible combination of K_i input values, the state of the regulated variable x_i. Thus, being K_i the number of input variables regulating a given node, since each of these inputs can be either at state 1 or 0, the number of combinations of states of the K_i inputs is 2^{K_i}. Furthermore, for each of these combinations, a specific Boolean function must output either 1 or 0. Therefore, the total number of Boolean functions over K_i inputs is $2^{2^{K_i}}$. When $K_i = 2$, some of these functions are well-known (AND, OR, XOR, NAND, etc.), but in the general case functions have no obvious semantics. Given the values of nodes V at time t, the Boolean functions are used to synchronously update the values of nodes at time $t + 1$.

To illustrate the regulation process, consider the network of Figure 6.1: the GRN is modeled as a RBN of $N = 3$ genes, with $K_i = 2$. The parents's set for nodes A, B and C are, respectively, {B,C}, {A,C} and {A,B}. A Boolean function is randomly assigned to each gene and the final regulatory rules are shown in Table 6.1: genes A and B are regulated by function OR, while gene C is regulated by a NAND function. Since the network has a fixed number of genes, each of which has a constant set of possible values, the network can assume just a finite set of states, given by 2^N, where N is the number nodes in the network. Therefore, given the Boolean functions from Table 6.1, Table 6.2 shows the resulting expression of each gene x_i at time $t + 1$ for all possible 2^3 states of network, based on their mapping functions and corresponding values in time t.

As system passes along a sequence of states, triggered from a random initial one, it may eventually re-enter a previously visited state due to its deterministic feature and finite set of possible states. This bring us the notion of *cycle* or *attractor*, which is the set of revisited states. The attractors and the states that lead into them comprise the *basins of attraction*. According to SHMULEVICH; DOUGHERTY (2010), the attractor represents the fixed-point of the system, thus capturing the system's long-term behavior. Starting from any state on an attractor, the number of transitions necessary for the system return to it is called the cycle length. If a state re-enters itself, then it is known as an equilibrium state or point attractor. According to KAUFFMAN (1969), a formal genetic network must contain at least one behavior cycle. In figure 6.2, one sees that there is only one attractor state for this example, namely `110`.

Random Boolean networks are randomly constructed in twofold directions. First, the K_i inputs of each node are defined at random. Second, one of the $2^{2^{K_i}}$ possible functions is randomly assign to each gene. After being assembled these networks are deterministic

$$\begin{tikzcd}
 & 000 \arrow[r] & 001 \arrow[d] & & \\
010 \arrow[r] & 101 \arrow[r] & 111 \arrow[r] & 110 \arrow[loop above] & \text{state cycle 1} \\
 & 100 \arrow[r] & 011 \arrow[u] & &
\end{tikzcd}$$

Figure 6.2: **State transition graph for the example Boolean network.** The state transitions in Table 6.1, concerning the network in Figure 6.1, may also be represented as a directed graph. For this example there is only one attractor state, namely `110`.

unless perturbations occur. We refer as perturbations the operations of flipping the value of a node or changing its function at random. Since GRN are highly stable in the presence of perturbations to genes, the Boolean network formalism should preferably be able to reproduce this same behavior. In fact, when a minimal number of genes transiently change value due to some external stimulus, the system usually transitions between states that compose the basin of attraction and eventually flows back to the same attractor, holding a so-called structural stability. In short, the structural stability captures the idea of a behavior that is not destroyed by small changes to the system (SHMULEVICH; DOUGHERTY, 2010).

6.2 Learning Random Boolean Networks

Although many different approaches for learning RBNs have been already proposed, as later discussed in Section 6.3, two problem formulations have been widely applied in the field: the Consistency Problem and the Best-fit Problem (AKUTSU; MIYANO; KUHARA, 1999; SHMULEVICH et al., 2003). In what follows we define both formulations focusing in their application for one function (gene) only, as specified in LÄHDESMÄKI; SHMULEVICH; YLI-HARJA (2003). Their extensions to a RBN can be obtained by repeating the same definition for all genes in the network.

Table 6.2: **State transition table for the example Boolean network.** Since the network has a fixed number of genes with a constant set of possible values, $x_i = \{0, 1\}$, the network can assume just a finite set of states, of size 2^N.

(t)			(t+1)		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	0

6.2.1 Consistency Problem

An issue related to inference of GRNs is to identify a network consistent with the observations in the given gene expression profile or determine if this network exists at all. This approach is known as Consistency Problem (or Extension problem) and resides in finding a Boolean function f from a class of functions C such that f is a perfect Boolean classifier, e.g. it correctly separates the given binary examples in true and false sets. In other words, the Consistency Problem entails a search for a rule from given examples.

A partially defined Boolean function, *pdBf*(T, F) is defined by two sets, T and $F \in \{0,1\}^n$, where T and F denote the set of true and false examples. These vector are determined by $T(f) = \{x \in \{0,1\}^n : f(x) = 1\}$ and $F(f) = \{x \in \{0,1\}^n : f(x) = 0\}$, respectively. The function f is said to be a consistent extension of *pdBf*(T, F) when $T \subseteq T(f)$ and $F \subseteq F(f)$. If a consistent extensions exists, then this function is returned.

However, as expression patterns exhibits uncertainty, and considering that real gene regulatory networks comprise many other elements beside genes, e.g proteins, one may argue that the simple Consistency Problem may not be used to infer network structure from data. In this case, it may seen more reasonable to conduct a search for Boolean functions that minimize the number of misclassification with respect to the given examples.

6.2.2 Best-Fit Extension Problem

As gene expression data are known to be noisy, another, new problem formulation has been proposed for the model inference: the Best-Fit Extension Problem. While the Consistency Problem aims to find the perfect Boolean classifier, the Best-Fit Extension Problem looks for the Boolean functions that causes as few misclassification as possible.

Suppose we are given two sets of binary vectors, T and F, as in the Consistency Problem. Let $T(f) = \{v \in \{0,1\}^n : f(v) = 1\}$ be called the on-set of function f and $F(f) = \{v \in \{0,1\}^n : f(v) = 0\}$ be the off-set of f. Consider, also, that positive weights (w) are available for all vectors $x \in T \cup F$ and that for a subset $S \subseteq T \cup F$. We have then the following definition: $w(S) = \sum_{x \in S} w(x)$. Thus, the magnitude of the error of function f is defined as:

$$\varepsilon(f) = w(T \cap F(f)) + w(F \cap T(f)). \tag{6.1}$$

The goal is to output subsets T^* and F^* such that $T^* \cap F^* = \varnothing$ and $T^* \cup F^* = T \cup F$ for which *pdBf*(T^*, F^*) has an extension in some class of functions C and so that $w(T^* \cap F) + w(F^* \cap T)$ is minimum. Consequentially, any extension $f \in C$ of g_{T^*,F^*} has minimum error size. The Consistency Problem can be, thus, defined as a special case of the Best-Fit Extension Problem, when $\varepsilon(f) = 0$.

6.3 Application to Gene Expression Data

Random Boolean networks have been used to explain adaptation and self-organization in complex systems. However, the groundbreaker suggestion of studying the behavior of gene regulatory systems by means of networks of Boolean functions was introduced by Kauffman in (KAUFFMAN, 1969). As the number of regulatory factor per genes is unknown, even nowadays, Kauffman studied topologies in which each gene has direct inputs from all genes, as well as topologies with $K = 1$, $K = 2$ and $K = 3$ input(s) per gene. One of the $2^{2^{K_i}}$ possible Boolean functions was randomly assigned to each gene based in the assumption that in nature there is no reason to suppose that all elements within an organism's underlying regulatory network perform the same activity, e.g. carry the same Boolean function. Kauffman's experiments suggested that large randomly connected feedback networks of binary nodes (genes) behave with stability comparable to that in living organisms. Furthermore, the author presents evidence that these systems undergo short stable cycles, which parallel and predicts the time required for cell replication in many species. As conclusion, Kauffman asserts that "*large, randomly assembled nets of binary elements behave with simplicity, stability, and order. It seems unlikely that Nature has made no use of such probable and reliable systems, both to initiate evolution and protect its progeny*".

Since Kauffman's seminal work, most of research on Boolean networks has focused on unraveling the structure of GRNs from gene expression data. LIANG; FUHRMAN; SOMOGYI (1998) investigated the viability of inferring a complex regulatory network architecture, modeled as a RBN, through a systematic analysis of mutual information between input/output states, such as state transition tables. The algorithm, named REVEAL, starts by testing if a particular gene is an effective one-input rule device. The Shannon entropy H regarding every possible single input is computed in turn. For genes whose output is not determined by a single input, the effective K for the rule of that gene is larger than one. Then, in the next step, the REVEAL algorithm tests whether the gene is determined by a rule with two effective inputs. This procedure continues until all genes have their rule established, gradually increasing the number of input whenever appropriated. The advantage of REVEAL algorithm is that simple networks can be calculated very quickly by a simple comparison between entropies of state transition pairs. The algorithm computes the entropy for higher K only when required. In this case, however, the computational cost is much higher and authors suggest the inclusion of parallel computation. LIANG; FUHRMAN; SOMOGYI (1998) suggested that only a small number of state transition pairs (around 100 pairs) were sufficient for inferring Boolean networks with 50 nodes, regulated by 3 other genes at most.

AKUTSU; MIYANO; KUHARA (1999) also proposed an algorithm for learning a Boolean network from data based on state transition tables and on the concepts of Consistency Problem, the BOOL-1 algorithm. The algorithm performs an exhaustive search: for each pair of nodes (v_k, v_h) in V and for each Boolean function $f \in F$, it checks whether or not $O_j(v_i) = f_i(I_j(v_k), I_j(v_h))$ holds for all $j = 1, \ldots, m$, e.g. if f is *consistent*, in which I refers to the input and O to the output. If there is only one consistent network, the algorithm returns the function that satisfies the above condition. According to the authors, this algorithm is much simpler than REVEAL, enabling its mathematical analysis, although it may not be so computationally efficient as the latter: it works in $O(n^3m)$ time for $K = 2$ and in $O(n^4m)$ time for $K = 3$, , where n is the number of nodes and m the number of samples. The authors proved that for a Boolean network with in-degrees bounded by a constant K and transition pairs given uniformly randomly from 2^N possible pairs, the algorithm requires $O(\log n)$ input/output pairs to correctly identify its structure.

One of the most common criticisms regarding the use of the Consistency Problem for inference of regulatory networks is that gene expression data are inherently noisy. Therefore, aiming to cope with the noise in data, AKUTSU; MIYANO; KUHARA (2000) modified the BOOL-1 algorithm, proposing a new robust inference algorithm, BOOL-2, for the so-called noisy Boolean networks. A noisy Boolean network consists of a graph $G(V, F)$ and p_{noise}, where p_{noise} is a constant taken over all possible input patterns I_j such that $0 < p_{noise} < 1$. The inference of the network structure is similar as in BOOL-1, except that the new algorithm checks if the condition $O_j(v_i) = f_i(I_j(v_), I_j(v_h))$ holds with probability p_{noise} for each node in a noisy Boolean network. Thus, instead of discarding all inconsistent functions during the exhaustive search, BOOL-2 discards only functions inconsistent with at least $\theta \cdot m$ patterns, where $\theta = \frac{1}{2^{2K+1}}$. According to the authors, BOOL-2 requires a larger number of data sets, which must comprise different experimental environments or conditions, when compared to BOOL-1. Regarding its computational complexity, computation time is not a serious problem. As REVEAL and BOOL-1, BOOL-2 is suitable for handling several hundred of genes simultaneously, which may be improved by the application of different search heuristics and parallel computation.

Following the direction of aggregating probability to random Boolean networks as a way to cope with uncertainty, SHMULEVICH et al. (2002) proposed an extension of random Boolean networks. According to the authors, given several 'good' competing functions for a given gene, there is no reason to put all the faith in just one of them. Therefore, they extend the classical RBN model in order to allow a single node to hold more than one possible function, creating the so-called probabilistic Boolean networks (PBN). In PBN, at any given point in time, the value of each node is determined by one of the possible functions, chosen according to its corresponding probability. The inference process in based on the coefficient of determination (COD), which measures the degree to which the transcriptional levels of an observed gene set can be use to improve the prediction of the transcriptional level of a target gene relative to the best possible prediction in the absence of observations (SHMULEVICH et al., 2002). The COD is in interval $[0, 1]$ and is estimated from the data themselves. The main advantage of PBN formalism is the fact that it is more flexible and powerful while retaining all the appealing properties of RBNs.

Apart from heuristic search, an alternative method for inference of random Boolean networks is correlation measurement, for example, the information-theoretic approach. MAUCHER et al. (2011) developed a method suitable for learning the network structure from large-scale data. Authors rely on the observation that most transcriptional regulators will be either activators or inhibitors of a certain gene in a specific cell type. Furthermore, the activating or repressing effect of a transcription factor monotonically depends on its cellular concentration, such that an increase in the concentration of an activator (repressor) will increase (decrease) but never decrease (increase) transcription of its target. According to MAUCHER et al. (2011), this kind of transcriptional regulation can be modeled mathematically in a very simple manner by the use of monotone Boolean functions. In order to detect the directed causal regulatory dependencies in a network, authors examined how the expression of different genes correlates with the successive states of potential target genes by means of Pearson correlation. The proposed algorithm performs a faster and more reliable identification of interactions than the best-fit problem with an overall running time of the order $O(n^2m)$, where n is the number of nodes and m the number of samples.

In addition to the Consistency Problem, the Best-Fit Problem and their extensions, the other class to infer Boolean networks is machine learning-based algorithms, in which genetic algorithm[1] (GA) is perhaps the most applied method. Discrete models were used in REPSILBER; LILJENSTROM; ANDERSSON (2002) to develop a ranking method

[1]Genetic algorithms consists of an optimization process inspired by the natural phenomenon of adaptation. A population of candidate solutions is evolved based on operations inspired from biology, i.e. selection, crossover and mutation, and the principle of *survival of the fittest* is applied such that fitter solutions have more probability of surviving through generations, while weaker ones perish.

of alternative hypothesis models to a target GRN. Since experimental data has limited availability, authors propose a more realistic approach, in which the set of alternative hypotheses is given as input file to a GA together with the gene expression data. Through the minimization of a quadratic error function between expression profiles obtained from generated and target networks, the method successfully ranked the alternative hypothesis, identifying the most probable class of network structures given the input gene expression patterns.

GAs were also used in MENDOZA; BAZZAN (2011) to explore the search space of all possible network structures coded as RBNs, rather than using exhaustive search, which usually requires a prohibitive amount of computation time for high dimensional problems. In this work, authors evaluate the power of inference of this approach and how far is possible to reconstruct an accurate model using solely experimental data, e.g. without supplying any biological prior knowledge. This is useful when no such prior knowledge is available, which is a common situation. The implemented algorithm reached a good accuracy level, representing a valuable start point for biologists in the investigation of gene interactions. The reported precision is low, a fact associated by the authors not only to the high frequency of false positives, but also to the stochastic nature of GA, which allows individuals to explore different sites of search space and, therefore, to have different topologies between themselves. However, exploring different solutions simultaneously is extremely advantageous when the solutions found by the GA are combined into a consensus network: almost all interactions in the target GRN have been correctly inferred by the model.

6.4 Advantages and Disadvantages

Perhaps the main advantages in using random Boolean networks for the reverse engineering of regulatory networks are the dynamic and rule-based properties of this formalism, which are common features of real GRN. In addition, Boolean networks are computationally simple, allowing their exploitation on larger scales more easily than in other modeling frameworks. Despite their simplicity, RBN are able to capture much of the complex dynamics of gene networks and allow the extraction of meaningful biological information (LÄHDESMÄKI; SHMULEVICH; YLI-HARJA, 2003). When the interest lies in the qualitative features of the network, RBN is indeed a suitable and efficient tool.

In contrast, the binary property of the devices in RBN is a strong abstraction, which can cause loss of information and interfere with the quality of reconstruction. Furthermore, RBN are inherently deterministic, which goes against the stochasticity observed in real biological systems (ARKIN, 1999). The assumption of only one logical rule per node may lead to incorrect conclusions when inference is based on gene expression data, as the latter are typically noisy and the number of samples is usually much lower than the number of parameters to be inferred (SHMULEVICH et al., 2002). In PBN, an extension of the standard Boolean networks, the determinism has been relaxed allowing the identification of a set of functions together with their corresponding selection probabilities and the quantification of the influence of genes on other genes. This adaptation not only allows the modeling of stochastic processes, but also the handling and learning from noisy data, providing a good balance between computational complexity and inference performance.

6.5 Tools

The BN/PBN toolbox is a MATLAB Toolbox maintained by Harri Lähdesmäki and Ilya Shmulevich. The software works with both random Boolean networks and probabilistic Boolean networks and it includes functions for performing a wide variety of simulations and network analysis. Some examples of features available are: simulation of network dynamics, computation of network statistics (numbers and sizes of attractors, basins, transient lengths, influence matrices), computation of state transition matrices and stationary distributions, inference of networks from data, generation of random networks and functions, visualization, intervention and membership testing of Boolean functions. The toolbox is available for download in a webpage[2] that comprises a comprehensive source of information about research work on probabilistic Boolean networks and related topics.

6.6 Discussion

The Boolean network formalism for modeling GRN discussed in the current chapter is the simplest and the first applied technique for inference of regulatory networks. Experiments held by Stuart Kauffman in (KAUFFMAN, 1969) suggested that random Boolean networks are as stable, ordered and simple as biological systems and since then they have been widely applied in the study of gene interactions and regulation. The simplicity built into random Boolean networks offers more benefits than disadvantages in the process of reverse engineering in which we are interested. For instance, the lower complexity of learning algorithms allows their application to high dimensional problems more easily and efficiently than other approaches. Furthermore, as the main interest is often the extraction of a global picture of gene regulation, due specially to the restricted information available for creating a detailed description of such mechanism, RBNs are a suitable and practical tool as they summarize the entire set of possible gene activities and states in only two basic conditions: a gene can be either expressed or not in a given time. Although this may seem a strong abstraction, is in fact one of the most appealing features of RBNs, making it a robust modeling framework.

Notwithstanding all the advantages introduced by this method, RBNs have a relevant shortcoming of being extremely vulnerable to noisy data. In this special case, the method is not robust enough to deal with uncertainty as it is based on deterministic transitions between possible network states. To overcome this limitation, a probabilistic extension of RBNs have been proposed, allowing each gene to hold more than one Boolean function and computing its value by choosing one of the possible functions according to its corresponding probability. The need for discretized data is also a disadvantage of Boolean networks as it causes information loss and introduces even more uncertainty to training data, interfering with the quality of reverse engineering process.

Nowadays, most of researches on the application of random or probabilistic Boolean networks to the modeling of regulatory networks focus on more efficient algorithms to identify the best network structure. The sought-after efficiency refers to both the computational time and the results' quality, often assessed in terms of metrics like precision, accuracy and area under the ROC curve. Although the most commonly used techniques are learning algorithms and search methods, statistical techniques such as correlation analysis have proven themselves a reliable and fast alternative to find dependencies in the

[2]http://personal.systemsbiology.net/ilya/PBN/PBN.htm

network. Once captured the structure and dynamics of GRN by means of Boolean networks, in silico experiments and hypothesis-testing may be easily performed, elucidating the functional mechanisms and basic dependencies of gene regulation in a given organism. The generated model represents a coarse-grained description of GRN for studying large scale gene networks through macroscopic variables in a global fashion. Even when not able to completely explain the process of gene regulation, the modeled Boolean network is at least a good source of information to define new experimental targets and conditions in order to collect more data for reverse engineering gene regulatory networks.

7 NEURAL NETWORKS

Despite the wide application in reverse engineering of GRN, the computational methods presented so far have inherent performance limitations. Bayesian networks, introduced in Chapter 5, are known to cope well with noise, incompleteness and stochastic aspects found in gene expression data. However, they do not consider the dynamical aspects of gene regulation and leave temporal information unaddressed (XU; WUNSCH; FRANK, 2007). Random Boolean networks, discussed in Chapter 6, are useful in exploring the dynamics of GRN in a global fashion. Yet, they ignore the effect of genes at intermediate levels and assume transitions between genes' states to be synchronous, which is biologically implausible.

Neural networks, as Boolean networks, are a coarse-grained approach to analyse large gene regulatory networks, but differently from those, they work at an intermediate level. In this formalism, nodes still represent genes, while connections between nodes denote regulatory influences on gene expression. However, in contrast to the latter, the gene expression and regulation is measured in a continuous range in order to capture properties that are not identified by the discrete models.

7.1 Artificial Neural Networks

The formal definition of an Artificial Neural Network (ANN) is given according to HAYKIN (1998).

> A neural network is a massively parallel distributed processor made up of simple processing units, named artificial neurons, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two aspects:
>
> 1. Knowledge is acquired by the network from its environment through a learning process.
> 2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The structure of an ANN is uniquely determined by the number of nodes and the wiring, e.g the connections among these nodes. Figure 7.1 shows the basic elements of an individual neuron. Each neuron has a set of links or synapses characterized by a weight, such that a signal x_j at input j of neuron k is multiplied by the synaptic weight w_{kj}. All the input signals are summed up by an adder according to Equation 7.1, which constitutes

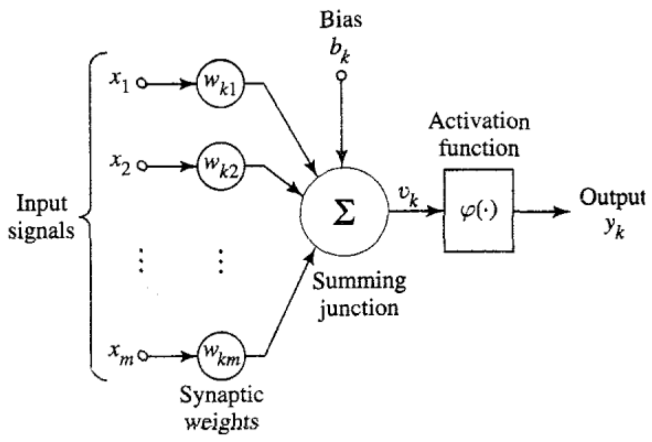


Figure 7.1: **Elements of the neuronal model.** Each neuron in the ANN is composed by a set of synapses with individual weights, a linear adder, a bias and an activation function. Reproduced from HAYKIN (1998).

a linear combiner:

$$u_k = \sum_{j=0}^{m} w_{kj} x_j \tag{7.1}$$

Also, the neuronal model includes a bias b_k, which has the effect of applying an affine transformation to the output u_k of the adder. The bias increases or decreases the input to the activation function, depending on whether it is a positive or negative factor, respectively. Considering the bias and the output of the linear adder, the input to the activation function is thus defined as:

$$v_k = u_k + b_k \tag{7.2}$$

Finally, a neuron has an activation function $\varphi_k()$ for limiting the amplitude of a neuron's output y_k, transferring it to a normalized transcriptional response. Three common used activation functions are the threshold function, the piecewise linear function and the sigmoid function (HAYKIN, 1998). The latter is shown in Equation 7.3.

$$\varphi_k() = \frac{1}{1 + e^{-v_k(t)}} \tag{7.3}$$

When applied to gene expression data, the nodes represent genes, the value of the node is the corresponding gene expression value and the connections define the regulatory interactions. Also, the number of nodes is often defined as the number of genes observed, although it may also include other factors involved in the regulatory process.

It is assumed that the state of gene expression at time $t + dt$ depends on the state of expression at time t and on the synaptic weights. Let a N-dimensional vector $x(t)$ be the expression state of a GRN with N genes, such that the element $x_k(t)$ denotes the expression level of gene k at time t. Also, consider w to be the matrix of synaptic weights of all connections in the network.A positive weight implies a stimulating effect (positive feedback) while a negative weight implies repression (negative feedback). Given these definitions, the expression level of gene k at time t may be computed as:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j + b_k \tag{7.4}$$

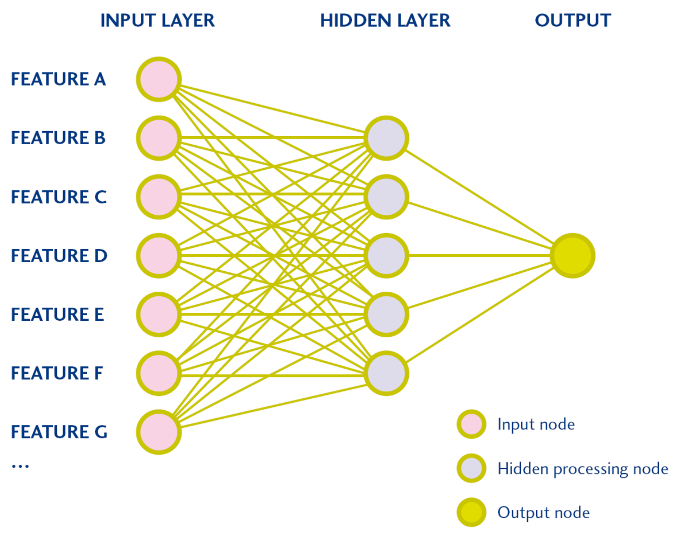


Figure 7.2: **Multilayer artificial neural network structure.** ANN are organized in the form of layers. The input layer receives information from the input vector, which constitute the input signals to the second layer (i.e. the first hidden layer). The model may have more than one hidden layer. The set of output signals of the neurons in the output (final) layer are the overall response of the network to the activation pattern supplied by the nodes in the input layer. Reproduced from the SANY IP website, available at `http://www.sany-ip.eu/book/export/html/3271`.

where, regarding the gene expression context, the bias b_k represents the influence of external inputs or reaction delay (TIAN; BURRAGE, 2003). After the application of the activation function, the output y_k is a value between 0 and 1, where 0 represents complete repression and 1 represents maximal expression.

Nodes are organized in layers, whose number and arrangement defines the classification of the ANN. In Figure 7.2, an example of a multilayer ANN is given. The input layer receives information from the input vector, which constitute the input signals to the second layer (i.e. the first hidden layer). In this figure, the input information corresponds to different features (feature A, feature B,...). However, when applying this model to gene expression data, the input refers to the expression level of genes in the target system. Despite the illustration of only one hidden layer, the model may have more than one hidden layers, providing better tools for extracting higher-order statistics. The set of output signals of the neurons in the output (final) layer are the overall response of the network to the activation pattern supplied by the nodes in the input layer. As the information is conducted in a single direction, from the input nodes to the hidden nodes and finally to the output layer, this network is referred to as feedforward ANN. Feedforward networks do not contain any cycles or loops. In contrast, recurrent neural networks (RNN) have at least one feedback loop, which has a deep impact in the learning capacity of the network and its performance.

7.2 Learning Artificial Neural Networks

In most cases, ANNs are adaptive systems, which means that their structure, or more specifically their synaptic weights, change with time based on the information flowing through the network. This process is known as learning and is a relevant property in the performance improvement of neural networks. At the end of each time step, the network becomes more knowledgeable about its environment, providing more precise information on the data being analysed.

According to HAYKIN (1998), the learning process comprises three steps:

1. The neural network is stimulated by the environment in which is embedded.
2. The neural network undergoes changes in its free parameters as a result of the stimulation.
3. The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

In what follows the fundamental learning paradigms are overviewed and the most well known learning algorithms in the neural network field are introduced.

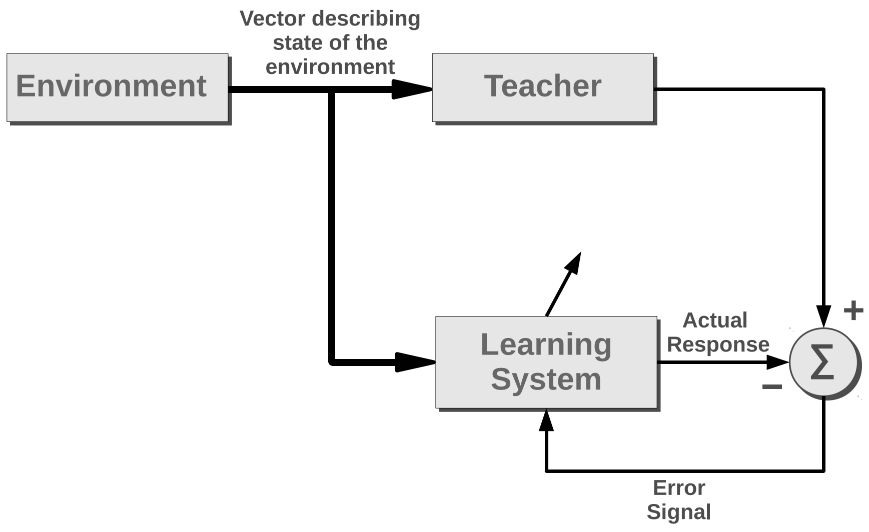
7.2.1 Paradigms

All learning methods used not only for adaptive neural networks, but also for other classes of modeling frameworks, can be classified into three major categories: supervised, unsupervised and reinforcement learning.

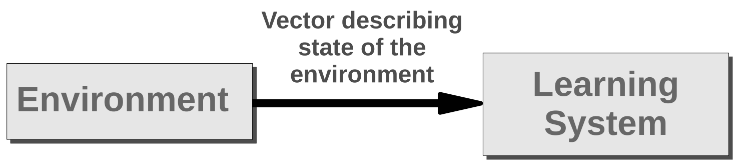
Supervised learning assumes the existence of a teacher, or expert, who has knowledge about the environment represented as a set of examples (input–output data) and provides the neural network with a desired response for a given instance of a training vector. According to HAYKIN (1998), the desired response is the optimum action to be performed by the neural network. The synaptic weights are iteratively adjusted under the combined influence of the training vector and the error signal, which is defined as the difference between the desired and the actual response of the network. Therefore, the knowledge available to the teacher is gradually captured by the neural model, such that at each time step the ANN is more likely to produce the appropriate response to a given input. This process is depicted in Figure 7.3(a).

In contrast, unsupervised and reinforcement learning methods do not rely on an expert to oversee the learning process. The mechanism underlying unsupervised learning is illustrated in Figure 7.3(b). In unsupervised methods, the learning is based only on local information and is usually referred to as self-organization. The parameters adaptation is performed based on the similarities and differences among the input patterns, given by a task-independent measure of the quality of representation that the network is required to learn.

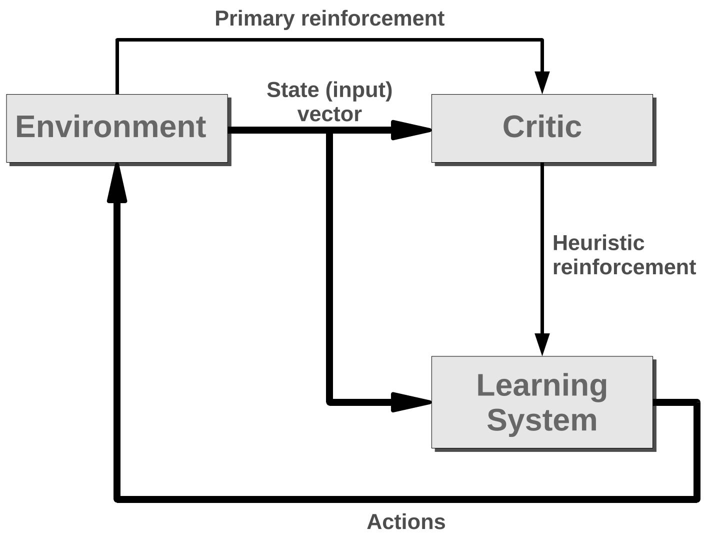
Likewise, reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented by an expert. Although an interaction with the environment exists, the learning process happens due to the observation of the consequences of the previously chosen actions, rather than due to a explicit teaching by an expert. The learning is essentialy based on the “trial and error" principle: for each selected choice a numerical reward is received, which encodes the success of an action’s outcome. In the sequence, new decisions are made such that the accumulated reward is maximized over time. The described process is depicted in Figure 7.3(c).



(a)



(b)



(c)

Figure 7.3: **Learning paradigms** Learning methods used not only for adaptive neural networks, but also for other classes of formalisms, can be classified into three major categories: (a) supervised, (b) unsupervised and (c) reinforcement learning. Adapted from HAYKIN (1998).

7.2.2 Error-Correction Learning

Error correction learning is a supervised learning method which consists in comparing the system output to the desired output value and directing the training based on the feedback from this comparison. The error signal $e_k(n)$ is defined as

$$e_k(n) = d_k(n) - y_k(n) \tag{7.5}$$

where $d_k(n)$ is the expected output and $y_k(n)$ is the actual output, ans acts as a control mechanism, which applies a sequence of corrective adjustments to the synaptic weights of neuron k. The consequence is that the output $y_k(n)$ gradually becomes closer to the desired response $d_k(n)$ (HAYKIN, 1998). This is achieved by the minimization of a cost function, defined in terms of the error signal:

$$E_n = \frac{1}{2}e_k^2(n) \tag{7.6}$$

E_n is the instantaneous value of the error energy. The minimization of the cost function E_n leads to a learning rule referred to as *delta rule*, which adjusts the synaptic weights of neurons proportionally to the computed error signal and to the input signal. The delta rule is computed according to the following equation:

$$\Delta w_{kj}(n) = \eta e_k(n)x_j(n) \tag{7.7}$$

where η is a positive constant that determines the rate of learning. The synaptic weights are then updated adding up the synaptic adjustment $\Delta w_{kj}(n)$ to the weights in the current time step, as shown in what follows:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{7.8}$$

7.2.2.1 Least Mean Square

The least mean square algorithm applies the error-correction learning to single layer ANNs, whose simplest form is known as perceptron. The algorithm incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector, which eventually leads to the minimum mean square error.

Algorithm 7.1: Least mean square algorithm.

```
$w(0) = 0$;
repeat
  for each training example $n$ do
    Activate the perceptron by providing input vector $x(n)$ and desired output $d(n)$;
    Compute the output value $y(n) = w^T(n)x(n)$;
    Adjust the perceptron's weight vector: $w(n+1) = w(n) + \mu[d(n) - y(n)]x(n)$;
  end for
until no more updates are made in the weight vector
```

Algorithm 7.2: Backpropagation algorithm.

```
Initialize weights with non-zero arbitrary values
repeat
  for each training example $n$ do
    Activate the multilayer ANN by providing input vector $x(n)$ and desired output $d(n)$;
    Propagate $x(n)$ until the output layer;
    Compute the instant error $e_k(n)$ in the output layer;
    Compute the local gradients of output layer $\delta_k^o(n)$;
    Adjust the weights of nodes in the output layer according to equation:
    $w_{kj}^o(n+1) = w_{kj}^o(n) + \mu\delta_k^o(n)i_j(n)$;
    Compute the local gradients of hidden layer $\delta_j^h(n)$;
    Adjust the weights of nodes in the output layer according to equation:
    $w_{ji}^h(n+1) = w_{ji}^h(n) + \mu\delta_j^h(n)x_i(n)$;
  end for
until the mean square error is still under a given threshold
```

7.2.2.2 Backpropagation

For the multilayer neural network architecture, in which the input signal propagates through the network, layer by layer, in a forward direction, a commonly used training algorithm is called the error backpropagation algorithm. The backpropagation (BP) algorithm is also based on the error-correction learning rule and consists of a generalization of the least mean square method. In contrast to the latter, the BP algorithm passes twice through the network's layers: in the forward pass a stimulus is applied to the input layer and its effect is propagated through the network, layer by layer, producing a set of outputs. In the backward pass, the synaptic weights are adjusted according to the error correction rule. Henceforth, at each pass, the synaptic weights become closer to the desired output.

The major difference between learning a single-layer ANN and a multiple-layer ANN is that in the first one there are no hidden layers and the error at the output layer is very clear and of easy computation. Nonetheless, error at the hidden layers are not obvious as the training data does no provide us with the expected values of the nodes in these layers. Therefore, the practice is to back-propagate the error from the output layer to the hidden(s) layer(s), since the nodes in the hidden layers are somehow “responsible" for a fraction of the error in each of the output nodes in which is connected.

In multilayer ANNs, the update equation of nodes in output layer is similar to the one defined in the least mean square algorithm:

$$w_{kj}^o(n+1) = w_{kj}^o(n) + \mu\delta_k^o(n)i_j(n) \tag{7.9}$$

where the notation o denotes the elements in the output layer, μ is the learning rate, i_j is the input provenient from the neuron j in the previous layer and the local gradient δ_k^o is computed as:

$$\delta_k^o \equiv (d_k(n) - y_k(n))(1 - y_k(n)^2). \tag{7.10}$$

Similarly, the nodes in the hidden layer are updated according to the following equation:

$$w_{ji}^h(n+1) = w_{ji}^h(n) + \mu\delta_j^h(n)x_i(n) \tag{7.11}$$

where the notation h denotes the elements in the hidden layer, the term $x_i(n)$ refers to the stimulus in the neuron i of the input layer and the gradient $\delta_j^h(n)$ is defined as:

$$\delta_j^h \equiv (1 - i_j(n)^2) \sum_{k=1}^{M} (\delta_k^o(n) w_{kj}^o(n)). \tag{7.12}$$

7.3 Application to Gene Expression Data

One of the seminal works establishing the capability of ANN to describe the dynamic behavior of gene regulatory networks was carried out by Vohradsky in 2001 (VOHRADSKY, 2001). Vohradsky states that ANNs are able to explain the experimental observations and, more importantly, to predict the specific functions of the system in experimentally inaccessible situations, allowing the extraction of conclusions about the stability and functionality of GRNs. Moreover, the author highlights the suitability of recurrent neural networks for this specific application, due to their ability to cope with feedbacks and their flexibility to fit the data. The RNN model is applied to simulate how the virus bacteriophage λ chooses the pathway for growing (lytic or lysogenic growth) soon after infection of *E. Coli*.

RNN models were also used by XU; WUNSCH; FRANK (2007) in the inference of GRNs. Authors propose a two-step algorithm for learning GRN modeled as recurrent neural network. First, the algorithm unveils potential genetic network architectures that fit well with the time-series data by means of a particle swarm optimization [1] (PSO) algorithm. In other words, this step determines which weights of the ANN have a nonzero value. The PSO-based search avoids making an exhaustive enumeration of all possible connectivity, which is very time demanding. After defining the optimum structure, PSO is applied in the RNN training. In this step, the algorithm performs the evolution of the weight matrix such that the nonzero weights can be fine-tuned, whereas the nonsignificant weights remain equal to zero. The RNN/PSO approach was tested with both synthetic and real data sets and the results suggest that, as the techniques presented in previous chapters, RNNs are meaningful in revealing potential regulatory interactions between genes. Nonetheless, in contrast to those, RNN is also very promising to capture the nonlinear dynamics of gene regulatory systems.

HACHE et al. (2007) proposed to applied the Backpropagation Through Time (BPTT) algorithm, described by WERBOS (1990), to reconstruct GRNs modeled as RNNs. The BPTT is an extension of the standard Backpropagation algorithm: it is also a gradient based parameter learning method which minimizes the error function

$$E(y(t), \hat{y}(t)) = \frac{1}{2} \sum_t \sum_i [y_i(t) - \hat{y}_i(t)]^2 \tag{7.13}$$

by varying the parameters of the model during every iteration step. This way, the estimated value $y(t)$ of each node converges to the data points $\hat{y}(t)$. However, the BPTT unfolds the temporal operation of a network in a multilayer network, to whose topology is added a new layer at each time step. Therefore, considering a RNN with two layers, f and g, when the network is unfolded through time, the unfolded network contains k

[1]Particle swarm optimization is a computational optimization method that improves a population of potential solutions, called particles, with regard to a given measure of quality by iteratively changing the velocity of (accelerating) each particle towards the coordinates in the problem space which are associated with the best local and global solutions so far.

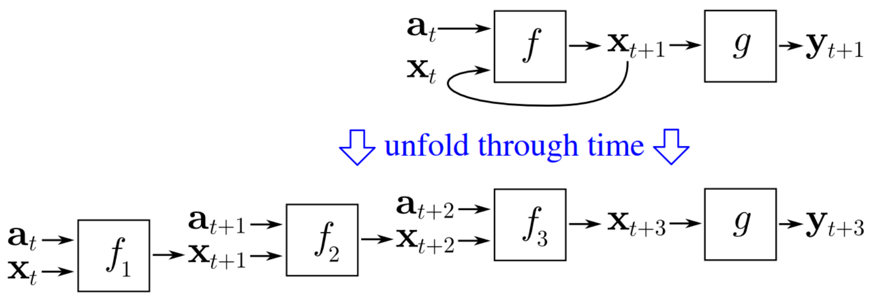


Figure 7.4: **Backpropagation Through Time.** The temporal operation of the RNN is unfolded in a multilayer network, to whose topology is added a new layer at each time step. Considering a RNN with two layers, f and g, when the network is unfolded through time, the unfolded network contains k instances of f and one instance of g.

instances of f and one instance of g. Figure 7.4 shows a graphical representation of this example, available at Wikipedia[2].

HACHE et al. (2007) resort to the simulation tool PyBioS to generate data for testing purposes. Both simple and complex networks were implemented, in which the latter reflect the features of a real biological system. The interest lies particularly in the evaluation of performance with respect to the reconstruction of network motifs of different sizes. As an example of complex network, authors modeled part of the gene developmental regulatory network of sea urchin.

Results were compared with the ARACNe software, developed by BASSO et al. (2005) based on mutual information, with a Dynamic Bayesian Model and a linear least square fit approach. Since the output of BPTT is not unique, the BPTT result is computed as the mean of ten learning processes with the same data set. In addition, the weight matrix generated by BPTT algorithm was discretized in three levels (activation, inhibition and non-regulation) based on an optimal threshold, so that comparison with the other mentioned methods is possible. Authors found that BPTT gives the best result in term of a distance measured defined as

$$d(sen, spe) = \sqrt{(1 - sen)^2 + (1 - spe)^2} \tag{7.14}$$

where sen is the computed sensitivity and spe is the computed specificity. The BPTT found the most true regulations and true non-regulations, having the highest sensitivity among all methods. However, as it also erroneously associated a lot of weights to regulations, its specificity is lower than ARACNe and DBN approaches. HACHE et al. (2007) states that a compromise among Bayesian methods and ANN learning techniques may introduce promising improvements.

7.4 Advantages and Disadvantages

Due to their nonlinearity feature, ANNs can perform complex tasks and solve difficult problems that are not feasible by linear methods. In addition, they deal with important aspects of real GRNs that are not addressed by the methods reviewed in previous chapters. First of all, ANNs allow a dynamic and temporal analysis of gene expression data, which is not possible with Bayesian networks. Second, since this is a continuous method,

[2]http://en.wikipedia.org/wiki/File:Unfold_through_time.png

it considers the intermediate level of gene expression ignored in the Boolean network formalism. As a consequence, data discretization is not required. Another appealing feature is the network ability to learn from example data and be later used to predict the response of the system under a new conditions. Last but not least, ANNs enables an asynchronous simulation of the system, which is not covered by Boolean networks and is still in its infancy in Boolean networks.

Concerning their drawbacks, ANNs need to be trained before actually beginning to operate, which demands time and a large training data set. Also, the training stage must be executed only as much as the network needs for learning, and not more, because ANNs have proneness to overfitting, becoming unable to detect new patterns or make new predictions in respect to original data. The processing phase is also time consuming, specially for large neural networks. Finally, the right decision about the network architecture, e.g. the number of layers and the number of nodes in each layer, is not obvious and is by itself a challenge.

7.5 Tools

Hendrik Hache and his group developed GNRevealer (HACHE et al., 2007), a software based on neural networks for reverse engineering of gene regulatory networks. GNRevealer is available for download[3] and is executed by C++ command line. No manual with user instructions and details about software implementation was found for download. Therefore, information about the use of GNRevealer is very restricted.

7.6 Discussion

Artificial neural network is a recent modeling framework for gene regulatory networks and its application has not been so widely explored as the use of clustering algorithms, Bayesian networks and Boolean networks, which were reviewed in previous chapters. While the seminal work about the application of Boolean networks to GRNs context dates to 1969 in KAUFFMAN (1969), ANNs were proposed as an alternative solution only in the beginning of the 21st century, by Vohradsky in VOHRADSKY (2001). In these ten years since Vohradsky's work, a greater attention has been turned to the learning of gene regulatory networks by means of Bayesian networks, which have been introduced to this context in FRIEDMAN et al. (2000) just one year before the publication of Vohradsky's paper. Bayesian networks seemed so promising at first sight that it was exhaustively exploited in several angles, thus rapidly establishing itself as a powerful solution to the problem. Hence, although neural networks are broadly known as an efficient method for solving complex problems, they are not as consolidated as Boolean networks and Bayesian networks in the context of gene regulatory networks modeling due to a matter of time and lack of efforts, which does not means that it do not offer important advantages in contrast those formalisms.

In fact, ANNs provide some features that are not covered by Boolean and Bayesian networks. For instance, ANNs are specially suitable for drawing meaning from domains with many parameters, interdependencies and uncertainty. Their most appealing feature is perhaps the ability to learn from data through a training stage and later perform predictions about new situations based on acquired knowledge. Their recurrent topology is able to

[3]http://www.molgen.mpg.de/ hache/GNRevealer/

cope with feedback loops and is flexible enough to fit well the data. The model is also noise-resistant and very robust: a failure in one neuron does not necessarily causes the collapse of the whole system.

Differently from Boolean networks, artificial neural networks is a continuous method and therefore considers the intermediate level of gene expression in the model inference, which improves its learning capacity. This factor also brings the advantage of not being necessary to discretize data, which is by itself a process that introduces uncertainty to the domain. When compared to Bayesian networks, ANNs introduce the benefit of allowing the analysis of the network's dynamic behavior and an asynchronous simulation of the system. The latter, although also possible to be performed with Boolean networks, is not an inherent feature of these and is still in an early stage of experimentation.

Notwithstanding all these advantages, ANNs have meaningful weaknesses which perhaps have influenced the low popularity of the method to this specific area of interest. First of all, due to its computational complexity, this modeling approach can currently only be applied to very small systems. The training stage is very time demanding, with no warranty of convergence, and the computation of the output may require too much time for large data sets and networks. In addition, figuring out what network structure will give the optimal solution is a big issue: even when trying out many different architectures, which is obviously extremely time consuming, is not possible to assure that the chosen network if the best solution. This is by itself an important research area concerning neural networks.

8 CONCLUSION

The recent improvements in biological experiments and the consequent amazing increase in the volume of biological data have introduced a new interesting research area in the field of bioinformatics: the reconstruction and analysis of gene regulatory networks. Nowadays it is known that organisms are regulated by intricated networks which interconnect all functioning entities within an organism, such as genes, proteins and RNA transcripts. Scientists believe that the pathway to effective drugs and treatments is to learn the functioning of these networks and how to control them.

The great amount of gene expression data available nowadays have brought the necessity of an interface between biological and computer sciences. More specifically, the use of machine learning methods for the analysis of such data aiming the reconstruction of the corresponding gene regulatory networks have become a common practice. The present work have outlined and reviewed some widely used methods for this purpose: clustering algorithms, Bayesian networks, Boolean networks and neural networks.

The survey shows that all methods have their own drawbacks and benefits and that each method covers different aspects of the reverse engineering problem. Clustering algorithms, for instance, identify similarities between genes, grouping them together according to these features. However, clustering algorithms do not provide the exact relationship among a group of genes and hence are not self-contained methods for the reconstruction of the gene regulatory network. In practice, they are very useful when combined with other methods as a pre-processing stage: once groups of closely related genes are identified, the network underlying these genes may be identified by the application of other modeling frameworks.

Bayesian networks, Boolean networks and neural networks also differ between themselves in respect to the type of information recovered from data. While Boolean networks provide a discrete and deterministic description, neural networks are continuous and adaptive systems, being specially suitable for capturing properties from domains with many parameters, interdependencies and uncertainty. This comes with a price tough: the complexity and the amount of data required by the learning algorithms are significantly higher in neural networks than in Boolean networks. Also, both modeling frameworks allow a temporal simulation of the biological system.

In contrast, Bayesian networks are limited for applications involving steady-state data, since its underlying graphical model, the directed acyclic graph, does not allow the occurrence of cycles. The niche occupied by Bayesian networks refers to the formulation of the domain's probability distribution, which are easily used to perform in silico prediction by computing the predictive distribution on the outcomes of possible actions, such as system interventions. The probabilistic nature of Bayesian networks is a favorable feature when dealing with missing and noisy data.

Given this brief review of the methods discussed, one can conclude that is difficult to assert that a particular method is better or worse for the inference of gene regulatory networks. All methods own relevant weaknesses and advantages: the decision about which one to apply depends on one's goal for the study. Furthermore, the method's performance is highly dependent on the scenario, on the target network's features and on the a priori knowledge available during the reverse engineering process.

Researchers have already suggested the integration of different types of biological data or even biological data collected under distinct experimental conditions as an interesting solution for the improvement of the model's accuracy. By joining data sets concerning different aspects of the same domain we provide further support for the inference process, which in turn will result in a richer computational model for the gene regulatory network. What was observed during this survey is that the natural course for the application of the reviewed modeling frameworks to this specific issue follows the same direction: the aggregation of distinct machine learning methods becomes a good strategy as it combines their particular strenghts and circumvents as much as possible their weaknesses. By joining efforts one would expect to improve the overall result of the inference process, forming more robust and accurate emsemble predictions.

REFERENCES

AKUTSU, T.; MIYANO, S.; KUHARA, S. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In: PACIFIC SYMPOSIUM ON BIOCOMPUTING, Hawaii. **Proceedings...** [S.l.: s.n.], 1999. v.4, p.17–28.

AKUTSU, T.; MIYANO, S.; KUHARA, S. Inferring qualitative relations in genetic networks and metabolic pathways. **Bioinformatics**, [S.l.], v.16, p.727–734, 2000.

ALI NAHVI, S. D. nad; GREEN, R. A Parsimonious Model for Gene Regulation by miRNAs. **Science**, [S.l.], v.331, p.550–553, 2011.

ARKIN, H. M. . A. It's a noisy business! Genetic regulation at the nanomolar scale. **Trends in Genetics**, [S.l.], v.15, p.65–69, 1999.

AULIAC, C.; FROUIN, V.; BUC, X. G. F. d'Alché. Evolutionary approaches for the reverse-engineering of gene regulatory networks: a study on a biologically realistic dataset. **BMC Bioinformatics**, [S.l.], v.9, 2008.

BAGIROV, A. M.; MARDANEH, K. Modified global k-means algorithm for clustering in gene expression data sets. In: WORKSHOP ON INTELLIGENT SYSTEMS FOR BIOINFORMATICS (WISB 2006), 2006. **Proceedings...** ACS, 2006. p.23–28. (CRPIT, v.73).

BALL, D. W.; HILL, J. W.; SCOTT, R. J. **The Basics of General, Organic, and Biological Chemistry**. [S.l.]: Flat World Knowledge, Inc., 2011.

BANSAL, M. et al. How to infer gene networks from expression profiles. **Molecular Systems Biology**, [S.l.], v.3, n.78, p.1–10, 2007.

BASSO, K. et al. Reverse engineering of regulatory networks in human B cells. **Nature Genetics**, [S.l.], v.37, n.4, p.382–390, 2005.

BERKHIN, P. **Survey of Clustering Data Mining Techniques**. [S.l.]: Accrue Software, 2002.

BROWN, T. A. **Genomes**. 2.ed. [S.l.]: Oxford: Wiley-Liss, 2002.

CHICKERING, D. M. Learning Bayesian networks is NP-complete. In: FISHER, D.; LENZ, H.-J. (Ed.). **Learning from Data**: Artificial Intelligence and Statistics V. [S.l.]: Springer-Verlag, New York, 1996. (Lecture Notes in Statistics).

EFRON, B.; TIBSHIRANI, R. **An Introduction to the Bootstrap**. [S.l.]: Chapman and Hall, London, 1993.

EISEN, M. B. et al. Cluster analysis and display of genome-wide expression patterns. **Proc. Natl. Acad. Sci. USA**, [S.l.], v.95, p.14863–14868, 1998.

FOGELBERG, C.; PALADE, V. Machine Learning and Genetic Regulatory Networks: A Review and a Roadmap. In: **Foundations of Computat. Intel.** [S.l.]: Springer-Verlag Berlin Heidelberg, 2009. v.1, p.3–34.

FRIEDMAN, N. et al. **Using Bayesian Networks to Analyze Expression Data**. [S.l.]: Hebrew University, 2000.

GEIGER, D.; HECKERMAN, D. **Learning Gaussian Networks**. Redmond, Washington: Microsoft Research, 1994. (MSR-TR-94-10).

GRZEGORCZYK, M.; HUSMEIER, D. Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. **Machine Learning**, [S.l.], v.71, p.265–305, 2008.

HACHE, H. et al. Reconstruction and Validation of Gene Regulatory Networks with Neural Networks. In: FOUNDATIONS OF SYSTEMS BIOLOGY IN ENGINEERING CONFERENCE (FOSBE 2007), 2. **Proceedings...** [S.l.: s.n.], 2007. p.319–324.

HACHE, H.; LEHRACH, H.; HERWIG, R. Reverse Engineering of Gene Regulatory Networks: A Comparative Study. **EURASIP Journal on Bioinformatics and Systems Biology**, [S.l.], v.2009, p.8:1–8:12, 2009.

HASTINGS, W. K. Monte Carlo sampling methods using Markov chains and their applications. **Biometrika**, [S.l.], v.57, p.97–109, 1970.

HAYKIN, S. **Neural Networks**: a comprehensive foundation. [S.l.]: Prentice Hall, 1998.

HECKER, M. et al. Gene regulatory network inference: Data integration in dynamic models – A review. **Biosystems**, [S.l.], v.96, p.86–103, 2009.

HECKERMAN, D. **A Tutorial on Learning with Bayesian Networks**. [S.l.]: Microsoft Research, 1995. (MSR-TR-95-06).

HUSMEIER, D. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. **Bioinformatics**, [S.l.], v.19, p.2271–2282, 2003.

IMOTO, S. et al. Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. In: IEEE COMPUTER SOCIETY BIOINFORMATICS CONFERENCE. **Proceedings...** IEEE Computer Society, 2003. p.104–113.

IMOTO, S. et al. Error tolerant model for incorporating biological knowledge with expression data in estimating gene networks. **Statistical Methodology**, [S.l.], v.3, n.1, p.1–16, 2006.

JIANG, D.; TANG, C.; ZHANG, A. Cluster Analysis for Gene Expression Data: A Survey. **IEEE Transactions On Knowledge and Data Engineering**, [S.l.], v.16, n.11, 2004.

KAUFFMAN, S. A. Metabolic stability and epigenesis in randomly constructed genetic nets. **J Theor Biol**, [S.l.], v.22, n.3, p.437–467, March 1969.

KOHONEN, T. Self-organized Formation Of Topologically Correct Feature Maps. **Biological Cybernetics**, [S.l.], v.43, p.59–69, 1982.

KOHONEN, T. et al. Self Organization of a Massive Document Collection. **IEEE Transactions On Neural Networks**, [S.l.], v.11, n.3, p.574–585, 2000.

LÄHDESMÄKI, H.; SHMULEVICH, I.; YLI-HARJA, O. On Learning Gene Regulatory Networks Under the Boolean Network Model. **Machine Learning**, [S.l.], v.52, p.147–167, 2003.

LI, L. et al. Computational approaches for microRNA studies: a review. **Mammalian Genome**, [S.l.], v.21, p.1–12, 2010.

LIANG, S.; FUHRMAN, S.; SOMOGYI, R. REVEAL, A Generak Reverse Engineering Algorithm for Inference of Genetic Network Architectures. In: PACIFIC SYMPOSIUM ON BIOCOMPUTING. **Proceedings...** [S.l.: s.n.], 1998. v.3, p.18–29.

LIKAS, A.; VLASSISB, N.; VERBEEKB, J. J. The global k-means clustering algorithm. **Pattern Recognition**, [S.l.], v.36, p.451–461, 2003.

MACLEAN, D.; JONES, J. D. G.; STUDHOLME, D. J. Application of ’next-generation’ sequencing technologies to microbial genetics. **Nature Reviews Microbiology**, [S.l.], v.7, p.287–296, 2009.

MADIGAN, D.; YORK, J. Bayesian graphical models for discrete data. **International Statistical Review**, [S.l.], v.63, p.215–232, 1995.

MARBACH, D. **Evolutionary Reverse Engineering of Gene Networks**. 2009. PhD thesis — École Polytechnique Fédérale de Lausanne.

MARDIS, E. R. Next-Generation DNA Sequencing Methods. **Annual Review of Genomics and Human Genetics**, [S.l.], v.9, p.387–402, 2008.

MAUCHER, M. et al. Inferring Boolean network structure via correlation. **Bioinformatics**, [S.l.], v.27, n.11, p.1529–1536, 2011.

MENDOZA, M. R.; BAZZAN, A. L. C. Evolving Random Boolean Networks with Genetic Algorithms for Regulatory Networks Reconstruction. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO’11). **Proceedings...** ACM, 2011.

MENDOZA, M. R.; WERHLI, A. V. Inferring Genetic Regulatory Networks with an Hierarchical Bayesian Model and a Parallel Sampling Algorithm. In: ELEVENTH BRAZILIAN SYMPOSIUM ON NEURAL NETWORKS, 2010. **Proceedings...** IEEE Computer Society, 2010. p.91–96.

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. **Journal of Chemical Physics**, [S.l.], v.21, p.1087–1092, 1953.

MILONE, D. et al. Analysis and Integration of Biological Data: a data mining approach using neural networks. In: **Knowledge Discovery Practices and Emerging Applications of Data Mining**: trends and new domains. [S.l.]: IGI Global, 2011. p.287–314.

MILONE, D. H. et al. *omeSOM: a software for clustering and visualization of transcriptional and metabolite data mined from interspecific crosses of crop plants. **BMC Bioinformatics**, [S.l.], v.11, 2010.

NARIAI, N. et al. Using protein-protein interactions for refining gene networks estimated from microarray data by Bayesian networks. In: PACIFIC SYMPOSIUM ON BIOCOMPUTING (PSB 2004). **Proceedings...** [S.l.: s.n.], 2004. p.336–347.

PEARL, J.; VERMA, T. S. A theory of inferred causation. In: SECOND INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING. **Proceedings...** [S.l.: s.n.], 1991. p.441–452.

PE'ER, D. et al. Inferring subnetworks from perturbed expression profiles. **Bioinformatics**, [S.l.], v.17, p.S215–S224, 2001.

PEROU, C. M. et al. Molecular portraits of human breast tumours. **Nature**, [S.l.], v.406, p.747–752, 2000.

REPSILBER, D.; LILJENSTROM, H.; ANDERSSON, S. G. Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses. In: **BioSystems**. [S.l.]: Elsevier, 2002. v.66, n.1–2, p.31–41.

SÁNCHEZ, A.; VILLA, M. C. R. de. **A Tutorial Review of Microarray Data Analysis**. 2008.

SCHLITT, T.; BRAZMA, A. Current approaches to gene regulatory network modelling. **BMC Bioinformatics**, [S.l.], v.8, n.S9, 2007.

SHMULEVICH, I.; DOUGHERTY, E. R. **Probabilistic Boolean Networks**: The Modeling and Control of Gene Regulatory Networks. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010.

SHMULEVICH, I. et al. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. **Bioinformatics**, [S.l.], v.18, n.2, p.261–274, 2002.

SHMULEVICH, I. et al. Inference of Genetic Regulatory Networks Via Best-Fit Extensions. In: ZHANG, W.; SHMULEVICH, I. (Ed.). **Computational and Statistical Approaches to Genomics**. [S.l.]: Springer US, 2003. p.197–210.

SWAIN, M. T.; MANDEL, J. J.; DUBITZKY, W. Comparative study of three commonly used continuous deterministic methods for modeling gene regulation networks. **BMC Bioinformatics**, [S.l.], v.11, n.1, p.459, 2010.

TAMADA, Y. et al. Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection. **Bioinformatics**, [S.l.], v.19, p.227–236, 2003.

TAMADA, Y. et al. Utilizing evolutionary information and gene expression data for estimating gene networks with Bayesian network models. **Journal of Bioinformatics and Computational Biology**, [S.l.], v.3, n.6, p.1295–1313, 2005.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Addison-Wesley, 2005. p.487–568.

TIAN, T.; BURRAGE, K. Stochastic Neural Network Models for Gene Regulatory Networks. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC'03), 2003. **Proceedings...** [S.l.: s.n.], 2003. v.1, p.162–169.

TÖRÖNEN, P. et al. Analysis of gene expression data using self-organizing maps. **FEBS Letters**, [S.l.], v.451, p.142–146, 1999.

VESANTO, J.; ALHONIEMI, E. Clustering of the Self-Organizing Map. **IEEE Transactions on Neural Networks**, [S.l.], v.11, n.3, p.586–600, 2000.

VOHRADSKY, J. Neural Model of the Genetic Network. **The Journal of Biological Chemistry**, [S.l.], v.276, n.39, p.36168–36173, 2001.

WERBOS, P. J. Backpropagation through time: what it does and how to do it. In: IEEE. **Proceedings...** [S.l.: s.n.], 1990. v.78, n.10, p.1550–1560.

WERHLI, A. V. **Reconstruction of gene regulatory networks from postgenomic data**. 2007. PhD thesis — School of Informatics of University of Edinburgh.

WERHLI, A. V.; HUSMEIER, D. **Bayesian integration of biological prior knowledge into the reconstruction of gene networks with Bayesian network**. [S.l.]: The Berkeley Electronic Press, 2007. v.6.

WERHLI, A. V.; HUSMEIER, D. Gene regulatory network reconstruction by Bayesian integration of prior knowledge and/or different experimental conditions. **Journal of Bioinformatics and Computational Biology**, [S.l.], v.6, p.543–572, 2008.

XU, R.; WUNSCH, D. C.; FRANK, R. L. Inference of Genetic Regulatory Networks with Recurrent Neural Network Models Using Particle Swarm Optimization. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, [S.l.], v.4, n.4, p.681–692, 2007.

YU, J. et al. Advances to Bayesian network inference for generating causal networks from obervational biological data. **Bioinformatics**, [S.l.], v.20, n.18, p.3594–3603, 2004.

ZAHA, A. et al. **Biologia Molecular Básica**. 3.ed. Porto Alegre - RS: Editora Mercado Aberto, 2003.