Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologi doktorsexamen i datalogi fredagen den 24 maj 2013 klockan 13.30 i Petrén Salen, Wargentinhuset, Nobels Väg 12A, Solna.

© Hossein Shahrabi Farahani, May 2013

Tryck: Eprint AB 2013
Abstract

Cancer is a multi-stage process resulting from accumulation of genetic mutations. Data obtained from assaying a tumor only contains the set of mutations in the tumor and lacks information about their temporal order. Learning the chronological order of the genetic mutations is an important step towards understanding the disease. The probability of introduction of a mutation to a tumor increases if certain mutations that promote it, already happened. Such dependencies induce what we call the monotonicity property in cancer progression. A realistic model of cancer progression should take this property into account.

In this thesis, we present two models for cancer progression and algorithms for learning them. In the first model, we propose Progression Networks (PNs), which are a special class of Bayesian networks. In learning PNs the issue of monotonicity is taken into consideration. The problem of learning PNs is reduced to Mixed Integer Linear Programming (MILP), which is a NP-hard problem for which very good heuristics exist. We also developed a program, DiProg, for learning PNs.

In the second model, the problem of noise in the biological experiments is addressed by introducing hidden variable. We call this model Hidden variable Oncogenetic Network (HON). In a HON, there are two variables assigned to each node, a hidden variable that represents the progression of cancer to the node and an observable random variable that represents the observation of the mutation corresponding to the node. We devised a structural Expectation Maximization (EM) algorithm for learning HONs. In the M-step of the structural EM algorithm, we need to perform a considerable number of inference tasks. Because exact inference is tractable only on Bayesian networks with bounded treewidth, we also developed an algorithm for learning bounded treewidth Bayesian networks by reducing the problem to a MILP.

Our algorithms performed well on synthetic data. We also tested them on cytogenetic data from renal cell carcinoma. The learned progression networks from both algorithms are in agreement with the previously published results.

MicroRNAs are short non-coding RNAs that are involved in post transcriptional regulation. A-to-I editing of microRNAs converts adenosine to inosine in the double stranded RNA. We developed a method for determining editing levels in mature microRNAs from the high-throughput RNA sequencing data from the mouse brain. Here, for the first time, we showed that the level of editing increases with development.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contents</strong></td>
<td>iv</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2 Biological Background</td>
<td>7</td>
</tr>
<tr>
<td>2.1 A brief history of modern genetics</td>
<td>7</td>
</tr>
<tr>
<td>2.2 DNA</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Genes and genomes</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Variations</td>
<td>10</td>
</tr>
<tr>
<td>2.5 What is cancer?</td>
<td>11</td>
</tr>
<tr>
<td>3 Computational Techniques</td>
<td>13</td>
</tr>
<tr>
<td>3.1 Bayesian networks</td>
<td>13</td>
</tr>
<tr>
<td>3.2 EM algorithm</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Structural EM</td>
<td>27</td>
</tr>
<tr>
<td>3.4 Branch and bound algorithm</td>
<td>28</td>
</tr>
<tr>
<td>4 Modeling cancer progression</td>
<td>33</td>
</tr>
<tr>
<td>4.1 Path and tree-based models</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Network-based models</td>
<td>36</td>
</tr>
<tr>
<td>5 A-to-I editing of microRNAs</td>
<td>41</td>
</tr>
<tr>
<td>5.1 microRNAs and their role in the cellular machinery</td>
<td>41</td>
</tr>
<tr>
<td>5.2 microRNA biogenesis</td>
<td>41</td>
</tr>
<tr>
<td>5.3 microRNAs editing</td>
<td>42</td>
</tr>
<tr>
<td>6 Description of the papers</td>
<td>43</td>
</tr>
<tr>
<td>Bibliography</td>
<td>47</td>
</tr>
</tbody>
</table>
List of Publications

• **Paper I:** Learning oncogenetic networks by reducing to MILP
  Hossein Shahrabi Farahani and Jens Lagergren.
  PLoS ONE 2013 (Accepted).

• **Paper II:** A linear programming approach for learning bounded treewidth Bayesian networks
  Hossein Shahrabi Farahani, Pekka Parviainen, and Jens Lagergren
  Manuscript.

• **Paper III:** A structural EM algorithm for learning hidden variable oncogenetic networks
  Hossein Shahrabi Farahani and Jens Lagergren.
  Manuscript.

• **Paper IV:** A-to-I editing of microRNAs in the mammalian brain increases during development
  Ylva Ekdahl*, Hossein Shahrabi Farahani*, Mikaela Behm, Jens Lagergren, and Marie Öhman
  Genome Research 22, 1477-1487 (2012).
  (*) These authors contributed equally to this work.
Acknowledgments

When I started my undergraduate studies in engineering, I used to look at biology as a science consists of memorizing some facts with no relation to mathematics. Years later, and quite by chance, I found out about modern molecular biology, evolution, and mathematical techniques for modeling biological systems. It seemed to me like a wonderland. Now, after some years of working in this wonderland, I know that new discoveries in biology — if one is lucky enough to make any — comes after a lot of sweating and hard work. I was lucky to have the best professor I have ever met as my guide through this journey. My thanks, first and foremost, goes to my supervisor Jens Lagergren for his constant support, his calm attitude in dealing with all sort of research problems, and his brilliant scientific ideas. Always when I was about to give up on a project and surrender, one of Jens’ marvelous ideas came to my rescue.

I would also like to thank all collaborators in various projects. Marie Öhman, Ylva Ekdahl, Gilad Silberberg, Hugh Salter, and Ying Sun, it was a pleasure working with you. I would like to specially thank Pekka Parviainen for his constant support and our many discussions about bounded treewidth Bayesian networks. I am also grateful to Pekka for reading this thesis. His comments contributed a lot in improving the thesis.

I was very fortunate to get to know many smart and interesting people during my doctoral studies. I specially thank Mattias Frånberg for his friendship, all the mean jokes and pranks we shared, and for our GWAS discussions, and Benjamin Auffarth, for his friendship and keeping my spirit high whenever I was down. Joel Sjöstrand, we started this PhD journey together, without your friendship this journey would have been very difficult. Ali Tofigh, the true perfectionist, your help made the start of my PhD a much smoother ride, and our Skype conversations from across the Atlantic were always a delight. Bengt Sennblad, your curiosity in everything scientific and our discussions about various topics over the years were very inspiring. Lars Arvestad, a very good teacher and scientist, I believe your bioinformatics course is the most well organized course I ever had. Erik Sjölund, thanks for your friendship and your help with various network and computer related issues, and of course for trying hard, but with no success, to make me take lessons as an opera singer. Afshin Ahmadian thanks for all nice discussions we had about science and politics, and for giving me advice in many practical matters in academic life. Peter
Andersson, thanks for the nice time over many lunches we had together. Your grumpy attitude toward everything non-scientific, like celebrating the new year, is very entertaining. Viktor Granholm, you are the perfect barista, thanks for all nice fika times.

I am also very grateful to Harriett Johansson, Birgit Ekberg-Eriksson, and Linda Kurdari for their excellent support in all administrative matters at CSC.

Especially during the last two years at SciLife lab I got to know many inspiring people with whom I shared many discussions about science and life over a coffee or drinking a beer. Not in an order I would like to thank Ikram Uddin, Luminita Moruz, Pelin Akan, Roman Valls, Jimmie Hällman, Oliver Frings, Thomas Schmitt, Erik Sonnhammer, Erik Hammar, Miguel Simões, Mikael Lindahl, Iman Kamali, Kristoffer Forslund, Kristoffer Sahlin, Awn Sayyed, Owais Mahmudi, Nemanja Nemo, Mehmoed Alam Khan, Raja Hashim Ali, Simon Sundling, Walter Basile, Bahram Amini, Frances Vega, Andreas Tjärnberg, Gabriel Östlund, Sanjit Roopra, Erik Aurell, Stefan Ek, Ino de Bruijn, Carlos Talavera-Lopez, Fredrik Salmen, Mohammad Pirmoradian, Mahya Dezfooli, Arman Ardalan, Lukas Käll, and Olof Emanuelsson.

Thank you also to my friends Wiktor Hubar, Rouzbeh Asssadi, Pooya Mohebbi, Samin Sallek, Falko Meisner, David Tingdahl, and Qi Zhe. I would like to specially thank Jana Mulacova for her friendship and support all these years.

And last but not least, my family deserves very special thanks for their constant support.
Chapter 1

Introduction

Cancer is a result of accumulation of different genetic mutations such as chromosomal aberrations.

Accumulation of the aberrations happens over a period of time and some earlier aberrations yield later aberrations by making the latter favorable to the tumor. In other words, at each stage of cancer progression the probability of the next set of events is determined by the existing aberrations. Learning the temporal order of the events in tumors is essential for understanding the disease. Unfortunately, the data obtained from assaying tumors are cross-sectional, i.e., the data do not contain information about the temporal order of the events. Developing realistic and scalable models for progression of cancer and algorithms for learning such models is the focus of this thesis.

A good model for cancer progression should have four important characteristics. First, the structure of the model should be as unrestrictive as possible. For example, trees are restrictive structures but Directed Acyclic Graphs (DAGs) are more general in this context.

The second property is monotonicity. In cancer progression the probability of occurrence of an event increases if a certain number of aberrations have happened before it. So, if all the events preceding a certain event have not occurred, that event has less chance to be introduced to the tumor. A good model should be able to take this property into account.

In recent years, high-resolution tumor data from high-throughput sequencing technologies has become available. Comparing to older cytogenetic data the number of variables that can be measured in each tumor has increased. So, as the third property, a good algorithm should be scalable and work on datasets with more variables.

The fourth property is about the issue of experimental noise in the biological data. A good model should be able to take this issue into account.

This thesis contains two models for cancer progression and necessary algorithms for learning them. The models are based on the Bayesian network framework. In
Paper I, a special class of Bayesian networks that is customized for modeling cancer progression is introduced. We call them Progression Networks (PNs). In learning PNs the issue of monotonicity is taken into consideration. Paper I also contains an algorithm, DiProg, for learning PNs. In DiProg, the problem of learning PNs is reduced to Mixed Integer Linear Programming (MILP). MILP is a NP-hard problem for which good heuristics exits.

An algorithm for learning Bayesian networks with bounded treewidth is introduced in Paper II. As it will be explained, this algorithm is essential for Paper III.

In Paper III the issue of experimental noise is addressed by introducing Hidden variable Oncogenetic Networks (HONs), and a structural EM algorithm for learning HONs is introduced. A substantial number of inference tasks is performed in each iteration of the structural Expectation Maximization (EM) algorithm in Paper III. Exact inference can be performed efficiently only on BNs with bounded treewidth. We utilized the algorithm for learning bounded treewidth BNs from Paper II in the M-step of the EM algorithm.

MicroRNAs are small non-coding RNAs involved in post-transcriptional regulation of mRNAs. Each microRNA has a set of mRNAs as its targets. Binding of a microRNA to its target mRNA results in gene silencing by translational repression or target degradation. It has been previously shown that microRNAs are subjected to adenosine-to-inosine (A-to-I) editing. Inosine in turn is read as guanosine (G) by the cellular machineries. In Paper IV, we used high-throughput sequencing data from mature microRNA obtained from the mouse brain to determine the level of A:G editing in three stages of development. For separating the genuine editing events from false positives, we developed the K-safety filter. Most false positives are known or unknown microRNAs with an A:G mismatch to the microRNA that we are determining its editing level. K-safety filter is capable of finding such false positives with a desired degree of stringency. In Paper IV, we show for the first time that A-to-I editing of microRNAs increases during development.

The outline of this thesis is as follows. In Chapter 2, some concepts from biology that are necessary for understanding this thesis are provided. Chapter 3 contains a review of the mathematical toolbox that is used in this thesis. Bayesian networks, the EM algorithm, and the branch and bound algorithm are introduced briefly in this chapter. Chapter 4 provides an overview of the existing algorithms for modeling cancer progression. In section 4.1 path and tree-based models, and in section 4.2 network-based models for cancer progression are reviewed. Chapter 5 contains a description of microRNAs, their function, their biogenesis, and a brief description of A-to-I editing mechanism. Finally, Chapter 6 contains a more detailed description of the papers that are included in this thesis.
Chapter 2

Biological Background

This chapter starts with a brief introduction of genetics and genomics and continues with an introduction of genetic variations such as structural variations and point mutations. Because the focus of this work is developing algorithms for modeling cancer progression, knowledge of different genetic variations and experimental techniques for measuring them is essential for understanding the content of the thesis. We conclude this chapter by a discussion about cancer.

2.1 A brief history of modern genetics

Gregor Mendel in mid 19th century made one of the earliest scientific attempts for understanding inheritance. Mendel performed hybridization experiments on garden peas [70, 71]. He used the color of the flowers of the pea plants as the subject of his study and deduced what we call Mendel’s principles of heredity. Mendel’s principles of heredity consist of two laws, which are called law of segregation and law of independent assortment. By crossing purebred white flower and purple flower plants, Mendel discovered that the offspring is not a blend of the parents, but it is only a purple flowered plant. He coined the term factors for the heredity units. One factor is always recessive and the other one is dominant. Factors are what we call genes today. The law of segregation states that each individual has two alleles of every gene. Each parent passes one allele of each of its genes to its offspring and the allele that is passed to the offspring is randomly selected. The dominant allele in the offspring defines the trait that is expressed. The law of independent assortments states that the gene for each trait is passed from parent to offspring independently of the gene for any other trait. Biologists in the 19th century were mostly trained for categorizing different forms of life. Mendel’s abstract and revolutionary idea of a gene was not appreciated by them and largely ignored and forgotten until the early 20th century. Ignoring Mendel’s work was mostly due to the fact that the physical entities corresponding to genes and their location in the living cells were unknown.
A few discoveries changed that notion. First, in 1905 Nettie Stevens and Edmund Wilson discovered that the gender of an individual is determined by special chromosomes \[84\]. This finding helped Thomas Morgan and his students to discover the physical location of the genes in the cell. They discovered that a cross between mutant white-eyed male fruit flies and normal red-eyed females results in normal red-eyed fruit flies in the first generation. In the second generation only male descendants have white eyes. They concluded that the gene for eye color is physically located on the X chromosome. For this discovery, Thomas Morgan was awarded with the Nobel Prize in medicine in 1933.

Morgan’s discovery revealed that chromosomes are the carriers of the genetic information, but the composition of the genetic material was still unknown. DNA was isolated as a major chemical of the nucleus by Friedrich Miescher in 1869. But by early 1900s, proteins were considered as carriers of the genetic information and the role of DNA molecules were largely ignored. It was about to change by discovery of the transforming principle in 1920s. By mixing a harmless strain of bacteria with dead virulent strain of the same bacteria, scientists found out that the harmless strain turns infectious. This happens by transfer of chemicals from the dead infectious strain to the harmless strain. Oswald Avery performed a set of experiments on the transforming principle in 1940s. In order to find out if the genetic material that moves between two strains of the bacteria consists of protein molecules or DNA, he treated the mixture with protein-digesting enzyme. After the treatment, the genetic material was still flowing between the two bacterial strains. But when he added the DNA-digesting enzyme the transfer of the genetic material stopped. Avery concluded that genes are made of DNA \[6\]. Avery’s results were later replicated by Hershey et al. in 1952 \[44\].

The 3-dimensional structure of DNA molecules was still not known. In 1953, James Watson and Francis Crick discovered the accurate structure of the DNA molecule \[90\]. For a more detailed discussion about the structure of DNA see section 2.2.

In 1961, Nirenberg and Mattaei \[74\] cracked the genetic code and discovered that DNA words are three letters long.

Francis Crick formulated the central dogma of molecular biology, which says DNA is a storehouse of the genetic information and RNA transfers this information to the cytoplasm where proteins are made \[23\]. In short, the central dogma says that information in the cell flows from DNA to mRNA and then to protein and never from a protein to a nucleic acid.

In 1980, Fredrick Sanger won the Nobel Prize in chemistry for inventing the Sanger sequencing technique for obtaining the sequence of DNA molecules. By invention of massively parallel sequencing techniques, it is possible to sequence the entire 6 billion bases of a human genome in a few days time.
2.2 DNA

As mentioned in section 2.1, DNA is the storehouse of genetic material. In all organisms, from simple prokaryotes to eukaryotes, DNA carries information about function and development of the organism. DNA molecules in eukaryotes are very long. In order to fit in the small space inside the nucleus, they are folded up into chromosomes.

As it is shown in Figure 2.1, DNA molecule has a twisted-ladder structure that is called double helix. Each side of the twisted-ladder structure is called a strand. Chemically, a DNA molecule contains two long chains, each composed of units called nucleotides. There are four types of nucleotides: Adenine (A), Guanine (G), Thymine (T), and Cytosine (C). A,C,T, and G are called bases. The sequence of these bases in an organism is called its genome.

2.3 Genes and genomes

As mentioned before the sequence of nucleotides in an organism is called its genome. Length of the genome varies in different organisms. Except for prokaryotes and lower eukaryotes, the genome size is not positively correlated with the morphological complexity of organisms. For example, some plant genomes are order of magnitude longer than the human genome.

Parts of the genome contain instructions about various cell functions. These parts are called genes. The cellular machinery translates the instructions encoded in a gene sequence to produced proteins, and proteins in turn perform the cellular functions.
Synthesizing of protein molecules from instructions in the genes consists of two steps. First, the instructions in the gene are transcribed into messenger RNA (mRNA). The same as DNA, mRNA is also a sequence of nucleic acids; however, there are two differences between DNA and mRNA, (1) mRNA has one strand, (2) in mRNA, Thymine (T) is replaced by Uracile (U). Copying information from the DNA to mRNA is called transcription.

The next step in synthesizing the proteins is called translation. In the translation step, the cellular machinery produces the protein molecules using the information in the mRNA.

Protein molecules are chains of amino acids. The sequence of amino acids in a protein is determined by the sequence of its corresponding gene. The amino acids are encrypted by the genetic code. There are 20 amino acids and each of them is encoded by 3 nucleotides in the genome.

2.4 Variations

The main focus of this thesis is modeling cancer progression. Cancer is a result of accumulation of different genetic variations such as point mutations and structural variations. In this section various genetic variations that can result in carcinogenesis are briefly discusses. Also, a number of experimental techniques for measuring such variations are reviewed.

Single nucleotide polymorphism

A Single nucleotide polymorphism (SNP) is a change in a single base in the genome that is present in more than 1% of a given population. The set of SNPs in an individual’s genome is an important factor for determining the response of the individual to certain drugs and susceptibility to genetic diseases and various forms of cancer. A single SNP can cause a Mendelian disease. More complex diseases are generally caused by a group of SNPs, rather than a single SNP. dbSNP is an online repository of genetic variations including SNPs [82]. As of June 2012, more than 187 million SNPs are listed for humans.

Structural variation

Variations that extend to a large region of an organism’s chromosome are called structural variations. There are different categories of structural variations.

Copy number variations (CNVs) are the largest category of structural variations. It consists of insertions, deletions, and duplications. CNVs are as important as SNPs in determining the difference between humans [8]. Changes in the copy number allows for more fine-tuning of expression level of the genes in the regions with variable copy number. The redundant additional copies of a gene can get new functions, while other copies fulfill the original function. This is a driving force for evolution. Many of CNVs are unfavorable. Some CNVs are involved in creation
2.5. WHAT IS CANCER?

and progression of cancer [35, 89]. Two main categories of experimental techniques are used for discovering CNVs. Older methods are based on cytogenetic techniques such as Fluorescent In Situ Hybridization (FISH) [5, 65], Comparative Genomic Hybridization (CGH) [54], and array comparative genomic hybridization [79, 77]. Recently more modern techniques based on next-generation sequencing are also used for measuring CNVs [73, 85, 60].

Experimental techniques for measuring genetic variations

In this section experimental techniques for discovering CNVs are briefly reviewed. The data that are used in this thesis are CNV data generated by such experimental methods.

In the CGH method, first test DNA and normal DNA are differentially labeled with fluorescence. Then the two differentially tagged DNA samples are hybridized simultaneously to normal chromosome spreads in chromosomal CGH experiments, or to a slide containing hundreds of thousands DNA probes, for array CGH experiments. Then using the epifluorescence microscopy the regions of insertion and deletion can be detected. In chromosome areas with an insertion or deletion there is a change in the ratio of two fluorochromes along the target chromosome.

There are limitations to the CGH method. CGH can only detect unbalanced chromosomal changes. For example CGH is unable to detect inversions and balanced reciprocal translocations, because the copy number stays unaffected by such variations. Chromosomal CGH can detect gains or losses of at least 5-10 Mb long. The resolution is improved in array based CGH.

CGH methods are widely used in measuring CNVs in cancer tumors.

2.5 What is cancer?

Survival of any organism depends on the cooperation between all its somatic cells. The ultimate aim is survival of the organism and not the single cell. So, for the cells, survival of the fittest is not the rule. Cells should sacrifice themselves for the sake of the organism. All cells are supposed to grow, divide, and eventually die in a programmed manner. If a set of mutations gives a cell a selective advantage to survive beyond the point that it was intended to, and the cell uses more resources than its neighbors, this cell can start a colony of the mutant cells. Over time, this colony can accumulate even more mutations and endanger the survival of the entire organism.

In short, a colony of cancer cells must have two properties and be able to pass these properties to their descendants: (1) they must defy the normal restraints on division and growth, and (2) they proliferate and move to other tissues.

Study of cancer cells shows that in many types of cancer, cells in a tumor share one or more aberrations in their DNA comparing to the cells in tissue around the tumor. This suggests that the cancer can be a result of somatic mutations. The
fact that mutagenesis-causing agents such as some chemicals or radiation also cause carcinogenesis strengthens this argument further.

Figure 2.2: The chromosomes 9 and 22 before and after the translocation [2].

Evidence also suggests that in most forms of cancer the malignancy starts from a single aberrant cell. In other words, all the cells in a tumor are descendants of a single cell that acquired a certain set of somatic mutations. For example, in Chronic Melogenous Leukemia (CML), the analysis of chromosomes of all patients revealed a translocation between the chromosomes 9 and 22 [25, 61]. Figure 2.2 shows this translocation. Sequencing the DNA at the site of translocations showed that in each patient these sites differed only by hundreds or thousands of bases. This shows, there is a possibility that CML starts from a single cell.

It is likely that more than one mutation are needed for starting the carcinogenesis. In an average lifetime, there are approximately $10^{16}$ cell divisions in a normal human body [4]. We are a viable organism, so most likely one mutation cannot start the tumorigenesis. Also, for most of cancers the prevalence of the disease is higher in older population. This fact also supports the idea that cancer is the result of gradual accumulation of various mutations rather than only a single mutation.

Not every mutation can cause cancer. For each type of cancer a certain group of mutations is essential for starting the oncogenesis. Generally, mutations in the genes that are involved in controlling cell division or growth can increase the risk of oncogenesis. Cancer causing genes are categorized into two major groups. If an excess activity of the gene increases the cancer risk, it is called proto-oncogene. The mutant overactive forms of proto-oncogenes are called oncogene. Genes whose loss of function or mutation can cause cancer are called tumor suppressor gene [4].
Chapter 3

Computational Techniques

This chapter contains an overview of the main computational techniques that are used in this thesis. Using Bayesian networks for modeling cancer progression is the center of this thesis. We start this chapter by introducing Bayesian networks and score-based methods for learning them. We have developed three algorithms. In Paper I, we reduced the problem of learning a Bayesian network with bounded number of parents to Mixed Integer Linear Programming (MILP). The software that we used for solving the MILPs, CPLEX, is based on a branch and cut methodology. Section 3.4 contains a brief description of the branch and cut algorithm. Algorithms for exact inference on Bayesian networks have running times that in the worst case are exponential in the size of the network. However, in Bayesian networks with a moralized graph with bounded treewidth, the inference is linear in the size of the network. In Paper II, we have developed a MILP for learning bounded treewidth Bayesian networks. Section 3.1 contains the definition of the treewidth and a brief review of inference in Bayesian networks. The above-mentioned algorithms are designed for learning Bayesian networks from complete data. In Paper III, we developed an algorithm for learning Bayesian networks from incomplete data. We used the Expectation Maximization (EM) algorithm in this algorithm. Section 3.2 contains an overview of the EM algorithm.

3.1 Bayesian networks

Introduction

Bayesian networks (BNs) [76] are a class of the probabilistic graphical models. The number of parameters in discrete and multinomial probability distribution increases exponentially with the number of variables. BNs offer a compact way for representing joint probability distributions. The compact representation of the joint probability distributions in the BNs is feasible by exploiting the conditional independencies between the variables. By using BNs we can often achieve a significant reduction in the number of parameters needed for representing a joint distribution.
A Bayesian network has two parts, structure and parameters. We can represent the structure of a BN with a directed acyclic hypergraph. Hypergraphs are a generalization of the graphs. Edges in graphs connect at most two vertices. Generalized edges in hypergraphs are called hyperedges and may connect more than one vertex. A hypergraph is a pair \( H = (V, E) \), where \( V \) is the set of vertices and \( E \) is a set of non-empty subsets of \( V \) that are called hyperedges. Notice in a simple graph \( \forall e_i \in E, |e_i| = 2 \). In a directed hypergraph each hyperedge \( e \) is an ordered pair \( e = (Pa(e), c(e)) \) where \( c(e) \) which is a single vertex is called the child and the set of the remaining vertices, \( Pa(e) \), is called the parents, and each vertex can be a child in exactly one hyperedge. When there is no cycle in a hypergraph, that hypergraph is called acyclic. An acyclic hypergraph is called a hyperDAG.

The parameters of a BN are the Conditional Probability Distributions (CPDs) of each variable given the variables corresponding to its parents.

Formally, a Bayesian network for a set of random variables \( Y = \{Y_1, \ldots, Y_n \} \) is a pair \((H, \Theta)\) where \( H = (V, E) \) is a hyperDAG. For each vertex \( v \in V \), there is an associated random variable \( Y_v \in Y \). \( \Theta \) is a set of CPDs associated with each random variable. For each hyperedge \( e \in E \), \( \Theta(e) \) is the CPD \( \Pr[Y(c(e))|Y(Pa(e))] \). In a Bayesian network \( B = (H, \Theta) \) the joint probability distribution over \( Y \) is

\[
\Pr[Y] = \prod_{e \in E} \Pr[Y(c(e))|Y(Pa(e))],
\]

and each variable is conditionally independent of its non-descendants, given its parents.

**Learning Bayesian networks from complete data**

Learning Bayesian networks consists of two tasks, learning the parameters of a given network structure and learning the structure.

There are two types of methods for learning the structure of the Bayesian networks: constraint-based methods and score-based methods.

Constraint-based methods [75, 83] are designed to find a Bayesian network structure that implies the independence constraints that matches those of the data. The problem with constraint-based methods is their sensitivity in independence tests. Failing even a single independence test can lead to learning an incorrect network.

In score-based methods [22, 43] a score, which is a measure of how well a BN describes the data, is assigned to the BN. Score-based methods learn a Bayesian network structure that represents the distribution of the data. Because the methods that are used in this thesis come from the score-based learning category, we discuss the score-based methods in more detail.

The aim of score-based learning algorithms is finding the BN structure among a set of network structures with the highest value of the scoring function. The space of the BN structures consists of a super-exponential number of structures, i.e., with \( n \) vertices there are \( 2^{O(n^2)} \) structures. Searching for the structure with the highest
score is NP-hard and heuristics techniques must be used for searching. In contrary to constraint-base methods, that rely on local and individual independence tests, score-based methods consider the entire structure at once. As explained before the main problem with score-based methods is the search problem.

In score-based methods, the choice of the scoring functions is one of the most important steps in designing the learning algorithms. Different scoring functions can result in learning different BNs. Depending on the problem at hand, we may need to learn a sparser or denser network. With the right choice for the scoring function we can achieve such goals.

The most obvious choice of scoring function is the likelihood function. The likelihood function is used for calculating the probability of the data given a certain model. Assume that we have a dataset \( D \) and we want to find a BN \( B \) with hyperDAG \( H \) as its structure and \( \Theta_H \) as its parameters, i.e., \( B = (H, \Theta_H) \). For a given hyperDAG \( H \) the likelihood score is defined as

\[
L((H, \Theta_H) : D) = \Pr[D : (H, \Theta_H)]
\]

Denoting the maximum likelihood parameters by \( \hat{\Theta}_H \), we have

\[
L((H, \hat{\Theta}_H) : D) = \max_{\Theta_H} L((H, \Theta_H) : D).
\]

So, for learning the Bayesian network \( B = (H, \Theta_H) \), we first calculated the maximum likelihood estimate of the parameter, \( \hat{\Theta}_H \), and then we search for a hyperDAG \( H \) with the highest likelihood score. The log-likelihood score is defined as

\[
\text{Score}_{ML}(H : D) = \mathcal{L}(\hat{\Theta}_H : D),
\]

where \( \mathcal{L}(\hat{\Theta}_H : D) \) is the logarithm of the likelihood function.

It is important to point out that the hyperDAG \( H \) is the best structure only if we use maximum likelihood score. Under a different scoring function, the network with the highest score most likely differs from \( H \).

As explained before, the choice of the scoring function directly affects the structure of the learned network. It will be shown later that using the maximum likelihood score in learning the structure of a BN results in learning a dense graph and it consequently overfits the data. One of the purposes of learning a BN is acquiring knowledge about new data that is not a part of the dataset from which the network is learned, but sampled from the same underlying distribution. Overfitting happens when the learned network does not perform well on the datasets different from the dataset that it is used for training. To investigate this issue further, we first need a few more definitions.

Mutual information between two discrete variables \( X \) and \( Y \) with instantiations \( x \) and \( y \), under a distribution \( P \) is defined as follows

\[
I_P(X; Y) = \sum_{x,y} Pr[x, y] \log \frac{Pr[x, y]}{Pr[x]Pr[y]}.
\]
Mutual information of two random variables is a measure of their dependence to each other.

Entropy of a variable $X$ under a distribution $P$ is

$$H_P(X) = - \sum_x Pr[x] \log Pr[x].$$

Entropy is a measure of uncertainty in a random variable.

It can be proven [59] that the maximum likelihood score can be decomposed as

$$\text{Score}_{ML}(H : D) = |D| \sum_{e \in E} I_P(X(c(e)); X(Pa(e))) - |D| \sum_{e \in E} H_P(X(c(e))), \quad (3.1)$$

where $|D|$ is the number of observations. The value of the second term in equation 3.1 is independent of the network structure, so for any two networks learned from the same dataset the value of this term is similar. We can ignore this term when we only intend to compare two structures.

It is easy to prove that adding an extra edge never decreases the mutual information. Assume variables $X, Y, Z$, and a distribution $P$, then the following always holds:

$$I_P(X; Y \cup Z) \geq I_P(X; Y). \quad (3.2)$$

Intuitively, adding an extra parent to a hyperedge never decreases the amount of information that parents provide about the child in the hyperedge. The only case in which adding the new parent, $Z$, does not increase the mutual information is when $Z$ is conditionally independent from $X$ given $Y$. Due to the noise in the empirical data, exact independence almost never holds. Using maximum likelihood score usually leads to learning a very dense graph that in many cases its underlying undirected graph resembles a complete graph. As explained before, the networks that are learned by the maximum likelihood score tend to overfit the data and do not generalize well with novel datasets. To address this issue, the concept of scoring function should be viewed from a Bayesian perspective.

In Bayesian approach, a distribution is used over anything with a degree of uncertainty. In learning Bayesian networks, there are uncertainties over both structure and parameters. We denote the structure and the parameter priors with $Pr[H]$ and $Pr[\Theta_H | H]$, respectively. Using Bayes’ rule, we have

$$Pr[H | D] = \frac{Pr[D | H] Pr[H]}{Pr[D]}. \quad (3.3)$$

The denominator in the equation 3.3 is only a normalization factor so we can ignore it. We define the Bayesian score as follows

$$\text{Score}(H : D) = \log Pr[D | H] + \log Pr[H]. \quad (3.4)$$
In equation 3.4 there is no term that let us apply a prior over the parameters. In calculating $Pr\left[D \mid H\right]$, the uncertainty over the parameters can be taken into consideration as follows

$$Pr\left[D \mid H\right] = \int_{\Theta} Pr\left[D \mid \Theta, H\right] Pr\left[\Theta \mid H\right] d\Theta.$$ (3.5)

The factors $Pr\left[\Theta \mid H\right]$ and $Pr\left[D \mid \Theta, H\right]$ in the equation 3.5 are refer to the priors over the parameters in a structure $H$ and the likelihood of the data $D$ given the Bayesian network $(H, \Theta)$ respectively. Because for calculating $Pr\left[D \mid H\right]$ we marginalize out all the parameters, we call it marginal likelihood. There is a very basic difference between the marginal likelihood and the maximum likelihood. While the maximum likelihood score is $\max_{\Theta} Pr\left[D \mid \Theta, H\right]$, the marginal likelihood is the average of $Pr\left[D \mid \Theta, H\right]$ weighted on the prior over the parameters $Pr\left[\Theta \mid H\right]$.

One of the motivations for introducing the Bayesian score is the tendency of the networks that are learned using the maximum likelihood score to overfit the data. Now we can discuss the issue of overfitting in more detail. The maximum likelihood score returns a set of parameters that maximize the likelihood function only for the training data. The set of parameters learned using the maximum likelihood score are useful if they fit the data in general and not only the training data. This situation is rare. That is why the networks learned using the maximum likelihood score tend to overfit the data. In contrary to the maximum likelihood score, the Bayesian score uses a range of parameters and integrates over them. Because the likelihood function is not strictly maximized for the training data and only one set of parameters is not picked, it is less likely that the model learned using the Bayesian score overfits the training data.

Schwarz [81] proposed the Bayesian Information Criterion (BIC) score. The BIC score of a model $H$ learned from a dataset $D$ is as follows

$$Score_{BIC}(H : D) = L(\hat{\Theta}_H : D) - \frac{\log M}{2} Dim(H),$$ (3.6)

where $L(\hat{\Theta}_H : D)$ is the logarithm of the maximum likelihood score of the model $H$ learned from dataset $D$, $Dim(H)$ is the number of the independent parameters in $H$, and $M$ is the number of samples.

Equation 3.6 guarantees a good balance between the complexity of the learned model and its fitness to the data. For complex models the value of $Dim(H)$ increases and reduces the overall score of the model. When there is a strong dependence between the child and the parents in the model, the value of $L(\Theta_H : D)$ increases and consequently the model gets a higher BIC score. The BIC score addresses the problems of overfitting better than the maximum likelihood score.

Another important concept in learning Bayesian networks with score-based methods is score decomposability. A score function is decomposable if the score for a Bayesian network $B = (H, \Theta)$ can be decomposed into sum of different terms, each for a single hyperedge $e \in E(H)$. In other words
Equations 3.6 and 3.7 show that both maximum likelihood score and BIC score are decomposable.

As mentioned before, in the score-based methods for learning the Bayesian networks, the aim of the learning algorithm is finding a structure with the maximum score. Using a decomposable score function is very important in developing efficient learning algorithms. When we use a decomposable score, changes that we make on a part of a structure, such as adding a new edge or removing an edge do not change the score of other parts of the structure.

Inference in Bayesian networks

In many applications only learning the Bayesian network from the data is not enough. After learning the network, depending on the application, we may need to calculate conditional probabilities from the network efficiently. This task is called inference. Both exact and approximate inference are \( NP \)-hard \([24, 21]\). Although if a property of the network called treewidth is bounded then, assuming a proper elimination order is known, inference is guaranteed to be tractable in the worst-case \([18, 63]\). A Bayesian network always represents a distribution. Each edge in a Bayesian network corresponds to a dependency between the random variables in the underlying distribution. A Bayesian network that is learned with additional condition of having a bounded treewidth rarely can represent all dependencies on the underlying distribution. Such a network only approximates the underlying distribution. In many applications the ability to perform fast inference on the learned network is so important that learning an approximate network with bounded treewidth is preferred.

In this section, first an example of a typical inference question is presented. Then we motivate the Variable Elimination (VE) \([92]\) algorithm with an example. Afterwards we present the general form of the VE algorithm. This section continues with a simple complexity analysis of the algorithm and in search for the network structures with a possibility of efficient inference, we introduce the concept of the treewidth.

An example of the type of queries that require performing inference on a Bayesian network is conditional probability queries. Assume that \( P \) is a probability distribution over a set of random variables \( S \). A conditional probability query consists of two parts: (1) the evidence \( E \) that is a subset of \( S \) and an instantiation \( e \) of the variables in \( E \), (2) a subset \( X \) of \( S \) which we call query variables. In conditional probability queries we want to find \( Pr[X|E = e] \). From the definition of the conditional probability we have

\[
Pr[X|E = e] = \frac{Pr[X, E = e]}{Pr[E = e]},
\]

(3.8)
3.1. BAYESIAN NETWORKS

To calculate the numerator and denominator in equation 3.8, we use marginalization. For the numerator, we marginalize all the variables in $S$ that are not in $X \cup E$. Assume $U = S \setminus X \setminus E$. For each instantiation of the numerator we have

$$Pr[x, e] = \sum_u Pr[x, e, u].$$

(3.9)

For calculating the denominator, we only need to sum out $x$ from the numerator, that is,

$$Pr[e] = \sum_x Pr[x, e].$$

(3.10)

One way for performing the calculations in the equation 3.9 is generating the joint distribution and sum out $u$. This approach for performing inference results in an exponential blowup.

In this thesis, the algorithm that is used for inference and calculating the CPDs for different queries is called the Variable Elimination (VE) algorithm. Before introducing the general form of the VE algorithm, we start with an example.

![Figure 3.1: Structure of a Bayesian network with five vertices](image)

Figure 3.1 shows a Bayesian network with five vertices. Say we want to calculate $Pr[E]$. The naive way is merely summing out the variables $A, B, C,$ and $D$ from the joint probability distribution $Pr[A, B, C, D, E]$, as follows

$$Pr[E] = \sum_D \sum_C \sum_A \sum_B Pr[A, B, C, D, E].$$

With $k$ variables that each of them can take $v$ values the number of the rows in the joint probability distribution table is $v^k$. So using the joint probability distribution table is not efficient. Instead, it is possible to take advantage of the factorization of the Bayesian network,

$$Pr[E] = \sum_D \sum_C \sum_A \sum_B Pr[D] Pr[E|C, D] Pr[C|A] Pr[B|A] Pr[A].$$

(3.11)
Before introducing an efficient way for calculating the sum in equation 3.11, we need new notations. The set of values that a random variable $X$ takes is denoted by $Val(X)$. For a set of random variables $D$, $Val(D)$ denotes the Cartesian product of the set of the values of all variables in $D$. We also need to define the concept of facto and factor marginalization.

**Factor:** Assume that $D$ is a set of random variables. A factor $f$ is a function from $Val(D) \rightarrow \mathbb{R}$. The members of $D$ are the domain of $f$ and denoted as $d(f)$.

**Factor marginalization:** Let $A$ be a set of random variables, $b \notin A$ a random variable, and $f(A, b)$ a factor. We call a factor $S(A)$ the factor marginalization or summing out of $b$ in $f$ if

$$S(A) = \sum_b f(A, b).$$

We frequently use a very important rule in this section. This rule allows us to exchange the order of summation and product. Assume we have two factors $f_1$ and $f_2$ and a variable $a$ that is not in the domain of $f_1$, i.e., $a \notin d(f_1)$, we have

$$\sum_a (f_1 \cdot f_2) = f_1 \cdot \sum_a f_2.$$ (3.12)

Equation 3.12 plays an important role in efficient computation of sums like the sum in equation 3.11. Because the domain of each factor is limited, equation 3.12 can be applied in order to push in the summations. By pushing in the summations, they only need to be computed over product of a subset of the factors. Equation 3.11 can be reorganized as follows

$$Pr[E] = \sum_D \sum_C \sum_A \sum_B f_1(D) f_2(E, C, D) f_3(A) f_4(C, A) f_5(B, A)$$ (3.13)

$$= \sum_D f_1(D) \sum_C f_2(E, C, D) \sum_A f_3(A) f_4(C, A) \sum_B f_5(B, A).$$ (3.14)

Equation 3.14 shows the basic idea behind the VE algorithm. Variables are summed out one at a time. Before summing out each variable, first all the factors that have that variable in their domains are multiplied. Then the variable is summed out from the factor that was generated in the multiplication step, this results to a new factor that in turn is added to the set of the factors. These steps are repeated for all the variables that are to be eliminated.

In equation 3.14, variables are eliminated in the following order, $B, A, C, D$. As we will see later in this section, the order of eliminating the variables plays an important role in the efficiency of our computations.

Algorithm 1 shows the VE algorithm in detail. The inputs of Algorithm 1 are a set of factors, $\mathcal{F}$, that are basically local CPDs in the BN, a set of variables $X$ that will be eliminated, and an ordering on the variables in $X$. The output of the algorithm is a factor $\mathcal{F}^*$. The domain of $\mathcal{F}^*$ does not include any of the random variables in $X$. 


Algorithm 1 VE algorithm

Input: \( F = \{f_1, \ldots, f_n\} \): a set of factors
\( X = \{X_1, \ldots, X_m\} \): a set of random variables that we want to eliminate
\( \prec \): an ordering on \( X \)

1: for \( i = 1, \ldots, m \) do
2: \( F' \leftarrow \{f \in F : X_i \in d(f)\} \)
3: \( F'' \leftarrow F \setminus F' \)
4: \( \pi_i \leftarrow \prod_{f \in F''} f \)
5: \( S_i \leftarrow \sum_{X_i} \pi_i \)
6: \( F \leftarrow F'' \cup \{S_i\} \)
7: end for
8: \( F^* \leftarrow \prod_{f \in F} f \)
9: return \( F^* \)

For eliminating each variable \( X_i \) in algorithm 1, first all the factors that have \( X_i \) in their domain, are multiplied. The result of this multiplication is a large factor \( \pi_i \). The domain of \( \pi_i \) is the union of the domains of all factors that are multiplied to make \( \pi_i \). Then the algorithm sums out \( X_i \) from \( \pi_i \). The resulting factor is \( S_i \) which its domain is the same as the domain of the factor \( \pi_i \) excluding the variable \( X_i \). Before proceeding further, we investigate the number of operations that are executed in Algorithm 1. In a Bayesian network with \( n \) variables, there are always \( n \) factors. In the worst case all variables but one, i.e., \( n - 1 \) variables, are to be eliminated. Let \( m(\pi_i) \) denote the number of variables in the domain of factor \( \pi_i \), and let \( N = \max_{i \in [n]} m(\pi_i) \) be the size of the domain of the largest factor where \([n] = \{1, \ldots, n\}\). When starting the algorithm, the set of factors, \( F \), has size \( n \). After eliminating a variable \( X_i \), a new factor \( S_i \) is generated in line 5 and is added to the set of factors \( F \) in line 6. So the total number of factors that enter \( F \) is \( n + (n - 1) = 2n - 1 \). Because each entry of a factor \( f \) multiplies only to one entry of \( \pi_i \), the cost of multiplication is at most \( (2n - 1)m(\pi_i) \leq (2n - 1)N = O(nN) \). There are two major operations in algorithm 1; multiplication of different factors in line 4 and marginalization of the variables in line 5. We already investigated the multiplications. The cost of the marginalization in a factor \( \pi_i \) is \( m(\pi_i) \) and at most \( n \max_{i \in [n]} m(\pi_i) \). So considering both factor multiplication and variable marginalization, the total amount of necessary operations is \( O(nN) \).

The motivation for designing algorithms like the VE algorithm is using the factorization of the joint distribution in a BN to avoid the exponential blowup while performing inference tasks. Although, this algorithm is one step forward from using the joint probability distribution in inference tasks, inside this algorithm there is still a source of exponential blowup. This source is factors \( \pi \). If \( \max_{i \in [n]} |\text{Val}(X_i)| = c \), then for a factor \( \pi_i \) with \( q \) variables in its domain, there are at most \( c^q \) entries or \( m(\pi_i) \leq c^q \). So the sizes of the intermediate factors are the major contributors to the computational cost of the VE algorithm. As it is shown,
the size of an intermediate factor grows exponentially with the size of the domain of the factor.

As explained before, the size of the largest factor in the variable elimination algorithm is the determining factor in the computational cost of the algorithm. So, the key to make this algorithm efficient is reducing the size of the factors in general and size of the largest factor in particular. Notice that in Algorithm 1 only inputs are the set of factors, \( \mathcal{F} \), and an elimination order, \( \prec \). We can assume that the algorithm works on an undirected graph.

Let \( \mathcal{F} = \{ f_1, \ldots, f_n \} \) be a set of factors. Let \( \mathcal{G}_\mathcal{F} = (\mathcal{V}, \mathcal{E}) \) an undirected graph, such that \( V(\mathcal{G}_\mathcal{F}) = d(\mathcal{F}) \) where \( d(\mathcal{F}) \) is the union of the domains of all factors \( f_i \in \mathcal{F} \). There is an edge \( (a, b) \in \mathcal{E} \) if and only if \( a \) and \( b \) are in \( d(f) \) for some \( f \in \mathcal{F} \).

We can easily conclude that the vertices in the graph \( \mathcal{G}_\mathcal{F} \) corresponding to the variables in the domain of each factor \( f \in \mathcal{F} \), form a clique.

In algorithm 1, for eliminating a variables \( X_i \), we first multiply all the factors that contain \( X_i \) to form a new factor, \( \pi \). Then we sum out \( X_i \) from \( \pi \) to get the factor \( S \). The domain of the factor \( S \) is the union of the domains of all the factors that have \( X_i \) in their domain excluding \( X_i \). In line 6 of Algorithm 1, this factor is added to the set of factors. We call this new set of factor \( \mathcal{F}_{X_i} \). The construction of factor \( S \) introduces new edges to the graph \( \mathcal{G}_\mathcal{F} \). If we denote the set of variables that appear with \( X_i \) in at least one factor as \( K_{X_i} \), then variables in \( K_{X_i} \) make a clique in \( \mathcal{G}_{\mathcal{F}_{X_i}} \). It is possible that some of the edges between the vertices in \( K_{X_i} \) do not exist in \( \mathcal{G}_\mathcal{F} \), such edges are called fill-in edges.

The union of all the undirected graphs, resulting from different stages of the variable elimination algorithm is called the induced graph. Note that, for a single Bayesian network structure, depending on the elimination order, we can get different induced graphs. So Bayesian network structure and the elimination order both play a role in determining the structure of the induced graph. Formally, the induced graph is defined as follows

**Induced graph:** Assume that \( \mathcal{F} \) is a set of factors over a set of variables \( X = \{ X_1, \ldots, X_n \} \), and that \( \prec \) is an elimination ordering over \( Y \subset X \). The induced graph \( I(\mathcal{F}_Y, \prec) \) is an undirected graph in which the variables corresponding to incidents of each edge in \( I(\mathcal{F}_Y, \prec) \) appear together in the domain of at least one of the intermediate factors \( \pi \) in the variables elimination algorithm.

It is easy to deduce from the definition of the induced graph that each maximal clique in the induced graph is corresponding to an intermediate factor in the variable elimination algorithm, and each factor in the variable elimination algorithm corresponds to a clique in the induced graph (for a proof see [59]).

Our main purpose is reducing the size of the largest factor in the variables elimination algorithm to avoid the exponential blowup while performing inference. This translates to finding an elimination order that minimizes the size of the maximal clique in the corresponding induced graph.

To formulate this problem, we need two new definitions. The width of an induced graph is the size of the largest clique in the graph minus 1. For a graph \( \mathcal{G} \) and an
ordering $\prec$ the induced-width is the width of the induced graph. The treewidth of a graph $G$ is its minimal induced width.

So, the question of finding an elimination ordering that minimizes the size of the largest intermediate factor, $\pi$, can be casted into the problem of finding an elimination ordering that results in a minimum treewidth in the induced graph. Bounding the treewidth of Bayesian networks is essential for being able to perform inference efficiently [18, 63].

A graph is chordal if it contains no induced cycle of length greater than three. It is easy to prove that every induced graph is chordal [59]. A graph $G = (V, E \cup F)$ is a chordalization of a graph $H = (V, E)$ if $G$ is chordal.

Using the definition of the chordalization of a graph and the fact that every induced graph is chordal, we can conclude that the induced graph that is the result of an elimination ordering on a BN is the chordalization of the undirected underlying graph of the BN. The treewidth of a graph $G$ is the minimum width over all chordalizations of $G$ [15].

There is an alternative definition for treewidth. This definition stems from the concept of the tree decomposition of a graph [80]. We denote an undirected graph by $G = (V, E)$, where $V$ is the vertex set and $E$ is the edge set.

A tree decomposition of $G$ is a pair $(X, T)$, where $X = \{X_1, X_2, \ldots, X_m\}$ is a collection of subsets of $V$ and $T$ is a tree on $\{1, 2, \ldots, m\}$, such that

1. $\bigcup_{i=1}^{m} X_i = V$,
2. for all edges $(u, v) \in E$ there exists $i$ with $u \in X_i$ and $v \in X_i$ and
3. for all $i, j$ and $k$, if $j$ is on the (unique) path from $i$ to $k$ in $T$ then $X_i \cap X_k \subseteq X_j$.

The width of a tree decomposition is defined as $\max |X_i| - 1$. The tree-width of an undirected graph $G$ is the minimum width over all tree decompositions of $G$.

In this work, we mostly use the definition of the treewidth based on the elimination ordering.

Learning Bayesian networks from incomplete data

3.2 EM algorithm

In section 3.1, the problem of learning Bayesian networks from complete data is discussed. In many real world applications the data is not complete. In some experiments, there can be samples in which values of a few variables in the dataset can be missing. For example, in a medical study in which each patient has to undergo a few tests such as a MRI scan, a blood test, and a fitness test, some patients may miss a test. There are also situations that some variables are hidden, for example we cannot observe a special phenomena directly, we only can observe a variable that is dependent on a hidden cause, which can be represented by a hidden variable.
CHAPTER 3. COMPUTATIONAL TECHNIQUES

When there are missing values or hidden causes, we say that the data is incomplete. The problem of learning Bayesian networks from incomplete data is more challenging than that of learning networks from the complete data. This section starts with a brief discussion about calculating likelihood of a dataset with missing values or hidden variables. This discussion aims at explaining the reason for hardness of learning Bayesian networks from an incomplete dataset. Before a formal presentation of the EM algorithm we discuss the intuition behind it. We conclude this section by introducing the structural EM algorithm.

The likelihood function for incomplete data

In this section we address the particular difficulties that arise when calculating the likelihood score for the data with missing or hidden variables. As explained in section 3.1, when we only deal with complete data, the decomposability of the likelihood score facilitates convenient calculation of ML estimated parameters.

Assume we have a dataset $\mathcal{D}$ with $N$ samples and a Bayesian network $B = (\mathcal{H}, \theta)$ over a set of variables $X$. $X_i$ and $x_i$ denote the observed variables and their values in the $i^{th}$ instance, respectively. Hidden or missing variables in the $i^{th}$ instance are denote by $Z_i$. If we marginalize all the hidden variables, the probability of the observed data, $L(\theta : \mathcal{D})$ is as follows

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{N} Pr[x_i | \theta]. \tag{3.15}$$

Equation 3.15 looks very similar to the likelihood equation for the case of complete data. It seems that similarly to the case with complete data, the decomposability of the likelihood function facilitates calculating the parameters. In reality, because each missing value or hidden variable can get more than one value, we effectively lose the decomposability of the likelihood function. In order to see this, we rewrite equation 3.15 as follows

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{N} Pr[x_i | \theta] = \prod_{i=1}^{N} \sum_{z_i} Pr[x_i, z_i | \theta]. \tag{3.16}$$

Equation 3.16 also illustrates another problem when calculating the likelihood function for incomplete data. In order to calculate $Pr[x_i, z_i | \theta]$, we need to perform inference for each instance. As explained in section 3.1 depending on the network structure, the inference may or may not be intractable.

In summary, computing the likelihood functions from incomplete data is much more challenging compared with computing it from the complete data. With incomplete data we can not take advantage of decomposability and closed form representation of the maximum of the likelihood function.

In ML parameter estimation from incomplete data, we want to find the values $\hat{\theta}$ that maximizes the log-likelihood function, $\hat{\theta} = \arg \max_{\theta} \mathcal{L}(\theta : \mathcal{D})$, where $\mathcal{L}(\theta : \mathcal{D})$
3.2. EM ALGORITHM

is the logarithm of $L(\theta : D)$. As explained before, the likelihood function does not decompose due to incomplete data. So we need to use techniques that are developed for maximizing non-linear functions. There are two major techniques for maximizing such functions, gradient ascent and Expectation Maximization (EM). The gradient ascent algorithm is a general optimization technique while EM algorithm especially designed for optimizing the likelihood function. Because in this work we used the EM algorithm, it will be our focus in this section.

When the complete data are available, we can maximize the likelihood by gathering sufficient statistics for each CPD. In the presence of missing values of hidden variables such approach cannot be used. Simple tactics such as filling in all missing values with a default value or filling missing values randomly from a distribution are not very efficient, because the learned parameters will be biased. While learning parameters from incomplete data, we are solving two problems simultaneously, learning the parameters and estimating the values for the missing variables. If either parameters or values for the missing variables are given, the other one can be easily computed. Having values for missing variables, maximum likelihood parameters for the complete data can easily be computed. Having the maximum likelihood estimates, we can use the inference techniques to compute the likely values for the unobserved variables. The EM algorithm constitutes a solution to this problem. Different forms of this method were in use for a long period. The current formulation of the EM algorithm first published by Dempster and colleagues in 1977, see [26]. In [72], Meng and van Dyk gave a historical account of the EM algorithm.

The EM algorithm first initializes the parameters. These initial values for the parameters can be either random or be chosen carefully according to the problem at hand. After the initial assignment of the parameters the algorithm repeats the following two steps: (1) using the current parameters and inference algorithms estimates the missing variables, or in other words, completes the data, (2) uses the complete data from step 1 to estimates new values for the parameters. We continue with derivation of the EM algorithm.

$\theta_n$ denotes the estimates of the parameters in the $n$-th iteration of the EM algorithm. The observed data is denoted by $X$. The log-likelihood function is denoted as follows

$$
\mathcal{L}(\theta | X) = \log Pr[X | \theta].
$$

(3.17)

We denote hidden variables with $Z$ and an instantiation of them by $z$. We have

$$
Pr[X | \theta] = \sum_z Pr[X | z, \theta]Pr[z | \theta].
$$

(3.18)

The objective of the EM algorithm is maximizing the log-likelihood, $\mathcal{L}[\theta | X]$, so after the $n$-th iteration, we want to find a new set of parameters such that $\mathcal{L}[\theta_{n+1} | X] - \mathcal{L}[\theta_n | X] > 0$. Using equation 3.18, we have
\[ \mathcal{L}(\theta_{n+1}|X) - \mathcal{L}(\theta_n|X) = \log \sum_z \Pr[z|X, \theta_{n+1}] \Pr[z|\theta_{n+1}] - \log \Pr[X|\theta_n]. \quad (3.19) \]

Multiplying the first term on the right hand side of equation 3.19 by \( \frac{\Pr[z|X, \theta_n]}{\Pr[z|X, \theta_n]} \), we have

\[ \mathcal{L}(\theta_{n+1}|X) - \mathcal{L}(\theta_n|X) = \log \left( \sum_z \Pr[z|X, \theta_{n+1}] \Pr[z|\theta_{n+1}] \cdot \frac{\Pr[z|X, \theta_n]}{\Pr[z|X, \theta_n]} \right) - \log \Pr[X|\theta_n]. \quad (3.20) \]

Using Jensen’s inequality we have

\[ \log \left( \sum_z \Pr[z|X, \theta_n] \cdot \frac{\Pr[z|X, \theta_{n+1}] \Pr[z|\theta_{n+1}]}{\Pr[z|X, \theta_n]} \right) \geq \sum_z \Pr[z|X, \theta_n] \cdot \log \left( \frac{\Pr[z|X, \theta_{n+1}] \Pr[z|\theta_{n+1}]}{\Pr[z|X, \theta_n]} \right). \quad (3.21) \]

Using equation 3.20 and inequality 3.21, we have

\[ \mathcal{L}(\theta_{n+1}|X) \geq \sum_z \Pr[z|X, \theta_n] \cdot \log \left( \frac{\Pr[z|X, \theta_{n+1}] \Pr[z|\theta_{n+1}]}{\Pr[z|X, \theta_n]} \right) \]
\[ = \sum_z \Pr[z|X, \theta_n] \cdot \log \left( \frac{\Pr[z|X, \theta_{n+1}]}{\Pr[z|X, \theta_n]} \right) \quad (3.22) \]
\[ = \mathcal{E}_{z|X, \theta_n} [\log \Pr[X, z|\theta_{n+1}]] - \mathcal{E}_{z|X, \theta_n} [\log \Pr[z|X, \theta_n]]. \]

The first and second terms in equation 3.22 are called the Q-term and R-term, respectively,

\[ Q(\theta_n, \theta_{n+1}) = \mathcal{E}_{z|X, \theta_n} [\log \Pr[X, z|\theta_{n+1}]], \quad (3.23) \]
\[ R(\theta_n) = -\mathcal{E}_{z|X, \theta_n} [\log \Pr[z|X, \theta_n]]. \quad (3.24) \]

The objective is finding values for \( \theta_{n+1} \) that maximize \( \mathcal{L}(\theta_{n+1}|X) \). If we choose \( \theta_{n+1} = \theta_n \), from equations 3.22,3.23, and 3.24 we have

\[ \mathcal{L}(\theta_{n+1}|X) \geq Q(\theta_n, \theta_{n+1}) + R(\theta_n) \]
\[ \geq Q(\theta_n, \theta_n) + R(\theta_n). \quad (3.25) \]
3.3. STRUCTURAL EM

Now we show that $Q(\theta_n, \theta_n) + R(\theta_n)$ is actually $L(\theta_n | X)$. We want to prove the log-likelihood $L(\theta | X)$ can be written as the sum of two expectations,

$$L(\theta | X) = \log Pr[X | \theta] \sum_z Pr[z | X, \theta]$$

$$= \sum_z Pr[z | X, \theta] \log \frac{Pr[X, z | \theta]}{Pr[z | X, \theta]}$$

$$= E_{z|X,\theta} [\log Pr[X, z | \theta]] - E_{z|X,\theta} [\log Pr[z | X, \theta]].$$

Using equation 3.26, we can substitute $Q(\theta_n, \theta_n) + R(\theta_n)$ by $L(\theta_n | X)$ in equation 3.25. Then we have

$$L(\theta_{n+1} | X) \geq L(\theta_n | X).$$

Equation 3.27 shows that the log-likelihood never decreases. In the EM algorithm we need to find a set of parameters that maximizes the $Q$-term.

The EM algorithm has two steps:

- **E-step:** The conditional expectation $E_{z|X,\theta_n} [\log Pr[X, z | \theta_{n+1}]]$ is computed in this step.

- **M-step:** In this step the conditional expectation from the E-step is maximized with respect to $\theta_{n+1}$.

The EM algorithm is only guaranteed to reach a stationary point. In most of the case the stationary point is a local maxima. Depending on the initialization the EM algorithm finds one of the local maxima of the expected complete log-likelihood. In real applications, different initializations can result in convergence of the algorithm to very different likelihood values. One solution to this problem is starting the algorithm with different initial values and after some iteration only keep the iteration with the best likelihood and continue only that iteration. For a discussion about convergence of the EM algorithm see [91].

3.3 Structural EM

So far we have discussed the problem of learning Bayesian network structure and parameters from complete data (section 3.1) and the problem of learning the parameters of a fixed network from incomplete data (section 3.2). As explained before, in learning Bayesian networks from complete data with score-based methods, we can take advantage of the decomposability of the likelihood or BIC score. This means if we add or remove an edge to a network, the score of the unaffected parts of the network remains the same. This property is important in designing efficient algorithms for structural learning.
For learning Bayesian networks from incomplete data, the likelihood score is not decomposable and we need to do inference for computing it. Also, for any candidate structure, we need to use EM algorithm for calculating the parameters for that specific structure. In contrary to the case with complete data, if we make a change in one part of the network, this change may also affect other parts of the network and the likelihood score must be calculated again with the EM algorithm. Due to the high cost of calculating parameters for a new structure, it is only possible to evaluate the score of relatively few structures.

Friedman \[37, 36\] discovered how to combine the EM algorithm that is used for parameter estimation with structure search. In other words, in each iteration of the EM algorithm a better structure is found. A closer look at the E-step and M-step of the EM algorithm shows why for learning BNs from incomplete data, the structural EM algorithm is more efficient comparing to the case in which the EM is only used for parameter learning for a fixed structure. In the E-step of the EM algorithm, the expected value of all the statistics necessary for evaluating the structure is computed. In the M-step, the parameters that are computed in the previous step are used for maximizing the score of the structure. The second step in the EM algorithm has no difference with learning from the complete data. In the structural EM algorithm, in every iteration, an effort is being made to learn a structure with better score.

### 3.4 Branch and bound algorithm

Mixed Integer Linear Programming (MILP) is widely used in this thesis. There are various commercial and non-commercial solvers for MILP problems. All MILP solvers use branch and bound or branch and cut algorithms. Branch and cut algorithm is a combination of branch and bound and cutting plane methods. In this section we describe these algorithms.

Branch and bound procedure is an algorithmic paradigm that is widely used in solving optimization problems. In a branch and bound algorithm a given problem continuously divided into smaller sub-problems until reaching solvable sub-problems. The global optimal solution is the best solution between the sub-problems. The main trick that is used in the branch and bound algorithm for excluding substantial parts of the search space is using previous estimates of the objective function. Branch and bound algorithm originally proposed by Land and Doig \[64\].

Algorithm 2 shows the general procedure in the branch and bound algorithm. The aim of the algorithm is minimizing a problem $P$. In line 1 of the algorithm, we define $S$ as a set of (sub)-problems. First we add $P$ to set $S$, then we solve $p_{\text{relax}}$, which is a relaxation of the original problem. If the original problem is a MILP or ILP, then the relaxation is a Linear Programming (LP) version of the original problem. This can be achieved by removing the integral condition from the integer variables. Solving the LP problem is substantially easier. If the solution of the relaxed problem is feasible for the original problem, then the original problem,
Algorithm 2 Branch and bound algorithm

Input: Minimization of problem $P$
Output: Optimal value $k^*$ and optimal solution $s^*$ or there is no feasible solution

$k^* = \infty$
1: Let $S = \{P\}$ and $\hat{k} = \infty$
2: if $S = \emptyset$ then
3: Return $s^* = \hat{s}$ and $k^* = \hat{k}$
4: Stop
5: end if
6: Pick $p \in S$
7: $S \leftarrow S \setminus \{p\}$
8: Solve $p_{relax}$
9: if $p_{relax} = \emptyset$ then
10: $\hat{k} = \infty$
11: else
12: Let $\hat{k}$ and $\hat{s}$ be optimal objective value and solution of $p_{relax}$
13: if $\hat{k} \geq \hat{k}$ then
14: Goto step 2
15: end if
16: end if
17: if $\hat{s}$ is feasible for $P$ then
18: $\hat{s} \leftarrow \hat{s}$
19: $\hat{k} \leftarrow \hat{k}$
20: Goto step 2
21: end if
22: Split $p$, i.e., $p = p_1 \cup \ldots \cup p_n$
23: $S \leftarrow S \cup \{p_1 \cup \ldots \cup p_n\}$
24: Goto step 2

$P$, is solved, otherwise we continue by dividing the feasible set of the original problem into smaller sub-problems and adding them to $S$ in line 23. Breaking a larger problem into two or more smaller problem in line 23 is called branching. Throughout execution of the algorithm a branching tree, with sub-problems $p_i$ as its nodes, is created (Figure 3.2). The branching tree is a rooted and directed tree. The root of the branching tree is the original problem, $P$, and its leaves are smaller problems that are either solved or are waiting to be picked up in line 6 of Algorithm 2.

As mentioned before, branch and bound algorithm tries to avoid complete enumeration of the solution space. This is achieved by bounding in line 13. Access to good lower and upper bounds are essential for recognizing the branches in the branching tree that will not lead to an optimal solution. Pruning such branches shortens the searching time. In a minimization problem solution to the relaxed LP
CHAPTER 3. COMPUTATIONAL TECHNIQUES

problem is a lower bound. For each sub-problem, \( p \), we first relax and solve it (line 12). If the value of the relaxed objective function, \( \hat{k} \), is larger than the best value of the objective function in any of the sub-problems (line 13) so far, we prune that sub-problem, and stop branching from its corresponding vertex in the branching tree.

![Figure 3.2: Search tree in the branch and bound algorithm](image)

Cutting planes

As explained before, when solving MILP or ILP problems with branch and bound algorithm, we use a relaxed solution of each sub-problem to obtain a lower bound for the integer solution. We can further tighten the search space by introducing some cutting planes. Assume \( \hat{s} \) is the solution of the LP relaxation of a sub-problem \( p \). Finding additional linear constraints with two properties (1) the current solution \( \hat{s} \) violates them, and (2) they do not cut into the convex hull of the feasible integer solutions, tightens the search space. Designing effective cutting planes can shorten the running time of the algorithm to a large extent. Figure 3.3 shows the schematic of the cutting process. The outer polygon is the boundary of the relaxed LP problem and \( \hat{s} \) is its solution. \( p_{int} \) is the convex hull formed by the integer points of the current sub-problem, \( p \). The cutting plane tightens the search space by cutting off the LP solution \( \hat{s} \). Note that the cutting plane does not cut any integer solution.

For the first time Gomory published an algorithm for finding such cutting planes [41, 40]. His algorithm is not useful in practice because it can generate an exponential number of cutting planes. Combination of branching and cutting can give the best performance. This strategy is the choice in most of the modern solvers such as CPLEX and SCIP [3]. It is commonly called as branch and cut algorithm. For
3.4. BRANCH AND BOUND ALGORITHM

Figure 3.3: A cutting plane cuts non-integer LP solution $\bar{s}$

A review of the cutting plane techniques in integer and mixed integer programming see [69].
Chapter 4

Modeling cancer progression

This chapter gives an overview of the different models of cancer progression and algorithms that are designed for learning them. The relevant models can be categorized in various ways such as probabilistic versus non-probabilistic and based on topology such as, path-based, tree-based, or network-based models.

Here we categorize the models according to their topology. This chapter starts with introducing the path-based models, which are the earliest attempts for modeling cancer progression. Due to complexity of cancer progression, when the disease reaches a certain point it can diverge to different paths. Path-based models are not able to capture such divergence events. In an attempt to model the divergence of the progression pathways, tree-based models were introduced. Different authors have proposed various algorithms for structural and parameter learning of such models.

Although, the tree-based models are more general than the path-based models, they cannot capture convergence of the cancer progression pathways. In order to solve this shortcoming in the tree-based models, various models with network structure are proposed. Learning the network-based models is generally a more challenging task than learning the tree-based models.

In Papers I and III, we propose two network-based models for cancer progression.

4.1 Path and tree-based models

Modeling progression of cancer as a multi-stage and discontinuous process is an idea that goes back to 1958 (see [34]), but due to lack of experimental techniques for obtaining quantitative measures of genomic alterations, it took decades until it became feasible to test such a hypothesis. Vogelstein and his pioneering work on colorectal cancer [33] is one of the earliest attempts to make a quantitative multi-stage model of cancer progression. The Vogelstein’s model is a path-based model containing a path of four genetic events. Although the Vogelstein’s model is solely biomedical, it inspired development of many mathematical models in recent years.
Analysis of CGH data from breast cancer by Kuukasjärvi et al. [62] proved that a linear progression model like Vogelstein model for the colon cancer cannot capture the progression pathways in the breast cancer. So, models beyond the path-based models are necessary.

In order to address the problem with the path-based models, Desper et al. [27] proposed the oncogenetic trees model. Before introducing the model, some definitions are necessary. A rooted direct tree is a triple $T = (V, E, r)$, where $V$ is a finite set of vertices, and $E$ is a set of directed edges. An edge is denoted by $e = (u, v)$, where $u, v \in V$. $r \in V$ is called the root of the tree. Each vertex $v$ has at most one incoming edge, except $r$ which has no incoming edge. A labeled tree is a tree in which a positive real number is assigned to each edge. We denote a labeled tree by $T = (V, E, r, \alpha)$ where triplet $(V, E, r)$ is a rooted directed tree, and for each $e \in E$, $\alpha(e) > 0$ is assumed to be a positive real number. If $0 < \alpha(e) \leq 1$, the rooted labeled tree $T = (V, E, r, \alpha)$ is called an oncogenic tree. In oncogenic trees $V$ is the set of all genetic events and $\alpha(e)$ is the probability that edge $e$ is in the tree. Also, assume that all such events are independent. Assume that in an experiment, we choose a set of edges each with probability $\alpha(e)$ and independently of each other. This experiment creates a sub-tree $T'$ of $T$. The set of vertices $S$ that are reachable from the root in $T'$ is the result of the experiment. An oncogenic tree, $T$, generates a probability distribution over the set of all subsets of $V$.

For each $S \subseteq V$, if $S$ contains the root, $r$, and $E'$ is a subset of $E$ such that all vertices reachable from $r$ in the rooted tree $(V, E', r)$ are in $S$, then

$$Pr[S] = \prod_{e \in E'} \alpha(e) \cdot \prod_{(u, v) \in E, u \in S, v \notin S} (1 - \alpha(u, v)),$$

and otherwise $Pr[S] = 0$.

One of the drawbacks of the oncogenetic tree model is that some observed tumors may get probability 0 and not characterized by the model. This issue arises from inability of the oncogenetic tree model to address the issue of noise, e.g., false positives and false negatives in real biological datasets. Szabo et al. [86] proposed a modified version of the oncogenetic tree model that takes false positives and false negatives into consideration.

For structural learning of the oncogenetic trees, Desper et al. [27] assign a weight $w_{ij}$ to each edge $e = (i, j)$. Then the structural learning problem can be reduced to the classical maximum weight branching problem that can be solved with Edmond’s algorithm [29, 87] in $O(|V|^2)$ time, where $V$ is the set of the genetic events.

Inspired by the similarities between evolution of species and progression of cancer, Desper et al. [28] proposed another tree model for cancer progression. The tree models in [28] are called distance-based trees. Distance-based trees are rooted and directed trees. In contrary to branching trees in [27], where every vertex is a genetic event, in distance-based trees only leaves represent genetic events. Internal vertices are considered to be hypothetical hidden events. In distance-based trees
4.1. PATH AND TREE-BASED MODELS

each pair of leaves is assigned a distance, which is computed from the dataset of genetic events such as CNAs. Learning a distance-based oncogenetic tree from a dataset means learning a tree whose leaves are the genetic events and each edge of a tree is assigned a length such that leaf to leaf distances in the tree is as close as possible to the computed pairwise distances between leaves. Learning a tree from the matrix of pairwise distances between its leaves is one of the most comprehensively studied problems in the phylogenetics. Desper et al. [28] used the Cavender [17] and Farris [32] model of evolution and the algorithm proposed by Farrach et al. [31] for learning such models.

We denote a distance-based tree by $T = (V, E, r, L)$ where quadruple $(V, E, r, L)$ is a branching oncogenetic tree. $L \subseteq V$ is the set of the leaves. A distance-based oncogenetic tree generates a distribution on $2^L$.

The main advantage of the distance-based tree model over the branching tree model in [27] is that every subset of the event $S \subseteq V$ has a positive probability. Because both the internal vertices and leaves in the branching trees [27] represent a genetic event, inferring the order of the events from the branching trees is easy. In distance-based oncogenetic trees only the leaves correspond to the genetic events, so inferring the order of events is not as straightforward as branching trees. A possible solution for this problem is to use the negative of the logarithm of the probability of each edge in a distance-based tree. As suggested in [28], adding these logarithms for all the edges from a leaf node to the root node gives a number, the larger the number the farthest away is the event from the root. Relative order between the events in the leaves can easily be inferred from their distances from the root.

Oncogenetic tree model [27] and distance-based oncogenetic tree model [28] are unable to accommodate converging progression pathways due to topological restriction of the tree. In order to remedy this problem, two categories of models are proposed, mixture of trees and network-based models. The mixture tree models are reviewed in this section. Next section covers the network-based models.

In order to address the problem of monotonicity and structural limitations of the oncogenetic tree model, Beerenwinkel and colleagues [14, 9, 78] proposed a mixture model of the oncogenetic tree model of Desper et al. [27]. In the mixture model in [14], one of the components is restricted to have star-like structure. This component captures the noise in the biological data. Star-like component in the mixture model enables it to assign non-zero probability to all combination of the genetic events, in contrary to the oncogenetic tree model in [27]. For learning the mixture of oncogenetic trees, an EM-like algorithm is used in [14]. This algorithm is not guaranteed to deliver a locally optimal maximum likelihood solution.

In another attempt to address this issue, Tofigh et al. [88] introduced the Hidden variable Oncogenetic Tree (HOT) model and a mixture of HOTs (HOT-mixtures). In HOTs each vertex in the tree represents an aberration. Hidden variables assigned to each vertex and probabilities that assigned to each edge in HOTs are used for modeling the progression of cancer. The variables in the data (the observations) are
modeled with a different set of random variables. Values of the observable variables are conditioned on the values of the hidden variables.

A HOT is defined by a structure and its parameters. Formally, a HOT $T = (T, \Theta_X, \Theta_Z)$ consists of a structure $T$ that is a rooted and directed tree and two conditional probability distributions, $\Theta_X$ and $\Theta_Z$. Two binary random variables are assigned to each vertex $v$ of a HOT, an observable random variable, $X(v)$, and a hidden random variable, $Z(v)$. The root is denoted by $r$ and $Z(r) = 1$. Parent of a vertex $v$ is denoted by $Pa(v)$. There are also two Conditional Probability Distributions (CPDs) assigned to each non-root vertex, $\theta_X(v)$ and $\theta_Z(v)$. $\theta_X(v)$ is a conditional probability distribution on $Z(v)$ conditioned by $Z(Pa(v))$, and $\theta_Z(v)$ is a conditional probability distribution on $X(v)$ conditioned by $Z(v)$. For modeling reasons, it is assumed in [88] that $\Pr[Z(v) = 1 | Z(Pa(v)) = 0] = \epsilon_Z$ and $\Pr[X(v) = 1 | \epsilon_Z = 0] = \epsilon_X$ where $\epsilon_Z$ and $\epsilon_X$ are two small numbers.

Tofigh et al. [88] derived a global structural EM algorithm for learning HOTs and their mixtures. The EM algorithm in [88] is global in the sense that it maximizes the expected complete log-likelihood in the EM algorithm rather than merely improving it. Tofigh et al. [88] made a very important observation, i.e., if two trees $T_1$ and $T_2$ have a common edge, then the weight of this edge is the same in both trees. This fact facilitates maximizing the $Q$-term in the EM algorithm. Defining a complete arc-weighted graph, $G$, Tofigh et al. casted the problem of finding the HOT to finding the maximum weight spanning tree of $G$. This problem can be solved using Edmond’s algorithm [29, 87] in quadratic time with respect to the number of variables.

4.2 Network-based models

As explained in the previous section, although the tree-based models constitute a step forward compared to the path-based models, they still impose topological restrictions. In reality, cancer progression pathways can diverge and converge. Tree-based models are only able to capture diverging progression pathways. Various attempts have been made for developing models that can learn Directed Acyclic Graphs (DAG) topology, instead being limited to the tree topology. In this section network-based models for cancer progression are reviewed.

Hjelm et al. [45] proposed the Network Aberration Model (NAM). NAM is a Markov chain model for accumulation of genetic events. Let $[n] = \{1, \ldots, n\}$ denote a set of genetic events. The set of events that have already occurred in the tumor is denoted by $Q$. The event of discovery of tumor is $S$. If $S$ happens at time $t_i$, no additional event happens afterwards. In a NAM, if a set $Q$ contains the events that have occurred and $S \notin Q$, then all events in $Q^c = ([n] \setminus Q) \cup \{S\}$ compete to become the next event. Figure 4.1 shows a Markov chain with 2 events. They made two assumptions: (1) the time until an event $x \in Q^c$ occurs in state $Q$, $T_{x|Q}$, is exponentially distributed with intensity $\Lambda_x(Q)$, and (2) independence of all $T_{x|Q}$.
4.2. NETWORK-BASED MODELS

Figure 4.1: A Markov chain with 2 aberrations [45].

when \( x \in Q^c \). Using the lack of memory in the system, Hjelm et al. [45] inferred the transition probability between states \( Q \) and \( Q \cup \{x\} \) to be the following

\[
p_{Q,Q \cup \{x\}} = \frac{\Lambda_x(Q)}{\sum_{y \in Q^c} \Lambda_y(Q)}.
\]  (4.2)

Because a Markov chain needs one parameter for each pair of states, for a Markov model with transition probabilities as equation 4.2 an exponential number of parameters must be calculated. To avoid an exponential blow up in the number of the required parameters, Hjelm et al. [45] in their NAM model introduced three sets of parameters, i.e., aberration intensities, pairwise dependencies, and stop intensities. Formally, a NAM \( M = (\lambda, \delta, \psi) \) has tree sets of parameters. Aberration intensities \( \lambda = \{\lambda_i : i \in [n]\} \) represent the intensity of each aberration in the starting state, pairwise dependencies \( \delta = \{\delta_{ij} : i, j \in [n], i \neq j\} \) in which \( \delta_{ij} \) shows how much intensity of aberration \( j \) changes if aberration \( i \) has happened before \( j \), and finally, \( \psi = \{\psi_i : 0 \leq i \leq n\} \) is a set of stop intensities. Hjelm et al. [45] used the definition of \( \lambda \) and \( \delta \) and assumed that \( \Lambda_j(Q ) = \Lambda_j(Q ).\delta_{ij} \). Then it easily follows that

\[
\Lambda_j(Q) = \lambda_j \prod_{i \in Q} \delta_{ij}.
\]  (4.3)

Equation 4.3 shows that instead the exponential number of parameters in a Markov chain, using the parameters of a NAM, only a quadratic number of parameters are necessary.

For learning a NAM, Hjelm et al. [45] used the maximum likelihood approach. The likelihood function is so complex that the authors in [45] resorted to a heuristics. Another possibility for learning a NAM can be an EM algorithm, but an efficient EM algorithm is not known yet.

Originally motivated by modeling the mutation patterns in HIV virus, Beerenwinkel and colleagues introduced the Conjunctive Bayesian Networks (CBN) model.
CBNs are a special case of Bayesian networks with very strict monotonicity. In a CBN an event cannot occur unless all its parents have occurred. Although the monotonicity property in CBNs is limiting from a modeling point of view, it simplifies the parameter and structure learning of the CBNs. There is a closed form solution for learning maximum likelihood parameters of the CBNs.

CBNs can be defined both using poset theory or as Bayesian networks. Here, we use the Bayesian networks terminology for defining the CBNs. A CBN is denoted by $B = (H, \Theta)$, where $H = (V, E)$ is a directed hypergraph which is the structure of the CBN, and $\Theta$ maps each hyperedge $e \in E$ to a CPD. For each vertex $v \in V$ there is an associated binary random variable $X(v)$. Every hyperedge $e \in E$ is assigned a CPD $\Pr[X(c(e)) | X(Pa(e))]$. For a subset $Y \subseteq V$ of random variables, $X(Y) = 1$ denotes that all variables in $Y$ have value 1.

The probability of an observation $X$ is

$$
\Pr[X|B] = \prod_{e \in E} \Pr[X(c(e)) | X(Pa(e))]
= \prod_{\substack{e \in E: \\
X(c(e))=1 \quad X(Pa(e))=1}} \theta_e 
\prod_{\substack{e \in E: \\
X(c(e))=0 \quad X(Pa(e))=1}} (1 - \theta_e) 
\prod_{\substack{e \in E: \\
X(c(e))=0 \quad X(Pa(e)) \neq 1}} 1 
\prod_{\substack{e \in E: \\
X(c(e))=1 \quad X(Pa(e)) \neq 1}} 0. \tag{4.4}
$$

The last term in equation 4.4 represents the hyperedges in which not all parents are 1 but the child is 1. This term enforces the monotonicity, that is, if in a sample for at least one hyperedge, the child vertex is present in the sample while not all the parents are present, then the probability of that sample is zero. In our monotone progression network (MPN) model in Paper I, instead assigning probability 0 to such hyperedges, we apply an upper bound $\hat{\alpha}$ on their probability, i.e., $\Pr[X(c(e)) = 1 | X(Pa(e)) \neq 1] < \varepsilon$. This provides more flexibility from a modeling point of view. There is a closed form solution for calculating the ML score for each hyperedge in the CBNs [12, 13].

Beerenwinkel et al. [10] also proposed a continuous-time version of the CBN model. In the continuous-time CBN model, aberrations accumulate with exponentially distributed waiting times. The same as the NAM model, in the continuous-time CBN model the system is observed at a stopping time, and all aberrations that have happened before the stopping event are recorded in the observation.

In order to address the observation errors, Gerstung et al. [39] extended the continuous-time CBN model to capture errors. The new model is called Hidden-CBN (H-CBN). In the H-CBN model, a small probability, $\varepsilon$, is assigned to false positives and false negatives that are events in which an aberration is misdiagnosed or unobserved, respectively. Gerstung et al. [39] devised an EM algorithm for estimating the maximum likelihood parameters of the model. Simulated annealing was used for learning the structure of H-CBNs.

The algorithms that are reviewed above are automated and can learn the cancer progression pathway with no need for manual curation by a human expert. Höglund
and colleagues [46, 51, 52, 49, 48, 38, 50, 53, 47] performed a series of studies for inferring progression pathways and temporal order of events from chromosomal aberration data in various types of tumors. They defined Time of Occurrence (TO) as a statistical measure that indicates how early or late a certain aberration occurs in a tumor. Then they performed Principal Component Analysis (PCA) on TO and correlation between the aberrations. Result PCA was grouped into clusters. Based on PCA result and TO, the clusters were formed into pathways by a human expert.
Chapter 5

A-to-I editing of microRNAs

This chapter starts with an overview of microRNAs and their function in the cellular machinery. Biogenesis of microRNAs is reviewed in section 5.2. Adenosine to inosine editing of microRNAs is one of the post-transcriptional regulatory mechanisms in the cell. Paper IV contains a study about editing of microRNAs in the mouse brain during development. We showed that adenosine-to-inosine editing increases during development. Section 5.3 contains a brief overview of microRNA editing.

5.1 microRNAs and their role in the cellular machinery

MicroRNAs are small non-coding RNA molecules with an approximate length of 22 nucleotides. MicroRNAs found in plants and animals. They are involved in post-transcriptional regulation of gene expression [19]. MicroRNAs regulate gene expression by binding to nearly complementary sequences in their target mRNA. Binding of a microRNA to a target mRNA results in downregulation of the corresponding gene of the mRNA by degrading the mRNA or by repressing its translation [7].

The sequence of an animal microRNA is partially complementary to the sequence of its target mRNA. It is shown that 6-8 nucleotides in the 5’ end of an animal microRNA, called seed region, is an important factor for target determination [67, 66]. Due to short length of the seed region, a microRNA may have more than one target mRNA, and a mRNA can be a target for more than one microRNA. In contrary to animal microRNAs, plant microRNAs have prefect or near perfect pairing with their target mRNAs [16].

5.2 microRNA biogenesis

MicroRNAs mature from primary microRNA transcripts (pri-miRNA). A pri-miRNA sequence contains several short inverted repeats that can form stem-loop structures
The stem-loop is recognized by DiGeorge syndrome critical region 8 (DGCR8) protein. DGCR8 together with the enzyme Drosha cut the stem-loop from the pri-miRNA. The product is an approximately 70 nucleotides long stem-loop precursor microRNA (pre-miRNA) [42]. After production of the pre-miRNA in the nucleus, proteins exportin-5 and RanGTP transport it to the cytoplasm [68].

After transportation to the cytoplasm, the pre-miRNA is cleaved by the enzyme Dicer. Together with dsRNA binding protein TRBP, Dicer cuts the loop joining the 3' and 5' arms of the pre-miRNA, transforms the pre-miRNA to a mature miRNA*-miRNA duplex. Two strands of the mature miRNA*-miRNA duplex are then separated. Each strand can function as a microRNA, but generally only one strand integrates in the RNA induced silencing complex (RISC) [20].

5.3 microRNAs editing

Editing is defined as a post-transcriptional modification that changes adenosine (A) to inosine (I) in RNA molecules. Inosine is recognized and guanosine (G) by the cellular machineries. The editing is performed by adenosine deaminase that acts on RNA (ADAR). Two ADAR enzymes are active in mammals, ADAR1 and ADAR2. Both ADAR enzymes act on double stranded RNA. It has been shown that A-to-I editing affects microRNAs [57, 56, 30]. Micro-RNA editing can influence the processing of pri and pre-miRNA [55], it also can change the target specificity of the microRNA [57].
Chapter 6

Description of the papers

Paper I

The genomes of cancer and healthy cells differ in various ways. Cancer genomes contain different types of aberrations, such as structural rearrangements, which accumulate in the cancer cell during the lifetime of the tumor. The set of aberrations in a tumor is revealed only after removal and subsequent assaying of the tumor. So, although cancer progresses and aberrations occur sequentially in time, the data obtained from assaying the tumor do not contain information about the temporal order of the aberrations. In other words, cancer data is cross-sectional. Learning the temporal order of the aberrations is very important for understanding the disease.

Disease progression is most likely a monotone process, that is, an aberration cannot occur until the aberrations that normally precede it in the temporal order already have happened. To better capture the monotonicity when modeling disease progression, we propose the Progression Networks (PNs) model. PN is a special class of Bayesian networks that is tailored for modeling cancer progression. We also developed a program, DiProg, for learning the PNs.

DiProg uses a score-base approach for learning Bayesian networks. As explained in section 3.1, in score-based methods and with a decomposable score function, a score is assigned to each parent-child set (hyper-edge). Then, the problem of learning BNs is reduced to searching for a structure with the highest score.

There are two types of PNs, monotone and semi-monotone. In Monotone PNs (MPNs), we apply an upper bound on the probability of the events in which the child is present but not all the parents are occurred in the tumor. In Semi-Monotone PNs (SMPNs), such upper bound applies on the probability of parent-child sets that the child presents in the tumor, but none of the parents have happened. The above-mentioned upper bounds penalize the corresponding scores of such parent-child sets.

In DiProg, the problem of searching for the structure with the highest score is reduced to Mixed Integer Linear Programming (MILP). There are very good
heuristics and efficient softwares available for solving MILPs. After calculating the scores and generating the MILP problem, DiProg used CPLEX to solve the problem.

DiProg is tested both on synthetic data and chromosomal aberration data from renal cell carcinoma.

Paper II

The algorithm in Paper I learns Bayesian and Progression networks with bounded number of parents for each hyper-edge. After learning a Bayesian or progression network from data, depending on the application, we may need to calculate conditional probabilities. This problem is called inference. Both approximate and exact inference are known to be NP-hard.

If the network has bounded treewidth then the inference is tractable. In Paper II, we present an algorithm based on MILP for learning bounded treewidth Bayesian networks. This algorithm is necessary for expanding the algorithm in Paper I for learning cancer progression networks. The DiProg algorithm in Paper I, learns Progression networks from complete data. In order to address the experimental noise issue, the algorithm in Paper III learns Bayesian networks from incomplete data. In each iteration of the structural EM algorithm in Paper III, we need to perform a substantial number of inference tasks. Such inference tasks can be performed efficiently only when the treewidth of the learned network is bounded. Algorithm in Paper II is essential for learning such bounded treewidth networks.

Paper III

In Paper I, the issue of the experimental errors is not taken into consideration. The algorithm in Paper I learns the progression networks from complete data. This is based on the assumption that the observed aberrations are the same as the existing aberrations in the tumor. In Paper III, we introduce the Hidden variable Oncogenetic Networks (HONs). In HONs, a hidden variable and an observable variable are assigned to each vertex. The hidden variable in each vertex, designates whether the cancer has reached to that vertex, the observable variable in the same vertex indicates the discovery of the aberration related to that vertex. Due to hidden variables, learning HONs is more complicated that learning the PNs from the complete data in Paper I.

For learning HONs, we devised a structural EM algorithm. In the E-step of the algorithm, we compute the Expected Complete Log-likelihood (ECL) and in the M-step of the algorithm we select the parameters and the structure that maximizes the ECL.

For estimating the parameters in the E-step, a substantial number of inference tasks should be performed. The time complexity of the exact inference in Bayesian networks increases exponentially with the treewidth. So, exact inference only can
be performed efficiently on networks with small treewidth. So, in the M-step, we need to learn a Bayesian networks with bounded treewidth, otherwise the next iteration of the algorithm cannot be performed efficiently. The algorithm in Paper II is used for this purpose. We tested our algorithm in synthetic data and real cytogenetic data from cancer tumors.

**Paper IV**

Adenosine-to-Inosine (A-to-I) RNA editing is a co- or post-transcriptional regulatory mechanism that changes adenosine to inosine in the double stranded RNA. Inosine is read a guanosine (G) by the cellular machineries.

We used high-throughput sequencing data from the mouse brain in three developmental stages and, for the first time, showed that A-to-I editing increases during development. A read with an A:G mismatch to a known microRNA can be a known or unknown microRNA which its sequence only has one mismatch to the first microRNA. The mismatch can also be originated from a sequencing error. Due to high level of false positives, devising a method for differentiating between genuine A-to-I editing sites and false positives is essential. For recognizing the false positives, we developed a method, K-safety filter. Our method can filter out the false positives with a desired degree of rigor. Results from applying our method on the data show the effectiveness of the K-safety filter.
Bibliography


47


[73] Ryan E Mills, Klaudia Walter, Chip Stewart, Robert E Handsaker, Ken Chen, Can Alkan, Alexej Abyzov, Seungtae Chris Yoon, Kai Ye, R Keira Cheetham, Asif Chinwalla, Donald F Conrad, Yutao Fu, Fabian Grubert, Iman Hajirasouliha, Fereydoun Hormozdiari, Lilia M Iakoucheva, Zamin Iqbal, Shuli Kang,


BIBLIOGRAPHY


