Likelihood-free methods for tumor progression modeling

Dissertation zur Erlangung des Doktorgrades
der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Physik

der Universität Regensburg

vorgelegt von

Daniela Herold

aus Wiesau

im Jahr 2014

Promotionsgesuch eingereicht am: 10.06.2014
Die Arbeit wurde angeleitet von: Prof. Dr. Elmar Lang

Daniela Herold Regensburg, 10.06.2014

Contents

Introduction

Cancer is a disease which is responsible for a large number of deaths worldwide. The number of new cases of cancer continues to increase every year [31, 45]. According to estimates of the GLOBOCAN project [31] there were 14.1 million new cancer cases and 8.2 million deaths from cancer in 2012. Among the reasons for the growing number of cancer incidents is an increased life expectancy in many parts of the world population [45]. Other factors which influence the cancer risk are changes in lifestyle, e.g. smoking, alcohol consumption, diet, obesity, but also external sources like infections, environmental pollution and radiation, both radioactive and ultraviolet, can contribute to the development of cancer [6].

Tumor progression is the process where cancer develops towards more and more malignant states. It is driven by the accumulation of mutations and epigenetic alterations in the DNA. In most tumor types progression is highly complex. Therefore, the precise order in which genetic alterations occur and how their occurrence depends on the presence of other genetic events is still unclear for many cancer types.

Fearon and Vogelstein [29] were the first who linked the stages of colorectal cancer with specific genetic mutations. They were able to describe the progression of the tumor by a linear path model. However, more complex models than a linear path are necessary to describe the progression in other tumor types [34].

Therefore, in the last decades researchers developed various models to describe tumor progression. Tree models [24, 25, 46, 75] for example generalize the sequential models of Fearon and Vogelstein [29] by allowing for various pathways in the process of accumulating mutations. However, tree models are still very restrictive because in tree graphs only single parental nodes are allowed. Conjunctive Bayesian networks (CBNs) [12, 13, 34] are a generalization of tree models, there multiple parental nodes are allowed in the associated graphs [34].

In the maximum likelihood estimation (MLE) of the CBN model network inference mostly relies on the evaluation of the likelihood function. However, in cases where

there is no closed-form solution for the maximization problem, likelihood-based approaches become computationally very intensive. In the case of the MLE of the CBN model for example, computational time and memory usage start to increase dramatically from a network size of 12 nodes.

In this thesis I propose two alternative methods for inferring tumor progression models and compare their performance to that of the MLE of the CBN model. The two methods avoid the calculation of a likelihood and rely instead on simulation. Artificial data are generated from networks which potentially explain the underlying biological mechanism of an observed genomic dataset $\mathbf{D}_0$. To capture the sample variability of the observed dataset, multiple artificial datasets are generated from each network. A search of the network space is performed while the simulated data are compared to the observed data at each iteration of the search.

The first method is based on machine learning and consists of three nested steps: *simulation*, *decision* and *searching*. In the innermost step, the simulation step, datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are simulated on the basis of two competing networks, the current network G_1 and the candidate network G_2. In the decision step a support vector machine (SVM) is trained on the data and a decision is made about which of the two networks describes the observed dataset $\mathbf{D}_0$ better. For global model fitting the classification approach is embedded in a model search algorithm where at each iteration of the model search a new candidate network is proposed.

The second method is similar. The difference lies in the decision step. The decision between the two competing networks G_1 and G_2 is made using a method based on approximate Bayesian computation (ABC). Datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are simulated from the two networks and the mean Euclidean distances $\bar{\rho}_1$ and $\bar{\rho}_2$ of their summary statistics to the summary statistics of the observed dataset $\mathbf{D}_0$ are computed. The candidate graph G_2 is accepted if $\bar{\rho}_2 < \bar{\rho}_1$. The ABC based method is also embedded in a model search approach for global network inference.

The performance and computational time of the two methods are compared to that of the MLE of the CBN model. In addition, both methods are applied to three real tumor datasets for network inference. For reasons of comparability the same three datasets are used as in [34].

Thesis organization

This thesis is divided into three chapters. In **chapter 1** a biological introduction is given. Sections 1.1 and 1.2 describe the biological background of cancer development. In section 1.3 a cytogenetic method for the analysis of specific chromosome changes is presented and in section 1.4 three cancer datasets are described.

Chapter 2 describes the methods used throughout this thesis. In section 2.1 some of the basic notations of graph theory are described. **Section 2.2** explains the conjunctive Bayesian networks. Support vector machines are introduced in **section 2.3** and **section 2.4** outlines the theory of approximate Bayesian computation. **Section 2.5** describes model search algorithms and in **section 2.6** a heuristic approach to noisy subset relations is presented.

In **chapter 3** the two methods developed in this thesis are explained. The simulation of artificial data is described in **section 3.1**. The algorithm and the results of the machine learning based method are presented in **section 3.2** and **section 3.3** gives a description of the ABC based method and the results.

Einleitung

Krebs ist eine Krankheit die für eine sehr große Zahl von Todesfällen weltweit verantwortlich ist. Die Zahl der neuen Krebsfälle steigt jährlich [31, 45]. Schätzungen des GLOBOCAN Projekts [31] zu Folge gab es im Jahr 2012 14.1 Millionen neue Krebsfälle und 8.2 Millionen krebsbedingte Todesfälle. Zu den Gründen für die steigende Zahl von Krebsfällen gehören die gestiegene Lebenserwartung in vielen Teilen der Weltbevölkerung [45]. Andere Faktoren die das Krebsrisiko beeinflussen sind veränderte Lebensgewohnheiten, z.B. Rauchen, Alkoholkonsum, Ernährung, Übergewicht, aber auch äußere Einflüsse wie Infektionen, Umweltverschmutzung und Strahlung, sowohl radioaktiv als auch ultraviolett, können zur Entwicklung von Krebs beitragen [6].

Tumorprogression nennt man den Prozess während dem sich Krebs in Richtung immer bösartigerer Stadien entwickelt. Er wird getrieben von der Anhäufung von Mutationen und epigenetischen Veränderungen in der DNA. In den meisten Tumorarten verläuft die Progression äußerst komplex. Deshalb ist die genaue Reihenfolge in der genetische Veränderungen erfolgen und wie deren Auftreten von der Gegenwart anderer genetischer Ereignisse abhängt bei vielen Krebsarten immer noch unklar.

Fearon und Vogelstein [29] haben als erstes die verschiedenen Stadien bei Darmkrebs mit bestimmten genetischen Veränderungen in Verbindung gebracht. Sie konnten die Progression des Tumors mit Hilfe eines linearen Pfadmodells beschreiben. Um die Progression in anderen Tumorarten zu beschreiben sind jedoch komplexere Modelle als das lineare Pfadmodell notwendig [34].

Aus diesem Grund entwickelten Forscher in den letzten Jahrzehnten verschiedene Modelle zur Beschreibung der Tumorprogression. Baummodelle [24, 25, 46, 75] zum Beispiel verallgemeinern die sequentiellen Modelle von Fearon und Vogelstein [29] dadurch, dass sie verschiedene Pfade im Prozess der Anhäufung von Mutationen erlauben. Baummodelle sind aber immer noch sehr restriktiv, weil in den entsprechenden Graphen ein Knoten nur jeweils einen Elternknoten besitzen darf. Conjunctive

Bayesian networks (CBNs) [12, 13, 34] sind eine Generalisierung der Baummodelle, in den zugehörigen Graphen sind mehrere Elternknoten erlaubt [34].

In der Maximum-Likelihood-Schätzung (MLE) des CBN Modells basiert die Netzwerk-Inferenz hauptsächlich auf der Berechnung der Likelihood-Funktion. Wenn das Maximierungsproblem jedoch nicht in geschlossener Form gelöst werden kann, werden Likelihood-basierte Methoden sehr rechenzeitintensiv. Im Falle der MLE des CBN Modells zum Beispiel steigen Rechenzeit und Speicherplatzverbrauch ab einer Netzwerkgröße von 12 Knoten drastisch an.

In dieser Doktorarbeit schlage ich zwei alternative Methoden zur Inferenz von Tumorprogressionsmodellen vor und vergleiche deren Ergebnisse mit denen der MLE des CBN Modells. Die beiden Methoden verzichten auf die Berechnung einer Likelihood und basieren statt dessen auf Simulation. Daten werden simuliert auf der Basis von Netzwerken, die möglicherweise den zugrundeliegenden biologischen Mechanismus in einem beobachteten genomischen Datensatz $\mathbf{D}_0$ erklären. Um die Variabilität verschiedener Stichproben des beobachteten Datensatzes zu erfassen werden mehrere Datensätze von jedem Netzwerk simuliert. Es wird eine Suche im Netzwerkraum durchgeführt, während der die simulierten Daten und die beobachteten Daten in jeder Iteration der Suche verglichen werden.

Die erste Methode basiert auf maschinellem Lernen und besteht aus drei ineinander verschachtelten Schritten: *Simulation*, *Entscheidung* und *Suche*. Im innersten Schritt, dem Simulations-Schritt, werden die Datensätze $\mathbf{M}_1$ und $\mathbf{M}_2$ auf der Basis zweier konkurrierender Netzwerke simuliert, dem aktuellen Netzwerk G_1 und dem Kandidaten-Netzwerk G_2. Im Entscheidungs-Schritt wird eine Support-Vektor-Maschine (SVM) auf den Daten trainiert und eine Entscheidung darüber gefällt welches der beiden Netzwerke den beobachteten Datensatz $\mathbf{D}_0$ besser beschreibt. Für die globale Netzwerksuche wird die Klassifikationsmethode in einen Netzwerk-Suchalgorithmus eingebettet, wo in jeder Iteration der Netzwerksuche ein neues Kandidaten-Netzwerk vorgeschlagen wird.

Die zweite Methode ist ähnlich. Der Unterschied liegt im Entscheidungs-Schritt. Die Entscheidung zwischen den zwei konkurrierenden Netzwerken G_1 and G_2 wird mit Hilfe einer Methode die auf ABC basiert gefällt. Datensätze $\mathbf{M}_1$ und $\mathbf{M}_2$ werden auf Basis der zwei Netzwerke simuliert und die mittleren Euklidischen Distanzen $\bar{\rho}_1$ und $\bar{\rho}_2$ ihrer Summary-Statistik zur Summary-Statistik der beobachteten Daten $\mathbf{D}_0$ werden berechnet. Das Kandidaten-Netzwerk G_2 wird angenommen wenn $\bar{\rho}_2 < \bar{\rho}_1$. Die

ABC basierte Methode wird ebenfalls zur globalen Netzwerksuche in einen Netzwerk-Suchalgorithmus eingebettet.

Die Ergebnisse und Rechenzeit der beiden Methoden werden mit denen der MLE des CBN Modells verglichen. Zusätzlich werden beide Methoden zur Netzwerk-Inferenz auf drei echte Tumordatensätze angewandt. Aus Gründen der Vergleichbarkeit werden die gleichen drei Datensätze wie in [34] verwendet.

Chapter 1

Biology

This chapter gives an introduction on the biological backgrounds of cancer development. Section 1.1 describes tumor progression as an age dependent process which requires the accumulation of multiple mutations. Section 1.2 presents a theory according to which cancer can be explained by a small number of underlying principles. Section 1.3 introduces a technique for the detection of chromosomal copy number changes, followed by section 1.4 which describes three tumor datasets used in this thesis to verify the simulation results.

1.1 The multi-step properties of tumor development

The development of a tumor can take years or even decades [29, 76]. Tumor progression is the process where cells undergo permanent and irreversible changes while transforming from normal, healthy cells into tumor cells [32, 56]. Several mutations and epigenetic alterations of the deoxyribonucleic acid (DNA), such as DNA methylation, are usually required [29] for a tumor to develop and the number of mutations varies between different cancer types. Wood et al. [78] investigated potential candidate cancer genes and found on average 15 and 14 mutated candidate cancer genes in colorectal and breast cancer, respectively. The increasing number of alterations of the DNA affects important functions of the cell such as proliferation and apoptosis so that unlimited growth of a tumor cell mass becomes possible [76].

Tumor progression happens in every healthy human simultaneously in many different parts of the body. However, most alterations of the DNA remain unnoticed. Cells maintain a series of barriers that lie between healthy cells and tumor cells [73, 76].

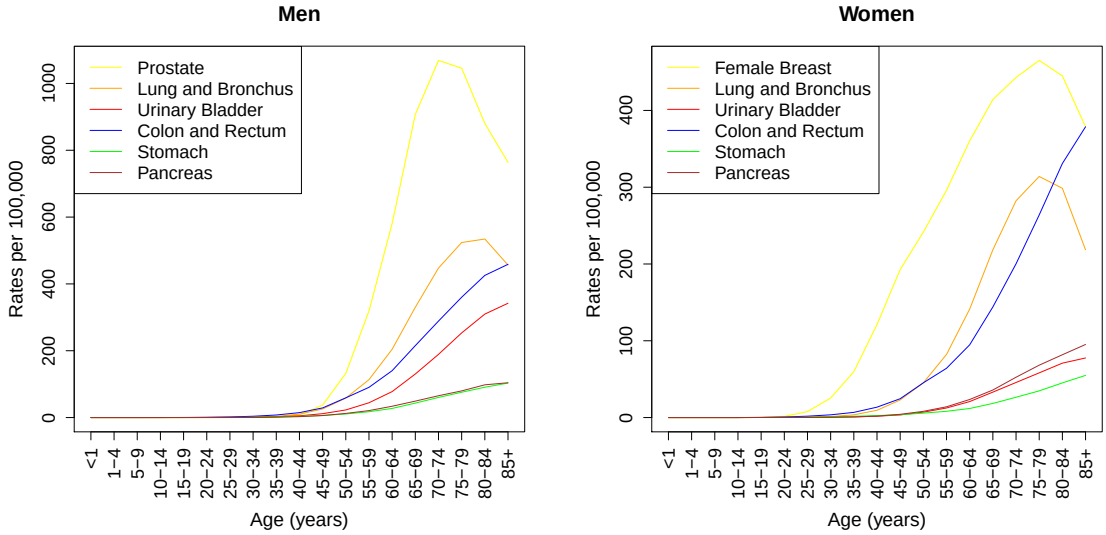

Figure 1.1: Age specific cancer incidence rates. The incidence of cancer grows steeply with increasing patient age. Data were obtained from the SEER Program of the US National Cancer Institute [1].

Among those barriers are both biological and biochemical defense mechanisms. Cells can be divided into stem cells and highly differentiated cells. The risk of DNA damage in the stem cells is minimized through two biological mechanisms: Stem cells divide only very rarely and the stem cell compartments protect them anatomically from external influences. The biochemical barriers include enzymes that destroy mutagens and DNA repair enzymes which find and repair structural changes in the DNA [76].

Epidemiologic studies, conducted e.g. by the US National Cancer Institute [1], showed that cancer incidence rates are highly age dependent. Figure 1.1 shows the age specific cancer incidence rates for American men and women for six different cancer sites. Data were collected between 1992 and 2010. The plots show that for both men and women the risk for cancer incidence rises steeply with increasing age. These data confirm that many cancer types that are frequent in adults develop over years or decades leading to the relatively high age of cancer incidence [76].

During the past several decades extensive evidence has been found which documents the accumulation of increasing numbers of mutations in cells during their development from healthy to malign cells [29, 56, 76]. This means that many of the steps in tumor progression can be associated with the accumulation of genetic alterations in the cells' genome. Colon cancer is very well studied because of the relatively good accessibility of the colonic epithelium. Therefore, the correlation between the accumulation of

mutations and an increasing malignancy of the cells has been studied most detailed for human colon carcinomas [29, 76].

As illustrated in figure 1.2 colon cancer develops in several distinct steps. Each step can be linked to a genetic change in the corresponding cells on the way to increasing malignancy. However, only a small number of all colon cancer patients obtain the mutations in the order shown in figure 1.2 [76]. While in about 90 % of all colon carcinomas the *APC* (adenomatous polyposis coli) gene on chromosome 5q is inactivated at a very early stage, only about 50 % have a K-*ras* mutation, show a loss of heterozygosity (LOH) on chromosome 17q involving *p53* and an LOH on chromosome 18q [29, 74]. The specific gene which is affected on chromosome 18q is not yet identified with certainty. Promising candidates are the *DCC* (deleted in colorectal cancer) gene and the *DPC4* (deleted in pancreatic cancer locus 4) gene [30, 62, 70].

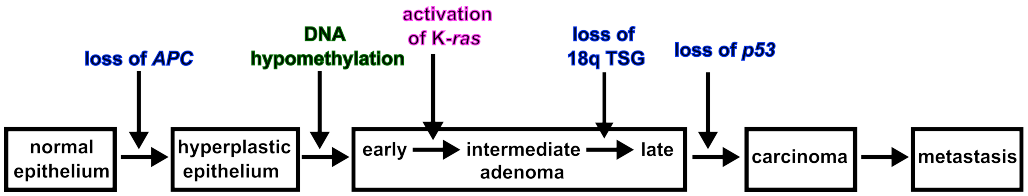

Figure 1.2: Tumor suppressor genes and the progression of colon cancer. Identified tumor suppressor genes (TSGs) are the *APC* gene on chromosome 5q and *p53* on chromosome 17p (blue). It is still unclear which TSG(s) are inactivated on chromosome 18q. About half of the colon carcinomas had a mutation on the K-*ras* gene (pink) and in most genomes of premalignant cells DNA hypomethylation had taken place (green). Figure reproduced from [76].

1.2 The hallmarks of cancer

In 2000 Hanahan and Weinberg [39] predicted that cancer research would develop into a logical science. This means that cancer could be explained by a small number of biological properties that are common to all cancer cells. The acquisition of such properties determines the development of cells into cancer cells. The authors suggest that these properties unbalance the cellular functions which are responsible for cell proliferation and homeostasis. They furthermore suggest, that all cancer-associated genes can be assigned to six alterations in cell physiology - the hallmarks - which all contribute to malignant growth:

1. **Self-sufficiency in growth signals**
   Normal cells can only start to proliferate when they receive mitogenic growth signals. Otherwise they stay in a passive non-proliferative state. This is in contrast to the behavior of tumor cells which are much less dependent on external growth signals. It has been shown for several human cancer types that tumor cells are able to synthesize stimulatory growth factors on their own [5, 36, 39] which makes them much less dependent on their environment.

2. **Insensitivity to antigrowth signals**
   In a healthy tissue, a cell can be kept in a quiescent state through antiproliferative signals. During the active proliferative phase healthy cells are receptive to signals from their environment and decide on the basis of these signals, if they go into a quiescent state (g_0 phase) or if they grow (g_1 and g_2 phase). Most antiproliferative signals are transmitted through the retinoblastoma protein (*pRb*) pathway. If this pathway is disrupted, E2F transcription factors are liberated and thereby cell proliferation is allowed even in the presence of antigrowth factors.

3. **Evasion of programmed cell death**
   The high cell proliferation rate is one of the main reasons for the enormous growth of tumor cell populations. But also the rate at which cell number decreases plays a major role. Apoptosis, i.e. programmed cell death, is primarily responsible for this decrease. There is evidence, e.g. from studies in mouse models, that *p53* plays an important role in the regulation of apoptosis which is in turn important barrier against tumor progression [14, 63, 67].

4. **Limitless replicative potential**
   The three acquired biological properties of growth signal autonomy, insensitivity to antigrowth signals and resistance to apoptosis allow cells to grow independently of external signals. It seems, however, that the acquisition of these properties is not sufficient to produce cells with unlimited growth. In addition to the signaling pathways between different cells, most cells have a specific intracellular program that controls their multiplication. When cells have divided for a certain number of times, they go into senescence, i.e. they stop dividing. Tumor cells must find a way to circumvent these limitations in number. By studies on cultured human fibroblasts it could be demonstrated that disabling of the *pRb* and *p53* TSGs allows cells to circumvent senescence and eventually achieve unlimited replicative potential [42].

5. **Sustained angiogenesis**
   In order to ensure the unlimited multiplication of tumor cells they must be able to create an *in vivo* environment where they can grow. This ensures that a sufficient amount of oxygen and nutrients is provided for proper cell function and survival. An important element for the creation of such an environment is the formation of new blood vessels. Tumor cells initially have no ability to trigger angiogenesis. There is extensive evidence that tumors must acquire angiogenic abilities to ensure further tumor growth [16, 38].

6. **Tissue invasion and metastasis**
   At some point during tumor development cells from the primary tumor go to adjacent tissues and eventually initialize the growth of a new tumor mass. The success of this new tumor colony depends of the presence of the other five acquired properties, as in the primary tumor. According to [65] it is not the enhanced cell proliferation rate, but rather tissue invasion and metastasis that are responsible for the many cancer deaths.

In 2011 Hanahan and Weinberg [40] further suggested that the acquisition of the six hallmarks is facilitated by two *enabling characteristics*:

1. **Genome instability and mutation**
   In order to acquire the above described hallmarks several alterations in the tumor genome are necessary. Cancer cells often have increased mutation rates [55, 61]. These are achieved because cancer cells are more sensitive to DNA damaging agents, because of a malfunction of parts of the DNA repair machinery, or both. The accumulation of mutations is additionally supported by damage in the systems that normally monitor genomic integrity. If the DNA can not be repaired these monitoring systems are responsible for killing cells with damaged DNA by triggering apoptosis or senescence, i.e. the cells cease to divide [43, 49, 64]. Genomic instability seems to be a property of the majority of human cancer cells. The defects in genome maintenance and repair must confer a selective advantage and thus support tumor progression, e.g. because they increase the mutation rates in premalignant cells [40].

2. **Tumor-promoting inflammation**
   Almost every tumor tissue contains cells of the immune system and thereby shows inflammatory conditions as they normally occur in healthy tissues [28]. The immune system might try to destroy the tumor cells by means of these

immune responses. Hence, in order for the tumor to escape immune destruction it must acquire new mutations. It has been repeatedly demonstrated that inflammation has the undesired effect of promoting tumor progression and thus helps tumor cells to acquire hallmark capabilities [20, 23, 37, 40, 59].

Other properties of cancer cells have been proposed to be important for tumorigenesis and might therefore be added to the list of cancer hallmarks. The two most promising among those are:

1. **Reprogramming energy metabolism**
   In order to maintain the enhanced proliferation in cancer cells their energy metabolism must be adjusted to ensure the cells' supply with nutrients. Normal cells produce their energy preferably through oxidative phosphorylation when aerobic conditions are given and only a small part of the cellular energy is obtained through the less effective glycolysis. Under anaerobic conditions the cells prefer energy production through glycolysis. Cancer cells are able to produce their energy mostly by means of glycolysis also under aerobic conditions. This state is termed "aerobic glycolysis". The reason why cancer cells switch to glycolysis is still partly unclear. According to a hypothesis by Potter [57], which was refined by Vander Heiden et al. [71], increased glycolysis is important in the formation of new cells and therefore supports active cell proliferation.

2. **Evading immune destruction**
   Research on immunodeficient mice suggests that the immune system plays an important role in preventing the formation of a tumor and its progression in non-virus induced cancer types [50, 69]. It could be observed that carcinogen-induced tumors grow more frequently and/or more rapidly in immunodeficient mice than in the immunocompetent controls. Similar results were obtained from transplantation experiments with mice. However, it is not yet clear to what extent antitumor immunity prevents the development of cancer in humans. In order to accept evading immune destruction as a core hallmark, more research will still be necessary [40].

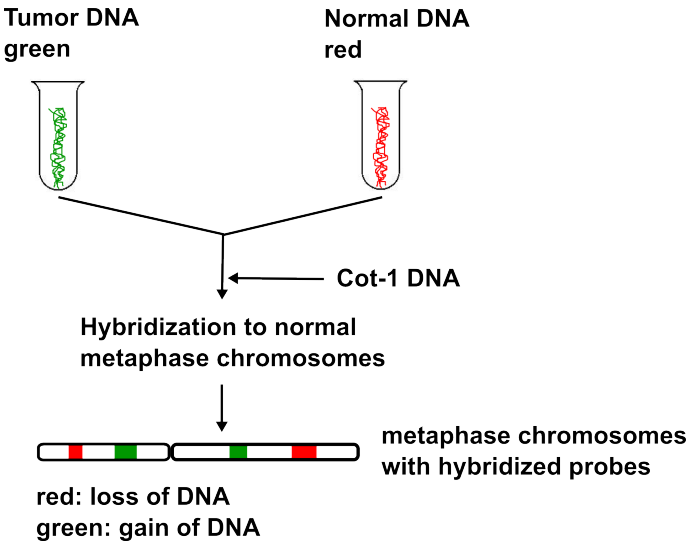

Figure 1.3: Comparative genomic hybridization. Tumor and reference DNA are labeled green and red, respectively. The differentially labeled DNA is mixed and Cot-1 DNA is added. The mix is then hybridized to normal metaphase chromosomes. Changes in copy number can be detected from shifts in the color, a shift to green means a gain, a shift to red represents a loss of chromosomes/chromosomal regions.

1.3 Comparative genomic hybridization

Comparative genomic hybridization (CGH) is a technique for the analysis of chromosomes which was first reported by Kallioniemi et al. [48] in 1992 and shortly after by du Manoir et al. [27]. It allows to find changes in the copy number of chromosomes without growing cell cultures [77]. CGH makes it possible to compare tumor and normal DNA and to detect chromosomal regions of gain or loss of DNA sequences in the tumor genome. With the help of these results one may then study the identified DNA regions in more detail using other molecular biological techniques in order to detect possible oncogenes or TSGs [77].

A schematic overview of the CGH technique is given in figure 1.3. Tumor DNA and normal reference DNA are labeled with two different fluorochromes - the tumor DNA in green, the normal DNA in red. The DNA is then mixed in a 1:1 ratio and put on normal human metaphase preparations where they hybridize to target chromosomes. A gain of chromosomal regions in the tumor DNA results in a higher green-to-red fluorescence ratio (> 1) in the concerned region, whereas a loss leads to a lower green-to-red ratio (< 1).

In their review, Weiss et al. [77] give a detailed description of the individual steps involved in the CGH technique and point out possible sources of error. After metaphase slides have been prepared and DNA has been isolated from the tissues, CGH involves the following steps:

1. **DNA labeling**
   The tumor and reference DNA is labeled and cut by means of nick translation which is a molecular biological tagging technique. In nick translation single-stranded DNA fragments ("nicks") are produced from the DNA with the help of DNase, a DNA degrading enzyme. Then DNA Polymerase I, an enzyme involved in DNA replication, is used to replace some of the nucleotides in the DNA with their labeled analogues [60]. In order to guarantee optimal hybridization, the fragment lengths of both test and reference DNA should be within the range of 500-1500 base pairs.

2. **Blocking**
   In many chromosomal regions there are short repetitive DNA sequences. These are particularly numerous at the centromeres and telomeres and some specific chromosome regions. The presence of these sequences can cause that some gains and losses remain undetected thus resulting in a reduced amplitude of the green-to-red ratio. Therefore, unlabeled Cot-1 DNA, i.e. placental DNA which is enriched for repetitive DNA sequences, is used to block the repetitive DNA sequences.

3. **Hybridization**
   For probe generation equal amounts of the labeled test and reference DNA are mixed and the Cot-1 DNA is added. Denaturation of the probe and the normal metaphase slides is done separately, then the probe is immediately added to the slides. The hybridization is left in a humid chamber at 40°C for two to four days. After hybridization the slides are washed, dried and colored with a blue-fluorescent stain for chromosome identification.

4. **Fluorescence Visualization and Imaging**
   Visualization of the green, red and blue fluorochromes is done using a fluorescence microscope. After capturing the digital images of the fluorescent signals, for each chromosome the ratios of several high quality metaphases are averaged and plotted along ideograms of the corresponding chromosome. This generates a relative copy number karyotype which presents chromosomal areas of dele-

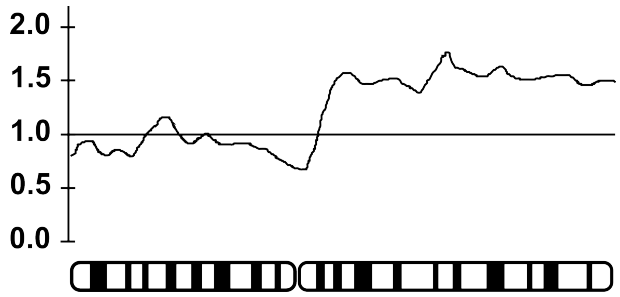

Figure 1.4: Relative copy number karyotype. The averaged green-to-red ratios are plotted against the chromosomal axis. Values greater than 1 represent a gain of chromosomal regions, values smaller than 1 stand for a loss.

tions or amplifications. An example is shown in figure 1.4. The ratio profiles can be interpreted in two ways using either fixed or statistical thresholds [77]. When using fixed thresholds, losses and gains are identified with the limits 0.75 and 1.25 or 0.85 and 1.15, while in the case of statistical thresholds the 95 % confidence interval (CI) limits of the ratio profile are taken. When using confidence intervals (CIs), gains or losses are present when the 95 % CI of the fluorescence ratio does not contain 1.0.

1.4 Tumor datasets

The performance of the methods developed in this thesis is verified using real tumor datasets. The same three datasets are used which were previously analyzed in [34] in order to compare the performance of these methods to that of the maximum likelihood estimation (MLE) of the conjunctive Bayesian network (CBN) model. The data were generated with CGH and were obtained from the Progenetix database [8]. Progenetix is a molecular-cytogenetic database which collects previously published chromosomal aberration data. A descriptive analysis of the Progenetix data can be found in [7].

Renal cell carcinoma

The first dataset is on renal cell carcinoma (RCC). Parts of this dataset (251 cases) have been published before in [46]. The most frequent gains in this dataset are: +5q(31) (25.2 %), +17q (21.2 %) and +7 (21.2 %). The most frequent losses are: –3p (59.4 %), –4q (29.9 %), –6q (25.5 %), –9p (24.4 %), –13q (23.1 %), –14q (17.9 %), –8p

(16.3 %) and –18q (14.7 %). As in [34] only the $N = 12$ copy number alterations (CNAs) used by Jiang et al. [46] are analyzed. They were selected using the method of Brodeur et al. [17], a statistical method to analyze structural aberrations in human cancer cells.

Among the CNAs listed above, the gain of chromosome 5p and the loss of 14q are not included. Instead, the gain of the Xp chromosome (9.6 %, often whole chromosome) is included as well as the gain of 17p (13.5 %). The dataset is reduced to the $N_P = 251$ cases (patients) published in [46].

Breast cancer

The breast cancer (BC) dataset consists of 817 cases and 10 CNAs. The most frequent (> 20 %) gains are: +1(q31) (59.7 %), +8(q23) (48.0 %), +17q (36.2 %), +20(q) (31.7 %), +16(p) (25.1 %), +11q13 (24.5 %) and +3q (22.4 %). The most frequent losses (> 20 %) are: –16(q) (29.0 %), –8p (27.8 %) and –13q (24.7 %) [34].

Colon cancer

The colorectal cancer (CRC) dataset consists of 570 cases and has 11 CNAs. The most frequent gains (> 20 %) in the CRC dataset are: +20(q13) (46.7 %), +13q (37.9 %), +8(q24), +7(q) (32.8 %) and +X(q24) (30.4 %). The most frequent losses are: –18(q22) (44.4 %), –8p(22) (34.2 %), –17p12 (25.3 %), –4(q) (23.3 %), –15q (19.2 %) and –1p (18.8 %) [34].

Each dataset is represented as a binary matrix $\mathbf{D}$ where rows are patients and columns are mutations or events. An element of the matrix is denoted as d_{ji}. Here, $i \in \{1, \ldots, N\}$ where N is the number of genomic events in the data and $j \in \{1, \ldots, N_P\}$ where N_P is the number of patients in the dataset. The j-th row in the matrix $\mathbf{D}$ is denoted as $\mathbf{d}_j = (d_{j1}, \ldots, d_{jN})$ and the i-th column is given by $\mathbf{d}_i^T = (d_{1i}, \ldots, d_{N_P i})^T$.

The data $\mathbf{D}$ are noisy, it is observed with both systematic and random errors which can be ascribed to the sampling and measurement procedures [66]. There are several different sources for the systematic error. One source of error lies in the process of DNA extraction. During the extraction of DNA e.g. from cancer cells one may not always succeed in perfectly separating the tumor cells from the surrounding tissue. Thus, the sample DNA might contain a small amount of DNA that actually comes

from healthy cells. Another possible source of error is intra-sample heterogeneity. The progression of the disease might be different among the sample cells, therefore resulting in a homogeneous sample where not all cells have identical alterations. A further potential source of systematic error are repeated DNA sequences that occur at more than one location in the genome. The repeated sequences may cross-hybridize therefore making it impossible to assign the measurements always to the correct genomic region. The random errors are caused by different kinds of measurement errors which lie in the experimental procedure [66].

Chapter 2

Methods

This chapter provides a description of the methods used throughout this thesis. Section 2.1 introduces some of the basic notations of graph theory and related terms which are used in the following sections. Section 2.2 explains the basics of conjunctive Bayesian networks (CBNs). Section 2.3 gives an introduction of support vector machines (SVMs) and section 2.4 describes approximate Bayesian computation (ABC). Both the SVM- and the ABC-based methods are combined with a number of searching algorithms which are discussed in section 2.5. The search is initialized using the heuristic approach of Jacob et al. [44] which is described in section 2.6.

2.1 Some basics of graph theory

The notations and definitions introduced in the following were taken from [26] and [35].

Graphs

A graph is a pair $G = (V, E)$ of sets where $E \subseteq [V]^2$. The notation $[V]^2$ means that the elements of E are subsets of V which consist of 2 elements. The elements of V are called the *vertices* or *nodes* of the graph G and the elements of E are called *edges*.

When drawing a graph, a node is usually depicted as a dot or circle and an edge between two nodes is represented by a line.

Subgraphs

Consider the graphs $G = (V, E)$ and $G' = (V', E')$. The graph G' is said to be a *subgraph* of G, written as $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$.

Paths and cycles

A *path* is a non-empty graph $P = (V, E)$ with nodes $V = \{v_1, v_2, \ldots, v_N\}$ and edges $E = \{v_1v_2, v_2v_3, \ldots, v_{N-1}v_N\}$ with the v_i being all distinct. The path P is also written as $P = v_1v_2 \ldots v_N$.

Given a path $P = v_1v_2 \ldots v_N$ with $N \geq 3$, a *cycle* is defined as $C = P + v_Nv_1$. The cycle C can also be denoted as $C = v_1v_2 \ldots v_Nv_1$.

Directed acyclic graphs

The graphs considered so far were undirected. A *directed* graph $G = (V, E)$ is a graph where every edge in E is an ordered pair of distinct nodes. When drawing a directed graph, an arrow normally marks the direction of an edge.

A directed acyclic graph (DAG) is a directed graph that does not contain cycles.

Adjacency matrix

The *adjacency matrix* $\mathbf{A}(G) = \mathbf{A}$ of a graph G with N nodes is an $N \times N$-matrix where an off-diagonal element $A(G)_{uv} = A_{uv}$ is equal to the number of edges from node u to node v, with $u, v \in V$.

In this thesis only DAGs are considered. In this case, the matrix $\mathbf{A}(G)$ is binary and for all off-diagonal elements we have that $A_{uv} = 1$ if there is a directed edge from u to v, and $A_{uv} = 0$ otherwise. Because there are no loops in DAGs, all the diagonal elements of $\mathbf{A}$ are zero.

Note that the adjacency matrix of a directed graph is not symmetric.

Transitive reduction

Following Aho et al. [3], the transitive reduction of a DAG G is defined as a smallest subgraph of G which contains a path from node u to node v whenever G contains a path from u to v. In other words, a DAG G^t is said to be a transitive reduction of G provided that

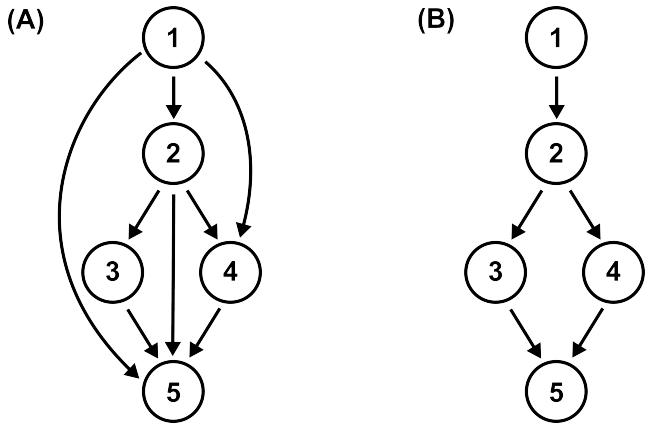

Figure 2.1: Transitive graph and transitive reduction. (A) Transitive graph. (B) Transitive reduction of the graph shown in (A).

(i) G^t has a directed path from node u to node v if and only if G has a directed path from node u to node v, and

(ii) there is no graph with fewer edges than G^t satisfying condition (i) [3].

Figure 2.1 (A) displays a transitive graph, the corresponding transitive reduction is shown in figure 2.1 (B). The transitive reduction of a finite DAG G is unique and is always a subgraph of G.

Intermediate states

An edge uw in a graph G is called a single intermediate state if there is exactly one node v, $v \in V \setminus \{u, w\}$, for which in the adjacency matrix of the graph we have that $A_{uv} = 1$ and $A_{vw} = 1$. To illustrate the single intermediate states, figure 2.2 (A) shows a subgraph of a graph G containing the nodes u, v and w.

An edge uw in a graph G is called a multiple intermediate state if there are two or more nodes v_i, $v_i \in V \setminus \{u, w\}$, $i \in \{1, \ldots, N-2\}$, for which in the adjacency matrix we have that $A_{uv_i} = 1$ and $A_{v_iw} = 1$. Figure 2.2 (B) gives an example of a multiple intermediate state. It shows a subgraph of a graph G that contains the nodes u, v_1, v_2 and w.

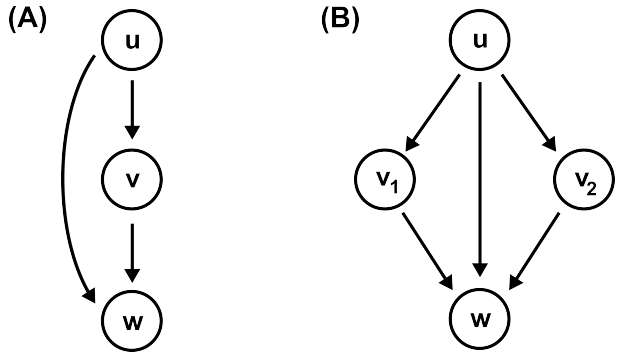

Figure 2.2: Intermediate states. (A) Single intermediate state. There is only one node, v, for which $A_{uv} = 1$ and $A_{vw} = 1$. (B) Multiple intermediate state. There are two (or more) nodes, v_1 and v_2, for which $A_{uv_i} = 1$ and $A_{v_i w} = 1$.

2.2 Conjunctive Bayesian networks

The most common way for learning the parameters of a statistical model is by analyzing the likelihood function. The optimal parameters for describing a specific dataset are found through maximization of the likelihood function. For models where the maximization can not be done in closed-form, parameter fitting is done via maximum likelihood estimation (MLE).

An example for such a model are the conjunctive Bayesian networks (CBNs) [11–13, 34]. Conjunctive Bayesian networks (CBNs) are probabilistic graphical models that can be used for describing the accumulation of events, e.g. genomic events. A CBN model is a specialization of a Bayesian network where an event can only occur if all of its parents have already occurred. CBNs are also a generalization of tree models [24, 25, 46, 75]. In contrast to oncogenetic tree models they allow for an event to depend on more than one predecessor event. Therefore, CBNs allow for an arbitrary partial order on the events which makes them able to model a larger range of biomedical problems than oncogenetic tree models.

A CBN can be used to model the accumulation of mutations in a tumor. CBNs assume that tumors progress along a transitively reduced DAG, where each node corresponds to a mutation event and parent nodes are mutations that typically precede a child mutation. CBNs are Bayesian networks that account for the variability and stochasticity of the timing of mutation events as well as of observation times. A graph of a CBN example is shown in figure 2.3. With every mutation i a random variable t_i, $i \in \{1, \ldots, N\} = [N]$ is associated which is the time it takes for the

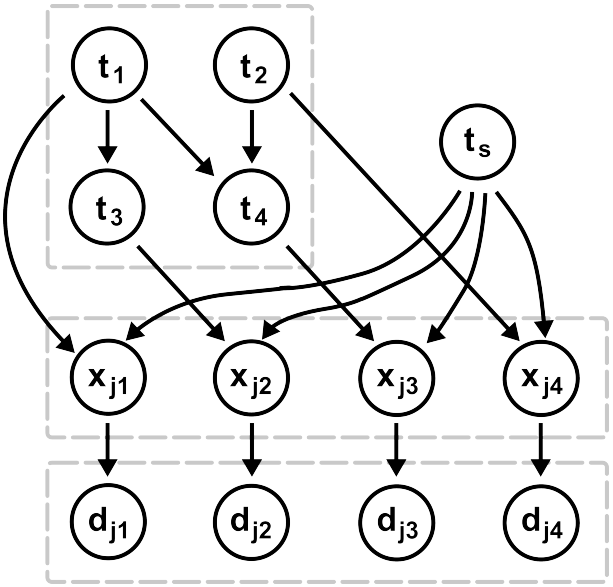

Figure 2.3: Graph of a CBN example. The CBN in the upper left part of the image shows the waiting times t_i of mutations $i = 1, \ldots, 4$ and the order in which the mutations occur: mutations 1 and 2 can occur independently; mutation 3 can only occur after mutation 1 occurred, $t_1 < t_3$, and the presence of mutation 4 is conditional on mutations 1 and 2 being present, i.e. $t_1, t_2 < t_4$. The observation time t_s is depicted in the upper right part of the image - it is independent of the waiting times t_i. A mutation i is present in the genotype $\mathbf{x}_j = (x_{j1}, \ldots, x_{j4})$ of a patient j if $t_i < t_s$. The observations $\mathbf{d}_j = (d_{j1}, \ldots, d_{j4})$ contain errors which occur independently for each mutation with probability ϵ. Figure reproduced from [34].

respective mutation to develop. Here, N is the total number of possible mutations. The set of parent mutations $\mathrm{pa}(i)$ is defined as the set of mutations that need to be present before a specific mutation i can occur. The waiting times t_i are defined to be exponentially distributed with parameter λ_i given that all parent mutations $\mathrm{pa}(i)$ are present:

$$t_i \sim \mathcal{E}(\lambda_i) + \max_{j \in \mathrm{pa}(i)} t_j. \tag{2.1}$$

The parameter $\lambda_i > 0$ is the parameter of the exponential distribution, often called the *rate parameter*.

The upper left part of figure 2.3 shows the waiting times t_i of mutations $i = 1, \ldots, 4$ and the order in which the mutations occur. Mutations 1 and 2 can occur independently, mutation 3 can only occur after mutation 1 occurred, i.e. $t_1 < t_3$, and the occurrence of mutation 4 is conditional on mutations 1 and 2 being present, i.e. $t_1, t_2 < t_4$.

CBNs furthermore include a time of diagnosis t_s for the tumor of each patient j which models the overall progression of that tumor. The time t_s which is independent of the waiting times t_i is shown in the upper right part of figure 2.3. The time of diagnosis is sampled from an exponential distribution $t_s \sim \mathcal{E}(\lambda_s)$ for each patient. Since the entire parameter vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N, \lambda_s)$ is not likelihood identified, the parameter λ_s for the observation time is set to 1.

The perfect data without observational errors are represented as a binary matrix $\mathbf{X} = (x_{ji})$, where $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, N_P\}$. Mutation i is present in patient j at diagnosis, i.e. $x_{ji} = 1$ if $t_i < t_s$, otherwise $x_{ji} = 0$. The genotype of a patient j is denoted as $\mathbf{x}_j = (x_{j1}, \ldots, x_{jN})$ and is formed by the j-th row in the matrix $\mathbf{X}$. The i-th column of $\mathbf{X}$ is denoted as $\mathbf{x}_i^T = (x_{1i}, \ldots, x_{N_P i})^T$.

As introduced in section 1.4, the actual observations are denoted as $\mathbf{D} = (d_{ji})$ and may differ from the perfect data $\mathbf{X}$ due to observational errors which are assumed to occur independently for each mutation with probability ϵ. Both the perfect genotype $\mathbf{x}_j$ and the observed genotype $\mathbf{d}_j$ of a patient j are shown in the lower part of figure 2.3.

Parameter estimation

For a given network G the log-likelihood of N_P independent observations (patients) $\mathbf{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_{N_P})$ is [34]

$$\ell_{\mathbf{D}}(\epsilon, \boldsymbol{\lambda}, G) = \sum_{j=1}^{N_P} \log\Big[\sum_{\mathbf{x}_l \in J(G)} \mathrm{Prob}_\epsilon[\mathbf{d}_j|\mathbf{x}_l] \; \mathrm{Prob}_{\boldsymbol{\lambda},G}[\mathbf{x}_l] \Big]. \tag{2.2}$$

The formula for the log-likelihood is composed by

- the prior probability that the genotype $\mathbf{x}_l$ occurs

$$\mathrm{Prob}_{\boldsymbol{\lambda},G}[\mathbf{x}_l] = \mathrm{Prob}_{\boldsymbol{\lambda},G}\Big[\max_{i:x_{li}=1} t_i < t_s < \min_{j:x_{lj}=0} t_j \Big],$$

- the conditional probability of an observation $\mathbf{d}_j$ given genotype $\mathbf{x}_l$

$$\mathrm{Prob}_\epsilon[\mathbf{d}_j|\mathbf{x}_l] = \epsilon^{d(\mathbf{x}_l,\mathbf{d}_j)}(1-\epsilon)^{n-d(\mathbf{x}_l,\mathbf{d}_j)},$$

- the sum $\sum_{\mathbf{x}_l \in J(G)}$ where $J(G)$ is the lattice of order ideals, containing all genotypes $\mathbf{x}_l$ compatible with the graph G (see [13]), and

- the sum $\sum_{j=1}^{N_P}$ over all N_P observations in $\mathbf{D}$.

For a fixed network G the parameters ϵ and $\boldsymbol{\lambda}$ are estimated in a nested expectation–maximization (EM)-algorithm which consists of two loops. In the inner loop the estimate of the noise level $\hat{\epsilon}$ is obtained which locally maximizes $\ell_{\mathbf{D}}(\epsilon, \hat{\boldsymbol{\lambda}}^{(k)}, G)$ for a given $\hat{\boldsymbol{\lambda}}^{(k)}$ estimated at the previous iteration and a given network G from the previous step of the simulated annealing (SA) procedure. The value $\hat{\epsilon}$ is then used in the outer loop to find the estimator $\hat{\boldsymbol{\lambda}}$. Here, k is the index of the outer loop.

In order to actually learn a network structure G from a given dataset $\mathbf{D}$, the EM-algorithm is embedded into a simulated annealing (SA) procedure for performing the search in the graph space. During this, one first computes the log-likelihood $\ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G)$ for a given G and $\mathbf{D}$. Then a new network G' is generated randomly and is accepted if either $\ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G') > \ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G)$ or, alternatively it is accepted with a probability $\exp(-[\ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G) - \ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G')]/T)$. During the process the annealing temperature T tends to zero and the probability to accept changes that decrease the log-likelihood also does.

The following two paragraphs describe how the proposal of a new graph topology is performed in the software package ct-cbn, version 0.1.04 (Oct 2011), available on the website http://www.bsse.ethz.ch/cbg/software/ct-cbn.

Compatibility

Consider the graph G depicted in figure 2.4 (A) and the dataset $\mathbf{D}$ shown in figure 2.4 (B). The example dataset consists of four genotypes and four mutations, where the rows represent the genotypes and the columns correspond to the mutations. In the figure, events that are present in a patient's genotype, i.e. $x_{ji} = 1$, are marked by black fields. White fields denote absent events. A genotype $\mathbf{x}_j$ of a patient j is called compatible with the graph G if for every event that is present in $\mathbf{x}_j$ all parent mutations are also present. Genotype $\mathbf{x}_1$ is compatible with the graph because the condition is fulfilled for all three events that are present. Event x_{11} can occur independently of any parent node, x_{13} can occur because its parent x_{11} is present, and the occurrence of event x_{14} is conditioned on the presence of x_{13}. Genotype $\mathbf{x}_2$ is also compatible with the graph G due to similar reasons. Genotype $\mathbf{x}_3$ is not compatible because here only event x_{33} is present, but its parent x_{31} is not. Genotype $\mathbf{x}_4$ is not compatible either. Here, the occurrence of events x_{41} and x_{42} is in agreement with the graph, x_{41} may occur independently because it has no parent events and x_{42}

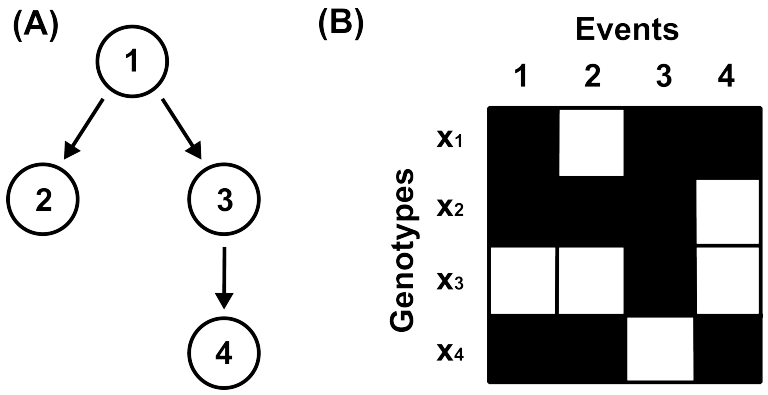

Figure 2.4: Example graph and dataset. (A) Example graph with four nodes. (B) Example dataset where every row represents a genotype and the columns correspond to the events. Black fields mark events that are present, absent events are shown in white.

can occur because its parent x_{41} is also present. However, the presence of event x_{44} without event x_{43} having occurred is not in agreement with the graph and therefore this genotype is not compatible with the graph.

More formally this means that a genotype $\mathbf{x}_j$ is not compatible with a graph G if for any edge ik in the graph we have that $x_{ji} = 0$ and $x_{jk} = 1$ for at least one pair $i, k \in 1, \ldots, N$.

For a dataset $\mathbf{D}$ that consists of N_P patients (or genotypes) and N events one can then calculate a measure α for the compatibility of the dataset with a graph G

$$\alpha = N_c/N_P, \tag{2.3}$$

where N_c is the number of genotypes in $\mathbf{D}$ that are compatible with the graph G and $0 < \alpha < 1$. The value α can be interpreted as the proportion of genotypes that are compatible with G among all genotypes in $\mathbf{D}$.

Proposal of a new graph topology

During each step of the simulated annealing (SA) a new graph G' is proposed with one edge difference to the previous graph G. To obtain G', a pair of numbers u, w is drawn uniformly from $\{1, \ldots, N\}$ with $u \neq w$ and the respective entry A'_{uw} in the adjacency matrix of G' is flipped, from 0 to 1 or vice versa. The graph is then transitively reduced. To assess the quality of the new graph, the compatibility α' of the dataset $\mathbf{D}$ with the graph G' is calculated according to eqn. (2.3).

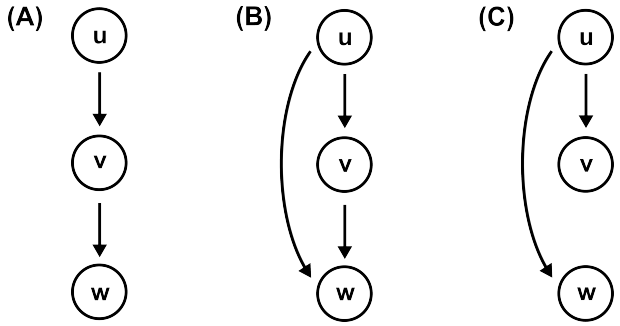

Figure 2.5: Edge replacement. (A) Original edge constellation in graph G with edges uv and vw. (B) Proposed edge uw is a transitive edge. (C) In the new graph G' the edge uw is kept and instead the edge vw is removed.

If in the new graph we have that $A'_{uw} = 1$, then an edge was added. If the graph contains a cycle or if the edge uw is a multiple intermediate state (see p. 23), α' is set to 0. If the edge uw is a single intermediate state (see p. 23), an edge replacement is made as shown in figure 2.5. The original edge constellation in graph G with edges uv and vw is shown in figure 2.5 (A). The edge uw would introduce a single intermediate state as illustrated in figure 2.5 (B). In this case, the edge uw is kept in the new graph G' and instead the edge vw is removed, as shown in figure 2.5 (C).

The compatibility α of the dataset $\mathbf{D}$ with the graph G is also computed following eqn. (2.3). $\alpha' > \alpha$ means that the proportion of genotypes in $\mathbf{D}$ compatible with the graph is higher in the case of G' compared to G. In this case the graph G' is accepted as a candidate graph. If $\alpha' < \alpha$, the graph G' is accepted as a candidate graph with probability $\exp\left[(\alpha' - \alpha)/0.05\right]$.

For the accepted candidate graph the likelihood $\ell_{\mathbf{D}}(\hat{\epsilon}, \hat{\boldsymbol{\lambda}}, G')$ is then calculated.

2.3 Support vector machines

Support vector machines (SVMs) [15, 21, 22, 72] belong to a class of supervised learning algorithms which can be used for classification and regression. When doing two-class classification an SVM takes a set of n-dimensional training samples with known class membership as input, where n is the number of attributes in each sample. The aim of the SVM is to find an (n-1)-dimensional hyperplane that separates the data best into the two classes.

In the following section the key properties of SVMs are explained for the case of linearly separable data. In the case of real world data the separation of the data is normally not possible without error. A modification of SVMs that accounts for this case is introduced in section 2.3.2. If the data can not be separated linearly it can be mapped to a feature space where linear separation is again possible. The explicit mapping of the data can be avoided by using the *kernel-trick*. This is explained in section 2.3.3.

2.3.1 Linearly separable data

The input to the training algorithm, the *training data*, is a set of training samples which is usually denoted by

$$\mathbf{D}_{train} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)). \tag{2.4}$$

The $\mathbf{x}_i$, $\mathbf{x}_i \in \mathcal{X}$, are called the *samples*, where $i = 1, \ldots, l$ and l is the number of samples. $\mathcal{X} \subseteq \mathbb{R}^n$ is called the *input space*. The $y_i \in \{-1, 1\}$ are the according class labels.

If the dataset is linearly separable, a hyperplane $h \in \mathbb{R}^{n-1}$ exists which divides the data correctly into its two classes. This splits the space into a positive and a negative halfspace. The hyperplane's normal vector $\mathbf{w}$ points by definition into the positive halfspace which contains class +1. The aim is to find the hyperplane with the maximal margin to the nearest data points of each class - the maximum-margin hyperplane. The nearest data points are called the *support vectors*.

Figure 2.6 shows an example dataset where the data from class -1 are marked in green and the data in class +1 are shown in red. The two boundary hyperplanes through the nearest data points $\mathbf{x}_+$ and $\mathbf{x}_-$, drawn as dashed lines, are chosen such that the geometric margin γ between them is maximized.

Any hyperplane h can be written as the set of points $\mathbf{x}_h$ satisfying

$$\langle \mathbf{w} \cdot \mathbf{x}_h \rangle + b = 0, \tag{2.5}$$

where $\mathbf{w}$ is the hyperplane's normal vector, b is a constant which determines the distance of the hyperplane to the origin of the coordinate system and $\langle \mathbf{w} \cdot \mathbf{x}_h \rangle$ is the *dot product* between the two n-dimensional vectors $\mathbf{w}$ and $\mathbf{x}_h$.

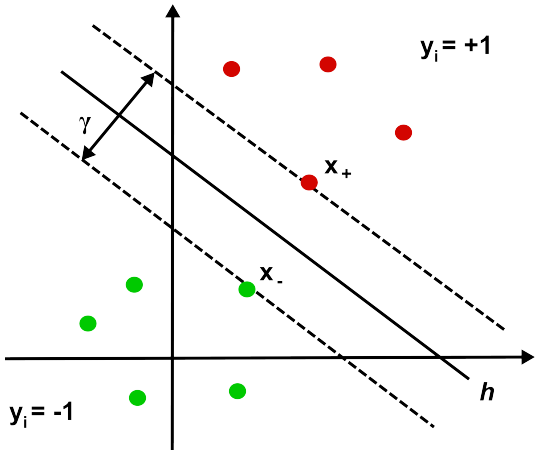

Figure 2.6: Optimal hyperplane. An example dataset is shown with data from class -1 marked in green and data from class +1 shown in red. The two dashed lines are the boundary hyperplanes through the nearest data points $\mathbf{x}_+$ and $\mathbf{x}_-$. They are chosen such that the geometric margin γ between them is maximized. The optimal hyperplane h lies exactly in between those two hyperplanes and separates the data perfectly into the two classes.

Having a solution for the optimal hyperplane h we can formulate the decision as

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b) \quad \text{primal form} \tag{2.6}$$

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b\right) \quad \text{dual form,} \tag{2.7}$$

where $\mathbf{x}$ is a test sample.

The optimal hyperplane h lies exactly in between the boundary hyperplanes and separates the data perfectly into the two classes. The geometric margin γ is invariant to rescalings of $\mathbf{w}$ and b. Therefore, $\mathbf{w}$ and b can be rescaled such that the two boundary hyperplanes through the nearest points $\mathbf{x}_+$ and $\mathbf{x}_-$ are described by

$$\langle \mathbf{w} \cdot \mathbf{x}_+ \rangle + b = 1 \quad \text{and} \quad \langle \mathbf{w} \cdot \mathbf{x}_- \rangle + b = -1. \tag{2.8}$$

Using geometry, the margin between these two hyperplanes can be calculated as $\frac{2}{\|\mathbf{w}\|}$. This means that the maximal margin of the hyperplane h can be found by minimizing $\|\mathbf{w}\|$. The expression $\|\mathbf{w}\|$ denotes the *norm* of the vector $\mathbf{w}$.

Solving the optimization problem described above is difficult because the norm of $\mathbf{w}$ involves a square root. One can substitute $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$ without changing the

solution. This results in a quadratic programming optimization problem:

$$\min_{(\mathbf{w},b)} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.9}$$

subject to

$$y_i\left(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b\right) \geq 1 \quad \text{for any } \; i = 1,\ldots,l. \tag{2.10}$$

The equality sign in eqn. (2.10) holds for the support vectors, the inequality holds for all the other samples $\mathbf{x}_i$ in the training dataset.

The solutions of (2.9) are obtained using a Lagrangian approach. The primal Lagrangian is given by

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i\left(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b\right) - 1\right], \tag{2.11}$$

with Lagrange multipliers $\alpha_i \geq 0$. The optimal solutions are denoted by $\mathbf{w}^*$, b^* and α_i^*. For non-zero α_i^* the *Karush-Kuhn-Tucker complementarity condition*

$$\alpha_i^*\left[y_i\left(\langle \mathbf{w}^*\cdot\mathbf{x}_i\rangle + b^*\right) - 1\right] = 0 \tag{2.12}$$

is only fulfilled by the support vectors. For all the other vectors $\mathbf{x}_i$ the α_i^* are 0.

In order to obtain the dual Lagrangian $\mathcal{L}$ is minimized with respect to $\mathbf{w}$ and b. Hence, setting

$$\frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad \text{and} \quad \frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \tag{2.13}$$

yields

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i. \tag{2.14}$$

Note that $\mathbf{w}$ is a linear combination of the training vectors as can be seen in eqn. (2.14). The constant b is obtained by

$$b = \frac{1}{r}\sum_{k=1}^{r} y_k - \langle \mathbf{x}\cdot\mathbf{x}_k\rangle, \tag{2.15}$$

where $k = 1,\ldots,r$ are the indices of the support vectors and r is the number of support vectors.

Substituting the conditions (2.14) into the primal form of the Lagrangian (2.11) leads

to the dual form of the Lagrangian

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{2.16}$$

Hence the optimization problem can be expressed as

$$\max_{(\boldsymbol{\alpha})} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{2.17}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \quad i = 1, \ldots, l \tag{2.18}$$

which is easier to solve than the problem in the primal form (equations (2.9) and (2.11)) because the constraints are less complex than those in eqn. (2.10).

2.3.2 Soft margin SVM

In practice, the training dataset will contain errors e.g. due to noise so that an optimal separation of the data with a linear SVM will not be possible. Therefore, Cortes and Vapnik [21] introduced a modification of the SVM algorithm. For each training vector $\mathbf{x}_i$ a *slack variable* $\xi_i \geq 0$ is introduced which measures the degree of misclassification of the data $\mathbf{x}_i$:

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{2.19}$$

The soft margin SVM is illustrated in figure 2.7. In the example dataset the data from class -1 are shown as green dots, the data in class +1 are drawn in red. The dashed lines are the two boundary hyperplanes, the solid line is the optimal hyperplane h. The green and red dots in between the two boundary hyperplanes are noisy data points that cannot be classified correctly.

Of course one can always fulfill the constraint in eqn. (2.19) by making the ξ_i large enough. This trivial solution is to be prevented by introducing a function which penalizes non-zero ξ_i. Then the optimization problem becomes

$$\min_{(\mathbf{w},\xi,b)} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i \right\} \tag{2.20}$$

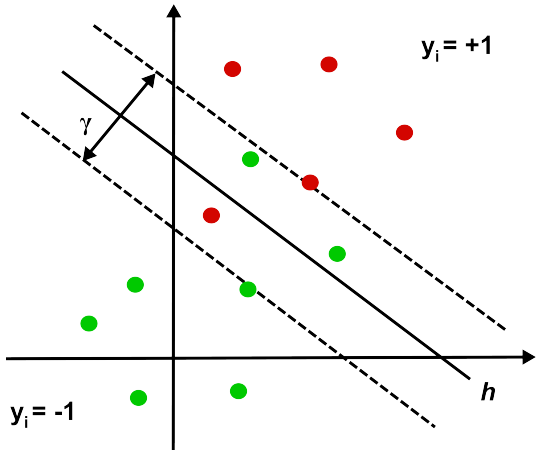

Figure 2.7: Soft margin SVM. An example dataset is shown with data from class -1 marked in green and data from class +1 shown in red. The data can not be separated perfectly with a linear SVM.

subject to

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0,$$

where $i = 1, \ldots, l$ and $C > 0$ is called the cost value. Equation (2.20) can again be solved using a Lagrangian approach. The new primal Lagrangian is given by

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i \\ &\quad - \sum_{i=1}^{l} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{l} \beta_i \xi_i, \end{aligned} \tag{2.21}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$. For obtaining the dual Lagrangian the derivatives of $\mathcal{L}$ with respect to $\mathbf{w}, b, \boldsymbol{\xi}$ are calculated. Setting the derivatives to zero yields

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \quad \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \quad \text{and} \quad C - \alpha_i - \beta_i = 0. \tag{2.22}$$

Inserting these relations into eqn. (2.20) we obtain the dual Lagrangian

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{2.23}$$

Here the non-zero coefficients α_i can only occur if the corresponding sample $\mathbf{x}_i$ satisfies

the equality in (2.19). Hence, the optimization problem reads

$$\max_{(\boldsymbol{\alpha})} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \tag{2.24}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0 \quad \text{and} \quad C \geq \alpha_i \geq 0, \quad i = 1, \ldots, l. \tag{2.25}$$

Interestingly, the optimization problem is the same as for the linearly separable case except for the upper bound for the coefficients α_i.

The optimal solutions are denoted by $\mathbf{w}^*$, b^*, $\boldsymbol{\xi}^*$, $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$. For non-zero α_i^* the Karush-Kuhn-Tucker complementarity conditions

$$\begin{aligned} &\alpha_i^* \left[y_i \left(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle + b^* \right) - 1 + \xi_i^* \right] = 0 \\ &\xi_i^* (\alpha_i^* - C) = 0 \end{aligned} \tag{2.26}$$

are only fulfilled by the support vectors. In this case, we need to have $\alpha_i^* = C$ to fulfill the second condition, which implies non-zero slack variables for the support vectors. For all the other vectors $\mathbf{x}_i$ the α_i^* are 0.

2.3.3 Kernels

Real world data can often not be separated linearly. A solution to this problem is to map the data $\mathbf{x}_i \in \mathcal{X}$ from the input space $\mathcal{X} \subseteq \mathbb{R}^n$ to a feature space $F \subseteq \mathbb{R}^m$ where linear separation of the data is again possible. The mapping, which is usually non-linear, is given by

$$\begin{aligned} \boldsymbol{\Phi} : \quad \mathcal{X} &\to F \\ \mathbf{x} &\mapsto \boldsymbol{\Phi}(\mathbf{x}). \end{aligned} \tag{2.27}$$

The mapping of the data to a feature space F is illustrated schematically in figure 2.8. On the left the data are shown in the input space $\mathcal{X}$ where they can not be separated linearly. On the right we see the data after mapping to a feature space F. Here, a linear separation of the data is possible.

The dimensions of the spaces $\mathcal{X}$ and F can be chosen to be equal, but in general they are different. One can choose $m < n$ if a dimension reduction of the data is useful, e.g. if there are redundant or irrelevant features that can be excluded. There might

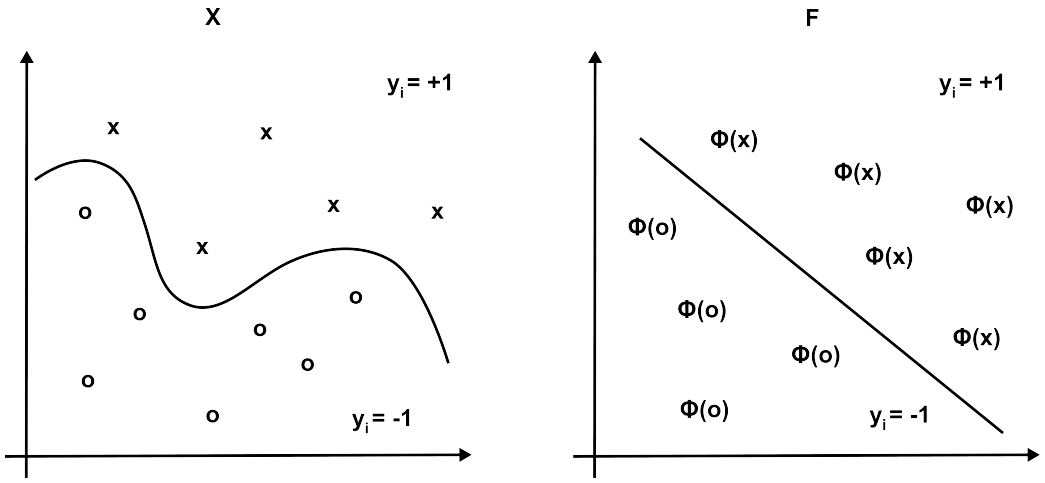

Figure 2.8: Feature map. Left: the data in the input space $\mathcal{X}$ can not be separated linearly. Right: After mapping the data to a feature space F linear separation is again possible. Figure adapted from [22].

be other cases, however, where an additional number of features is advantageous to extract more relevant information from the data.

Combining the dual form of the decision function given in equation (2.7) and the mapping (2.27) we obtain the following decision function in the feature space

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}) \rangle + b\right). \tag{2.28}$$

One way to avoid the explicit computation of the mapping is to calculate the inner product $\langle \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}) \rangle$ directly as a function of the input samples $\mathbf{x}_i$ and the test point $\mathbf{x}$. Such a method is called a *kernel* function which is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \rangle, \tag{2.29}$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ and $i, j \in \{1, \ldots, l\}$. The kernel function K allows the implicit mapping of the data into the feature space F [4, 15].

A function $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel if it is symmetric and the Gram matrix

$$\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{l} \tag{2.30}$$

is positive semi-definite.

Examples for kernels

The simplest kernel is the linear kernel which was already used in section 2.3.1 in the linearly separable case. It is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{2.31}$$

The radial kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{2.32}$$

for $\gamma > 0$.

The polynomial kernel is given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + c_0)^d, \tag{2.33}$$

where d is the degree of the polynomial.

The parameters γ, c_0 and d need to be tuned.

2.4 Approximate Bayesian computation

In most problems of parametric statistical inference knowledge of the likelihood function is crucial. However, for large datasets and complex probability models it might be impossible to derive an analytical formula for the likelihood function or its calculation might be computationally infeasible. Approximate Bayesian computation (ABC) techniques avoid the calculation of a likelihood function and thus make statistical inference possible for a wider range of models.

ABC techniques originally arose in the field of population genetics [10, 52, 58, 68], but have since also been applied in many other fields, as for example in ecology, epidemiology and systems biology. For an overview of the applications see e.g. Beaumont [9] and references therein.

Consider an observed dataset $\mathbf{D}$, a model G underlying the dataset and parameters θ which determine the model. In Bayesian computation one is usually interested in the posterior distribution $p(\theta|\mathbf{D})$ given by

$$p(\theta|\mathbf{D}) = \frac{p(\mathbf{D}|\theta)\pi(\theta)}{p(\mathbf{D})}. \tag{2.34}$$

Here, $p(\mathbf{D}|\theta)$ is the likelihood, i.e. the probability that the observed dataset $\mathbf{D}$ was generated with parameters θ, $\pi(\theta)$ is the prior and $p(\mathbf{D}) = \int p(\mathbf{D}|\theta)\pi(\theta)d\theta$ is the marginal likelihood. For problems where the calculation of the likelihood function is not possible, the posterior distribution $p(\theta|\mathbf{D})$ can not be calculated explicitly. Then ABC based methods offer a means to approximate the posterior distribution of the parameters θ by sampling observations from it.

Many ABC techniques are based on the simulation of data from a model G with parameters θ. One of the most basic approaches to approximate the posterior distribution is the following ABC rejection algorithm [9, 52]:

A1. Generate θ from the prior $\pi(\theta)$.
A2. Simulate data $\mathbf{D}'$ from model G with parameters θ.
A3. Accept θ if $\mathbf{D}' = \mathbf{D}$; return to A1.

This is usually repeated until a predefined number of parameter values have been accepted. The probability of accepting a simulated dataset $\mathbf{D}'$ depends on the size of $p(\mathbf{D})$. Therefore, in some cases the acceptance rate might not be sufficiently large [52]. Then one might use a more approximate algorithm such as e.g.:

B1. Generate θ from the prior $\pi(\theta)$.
B2. Simulate data $\mathbf{D}'$ from model G with parameters θ.
B3. Calculate the distance $\rho(\mathbf{D}, \mathbf{D}')$ between $\mathbf{D}$ and $\mathbf{D}'$.
B4. Accept θ if $\rho \leq \delta$; return to B1.

Here, δ is a parameter that has to be adjusted, the choice of δ is a tradeoff between computability and accuracy [52].

Algorithm B is useful when the data are discrete and low-dimensional. For high-dimensional or continuous data this algorithm might be impractical. The distance ρ can then be calculated using lower-dimensional summary statistics S of the data. This can be done since if a set of summary statistics $S = (S_1, \ldots, S_p)$ is sufficient for θ, i.e. $p(\mathbf{D}|S, \theta)$ is independent of θ, then the posterior distributions $p(\theta|\mathbf{D})$ and $p(\theta|S)$ are equal [52]. A rejection algorithm that uses summary statistics of the data is the following [33, 52]:

C1. Generate θ from the prior $\pi(\theta)$.
C2. Simulate data $\mathbf{D}'$ from model G with parameters θ.
C3. Compute the summary statistics S' corresponding to $\mathbf{D}'$ and calculate the distance $\rho(S, S')$ between S and S'.
C4. Accept θ if $\rho \leq \delta$; return to C1.

Rejection algorithms as the ones given above have several advantages, such as being easy to program, making parallel computation possible and allowing the estimation of Bayes factors for model comparison [52]. However, for more complex models sampling the parameters from the prior might not be sensible, because the prior and the posterior distribution are probably very different. Marjoram et al. [52] propose an alternative Markov chain Monte Carlo (MCMC) algorithm where the parameter θ is not drawn from the prior, but instead it is changed locally at each iteration of the algorithm based on the previous value of θ.

An example of an MCMC approach is given by the following algorithm:

D1. Given θ propose a move to θ' according to a transition kernel $q(\theta \to \theta')$.
D2. Simulate data $\mathbf{D}'$ from model G with parameters θ'.
D3. If $\mathbf{D}' = \mathbf{D}$, proceed to D4, otherwise reject θ' and return to D1.
D4. Calculate

$$h = h(\theta, \theta') = \min\left(1, \frac{\pi(\theta')q(\theta' \to \theta)}{\pi(\theta)q(\theta \to \theta')}\right).$$

D5. Accept θ' with probability h, otherwise reject θ'. Then return to D1.

Similar to algorithm A this algorithm might be impractical if e.g. acceptance rates are too small or if the data are high-dimensional and continuous. Then in analogy to algorithms B and C one can replace step D3 by:

D3′. If $\rho(\mathbf{D}, \mathbf{D}') \leq \delta$, proceed to D4, otherwise reject θ' and return to D1.

in the case of algorithm B or

D3″. If $\rho(S, S') \leq \delta$, proceed to D4, otherwise reject θ' and return to D1.

in the case of algorithm C.

2.5 Model search algorithms

2.5.1 Simulated annealing

Simulated annealing (SA) [18, 51, 53] is an algorithm that was developed for solving complex optimization problems in systems with many variables (up to several ten thousands). An example is the optimization of the physical design of a computer, which consists of many subsystems that all need to be optimized. Since these systems usually have a very large number of possible states, an exhaustive enumeration of all states is not feasible due to the enormous computational effort. Suboptimal solutions of the optimization problem can be obtained by a number of approximate methods. In some cases, algorithms based on SA offer an efficient way to solve such an optimization problem. Although it is unlikely that the method finds the global optimum, it often finds a *local optimum* which is a good approximation to the optimum.

The algorithm of SA consists of an iterative procedure. At each iteration a current state s and a candidate state s' are considered and a decision is made between the two states. The procedure is typically repeated until a good enough state of the system is obtained (according to a certain criterion or threshold) or until a given number of iterations has been performed.

A major property of the SA is the parameter T called the temperature which is gradually reduced during the simulation. The temperature T comes from an analogy to statistical mechanics. Here, one of the fundamental questions is the behaviour of

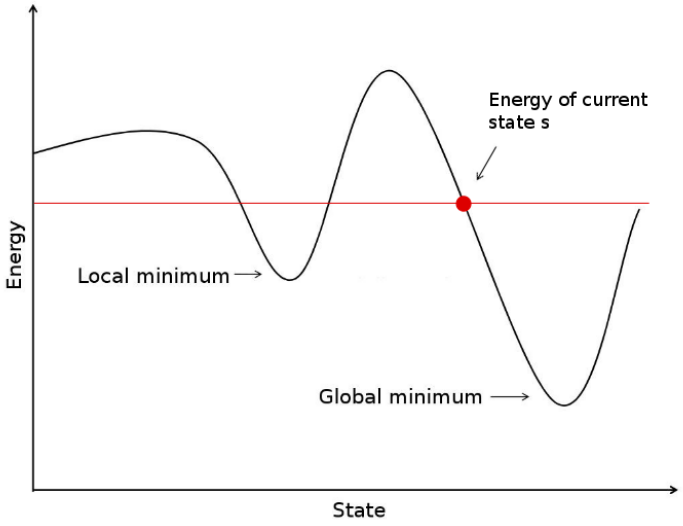

Figure 2.9: Energy landscape in simulated annealing. The possible states of the system are shown on the x-axis, the corresponding energies are depicted on the y-axis.

a system at very low temperatures, e.g. whether the atoms in a macroscopic body form a crystal or glass or if they stay fluid. This question is investigated by a careful annealing of the corresponding fluid material, doing a much slower cooling when the temperature is close to the freezing point.

This concept is also used in simulated annealing. The annealing is usually done relatively fast at high temperatures, while the stepsize is gradually reduced when the temperature comes close to zero. The specific annealing schedule can be chosen by the user. The SA algorithm is initialized with a high value of T, the temperature is then lowered by a small amount at each iteration. The temperature should be zero when the predefined number of iterations has been reached.

Again in analogy to statistical mechanics, each state s of the system is linked with an energy e. This is illustrated in figure 2.9. At high temperatures, the system can theoretically take every possible state, at low temperatures, however, low-energy states are increasingly favored.

In SA the decision between the two states s and s' is made probabilistically. The probability of moving the system to the candidate state s' is given by the function $P(e, e', T)$, where e and e' are the energies of the states s and s', respectively. In order to find a state of the system which is close to the global optimum, states with smaller energies are preferred to those with higher energies. In order to give the method

a chance to escape from a local minimum, the state e' is accepted with probability $P \neq 0$ even when $e' > e$.

In the original formulation of the simulated annealing by Kirkpatrick et al. [51] the function P is defined as

$$P(e, e', T) = \begin{cases} 1 & \text{if } e' < e \\ \exp[-(e' - e)/T] & \text{otherwise.} \end{cases} \tag{2.35}$$

This means that initially, when the temperature is still high, the system can take many different states within the search space. With decreasing temperature the acceptance of states with lower energy becomes much more likely than of those with higher energy. When the temperature is close to zero, the probability P of accepting a candidate state s' also tends to zero if its corresponding energy e' is higher than e.

2.5.2 Hill climbing

Hill climbing is another method for solving optimization problems where in contrast to simulated annealing only a local search is performed. Although finding the global optimum is not guaranteed, hill climbing is often able to find a good local optimum when only limited computational time is available.

Hill climbing consists of different iterations at each of which a current state s and a neighboring state s' are considered. A function $f(\mathbf{x})$ is associated with each state s of the system, where $\mathbf{x}$ is a vector that contains the parameters of the state s. The aim is to find a state which maximizes (or minimizes) the function $f(\mathbf{x})$.

The candidate state s' is generated by modifying one parameter in $\mathbf{x}$. The new state is accepted if it increases (or decreases) the value of the function $f(\mathbf{x})$, otherwise the system is left in the old state s. The procedure ends when no further improvement of the function $f(\mathbf{x})$ can be found.

2.6 A heuristic approach to noisy subset relations

A computational approach to detect hierarchical structure in binary datasets was developed by Jacob et al. [44]. Given a dataset where columns are events and rows

are genotypes or patients, the method finds the underlying structure by screening all pairs of events for potential noisy subset/superset relations.

This is illustrated in figure 2.10. Figure 2.10 (A) shows an example dataset where the columns correspond to the four events A, B, C, D and the rows represent the patients. Black fields mean that an event was observed in a patient, white fields encode events that were not observed. Figure 2.10 (B) depicts the subset/superset relations between the events in the dataset. Patients with event A are a (noisy) superset of patients with event B, which in turn are a (noisy) superset of patients with events C and D.

A parameter α is introduced which can be used to scale the sparseness of the graph. Small values of α result in graphs with many edges, while large values yield very sparse graphs. Considering any two events A and B, if the percentage of patients with event A that also have event B is α or higher, then B is said to be a noisy subset of A and an edge is drawn from A to B. All pairs of events in the dataset are screened for possible subset/superset relations using the above criteria. The relations are then encoded in a graph where nodes represent events and edges are drawn from supersets to subsets. If there are two or more events with identical cases, those events are joined in the graph. The graphical representation of the hierarchical structure in the example dataset is shown in figure 2.10 (C).

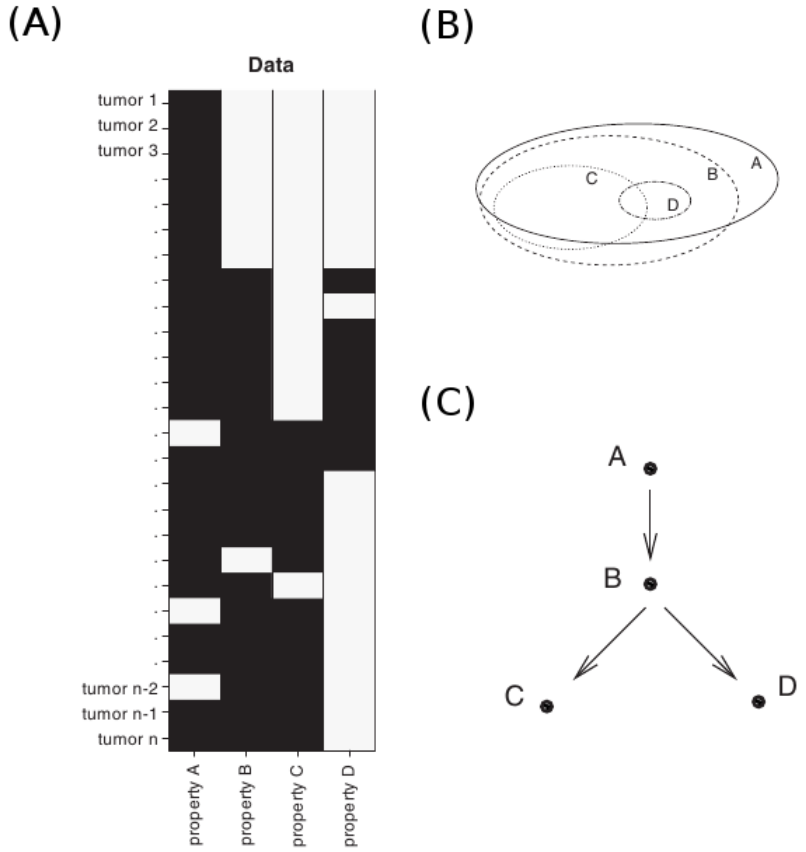

Figure 2.10: Subset/superset relations between different events observed in patients. (A) An example dataset. The columns correspond to the four events A, B, C, D, the rows represent the genotypes (or patients). Black regions mean that a genetic event was observed in a patient, white regions encode events that were not observed. (B) Subset/superset relations between genetic events. Patients with event A are a (noisy) superset of patients with event B, which in turn are a (noisy) superset of patients with events C and D. (C) Graphical representation of the subset/superset relations. Directed edges in the graph are drawn from supersets to subsets. Figure taken from [44].

Chapter 3

Likelihood-free methods for tumor progression modeling

In this chapter two methods are explained that were developed during this thesis for inferring tumor progression models from biological data. The methods rely on the simulation of artificial data based on competing networks. The data simulation is described in section 3.1. The first method is explained in section 3.2. It is based on machine learning and uses two different kinds of model search algorithms for global model fitting, simulated annealing and hill climbing. The second method that was developed is based on approximate Bayesian computation and global model search is done by means of hill climbing. A description of the method is given in section 3.3.

3.1 Data simulation

Since both methods developed in this thesis are based on the use of simulated data, the details of data simulation are explained first. Data simulation is done on the basis of a given graph topology G. The data are represented as a binary matrix $\mathbf{X} = (x_{ji})$, where $x_{ji} = 1$ if event i is diagnosed in patient j and $x_{ji} = 0$ otherwise. Here, $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, N_P\}$, where N is the number of genomic events in the dataset and N_P is the number of patients. A patient j is denoted as $\mathbf{x}_j = (x_{j1}, \ldots, x_{jN})$ and formed by the j-th row in the matrix $\mathbf{X}$. The i-th column of $\mathbf{X}$ is denoted as $\mathbf{x}_i^T = (x_{1i}, \ldots, x_{N_P i})^T$.

The following section describes how data are simulated in the CBN framework [34]. In order to generate the data $\mathbf{X}$, for each event i a waiting time parameter λ_i is

sampled from an exponential distribution $\lambda_i \sim \mathcal{E}(1)$. Then a set of waiting times t_i is generated according to equation (2.1) for every simulated patient $\mathbf{x}_j$. The set of waiting times defines the order in which the events occur in the simulated patient. In addition, a time of diagnosis is sampled from an exponential distribution $t_s \sim \mathcal{E}(\lambda_s)$ for each patient, where $\lambda_s = 1$. The data for one observation are then generated by comparison of t_s and the t_i. An event is present in a patient, i.e. $x_{ji} = 1$, if $t_i < t_s$ and it is absent otherwise. The matrix $\mathbf{X}$ represents the perfect data without noise. In order to obtain the actual observations $\mathbf{D}$, noise is then introduced with probability ϵ.

Data progression

A patient's tumor with many events is considered being further progressed than a tumor with only few events. Assuming a uniform distribution of the numbers of events n_E, $n_E \in \{1, \ldots, N\}$, of the simulated patients, there would be equally many patients in each progression level of a tumor. In case of an early event this means that both patients with less progressed tumors and patients with far progressed tumors would have this event. In contrast, an event that happens relatively late during tumor development will only occur in patients with further progressed tumors. This in turn means that events which happen very late during tumor development, i.e. events in the downstream parts of the network, occur very rarely in the simulated patients. Hence, in cases where the difference between two candidate graphs is in a downstream part of the network, the difference is hardly reflected in the data.

For this reason, data simulation is adapted in such a way that the distribution $\mathcal{F}$ of the number of events n_E of the simulated patients assigns a bigger weight to higher numbers of events. In the adapted model no time of diagnosis t_s is sampled. Instead, a random number n_E is sampled that follows a distribution $\mathcal{F}$. The choice of the distribution $\mathcal{F}$ offers a means to control the progression of every simulated patient's tumor. The distribution is chosen such that patients with further progressed tumors are more frequent than patients with less progressed tumors.

One possibility is to set the probability that n_E takes the value i, $i \in \{1, \ldots, N\}$, to

$$p(n_E = i) = p_i = z_i/N^2 + (N \bmod 2 - 1)/2N^2, \tag{3.1}$$

where $\sum_{i=1}^{N} p_i = 1$. The vector $z = (z_1, \ldots, z_N)$ is a sequence of numbers with stepsize

1, where the start and end values are defined as

$$\begin{aligned} z_1 &= N - [N - (1 - N \bmod 2) + 1]/2 \\ z_N &= N + [N - (1 - N \bmod 2) + 1]/2 + (1 - N \bmod 2) \end{aligned} \tag{3.2}$$

or in a more compact way

$$\begin{aligned} z_1 &= [(N - N \bmod 2)/2] + 1 \\ z_N &= (3N - N \bmod 2)/2. \end{aligned} \tag{3.3}$$

Equations (3.1) and (3.2) were found in an empirical way.

Noise

In reality the data will always contain some noise due to the fact that an event can be overlooked (false negative) or misdiagnosed (false positive). Therefore, noise is also introduced in the simulated data. False positives and false negatives are assumed to occur with the same probability ϵ. In the simulation process noise is introduced by flipping each binary component of the simulated data $\mathbf{X}$ with probability ϵ from 0 to 1 and vice versa. The noise is sampled from a beta distribution $\epsilon \sim \mathcal{B}(\alpha, \beta)$ with $\bar{\epsilon} = \alpha/(\alpha + \beta)$ and $\alpha = 2$. The noisy data are denoted by $\mathbf{D}$.

Summary statistics

Both the SVM and ABC based methods take as input a summary statistics of the data which is used to make a decision. There are many possible summary statistics and it is at this point difficult to say which might represent the properties of the underlying network best. The performance of several summary statistics, namely *covariance matrix*, *precision matrix* and *subset structures*, is compared in section 3.2.4.

Covariance matrix

Given a random vector $\mathbf{r} = (r_1, \ldots, r_N)$, the covariance matrix $\mathbf{\Sigma}$ is composed by the entries

$$\sigma_{i,k} = \mathrm{cov}(r_i, r_k), \tag{3.4}$$

where $\mathrm{cov}(r_i, r_k)$ is the covariance between two random variables r_i and r_k and $i, k \in \{1, \ldots, N\}$. Here, the random variables are the events i and k of the data. In order

to estimate an entry $\sigma_{i,k}$ of the covariance matrix $\mathbf{\Sigma}$, the columns $\mathbf{d}_i^T$ and $\mathbf{d}_k^T$ of the observed dataset $\mathbf{D}$ are used.

Due to the symmetry of the covariance matrix it is sufficient to consider the upper triangular matrix as input to the algorithm. For convenience it is then represented as a vector. The vectorized dataset $\mathbf{d}_{cov}$ is formed by the $m = (N^2 - N)/2$ entries of the upper triangular matrix according to

$$\mathbf{d}_{cov} = (\sigma_{12}, \sigma_{13}, \sigma_{23}, \ldots, \sigma_{N-1,N})^T. \tag{3.5}$$

Precision matrix
The precision matrix $\mathbf{\Sigma}^{-1}$ is the inverse covariance matrix. The elements of the precision matrix can be interpreted in terms of partial correlations. An element σ_{ik}^{-1} in the precision matrix reflects the correlation between two random variables r_i and r_k with the effects of all the other random variables r_l, $l \in \{1, \ldots, N\} \backslash (i, k)$, removed. In order to estimate an entry $\sigma_{i,k}^{-1}$ of the covariance matrix $\mathbf{\Sigma}^{-1}$, the columns $\mathbf{d}_i^T$ and $\mathbf{d}_k^T$ of the observed dataset $\mathbf{D}$ are used.

Again only the upper triangle of the precision matrix is provided as input to the algorithm. It is represented as a vector. The vectorized dataset $\mathbf{d}_{pinv}$ is formed by the $m = (N^2 - N)/2$ entries in the upper triangle according to

$$\mathbf{d}_{pinv} = (\sigma_{12}^{-1}, \sigma_{13}^{-1}, \sigma_{23}^{-1}, \ldots, \sigma_{N-1,N}^{-1})^T. \tag{3.6}$$

Subset structure
Consider a matrix $\mathbf{R}$ of random variables with elements r_{ji}, where $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, N_P\}$. The matrix of subset structures $\mathbf{\Omega}$ is then composed by the entries

$$\omega_{ik} = \begin{cases} \dfrac{\sum_{j=1}^{N_P} r_{ji}}{N_P} & \text{for} \quad i = k \\ \dfrac{\langle \mathbf{r}_i^T \cdot \mathbf{r}_k^T \rangle}{\sum_{j=1}^{N_P} r_{jk}} & \text{for} \quad i \neq k, \end{cases} \tag{3.7}$$

where $i, k \in \{1, \ldots, N\}$, the vectors $\mathbf{r}_i^T$ and $\mathbf{r}_k^T$ correspond to the i-th and k-th column of the matrix $\mathbf{R}$ and $\langle \mathbf{r}_i^T \cdot \mathbf{r}_k^T \rangle$ is the dot product between $\mathbf{r}_i^T$ and $\mathbf{r}_k^T$. In order to estimate the matrix of subset structures, the observed data $\mathbf{D}$ are used. The columns $\mathbf{d}_i^T$ and $\mathbf{d}_k^T$ are used instead of the random variables $\mathbf{r}_i^T$ and $\mathbf{r}_k^T$ and for the elements r_{ji} the data elements d_{ji} are used. For $i = k$ an element ω_{ik} corresponds to

the relative frequency of event i, while for $i \neq k$ an element ω_{ik} reflects the percentage of patients with event k that also have event i.

The vectorized dataset $\mathbf{d}_{sub}$ is formed columnwise by all $m = N^2$ entries of the subset structure matrix as

$$\mathbf{d}_{sub} = (\omega_{11}, \omega_{21}, \ldots, \omega_{N_P1}, \omega_{12}, \ldots, \omega_{N_P2}, \omega_{1N}, \ldots, \omega_{N_PN})^T \tag{3.8}$$

and given as input to the algorithm.

Simulation of multiple datasets

In order to capture the sample variability of the artificial data multiple datasets are simulated based on a given graph.

Covariance matrix

When using the covariance matrix as summary statistics, the columns of a *multiple dataset* $\mathbf{M}$ are given by the datasets $\mathbf{d}_{cov,i}$, where $i \in \{1, \ldots, N_d\}$ and N_d is the number of simulated datasets. The dimensions of the matrix $\mathbf{M}$ are $m \times N_d$, where $m = (N^2 - N)/2$ is the number of features (see also section 2.3).

Precision matrix

In case of the precision matrix, the columns of the multiple dataset $\mathbf{M}$ are composed by the datasets $\mathbf{d}_{pinv,i}$. The dimensions of $\mathbf{M}$ are $m \times N_d$, where again $m = (N^2 - N)/2$.

Subset structure

For the subset structure as a summary statistics the columns of the matrix $\mathbf{M}$ contain the datasets $\mathbf{d}_{sub,i}$. The dimensions of $\mathbf{M}$ are $m \times N_d$, where $m = N^2$.

3.2 Learning of tumor progression models via machine learning

3.2.1 The algorithm

Suppose we have an observed genomic dataset $\mathbf{D}_0$ of dimension $N_P \times N$ that contains tumor data of N_P patients and N events. Further suppose that there is a network G_0 underlying the dataset $\mathbf{D}_0$ which describes the process of accumulating genomic events for a certain cancer type. Many methods have been developed with the goal to reconstruct the true underlying network G_0 as closely as possible. Quite frequently those methods are based on likelihoods, this means that the decision about which of two candidate networks describes the dataset $\mathbf{D}_0$ better is made via maximum likelihood estimation (MLE).

The conjunctive Bayesian networks (CBNs) introduced in section 2.2 are an example for a model to describe the accumulation of genomic events. In order to use the CBN model to reconstruct a network underlying a genomic dataset a maximum likelihood estimation (MLE) is performed [34]. This means the decision between two candidate networks is made via the analysis of the likelihood function. For each network the likelihood function is maximized with respect to the noise ϵ and the parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N, \lambda_s)$ within an EM-algorithm. The parameters with the higher likelihood are kept. For global network inference this procedure is embedded into a simulated annealing approach. At each iteration of the simulated annealing a decision between two candidate networks is made based on the value of the likelihood function.

In cases where the maximization problem cannot be solved in closed form, likelihood-based approaches become computationally very intensive. In the case of the CBN model for example, model inference is increasingly difficult for networks with 12 or more nodes because the computational time and memory usage increase dramatically. Therefore, in this thesis two alternative methods are proposed which circumvent the calculation of likelihoods and instead rely on the simulation of artificial data. The networks define different dependencies between the observed aberrations, and these are reflected in the simulated data.

As in the likelihood-based approach, the machine learning based method presented in this section is embedded in a model search algorithm for global model fitting. Two

different model search algorithms are used, simulated annealing and hill climbing, and their performance is compared. Both model search algorithms consist of individual iterations, where at each iteration two competing networks G_1 and G_2 are compared at a time based on the corresponding simulated data. An SVM classifier is trained to discriminate between the datasets underlying the two different networks. Finally, the classifier is applied to assign the dataset $\mathbf{D}_0$ to one of the two networks.

The main part of the algorithm consists of the three nested steps *simulation*, *decision* and *searching*.

The simulation step is the innermost part of the algorithm. Here, data are simulated on the basis of the pair of competing networks G_1 and G_2 as described in section 3.1. Based on the networks multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are simulated, respectively. The data contain noise, each element of the binary data matrix is incorrect with probability ϵ. As described in section 3.1, the noise is sampled from a beta distribution $\epsilon \sim \mathcal{B}(\alpha, \beta)$ with $\bar{\epsilon} = \alpha/(\alpha + \beta)$ and $\alpha = 2$. The waiting time parameters λ_i, $i \in \{1, \ldots, N\}$, are sampled from an exponential distribution $\lambda_i \sim \mathcal{E}(1)$.

In the decision step the decision between the two competing networks is made by means of SVM classification. The SVM is trained on a training dataset $\mathbf{D}_{train}$ which consists of the multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ based on the two graphs and the class labels (see section 3.2.2 for details). A model is trained on the training dataset and used to classify the dataset $\mathbf{D}_0$. This means the classifier is used to decide which of the two networks is closer to the true underlying progression model G_0.

In the search step a model search algorithm is built around the classification approach for global model search, similar as in Gerstung et al. [34]. Model search is initialized with a non-empty graph topology G_1 which is obtained using a method adapted from Jacob et al. [44], as will be explained on page 53. At each iteration of the model search a new candidate network G_2 is generated by randomly changing an edge in the network G_1. If the new network wins the classification, the modified edge is accepted. If instead the old network wins, depending on which model search algorithm is used, the new network is either rejected or accepted with probability p. The algorithm stops after a predefined number of iterations, or earlier if the condition of the dynamic stopping criterion is fulfilled (see p. 69).

The pseudocodes of the method are given in Algorithms 1 and 2 in Appendix A.2, for model search done with simulated annealing and hill climbing, respectively.

3.2.2 Learning with support vector machines

The theory of support vector machines (SVMs) was introduced in section 2.3. In this thesis classification is performed using C-Support Vector Classification (C-SVC) [15, 21] in the R package "e1071" [54] which is based on the "libsvm" library [19]. C-SVC is referred to as soft margin SVM in this thesis and was explained in section 2.3.2.

SVM classification is used in the decision step of each iteration of the model search algorithm to make the decision between two candidate topologies G_1 and G_2.

The training dataset $\mathbf{D}_{train}$ is formed by the multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ simulated based on the two graphs and the class labels and has the following form:

$$\mathbf{D}_{train} = \begin{vmatrix} \mathbf{M}_1 & & \mathbf{M}_2 \\ y_1 & \cdots & y_{2N_d} \end{vmatrix}^T . \tag{3.9}$$

The $y_i \in \{\pm 1\}$, $i \in \{1, \ldots, 2N_d\}$, are the class labels. If a dataset was simulated based on graph G_1 then $y_i = -1$, otherwise $y_i = 1$. The model is trained on dataset $\mathbf{D}_{train}$ and used to classify the dataset $\mathbf{D}_0$. This means that the classifier is used to decide between the two candidate networks in order to find the network which describes the observed genomic dataset $\mathbf{D}_0$ better.

3.2.3 Model search algorithms

In the search step of the previously described method two different model search algorithms are used, simulated annealing and hill climbing. Both search algorithms are initialized with a graph G_1 obtained by the same heuristic method (explained on p. 53) and propose a new graph topology at each iteration according to the same scheme (described on p. 54).

Simulated annealing

In section 2.5.1 simulated annealing was introduced in a general way. In order to do model fitting via machine learning the simulated annealing procedure is used for global model search in the space of DAGs.

At each iteration of the simulated annealing, there is a current graph topology G_1 and a candidate topology G_2, corresponding to the states s_1 and s_2. The probability $P(e_1, e_2, T)$ of accepting graph G_2 is commonly set to 1 if $e_2 < e_1$ and to $\exp(-\Delta E/T)$ otherwise, where $\Delta E = e_2 - e_1$. However, since in the method developed in this thesis the calculation of a likelihood function or any other objective function is avoided, there is no energy corresponding to the states of the system. Therefore, the original formulation is adapted by setting $\Delta E = 1$. Then the probability P of accepting graph G_2 is

$$P = \begin{cases} 1 \quad \text{if} \quad G_2 \text{ wins in the classification and} \\ a \cdot \exp(-1/T) \quad \text{otherwise,} \end{cases} \tag{3.10}$$

where a is a constant that needs to be specified (see eqn. (3.16), p. 70).

As the simulation proceeds the temperature is reduced gradually. The algorithm is initialized with $T = 10$ and the temperature finally tends to zero according to a user-defined annealing schedule (see p. 70).

Hill climbing

The theory of hill climbing was described in section 2.5.2. Here, hill climbing is used with a negative sign. Due to the lack of an objective function, hill climbing can not be used in the usual way. However, the basic concept of always moving the system towards states which decrease the value of the objective function is overtaken. Here, the classification replaces this concept.

As the simulated annealing, hill climbing consists of different iterations. At each iteration, a current graph G_1 and a candidate topology G_2 are considered and a decision between the two graphs is made by means of SVM classification. The decision is made deterministically, i.e. graph G_2 is accepted only if it wins in the classification. Hence, in contrast to the simulated annealing new states of the system are always rejected if they do not lead to an improvement.

Model search initialization

Model search is initialized with a non-empty graph topology which is obtained using a method adapted from Jacob et al. [44] (see section 2.6). The method screens the data for subset/superset relations between pairs of events and encodes them by drawing directed edges from supersets to subsets. According to the method of Jacob et al.

[44], if there are two (or more) events with identical cases, those events are joined in the corresponding graph.

In the present application events with identical cases cannot be joined as this would change the total number of nodes in the graph. However, events with identical cases are represented by a double edge in the graph which is not allowed in DAGs. Therefore, the original method is modified in such a way that one edge direction in the double edge is removed randomly.

Based on the results found by Jacob et al. [44], the scaling parameter is set to $\alpha = 0.9$. This means an edge is drawn from A to B if the percentage of patients with event A that also have event B is 90 % or higher. There are cases where both 90 % or more of the patients with event A also have event B and vice versa. Since this would again introduce a double edge into the graph, one of the two edge directions is removed randomly.

The resulting graph is transitively reduced.

Proposal of a new graph topology

The proposal of a graph topology in the original CBN approach [34] is described on p. 28 ff. In the method developed in this thesis a new topology is proposed in a similar way.

At each iteration in the model search a new graph G_2 is proposed which differs from the previous graph G_1 in one edge. To obtain G_2, an edge is chosen randomly by drawing a pair of numbers u, w uniformly from $\{1, \ldots, N\}$ with $u \neq w$ and the respective entry $A(G_2)_{uw}$ in the adjacency matrix of G_2 is flipped, from 0 to 1 or vice versa. The graph is then transitively reduced. Here, in contrast to the search method for CBN in [34] no value α for the compatibility of the dataset $\mathbf{D}_0$ with the graph G_2 is calculated (see eqn. (2.3)).

If we have $A(G_2)_{uw} = 1$, then an edge was added. If the edge uw is a multiple intermediate state (see p. 23) or if the new graph contains a cycle, then the edge is rejected. In the case of a single intermediate state, an edge replacement is made as described on p. 28. This means, that in G_2 the edge uw is kept and the edge vw is removed instead (see figure 2.5).

Data is then simulated from the accepted candidate graph in order to decide between G_1 and G_2 in the SVM classification.

3.2.4 Parameter tuning

As already explained in section 3.2.1 the SVM based method consists of the three parts simulation, decision and searching. In order to achieve optimal results when inferring tumor progression models the parameters in the individual parts are optimized. In the following, parameter optimization in data simulation, classification and model search is described.

Optimal parameters in data simulation

We look for the optimal parameters in data simulation, especially we want to find out what is the best summary statistics and what level of data progression should be used in order that the classifier can best distinguish between two graphs and the corresponding data.

Summary statistics

In order to get a qualitative assessment of how the different summary statistics mirror the structure of the data a visualization of the summary statistics is made. We consider the covariance matrix, precision matrix and subset structure.

Three graph pairs are considered which are shown in figure 3.1. The graphs in each pair differ in one edge and have a network size of $N = 5$. Figure 3.1 (A) shows graph pair 1 where the difference between the two graphs is in a downstream edge. The graphs in graph pair 2, figure 3.1 (B), differ in an edge at medium height. In graph pair 3 the difference is in an upstream edge, as shown in figure 3.1 (C). Figure 3.2 (A) shows the adjacency matrices of graph pair 1, left: graph G_1, right: graph G_2. In the images black corresponds to a 1 in the matrix (an edge), white stands for a 0 (no edge). Figures 3.3 (A) and 3.4 (A) show the adjacency matrices of graph pairs 2 and 3, respectively.

For data visualization a simulation is performed where artificial data are generated based on the three graph pairs. For each graph pair N_d datasets are simulated with N_P patients based on both graph G_1 and G_2. Here, the set (N_d, N_P) takes the values $(100, 20N)$ and $(250, 100N)$. The data are simulated with uniform distribution of progression levels, i.e. n_E is drawn uniformly from $\{1, \ldots, N\}$, and the noise is drawn from a beta distribution $\epsilon \sim \mathcal{B}(\alpha, \beta)$ with $\bar{\epsilon} = 0.01$.

From the N_d simulated datasets of each graph pair, the summary statistics $\mathbf{S}_{1,i}$ and $\mathbf{S}_{2,i}$, $i \in 1, \ldots, N_d$, are computed which correspond to the graphs G_1 and G_2, respectively. As summary statistics $\mathbf{S}$ the covariance matrix $\mathbf{\Sigma}$, the precision matrix $\mathbf{\Sigma}^{-1}$ and the subset structure $\mathbf{\Omega}$ are taken. Then the averages $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$ of the summary statistics $\mathbf{S}_{1,i}$ and $\mathbf{S}_{2,i}$ are computed elementwise. This means, an element $\bar{s}_{1,kl}$ of the matrix $\bar{\mathbf{S}}_1$ is calculated as

$$\bar{s}_{1,kl} = \frac{\sum_{i=1}^{N_d} s_{1,i,kl}}{N_d},$$

where $s_{1,i,kl}$ is an element of the summary statistics $\mathbf{S}_{1,i}$ and $k, l \in \{1, \ldots, N\}$. In addition, the difference $\Delta\mathbf{S} = \bar{\mathbf{S}}_1 - \bar{\mathbf{S}}_2$ is calculated. Note that the averages $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$ and the difference $\Delta\mathbf{S}$ are square matrices.

The pseudocode of the simulation can be found in Algorithm 3 in Appendix A.2.

Figure 3.2 (B-D) shows the image plots of the summary statistics for the simulated data for graph pair 1, left: $\bar{\mathbf{S}}_1$, middle: $\bar{\mathbf{S}}_2$, right: $\Delta\mathbf{S}$. Data was simulated with the values $(100, 20N)$ used for the set (N_d, N_P). In figure 3.2 (B) the covariance matrix $\mathbf{\Sigma}$ was used as a summary statistics, figure 3.2 (C) shows the results for the precision matrix $\mathbf{\Sigma}^{-1}$ and figure 3.2 (D) corresponds to the subset structure $\mathbf{\Omega}$. The images on the left and in the middle are scaled pairwise. Their colors range from light grey (minimum value) to dark grey (maximum value). In the images on the right colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white.

Figures 3.3 and 3.4 show the analogous results for graph pairs 2 and 3. Image results obtained when $(250, 100N)$ is used for the set (N_d, N_P) can be found in figures A.1 to A.3 for graph pairs 1-3 in Appendix A.3.

In the case of graph pair 1, the difference between the graphs is in a downstream part of the network. This fact is clearly reflected in the upper left part of the right images in figure 3.2. There is mostly white color in this part of the images, whereas in the lower right parts of the images darker shades of red and blue are dominating.

In constrast, in graph pairs 2 and 3 the difference is further upstream in the networks. This means that effects of this difference can also be seen in the events that are further downstream in the graphs. Therefore, in general a larger effect is to be expected when the difference is further upstream. This fact is reflected in the image plots of $\Delta\mathbf{S}$ through the presence of darker colors in all parts of the images. In addition, in the right plot of figure 3.4(B) the unconnected node 1 can clearly be seen.

However, in order to make the decision about which of the three representations mirrors the structure in the data best and hence leads to the lowest training error, a larger analysis is necessary. This analysis is presented on p. 67. There, both the optimal summary statistics and kernel function are found.

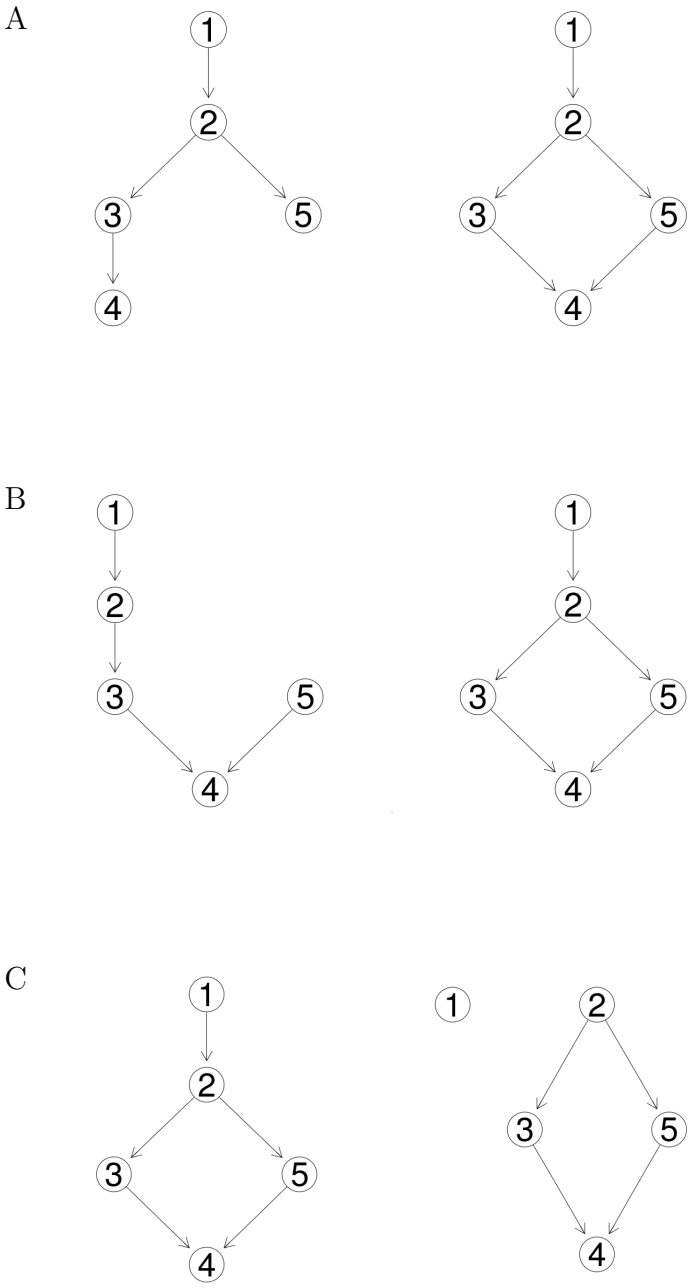

Figure 3.1: Graph pairs used for optimization. The figure shows the three graph pairs used for optimization of the summary statistics and the data progression. (A) In graph pair 1 the difference between the two graphs is in a downstream edge. (B) The graphs in pair 2 differ in an edge at medium height. (C) In graph pair 3 the difference is in an upstream edge.

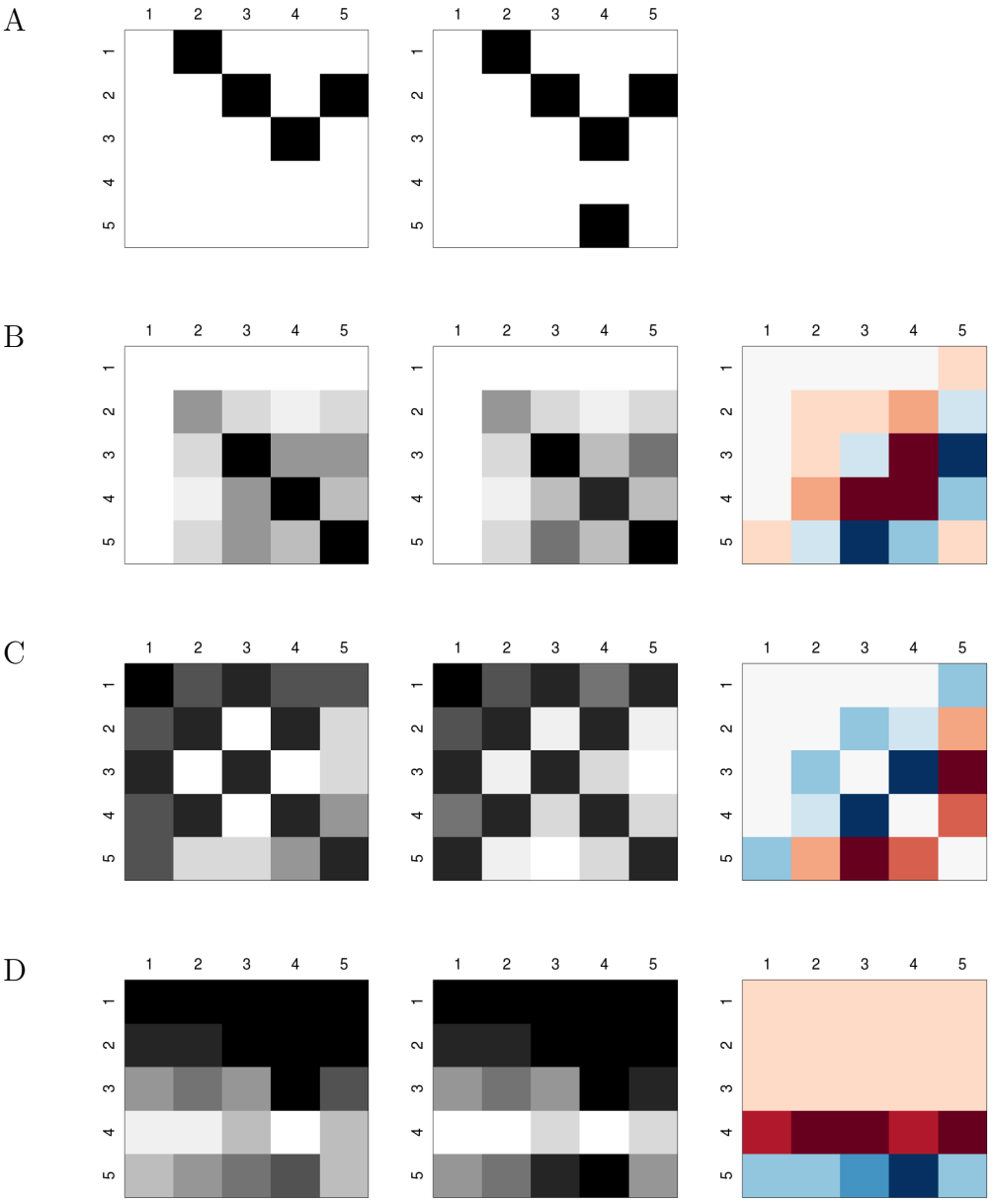

Figure 3.2: Different summary statistics for graph pair 1 with $(N_d, N_P) = (100, 20N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: $\Delta\mathbf{S}$. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

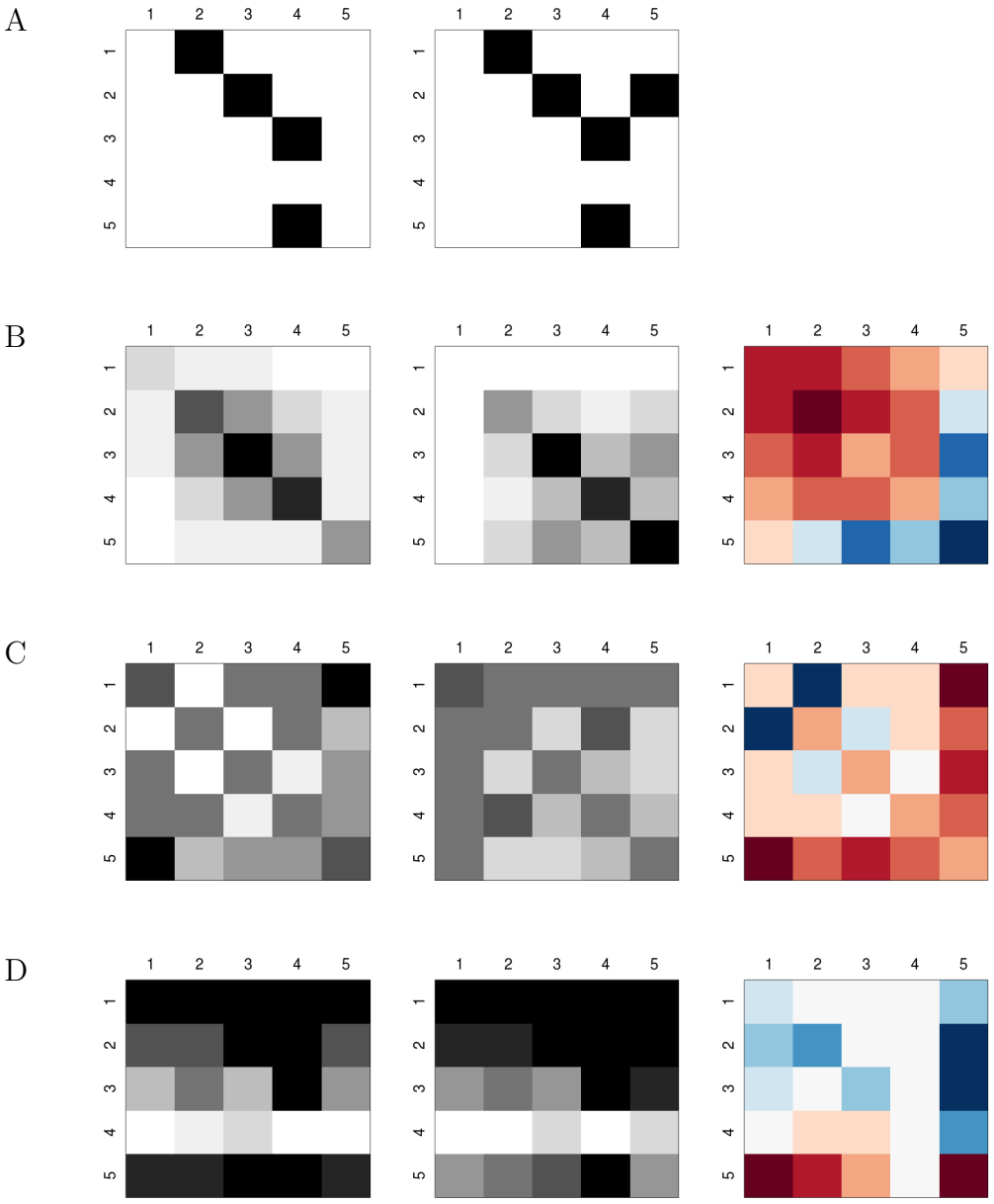

Figure 3.3: Different summary statistics for graph pair 2 with $(N_d, N_P) = (100, 20N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: $\Delta\mathbf{S}$. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

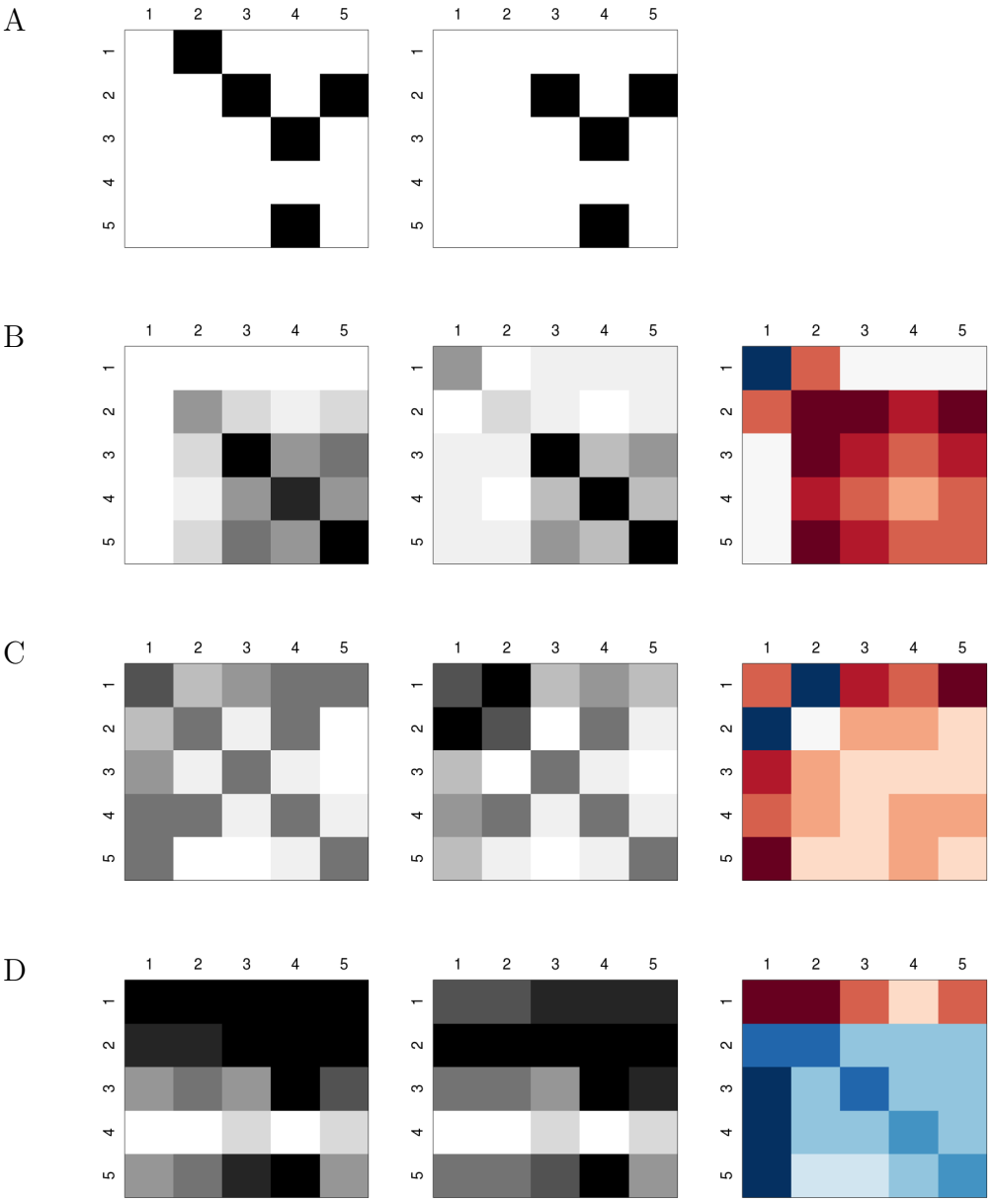

Figure 3.4: Different summary statistics for graph pair 3 with $(N_d, N_P) = (100, 20N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: $\Delta\mathbf{S}$. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

Data progression

In order to find the optimal level of progression in the simulated data, a simulation is performed where two different progression levels are compared. Data are either simulated with uniform distribution of progression levels, i.e. n_E is drawn uniformly from $\{1, \ldots, N\}$, or with a bigger weight assigned to higher numbers of events (see p. 46).

Again, the 3 graph pairs shown in figure 3.1 are considered. As summary statistics the covariance matrix, precision matrix and subset structure are considered. For each graph pair and summary statistics, $N_r = 25$ runs are performed. In each run, N_d datasets are simulated with N_P patients based on both graph G_1 and G_2. The set (N_d, N_P) takes the values $(100, 20N)$ and $(250, 100N)$. The noise is drawn from a beta distribution $\epsilon \sim \mathcal{B}(\alpha, \beta)$ with $\bar{\epsilon} = 0.01$.

Multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are computed from the N_d simulated datasets and a training dataset $\mathbf{D}_{train}$ is formed using $\mathbf{M}_1$ and $\mathbf{M}_2$. An SVM classifier with a radial kernel is then trained on the training data and the training error is determined via 10-fold cross validation. The average training error over the 25 runs is computed.

The pseudocode of the simulation can be found in Algorithm 4 in Appendix A.2.

Results for data with uniform distribution of progression levels are presented in tables 3.1 to 3.3. As expected, the training error is highest for graph pair 1 (see figure 3.1 (A)), because there the difference between the two graphs is in a downstream edge. This means that the difference between the graphs is reflected only in a very small part of the data, more precisely, only in the patients which have $n_E = 4$ events. When data are simulated based on the left graph, patients with 4 events can have two different genotypes, $(1, 1, 1, 1, 0)$ and $(1, 1, 1, 0, 1)$. When following the right graph, patients can have three different genotypes, $(1, 1, 1, 1, 0)$, $(1, 1, 1, 0, 1)$ and $(1, 1, 0, 1, 1)$.

For the other two graph pairs the training error is much lower. Surprisingly, the training error is not lowest for graph pair 3, although there the difference between the two graphs is in an upstream edge. This is very likely due to the fact that in the right graph of graph pair 3 node 1 is unconnected. Hence, event 1 can occur in a simulated patient completely independently of all other events. This makes it more difficult for the classifier to distinguish between data from the two graphs.

The results obtained when simulating data with a bigger weight on higher numbers of events are shown in tables 3.4 to 3.6. As expected, the training error is lower in

the case of graph pair 1, whereas for the other two graph pairs there is not much change.

For this reason, the modified version of the data simulation is used throughout the rest of this thesis.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.238	0.147
Precision matrix	0.219	0.130
Subset structure	0.225	0.148

Table 3.1: Average training errors for graph pair 1. The table shows the average training errors for three different summary statistics and for two different data sizes.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.029	0.019
Precision matrix	0.023	0.013
Subset structure	0.050	0.034

Table 3.2: Average training errors for graph pair 2. The table shows the average training errors for three different summary statistics and for two different data sizes.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.080	0.039
Precision matrix	0.173	0.099
Subset structure	0.139	0.090

Table 3.3: Average training errors for graph pair 3. The table shows the average training errors for three different summary statistics and for two different data sizes.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.222	0.137
Precision matrix	0.205	0.124
Subset structure	0.211	0.140

Table 3.4: Average training errors for graph pair 1 with further progressed data. The table shows the average training errors for three different summary statistics and for two different data sizes.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.026	0.019
Precision matrix	0.027	0.015
Subset structure	0.046	0.031

Table 3.5: Average training errors for graph pair 2 with further progressed data. The table shows the average training errors for three different summary statistics and for two different data sizes.

Summary statistics	$N_P = 20,\ N_d = 100$	$N_P = 100,\ N_d = 250$
Covariance matrix	0.095	0.049
Precision matrix	0.161	0.080
Subset structure	0.144	0.091

Table 3.6: Average training errors for graph pair 3 with further progressed data. The table shows the average training errors for three different summary statistics and for two different data sizes.

Optimal parameters in classification

In order to optimize the performance of the SVM classification we want to find out which kernel function yields the lowest training error. We look at the classifier's performance for three different kernel functions, namely radial kernel, linear kernel and polynomial kernel. The kernel function was introduced in section 2.3.3, the three kernel functions are given on p. 37.

Radial kernel

The radial kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad \text{for } \gamma > 0. \tag{3.11}$$

Parameter γ

For the parameter γ the default value is used which is $\gamma = 1/m$. The variable m is the number of features in the data. When using the covariance matrix as a summary statistics, the dimensions of the training dataset are $m \times N_d$, with $m = (N^2 - N)/2$. For the precision matrix, the dimensions are also $(N^2 - N)/2 \times N_d$ and for the subset structure they are $N^2 \times N_d$ (compare p. 49).

Parameter C

The parameter C occurs in the Lagrange formulation of the soft margin SVM (see equation (2.20)) and is called the cost value.

In order to optimize the parameter C a simulation is performed where $N_r = 100$ random graph pairs are generated with a network size of $N = 5$. The graphs G_1 and G_2 in each graph pair differ by one edge. Graph G_1 is generated with the randomDAG function in the package pcalg [41, 47] (see section A.1). There, the neighbor probability q_n is introduced which is a parameter that controls the sparsity of the graph. The neighbor probability is set to $q_n = 0.5$ throughout this thesis. The graph G_2 is obtained by randomly changing one edge in graph G_1 using the approach described on p. 54.

For each graph pair the set (N_d, N_P) takes the values $(100, 20N)$ and $(250, 100N)$. For each set of values (N_d, N_P) multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are simulated based on graphs G_1 and G_2 using the covariance matrix as a summary statistics. The data contain noise with probability $\bar{\epsilon} = 0.01$. A training dataset $\mathbf{D}_{train}$ is formed from the datasets $\mathbf{M}_1$ and $\mathbf{M}_2$. For each value of $C \in (0.1, 0.15, \ldots, 2)$ a 10-fold cross

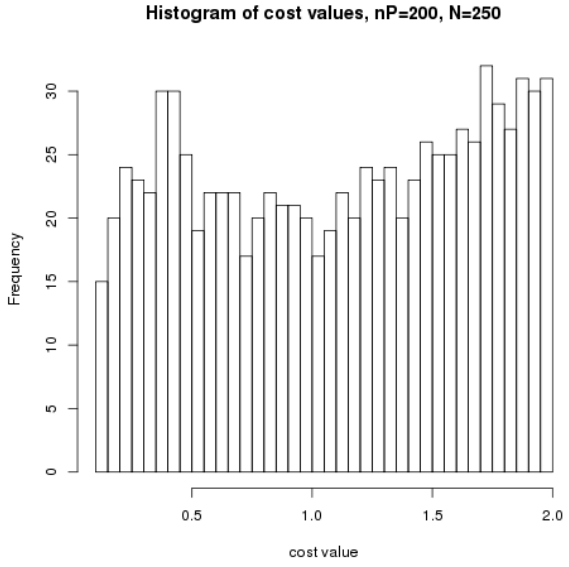

Figure 3.5: Histogram of the optimal cost values for 100 random graph pairs. Data were simulated with $N_P = 20$ and $N_d = 100$.

validation is performed using a radial kernel in order to obtain the training error. The value of C with minimal training error is recorded. In the case of several minima, all corresponding values of C are kept.

The pseudocode of the simulation can be found in Algorithm 5 in Appendix A.2.

The histogram of the optimal values of C over all 100 graph pairs is shown in figure 3.5. One can see that there is no clear most frequent value and hence no global optimum. This means that the parameter C would have to be optimized in every single decision step of the algorithm, which would make the machine learning based method very slow. Therefore, the default value of $C = 1$ is used.

Linear kernel

The linear kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{3.12}$$

Parameter C

The same simulation as described on p. 65 was done for the linear kernel in order to optimize the parameter C. Again, there was no clear most frequent value for

the parameter C. Therefore, the default value of $C = 1$ is also used for the linear kernel.

Polynomial kernel

The polynomial kernel is given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + c_0)^d, \tag{3.13}$$

where d is the degree of the polynomial.

Parameter γ
For the parameter γ again the default value $\gamma = 1/m$ was used.

Parameter C
For the parameter C, too, the default value $C = 1$ is used since no clear most frequent value could be found in the simulation.

Analysis to find optimal kernel and summary statistics

In order to decide which kernel and which summary statistics give the lowest training error a simulation is performed where three different network sizes $N \in \{5, 8, 10\}$ are considered and for each network size $N_r = 100$ random graph pairs are generated. In each graph pair the graphs G_1 and G_2 differ by one edge. Graph G_1 is again generated with the randomDAG function in the package pcalg [41, 47] (see section A.1) and graph G_2 is obtained by randomly changing one edge in graph G_1 as described on p. 54.

For each graph pair and network size, the set (N_d, N_P) takes the values $(100, 20N)$ and $(250, 100N)$. For each set of values (N_d, N_P) multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ are simulated based on graphs G_1 and G_2 using the covariance matrix, the precision matrix and the subset structure as summary statistics. The data contain noise with probability $\bar{\epsilon} = 0.01$.

From the data a training dataset $\mathbf{D}_{train}$ is formed. We consider the radial kernel, the linear kernel, and the polynomial kernel with degrees $d \in \{2, 3, 4\}$. For each of the 5 different kernels a 10-fold cross validation is performed using default values for γ and C in order to determine the training error. For each kernel and summary statistics the average training error over all 100 graph pairs is computed.

The pseudocode of the simulation can be found in Algorithm 6 in Appendix A.2.

Tables 3.7 to 3.9 show the average training errors for the radial kernel, the linear kernel and the polynomial kernel of degree 3 for data simulated with the values $(100, 20N)$ used for (N_d, N_P). Obviously, the classifier makes the lowest training error when a linear kernel is used. It can also be seen from table 3.8 that the training error is lowest when using the covariance matrix as summary statistics. Therefore, for the machine learning based method a linear kernel is used and the covariance matrix is used as summary statistics for the data.

For the polynomial kernel with degrees $d \in \{2, 4\}$ the training error was much higher. The results are shown in tables A.1 and A.2. The results obtained when $(250, 100N)$ is used for (N_d, N_P) can be found in tables A.3 to A.7. As expected, the error is much lower when the number of simulated datasets N_d and patients N_P is higher. These results confirm that the training error is lowest when using a linear kernel and when the covariance matrix is used as summary statistics for the data.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.156	0.202	0.198
Precision matrix	0.141	0.183	0.168
Subset structure	0.145	0.215	0.212

Table 3.7: Average training errors for radial kernel. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 100$ datasets were generated and the data was simulated with $N_P = 20$ patients. Training was done with an SVM using a radial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.101	0.160	0.155
Precision matrix	0.139	0.189	0.184
Subset structure	0.135	0.166	0.170

Table 3.8: Average training errors for linear kernel. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 100$ datasets were generated and the data was simulated with $N_P = 20$ patients. Training was done with an SVM using a linear kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.183	0.229	0.232
Precision matrix	0.189	0.240	0.249
Subset structure	0.195	0.258	0.262

Table 3.9: Average training errors for polynomial kernel of degree 3. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 100$ datasets were generated and the data was simulated with $N_P = 20$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Optimal parameters in model search

In both simulated annealing and hill climbing, model search stops after a certain predefined number of iterations. In addition, a dynamic stopping criterion is defined in order to reduce computation time. If the criterion is fulfilled before having reached the predefined number of iterations, the algorithm stops.

In the simulated annealing we also need to find an optimal annealing schedule and to specify the parameter a in eqn. (3.10) for the acceptance probability in order to maximize the chance of finding a good approximation to the global optimum.

Number of iterations and dynamic stopping criterion

At each iteration of the model search one edge is changed at random in the network. In order to guarantee that the graph space is explored well enough, the number of iterations is chosen to be at least twice the size of the adjacency matrix of the graph, i.e. $2 \cdot N^2$. So on average each edge can be changed twice, once removed and once added.

Since for small networks this would result in very few iterations and since computation time is not yet an issue, the number of iteration is chosen to be

$$N_{iter} = \max(250, 2 \cdot N^2). \tag{3.14}$$

In order to reduce computation time a dynamic stopping criterion is introduced for network sizes larger than 12. The criterion is found empirically by looking at the number of iterations during which the network stays the same. Towards the end of the model search, when we expect the system to be already close to a minimum, the

graph is very often constant over many iterations. Constant networks during more than $5 \cdot N$ iterations were in about 80 % of the cases followed by 2 or less further changes of the network before the algorithm ended.

Therefore the following rule is introduced as a dynamic stopping criterion for networks larger than 12 nodes: if the network has been constant during $5 \cdot N$ iterations, then the algorithm stops.

Annealing schedule and acceptance probability

The optimal annealing schedule and parameter a (see eqn. (3.10)) for the simulated annealing were found empirically. The temperature T decreases at each iteration according to

$$T = T \cdot (1 - 0.03\sqrt{k}/N), \tag{3.15}$$

where k is the iteration number. The probability of accepting the new network is $p = 1$ if the new network wins the classification, and it is

$$p = 0.3 \cdot \exp(-1/T) \tag{3.16}$$

if the old network wins. Figure 3.6 shows the annealing schedule and the corresponding acceptance probability. Figure 3.6 (A) shows the annealing temperature (y-axis) plotted against the iteration number (x-axis). In figure 3.6 (B) the acceptance probability (y-axis) is plotted against the iteration number (x-axis). The dots in the upper line correspond to an acceptance probability of $p = 1$ when the new network wins the classification, the dots in the lower line correspond to a probability of $p = 0.3 \cdot \exp(-1/T)$ in the case when the old network wins.

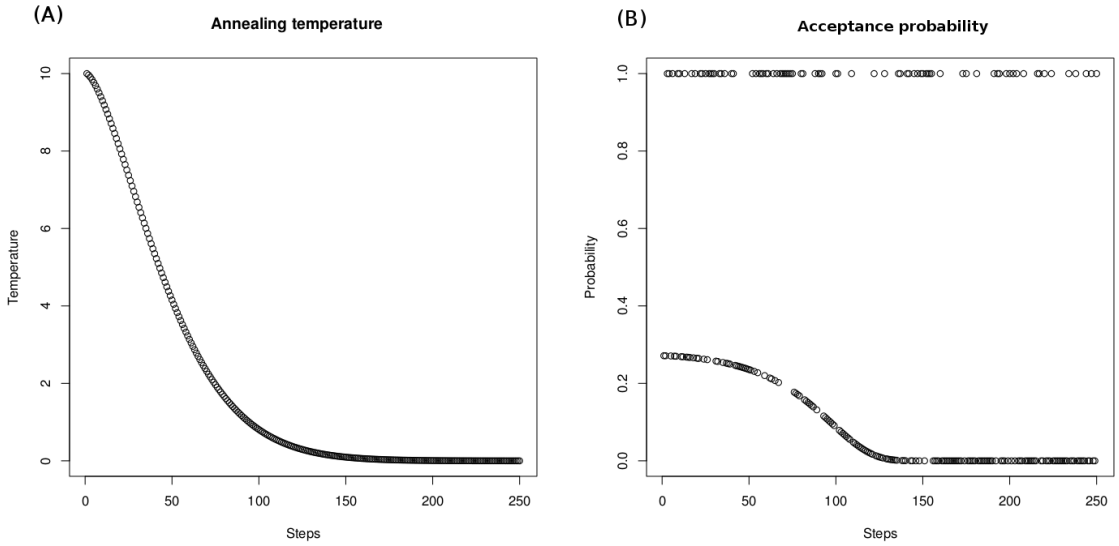

Figure 3.6: Annealing schedule and acceptance probability. (A) Annealing temperature (y-axis) plotted against the iteration number (x-axis). (B) Acceptance probability (y-axis) plotted against the iteration number (x-axis). The dots in the upper line correspond to an acceptance probability of $p = 1$ when the new network wins the classification, the dots in the lower line correspond to a probability of $p = 0.3 \cdot \exp(-1/T)$ in the case when the old network wins.

3.2.5 Simulations and results

In order to compare the performance of the SVM based method with that of the MLE of the CBN model a simulation is performed where the underlying network G_0 of a dataset $\mathbf{D}_0$ is reconstructed using both methods. When using the SVM based method both simulated annealing and hill climbing are used for model search. In the simulation we consider the network sizes $N \in \{5, 6, 8, 10, 12, 20, 30\}$ in the case of the SVM based method and the sizes $N \in \{5, 6, 8, 10, 12\}$ for the MLE of the CBN model. For each network size there are $N_r = 20$ runs. For each network size and run a random graph G_0 is generated using the randomDAG function in the package pcalg [41, 47] (see section A.1) and based on this graph a dataset $\mathbf{D}_0$ is simulated with noise $\epsilon_0 \sim \mathcal{B}(\alpha, \beta)$, where $\alpha = 2$ and $\bar{\epsilon}_0 = \alpha/(\alpha + \beta) = 0.01$, and $\lambda_{0,i} \sim \mathcal{E}(1)$.

Then a network G is inferred to describe the dataset $\mathbf{D}_0$ using three different methods: the MLE of the CBN model, the SVM based method with simulated annealing and the SVM based method with hill climbing. For each of the SVM based methods three different mean noise levels $\bar{\epsilon} \in \{0.01, 0.05, 0.1\}$ are used in the data simulation and $\lambda_{1,i}, \lambda_{2,i} \sim \mathcal{E}(1)$.

The pseudocode of the simulation is given in Algorithm 7 in Appendix A.2.

Results

In order to have a common means to assess the quality of an inferred network G in explaining the original dataset $\mathbf{D}_0$, likelihoods and distances are calculated. For each reconstructed graph the likelihood of the data $\mathbf{D}_0$ given the network G is computed using the likelihood function derived in [34]. Likelihoods are calculated using the graphs inferred with all three methods, namely MLE of the CBN model and the SVM based method both using simulated annealing and hill climbing. Furthermore, likelihoods are also computed when using the true graphs.

In addition, distances between the dataset $\mathbf{D}_0$ and data simulated from the network G are calculated. To this end, a multiple dataset $\mathbf{M}$ is simulated based on G which consists of $N_d = 100$ datasets. Data are simulated with $N_P = 20N$ patients and a mean noise level of $\bar{\epsilon} = 0.01$. For both datasets $\mathbf{M}$ and $\mathbf{D}_0$ the covariance matrix is used as summary statistics. Then the mean Euclidean distance $\bar{\rho}$ between the datasets $\mathbf{d}_{cov,i}$ in $\mathbf{M}$ and the vectorized covariance matrix $\mathbf{d}_{cov,0}$ of the dataset $\mathbf{D}_0$ is

calculated as

$$\bar{\rho} = \frac{\sum_{i=1}^{N_d} \rho(\mathbf{d}_{cov,i}, \mathbf{d}_{cov,0})}{N_d}, \tag{3.17}$$

where $i = 1, \ldots, N_d$ and $\rho(\cdot)$ is the Euclidean distance between $\mathbf{d}_{cov,i}$ and $\mathbf{d}_{cov,0}$ defined as

$$\rho(\mathbf{d}_{cov,i}, \mathbf{d}_{cov,0}) = \sqrt{\sum_{j=1}^{m} (\mathbf{d}_{cov,ij} - \mathbf{d}_{cov,0j})^2}. \tag{3.18}$$

As in the case of the likelihood, distances are computed for the graphs inferred with all three methods and for the true graphs.

The results for the likelihood and distance values are shown in figure 3.7, where figure 3.7 (A) corresponds to a mean noise level of 1%, in (B) a mean noise level of 5% was used and in (C) the mean noise level was 10%.

The results for the likelihood function are shown in figure 3.7 left. The x-axis shows the network size, whereas on the y-axis the likelihood is plotted. The likelihoods corresponding to the graphs inferred with the MLE of the CBN model are shown in black, the likelihood results for graphs obtained with the SVM based method using simulated annealing are depicted in green, and the blue line belongs to the results of the SVM based method using hill climbing. The likelihood results for the true graph are shown in orange. Every dot represents the average likelihood over $N_r = 20$ runs.

In figure 3.7 left one can see that the highest likelihoods for the observed dataset $\mathbf{D}_0$ are obtained in the case of the true graphs (orange line). The likelihoods obtained when using the graphs inferred with the MLE of the CBN model are only slightly lower. The likelihood values calculated for the graphs obtained with the SVM based method are lower both for simulated annealing and for hill climbing, with the likelihoods being higher when using hill climbing.

The results for the distance values are shown in figure 3.7 right. The x-axis shows the network size, the y-axis corresponds to the distance. The black line corresponds to the distance results for the MLE of the CBN model, green belongs to the results of the SVM based method using simulated annealing and the distance results obtained for the SVM based method using hill climbing are shown in blue. The distance results for the true graph are shown in orange. The dots represent average values over $N_r = 20$ runs.

In figure 3.7 right one can see that the distances are lowest between the observed

dataset $\mathbf{D}_0$ and data simulated from the graph obtained with the SVM based method using hill climbing (blue line). The distances of $\mathbf{D}_0$ to the data simulated from the true graph (orange line) and from the graph reconstructed with the MLE of the CBN model (black line) are higher but very close to the results obtained with the SVM based method using hill climbing. Starting from a network size of 20 in the case of $\bar{\epsilon} = 0.01$ and a size of 30 for $\bar{\epsilon} = 0.05$ the distances of the SVM based method using hill climbing are higher than those obtained for the true graph. The distances are highest for the SVM based method using simulated annealing (green line).

Calculation times

Simulations were performed on a high performance computing (HPC) cluster. The technical details can be found in Appendix A.7.

Figure 3.8 shows the average calculation times of the three methods for data simulated with a mean noise of $\bar{\epsilon} = 0.01$. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the calculation times of the MLE of the CBN model, green belongs to the machine learning based method using simulated annealing and the calculation times of the machine learning based method using hill climbing are shown in blue.

It can be seen that the calculation times of the MLE of the CBN model increase dramatically starting from a network size of 12 nodes, where the calculation time is close to 50 hours. The calculation time of the machine learning based method is close to 50 hours starting from a network size of 30 nodes. Calculation times are nearly equal for model search done with simulated annealing and with hill climbing.

The calculation times for mean noise levels $\bar{\epsilon} \in \{0.05, 0.1\}$ are very similar to those shown in figure 3.8. These are shown in figures A.4 and A.5 in Appendix A.5.

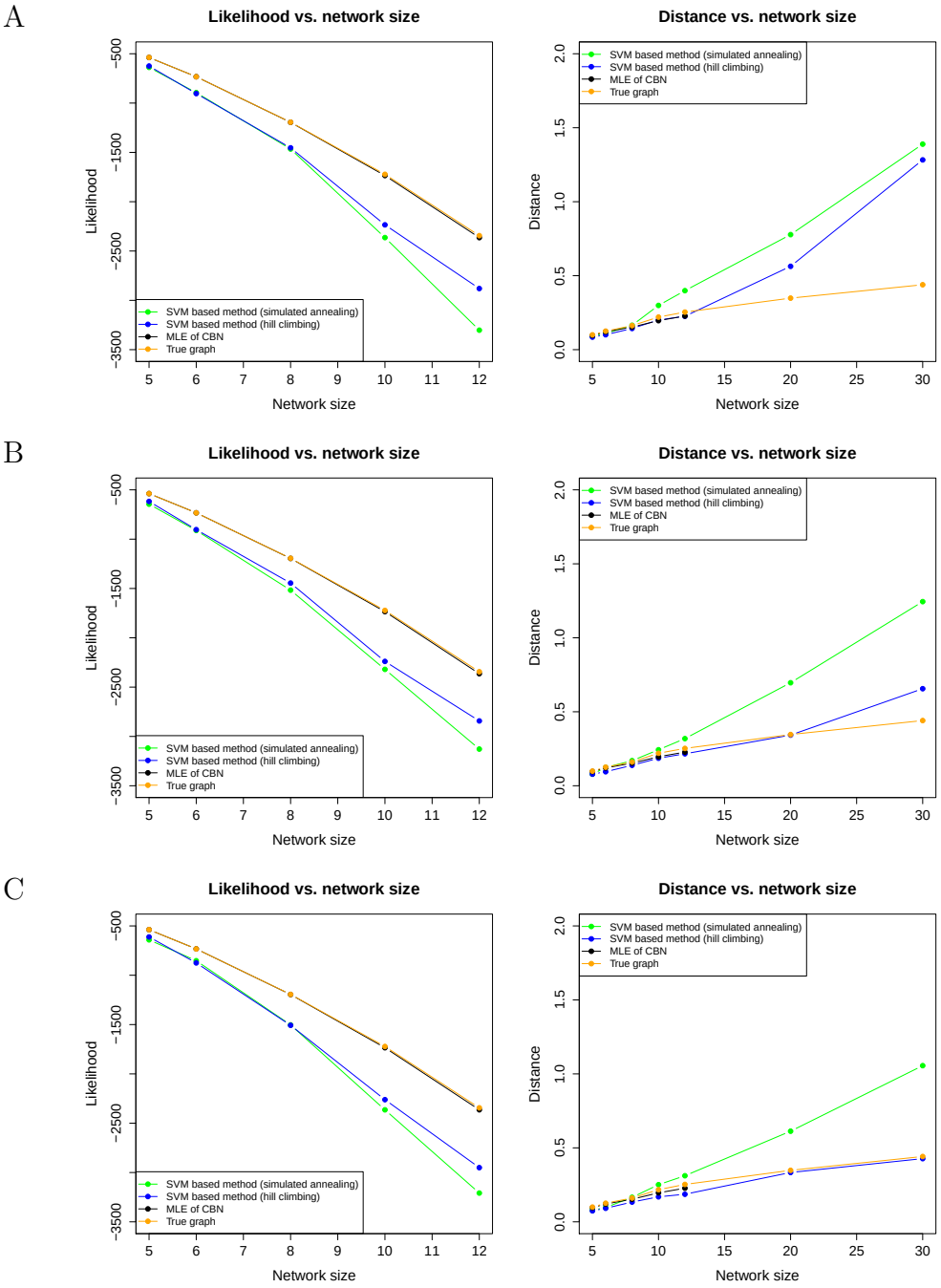

Figure 3.7: Comparison of the results for the SVM based method and the MLE of the CBN model. Left: likelihood results using a mean noise of (A) 1 %, (B) 5 % and (C) 10 %. The x-axis shows the network size, whereas on the y-axis the likelihoods are plotted. The dots represent average values over $N_r = 20$ runs. The green line corresponds to the SVM based method using simulated annealing, blue belongs to the SVM based method using hill climbing, the black line corresponds to the MLE of the CBN model and the results for the true graph are shown in orange. Right: distance results using a mean noise of (A) 1 %, (B) 5 % and (C) 10 %. The x-axis shows the network size, whereas on the y-axis the distances are plotted.

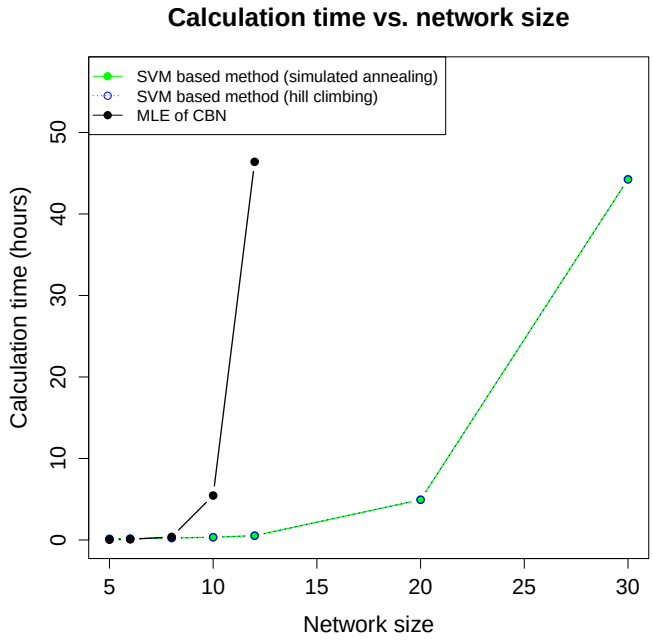

Figure 3.8: Calculation times of the SVM based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.01$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model, green belongs to the SVM based method using simulated annealing and the calculation times of the machine learning based method using hill climbing are shown in blue.

3.2.6 Inferring the underlying tumor progression models for real datasets

The two SVM based methods are applied to the three real datasets described in section 1.4 for inferring the underlying tumor progression models. For network inference, the noise levels as estimated in Gerstung et al. [34] are used and the waiting time parameters λ_i, $i \in \{1, \ldots, N\}$, are sampled from an exponential distribution $\lambda_i \sim \mathcal{E}(1)$.

Figures 3.9 and 3.10 show the graphs obtained when using the SVM based method with simulated annealing.

In figure 3.9 the two graphs obtained for renal cell carcinoma data are shown, where mean noise levels of $\bar{\epsilon} = 0.01$ (left) resp. $\bar{\epsilon} = 0.08$ (right) were used in data simulation. In Gerstung et al. [34] a noise level of $\bar{\epsilon} = 0.01$ was estimated. The resulting graph contained only two relations, namely -4q $\rightarrow$ -4p and +17q $\rightarrow$ +17p. These two relations were also found by the SVM based method using simulated annealing. In addition, several other relations were found. Due to the only small similarity of the graph found by Gerstung et al. [34] and the tree published by Jiang et al. [46] for the same dataset, an additional graph was inferred in [34] with an estimated noise level of $\bar{\epsilon} = 0.08$. This graph and the graph shown in figure 3.9 (right) have the relations -4q $\rightarrow$ -6q $\rightarrow$ -Xp and +17q $\rightarrow$ -Xp in common. In addition, the relation -3p $\rightarrow$ -9p is present indirectly through two intermediate steps in -3p $\rightarrow$ -Xp $\rightarrow$ +Xp $\rightarrow$ -9p.

In figure 3.10 (left) the graph inferred for breast cancer data is presented, where a mean noise level of $\bar{\epsilon} = 0.15$ was used. The graph coincides with the graph found in [34] in the relations +8q $\rightarrow$ -8p $\rightarrow$ -11q and +20q $\rightarrow$ -13q. The relation +8q $\rightarrow$ +3q is present indirectly through two intermediate steps in +8q $\rightarrow$ -8p $\rightarrow$ -11q $\rightarrow$ +3q.

The graph obtained for colorectal cancer data is shown in figure 3.10 (right). Here, a mean noise level of $\bar{\epsilon} = 0.11$ was used in data simulation. The inferred graph has two relations in common with the graph found in [34], namely -18q $\rightarrow$ -8p and -18q $\rightarrow$ -15q. The latter relation is only present indirectly in -18q $\rightarrow$ +13q $\rightarrow$ -15q.

Figures 3.11 to 3.12 show the graphs obtained when using the SVM based method with hill climbing.

Figure 3.11 presents the two graphs obtained for renal cell carcinoma data, where mean noise levels of $\bar{\epsilon} = 0.01$ (left) resp. $\bar{\epsilon} = 0.08$ (right) were used in data simulation.

The left graph contains four relations which were also found in [34]. These are +17q → +17p, -4q → -4p, -3p → -6q and -3p → -4q which is contained indirectly in the relation -3p → -13q → -4q. The right graph has only two relations which coincide with those found in [34], namely -4q → -4p through -4q → -6q → -4p and -4q → -6q.

In figure 3.12 (left) the graph obtained for breast cancer data is presented. Here, a mean noise level of $\bar{\epsilon} = 0.15$ was used. In this graph three relations are found to be in common with the graph inferred in [34]. These are +8q → -8p → -11q, +17q → +16p and +17q → +20q. The latter is contained indirectly in +17q → +1q → +20q.

The graph inferred for colorectal cancer data is shown in figure 3.12 (right), where a mean noise level of $\bar{\epsilon} = 0.11$ was used in data simulation. The graph has three relations in common with the graph inferred in [34], namely +20q → +13q, +20q → -18q → -15q and +20q → +8q indirectly through +20q → +13q → +8q.

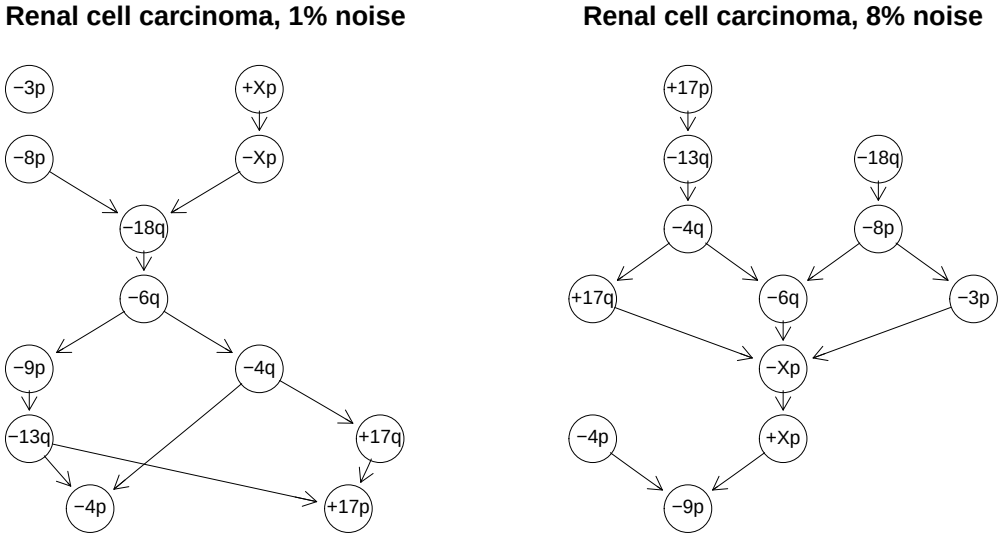

Figure 3.9: Tumor progression models inferred for renal cell carcinoma data with the SVM based method using simulated annealing. Left: progression model obtained using a mean noise level of $\bar{\epsilon} = 0.01$ in data simulation, right: progression model inferred when using a mean noise level of $\bar{\epsilon} = 0.08$.

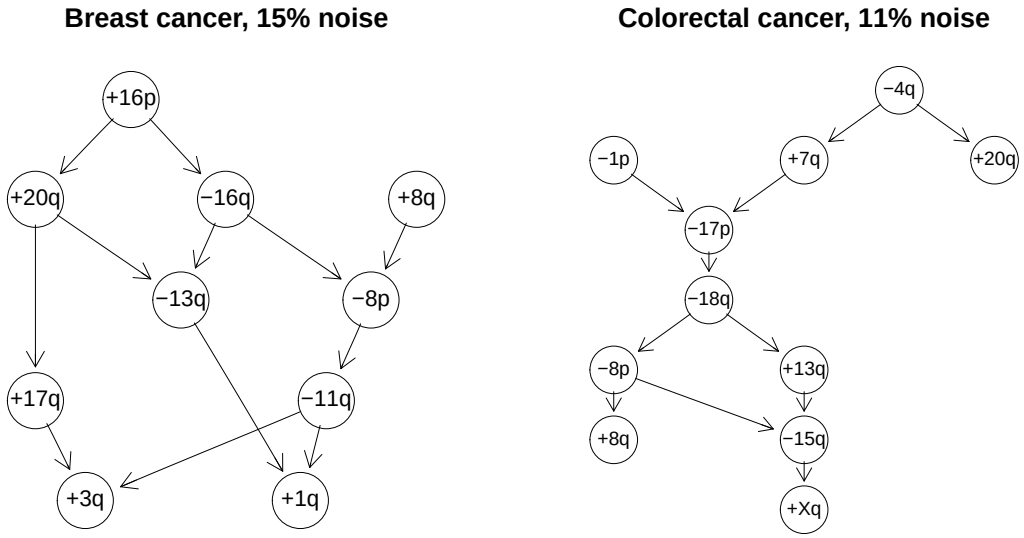

Figure 3.10: Tumor progression models inferred for breast cancer and colorectal cancer data with the SVM based method using simulated annealing. Left: progression model obtained for breast cancer data where a mean noise level of $\bar{\epsilon} = 0.15$ was used in data simulation. Right: progression model inferred for colorectal cancer data using a mean noise level of $\bar{\epsilon} = 0.11$.

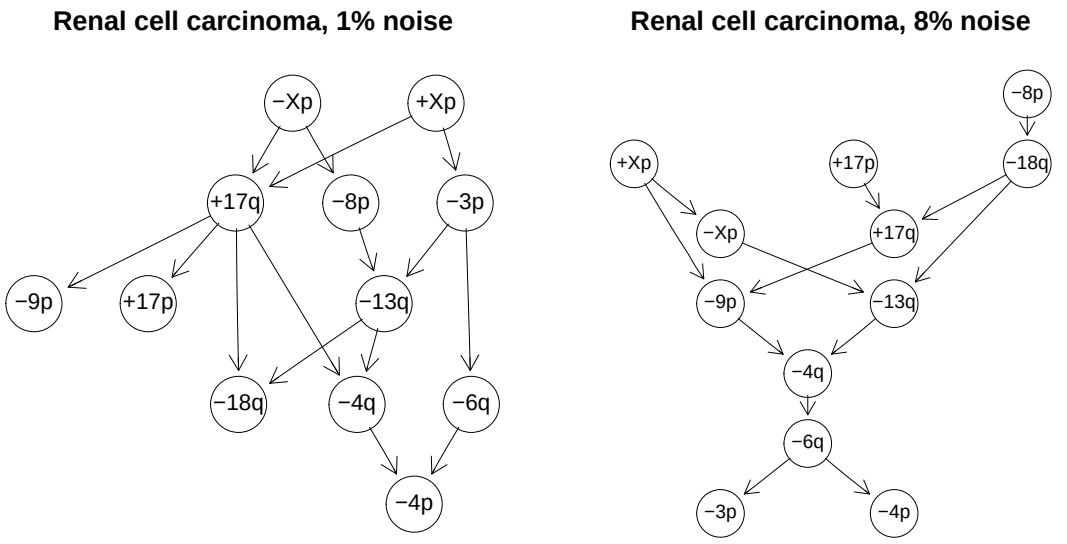

Figure 3.11: Tumor progression models inferred for renal cell carcinoma data with the SVM based method using hill climbing. Left: progression model obtained using a mean noise level of $\bar{\epsilon} = 0.01$ in data simulation, right: progression model inferred when using a mean noise level of $\bar{\epsilon} = 0.08$.

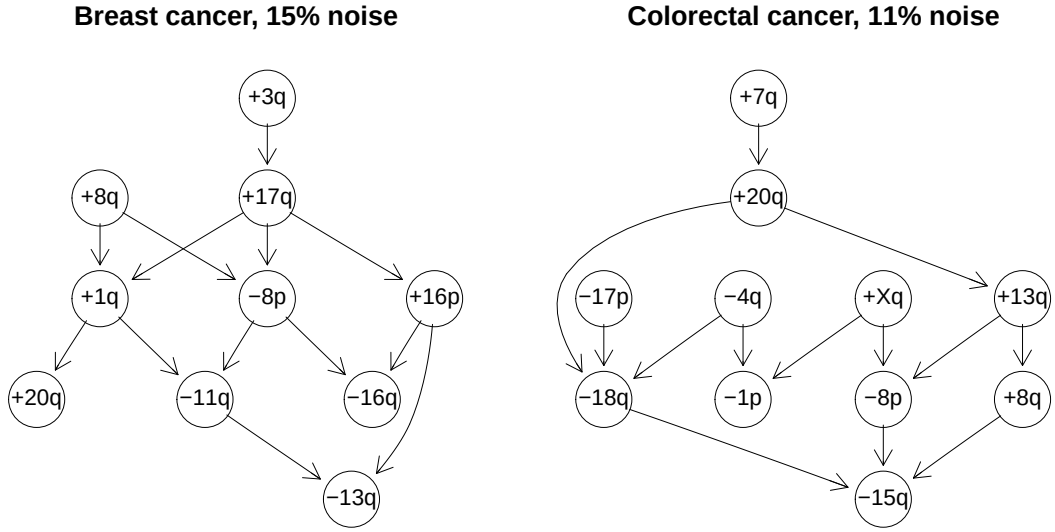

Figure 3.12: Tumor progression models inferred for breast cancer and colorectal cancer data with the SVM based method using hill climbing. Left: progression model obtained for breast cancer data where a mean noise level of $\bar{\epsilon} = 0.15$ was used in data simulation. Right: progression model inferred for colorectal cancer data using a mean noise level of $\bar{\epsilon} = 0.11$.

3.3 Learning of tumor progression models via approximate Bayesian computation

3.3.1 The algorithm

The second method proposed in this thesis also relies on the simulation of artificial data instead of the calculation of likelihoods. The main difference to the machine learning based method described in section 3.2 lies in the way the likelihood function is approximated. Here, network inference is done based on approximate Bayesian computation (ABC).

The algorithm is embedded in a hill climbing like approach for global model search. It consists of individual iterations, where at each iteration two competing networks G_1 and G_2 are compared at a time through their corresponding simulated data. In order to discriminate between the datasets underlying the two different networks, Euclidean distances of the summary statistics of the datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ to the summary statistics of dataset $\mathbf{D}_0$ are calculated. These distances are then used to decide between the two networks.

As in the machine learning based method, the main part of the algorithm can be divided into the three steps simulation, decision and searching. However, in the ABC based approach these steps are not nested.

In the simulation step multiple datasets $\mathbf{M}_1$ resp. $\mathbf{M}_2$ are simulated based on the pair of competing networks G_1 and G_2 as described in section 3.1. The data contain noise, each element of the binary data matrix may be incorrect with probability ϵ. The noise ϵ is sampled from a beta distribution $\epsilon \sim \mathcal{B}(\alpha, \beta)$ with $\bar{\epsilon} = \alpha/(\alpha + \beta)$ and $\alpha = 2$. The waiting time parameters λ_i, $i \in \{1, \ldots, N\}$, are sampled from an exponential distribution $\lambda_i \sim \mathcal{E}(1)$ (see section 3.1).

In the decision step the mean Euclidean distances $\bar{\rho}_1$ and $\bar{\rho}_2$ of the summary statistics of the simulated datasets $\mathbf{M}_1$ resp. $\mathbf{M}_2$ to the summary statistics of the observed dataset $\mathbf{D}_0$ are calculated according to equation (3.17). The decision between the two networks is then made on the basis of the distance values.

In the search step a hill climbing like approach is used for global model search. The method of hill climbing is introduced in section 2.5.2. Here again, hill climbing is used with a negative sign. As an objective function $f(\mathbf{x})$ the mean Euclidean distance $\bar{\rho}_1$

($\bar{\rho}_2$) between the summary statistics of dataset $\mathbf{M}_1$ ($\mathbf{M}_2$) and the summary statistics of $\mathbf{D}_0$ is used. The aim is to find a state G such that the distance $\bar{\rho}$ is minimized.

Model search is initialized with a non-empty graph topology G_1 obtained using the method adapted from Jacob et al. [44] (see p. 53). At each iteration of the hill climbing a current graph G_1 and a candidate topology G_2 are considered. The candidate network G_2 is generated by randomly changing an edge in the network G_1 using the method described on p. 54. The decision between the two graphs is made in a deterministic way, i.e. the graph G_2 is accepted only if $\bar{\rho}_2 < \bar{\rho}_1$. Otherwise graph G_1 is kept. The algorithm stops after a predefined number of iterations, or earlier if the condition of the dynamic stopping criterion is fulfilled (see p. 69).

The pseudocode of the method is given in Algorithm 8 in Appendix A.2.

3.3.2 Approximate Bayesian computation scheme

The algorithm described above is similar to the ABC rejection algorithm that uses summary statistics (algorithm C on p. 39) and to the MCMC approach by Marjoram et al. [52] (algorithm D on p. 39) when using D3″ instead of step D3. It can be summarized in the following ABC scheme:

E1. Given G_1 propose a move to G_2. Generate ϵ_2 and $\boldsymbol{\lambda_2}$ from the prior.
E2. Simulate data $\mathbf{M}_2$ based on graph G_2 using parameters ϵ_2 and $\boldsymbol{\lambda_2}$.
E3. Calculate the distance $\bar{\rho}_2$ between the summary statistics of $\mathbf{M}_2$ and the summary statistics of $\mathbf{D}_0$.
E4. If $\bar{\rho}_2 < \bar{\rho}_1$ accept the candidate graph G_2, otherwise keep graph G_1.
E5. Return to 1.

Step E1 of the scheme combines steps C1 and D1. A new graph topology G_2 is generated based on the previous topology, similar as in the MCMC approach. The parameters ϵ_2 and $\lambda_{2,i}$ are drawn from the prior as it is done in the ABC rejection algorithm.

The difference between algorithm E and the algorithms C and D lies in the way a candidate graph is accepted. Here, the candidate graph G_2 is accepted if $\bar{\rho}_2 < \bar{\rho}_1$, otherwise graph G_1 is kept.

The algorithm outputs a graph to describe the observed dataset $\mathbf{D}_0$ after a predefined number of iterations, or if the condition of the dynamic stopping criterion is fulfilled (see p. 69).

3.3.3 Simulations and results

In order to compare the performance of the ABC based method with that of the MLE of the CBN model a simulation is made where the underlying network G_0 of a dataset $\mathbf{D}_0$ is reconstructed using both methods. In the simulation the network sizes $N \in \{5, 6, 8, 10, 12, 20, 30\}$ are considered in the case of the ABC based method and the sizes $N \in \{5, 6, 8, 10, 12\}$ are used for the MLE of the CBN model. There are $N_r = 20$ runs for each network size. For each network size and run we generate a random graph G_0 using the randomDAG function in the package pcalg [41, 47] (see Appendix A.1) and simulate a dataset $\mathbf{D}_0$ based on this graph with noise $\epsilon_0 \sim \mathcal{B}(\alpha, \beta)$, where $\alpha = 2$ and $\bar{\epsilon}_0 = \alpha/(\alpha + \beta) = 0.01$, and $\lambda_{0,i} \sim \mathcal{E}(1)$.

Then a graph G is inferred describing the dataset $\mathbf{D}_0$ using two different methods: the MLE of the CBN model and the ABC based method. For the ABC based method three different mean noise levels $\bar{\epsilon} \in \{0.01, 0.05, 0.1\}$ are used in the simulated data and $\lambda_{1,i}, \lambda_{2,i} \sim \mathcal{E}(1)$.

The pseudocode of the simulation can be found in Algorithm 9 in Appendix A.2.

Parameter choice

Due to time constraints parameters are chosen to be the same as in the SVM based approach. This means that data progression is adapted such that in the data the number of patients with further progressed tumors is higher than of those with less progressed tumors (see section 3.1). As a summary statistics the covariance matrix is chosen.

In the model search the number of iterations is chosen as $N_{iter} = \max(250, 2 \cdot N^2)$ as in the SVM based approach. For network sizes of more than 12 nodes the same dynamic stopping criterion as in the previous method is used: the algorithm stops if the network has been constant during $5 \cdot N$ iterations.

Results

As in the SVM based method, in order to have a common measure for the quality of the inferred network G in describing the observed dataset $\mathbf{D}_0$, we calculate likelihoods and distances. This is done as described in section 3.2.5.

The results when using the likelihood function are shown in figure 3.13 left. On the x-axis the network size is shown, whereas on the y-axis the likelihood is plotted. The likelihoods corresponding to the graphs inferred with the MLE of the CBN model are shown in black and the results for graphs inferred with the ABC based method are depicted in green. The likelihood results for the true graph are shown in orange. The dots represent average values over $N_r = 20$ runs.

In figure 3.13 left one can see that the observed dataset $\mathbf{D}_0$ has the highest likelihoods in the case of the true graph (orange line). The likelihoods obtained when using the graphs inferred with the MLE of the CBN model (black line) are very similar but slightly lower. The likelihood values calculated for the graphs obtained with the ABC based method (green line) are lowest.

The results for the distance values are shown in figure 3.13 right. The x-axis shows the network size, whereas on the y-axis the distance is plotted. The black line corresponds to the distance results for the MLE of the CBN model, green corresponds to the results of the ABC based method. The distance results for the true graph are shown in orange. The dots represent average values over $N_r = 20$ runs.

In figure 3.13 right one can see that the distances are lowest between the observed dataset $\mathbf{D}_0$ and data simulated from the graph reconstructed with the MLE of the CBN model (black line). The distances obtained when using the true graph (orange line) are very close to these results. The distances are highest for the ABC based method (green line).

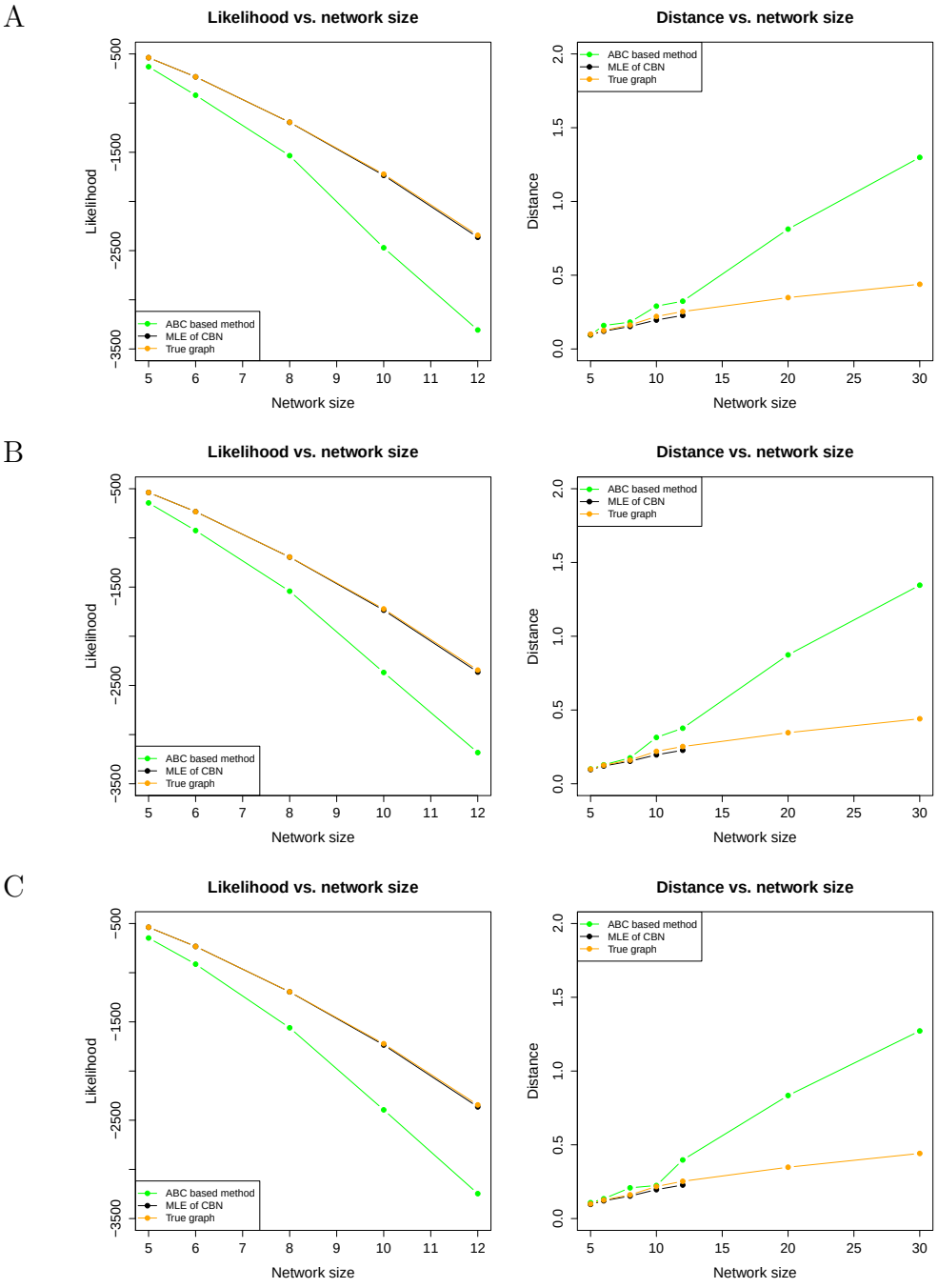

Figure 3.13: Comparison of the results for the ABC based method and the MLE of the CBN model. Left: likelihood results using a mean noise of (A) 1 %, (B) 5 % and (C) 10 %. The x-axis shows the network size, whereas on the y-axis the likelihoods are plotted. The dots represent average values over $N_r = 20$ runs. The green line corresponds to the ABC based method, the black line corresponds to the MLE of the CBN model and the results for the true graph are shown in orange. Right: distance results using a mean noise of (A) 1 %, (B) 5 % and (C) 10 %. The x-axis shows the network size, whereas on the y-axis the distances are plotted.

Calculation times

Simulations were performed on a HPC cluster. The technical details can be found in Appendix A.7.

Figure 3.14 shows the average calculation times of the two methods for data simulated with a mean noise of $\bar{\epsilon} = 0.01$. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the calculation times of the MLE of the CBN model, green belongs to the ABC based method.

It can be seen that the calculation times of the MLE of the CBN model increase dramatically starting from a network size of 12 nodes, where the average calculation time is close to 50 hours. The average calculation time of the ABC based method reaches only about 15 hours with a network size of 30 nodes.

The calculation times for mean noise levels $\bar{\epsilon} \in \{0.05, 0.1\}$ are very similar to those shown in figure 3.14. These are shown in figures A.6 and A.7 in Appendix A.6.

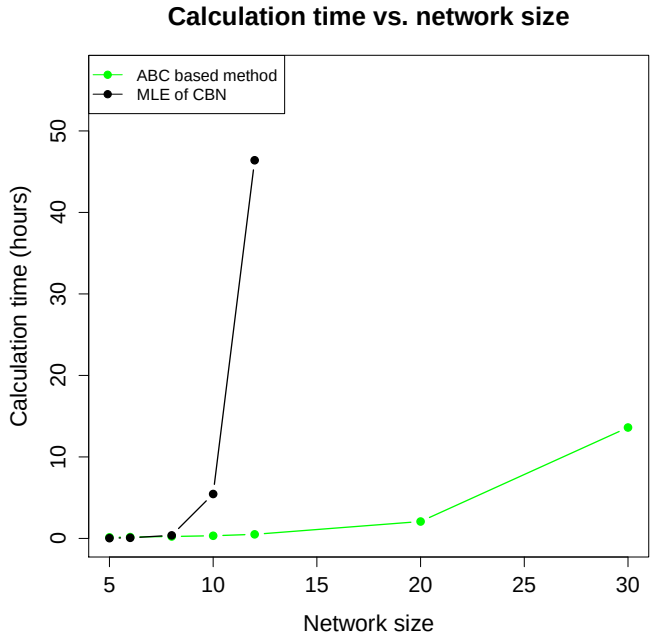

Figure 3.14: Calculation times of the ABC based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.01$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model and the green line belongs to the ABC based method.

3.3.4 Inferring the underlying tumor progression models for real datasets

The ABC based method is applied to the three real datasets described in section 1.4 for inferring the underlying tumor progression models. As in the SVM based method, for network inference the noise levels which were estimated in Gerstung et al. [34] are used and the waiting time parameters λ_i, $i \in \{1, \ldots, N\}$, are sampled from an exponential distribution $\lambda_i \sim \mathcal{E}(1)$.

Figures 3.15 and 3.16 show the graphs obtained when using the ABC based method with hill climbing.

Figure 3.9 shows the two graphs obtained for renal cell carcinoma data. The mean noise levels used in data simulation were $\bar{\epsilon} = 0.01$ (left) resp. $\bar{\epsilon} = 0.08$ (right). For both noise levels the ABC based method could only detect a single relation which is -4q $\rightarrow$ -4p. This relation was also found in [34] in the inferred networks both with an estimated noise of 0.01 and 0.08.

In figure 3.10 (left) the graph obtained for breast cancer data is shown. Here, a mean noise level of $\bar{\epsilon} = 0.15$ was used. The graph has three relations in common with the graph found in [34], namely +8q $\rightarrow$ +3q $\rightarrow$, +1q $\rightarrow$ +3q and +17q $\rightarrow$ +20q.

The graph inferred for colorectal cancer data is shown in figure 3.10 (right). Here, a mean noise level of $\bar{\epsilon} = 0.11$ was used in data simulation. The graph coincides with the graph found in [34] in two relations. These are -18q $\rightarrow$ -1p and -18q $\rightarrow$ -15q which is contained indirectly in -18q $\rightarrow$ +7q $\rightarrow$ -15q.

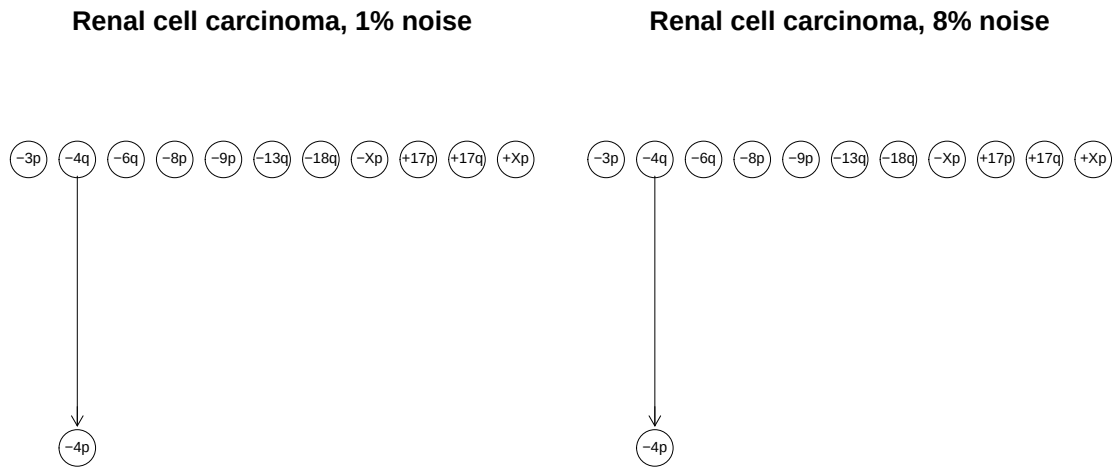

Figure 3.15: Tumor progression models inferred for renal cell carcinoma data with the ABC based method using hill climbing. Left: progression model obtained using a mean noise level of $\bar{\epsilon} = 0.01$ in data simulation, right: progression model inferred when using a mean noise level of $\bar{\epsilon} = 0.08$.

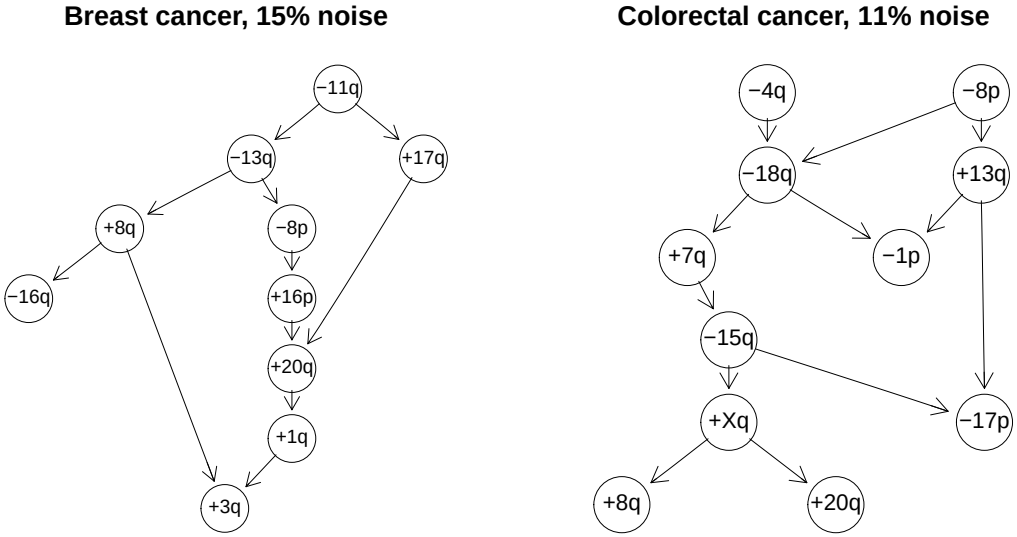

Figure 3.16: Tumor progression models inferred for breast cancer and colorectal cancer data with the ABC based method using hill climbing. Left: progression model obtained for breast cancer data where a mean noise level of $\bar{\epsilon} = 0.15$ is used in data simulation. Right: progression model inferred for colorectal cancer data using a mean noise level of $\bar{\epsilon} = 0.11$.

Chapter 4

Summary and outlook

Since it became clear that the number of newly diagnosed cancer cases and the number of deaths from cancer worldwide increases from year to year, a great effort has been put into the field of cancer research. One major point of interest is how and by means of which intermediate steps the development of cancer from an initially benign mass of cells into a large malignant and deadly tumor takes place. In order to shed light onto the details of this process, many models have been developed in the recent decades, the most prominent among them being the linear path model by Fearon and Vogelstein [29] which describes the stages of tumor progression in colorectal cancer.

Since then many more complex models have been developed for describing tumor progression. However, with increasing complexity of the model the computational time also increases. This is especially the case for models where network inference is based on the analysis of the likelihood function when the maximization of the likelihood can not be done in closed form.

In this thesis two alternative methods for the inference of tumor progression models were proposed which do not use a likelihood function but instead rely on simulated data. The two methods are based on support vector machines (SVMs) and approximate Bayesian computation (ABC), respectively, to decide between competing networks. In the SVM based method two different model search algorithms, simulated annealing and hill climbing, are used, while in the ABC based method only hill climbing is used. The performance and runtime of the two methods were compared to that of the maximum likelihood estimation (MLE) of the conjunctive Bayesian network (CBN) model.

The performance of the two methods in reconstructing an underlying network was compared to that of the MLE of the CBN model in a simulation study. In this study, a graph G_0 and corresponding simulated data $\mathbf{D}_0$ were generated and on the basis of the data each of the methods was used in turn to reconstruct the underlying graph.

In order to assess and compare the performance of the methods likelihood and distance values were calculated. The likelihood of the data $\mathbf{D}_0$ given the reconstructed graph G was computed using the likelihood function derived in [12]. To obtain the distance value, the mean Euclidean distance between the summary statistics of the dataset $\mathbf{D}_0$ and the summary statistics of data simulated from the graph G is calculated.

It was found that the likelihood values for the observed data $\mathbf{D}_0$ were always higher when using the MLE of the CBN model for network reconstruction compared to those obtained both for the SVM based method. Among the two model search algorithms used for the SVM based method the likelihood for the observed data $\mathbf{D}_0$ was higher when using hill climbing compared to the SVM based method using simulated annealing.

When looking at the distance values the performance of the methods depended on the noise level used in the simulated data. With a noise level of 10% the distances between the observed dataset $\mathbf{D}_0$ and data simulated from the networks reconstructed with the SVM based method using hill climbing and the MLE of the CBN model were very similar and were also very close to the results obtained for the true graph. For noise levels of 1% and 5% the distances of the SVM based method using hill climbing started to be higher than those obtained for the true graph from a network size of 20 and 30 nodes, respectively. The distances obtained for the graph reconstructed with the SVM based method using simulated annealing were always highest.

In the case of the ABC based method the distances obtained for the MLE of the CBN model were always close to those of the true graph, while the distances obtained for the networks reconstructed with the ABC based method were always higher.

The average calculation time for the MLE of the CBN model was found to be close to 50 hours when using networks with a size of 12 nodes, while both the SVM based and the ABC based method need only about 30 minutes for the same network size. It could further be shown in this thesis that in the case of the SVM based method an average calculation time of about 50 hours is reached only starting from a network

size of 30 nodes, while for the ABC based method the average calculation time is only about 15 hours for networks with 30 nodes.

In addition, both the SVM based method and the ABC based method were applied to three real tumor datasets for network inference. Although the reconstructed networks differ from those obtained when using the MLE of the CBN model, all networks have one or several relations in common with the CBN results. From those networks one can see that the two methods developed in this thesis produce results that are comparable to those obtained with the MLE of the CBN model.

Outlook

Different networks were obtained when applying the likelihood-free methods and the MLE of the CBN method to real tumor datasets. It is at this point difficult to say which of the inferred graphs better describes the true underlying progression model. For an impartial comparison of the performance of the different methods, a biological verification of the results is indispensable.

It should be noted that the MLE of the CBN method is implemented in C while the SVM based and the ABC based method are written in R. From experience it can be said that a program that is implemented in C in general runs faster than the same program written in R. Hence, the likelihood-free methods could be made even faster by implementing them in C.

Since the likelihood-free methods are much faster than the MLE of the CBN method, but the MLE of the CBN method can reconstruct an underlying network more accurately, I suggest to use a combination of the two methods to obtain fast and optimal results. This means to use the likelihood-free methods to find a starting point for the MLE based search. Then the MLE of the CBN method can be used to further refine the model.

Appendix A

Appendix

A.1 Generation of a random graph topology

Random subsets of the space of directed acyclic graphs (DAG) are generated using the randomDAG function in the R-package pcalg [41, 47]. In the resulting graph nodes are ordered having numbers from 1 to N. Construction of a graph always starts with node 1. Assuming that the number of nodes with higher order is k, the number of neighboring nodes is drawn from the binomial distribution $\mathcal{B}(k, q_n)$, where q_n is the probability that a node is connected to a node with higher topological order. Throughout this thesis we use $q_n = 0.5$. The corresponding nodes are then drawn without replacement among the nodes with higher order. The procedure is repeated for the next node in the original ordering until all nodes are placed in the graph.

In the obtained graph there will be situations where several paths of different length lead to a certain node. However, the graph would then describe a progression model in which, using the shorter path, certain events can occur even if some of their parent events have not happened yet. In order to avoid such situations the obtained graph needs to be transitively reduced to agree with the definition of CBNs.

A.2 Pseudocodes

Algorithm 1 Pseudocode of the machine learning approach (simulated annealing).

Initialize model search: infer graph G_1 from dataset $\mathbf{D}_0$ (Jacob et al. [44]).
Simulate multiple dataset $\mathbf{M}_1$ based on G_1 using parameters ϵ_1 and $\boldsymbol{\lambda}_1$.
Initialize annealing temperature: $T \leftarrow 10$.
for $i = 1 : N_{iter}$ **do**
    Generate graph G_2 by randomly changing one edge in graph G_1.
    Simulate multiple dataset $\mathbf{M}_2$ based on G_2 using parameters ϵ_2 and $\boldsymbol{\lambda}_2$.
    Form training dataset $\mathbf{D}_{train}$ using $\mathbf{M}_1$ and $\mathbf{M}_2$ and the class labels.
    Train an SVM model on $\mathbf{D}_{train}$.
    Use the model to get a class prediction $Pred$ for dataset $\mathbf{D}_0$.
    if $Pred == G_1$ **then**
        Calculate acceptance probability $P \leftarrow 0.3 \cdot \exp(-1/T)$.
        Draw a random number from a uniform distribution $\mathcal{U}(0, 1)$.
        if random number $< P$ **then**
            Accept new graph: $G_1 \leftarrow G_2$ and
            Copy data: $\mathbf{M}_1 \leftarrow \mathbf{M}_2$.
        else
            Keep old graph
        end if
    else // if $Pred == G_2$
        Accept new graph: $G_1 \leftarrow G_2$ and
        Copy data $\mathbf{M}_1 \leftarrow \mathbf{M}_2$.
    end if
    Update temperature: $T \leftarrow T \cdot (1 - (0.03 \cdot \sqrt{i}/N))$
    if $(N > 12)$ **AND** (graph constant during more than $5N$ iterations) **then**
        break
    end if
end for

Algorithm 2 Pseudocode of the machine learning approach (hill climbing).

Initialize model search: infer graph G_1 from dataset $\mathbf{D}_0$ (Jacob et al. [44]).
Simulate multiple dataset $\mathbf{M}_1$ based on G_1 using parameters ϵ_1 and $\boldsymbol{\lambda}_1$.
for $i = 1 : N_{iter}$ **do**
    Generate graph G_2 by changing one edge in graph G_1.
    Simulate multiple dataset $\mathbf{M}_2$ based on G_2 using parameters ϵ_2 and $\boldsymbol{\lambda}_2$.
    Form training dataset $\mathbf{D}_{train}$ using $\mathbf{M}_1$ and $\mathbf{M}_2$ and the class labels.
    Train an SVM model on $\mathbf{D}_{train}$.
    Use the model to get a class prediction $Pred$ for dataset $\mathbf{D}_0$.
    if $Pred == G_1$ **then**
        Accept new graph: $G_1 \leftarrow G_2$ and
        Copy data: $\mathbf{M}_1 \leftarrow \mathbf{M}_2$.
    else
        Keep old graph
    end if
    if $(N > 12)$ **AND** (graph constant during more than $5N$ iterations) **then**
        break
    end if
end for

Algorithm 3 Pseudocode of simulation to visualize data.

for $i \in$ (graph pairs) **do**
    for $j \in (N_d, N_P)$ **do**
        Simulate N_d datasets both based on graph G_1 and G_2 using parameters
          ϵ_1, ϵ_2, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$.
        for $k \in (\boldsymbol{\Sigma}, \boldsymbol{\Sigma}^{-1}, \boldsymbol{\Omega})$ **do**
            Compute summary statistics $\mathbf{S}_{1,(1,\ldots,N_d)}$ and $\mathbf{S}_{1,(2,\ldots,N_d)}$ from the data.
            Compute the averages $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$.
            Compute the difference $\Delta\mathbf{S} = \bar{\mathbf{S}}_1 - \bar{\mathbf{S}}_2$.
        end for
    end for
end for

Algorithm 4 Pseudocode of simulation to find optimal data progression.

```
for i ∈ (progression levels) do
    for j ∈ (graph pairs) do
        for k ∈ (N_d, N_P) do
            Simulate N_d datasets both based on graph G_1 and G_2 using parameters
              ε_1, ε_2, λ_1 and λ_2.
            for l ∈ (Σ, Σ^{-1}, Ω) do
                for m = 1 : N_r do
                    Compute multiple datasets M_1 and M_2 from the data
                      using summary statistics l.
                    Form training dataset D_train using M_1 and M_2.
                    Train an SVM model on D_train and compute the training error
                      using 10-fold cross validation.
                end for
                Compute the average training error over N_r runs.
            end for
        end for
    end for
end for
```

Algorithm 5 Pseudocode of simulation to optimize parameter C.

```
for i = 1 : N_r do
    Generate a random graph G_1 of size N = 5.
    Generate a graph G_2 by randomly changing one edge in G_1.
    for j ∈ (N_d, N_P) do
        Simulate multiple datasets M_1 and M_2 based on graphs G_1 and G_2 with
          parameters ε_1, ε_2, λ_1 and λ_2 using the covariance matrix as summary
          summary statistics.
        Form training dataset D_train using M_1 and M_2.
        for k ∈ C do
            Train an SVM model on D_train using C[k] as cost value and compute
              the training error via 10-fold cross validation
        end for
        Record the value of C with minimal training error. In case of several
          minima record all corresponding values.
    end for
end for
```

Algorithm 6 Pseudocode of simulation to find optimal kernel.

for $i \in N$ **do**
  for $j = 1 : N_r$ **do**
    Generate random graph G_1 of size $N[i]$.
    Generate graph G_2 by randomly changing one edge in G_1.
  end for
end for
for $i \in (N_d, N_P)$ **do**
  for $j \in N$ **do**
    for $k = 1 : N_r$ **do**
      Simulate multiple datasets $\mathbf{M}_1$ and $\mathbf{M}_2$ based on graphs G_1 and G_2
        using parameters ϵ_1, ϵ_2, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$.
      for $m \in$ (summary statistics) **do**
        Form training dataset $\mathbf{D}_{train}$ from $\mathbf{M}_1$ and $\mathbf{M}_2$ using summary
          statistics m.
        for $n \in$ (kernels) **do**
          Train an SVM model on $\mathbf{D}_{train}$ using kernel n and compute
            the training error via 10-fold cross validation
        end for
      end for
      Compute the average training error over 100 runs for each kernel
        and summary statistics.
    end for
  end for
end for

Algorithm 7 Pseudocode of simulation to compare the SVM based method with the MLE of the CBN model.

for $i \in N$ **do**
  for $j = 1 : N_r$ **do**
    Generate random graph G_0 of size $N[i]$
    Simulate dataset $\mathbf{D}_0$ based on G_0 using parameters ϵ_0 and $\boldsymbol{\lambda}_0$.
    for $k \in$ methods **do**
      for $m \in \bar{\epsilon}$ **do** // only for SVM based method
        Reconstruct the network G underlying dataset $\mathbf{D}_0$ using method k
        Compute the likelihood of the data $\mathbf{D}_0$ given G and the
          distance between data $\mathbf{D}_0$ and data simulated based on G
      end for
    end for
  end for
end for

Algorithm 8 Pseudocode of the ABC based method.

Initialize model search: infer graph G_1 from dataset $\mathbf{D}_0$ (Jacob et al. [44]).
Simulate multiple dataset $\mathbf{M}_1$ based on G_1 using parameters ϵ_1 and $\boldsymbol{\lambda}_1$.
for $i = 1 : N_{iter}$ **do**
    Generate graph G_2 by randomly changing one edge in graph G_1.
    Draw ϵ_2 and $\boldsymbol{\lambda}_2$ from the prior.
    Simulate multiple dataset $\mathbf{M}_2$ based on G_2 using parameters ϵ_2 and $\boldsymbol{\lambda}_2$.
    Compute the mean Euclidean distances $\bar{\rho}_1$ and $\bar{\rho}_2$.
    if $\bar{\rho}_2 < \bar{\rho}_1$ **then**
        Accept new graph: $G_1 \leftarrow G_2$ and
        Copy data: $\mathbf{M}_1 \leftarrow \mathbf{M}_2$.
    else
        Keep old graph
    end if
    if $(N > 12)$ **AND** (graph constant during more than $5N$ iterations) **then**
        break
    end if
end for

Algorithm 9 Pseudocode of simulation to compare the ABC based method with the MLE of the CBN model.

for $i \in N$ **do**
    for $j = 1 : N_r$ **do**
        Generate random graph G_0 of size $N[i]$
        Simulate dataset $\mathbf{D}_0$ based on G_0 using parameters ϵ_0 and $\boldsymbol{\lambda}_0$.
    end for
end for
for $i \in$ methods **do**
    for $j \in \bar{\epsilon}$ **do** // if not CBN
        for $k \in N$ **do**
            for $m = 1 : N_r$ **do**
                Reconstruct the network G underlying dataset $\mathbf{D}_0$ using method i
                Compute the likelihood of the reconstructed network and the
                  distance between data $\mathbf{D}_0$ and data simulated based on G
            end for
        end for
    end for
end for

A.3 Results for optimization in data simulation

The images shown in figures A.1 to A.3 contain the results for simulations described in section 3.2.4 (p. 55 ff.). The difference to figures 3.2 to 3.4 is that here, simulations were done with more simulated datasets ($N_d = 250$) and with larger data ($N_P = 100N$).

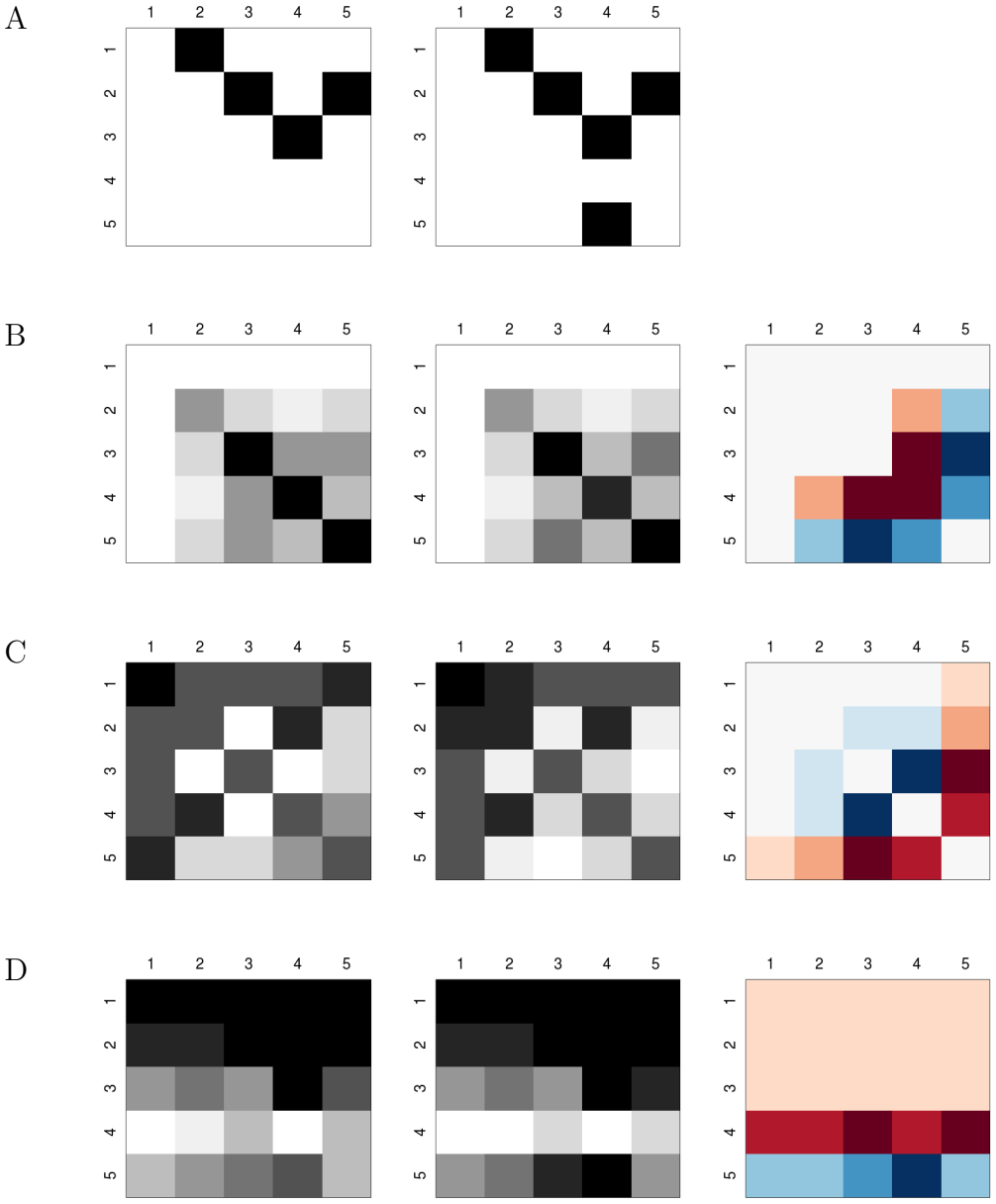

Figure A.1: Different summary statistics for graph pair 1 with $(N_d, N_P) = (250, 100N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{S}_1$ and $\bar{S}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: ΔS. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

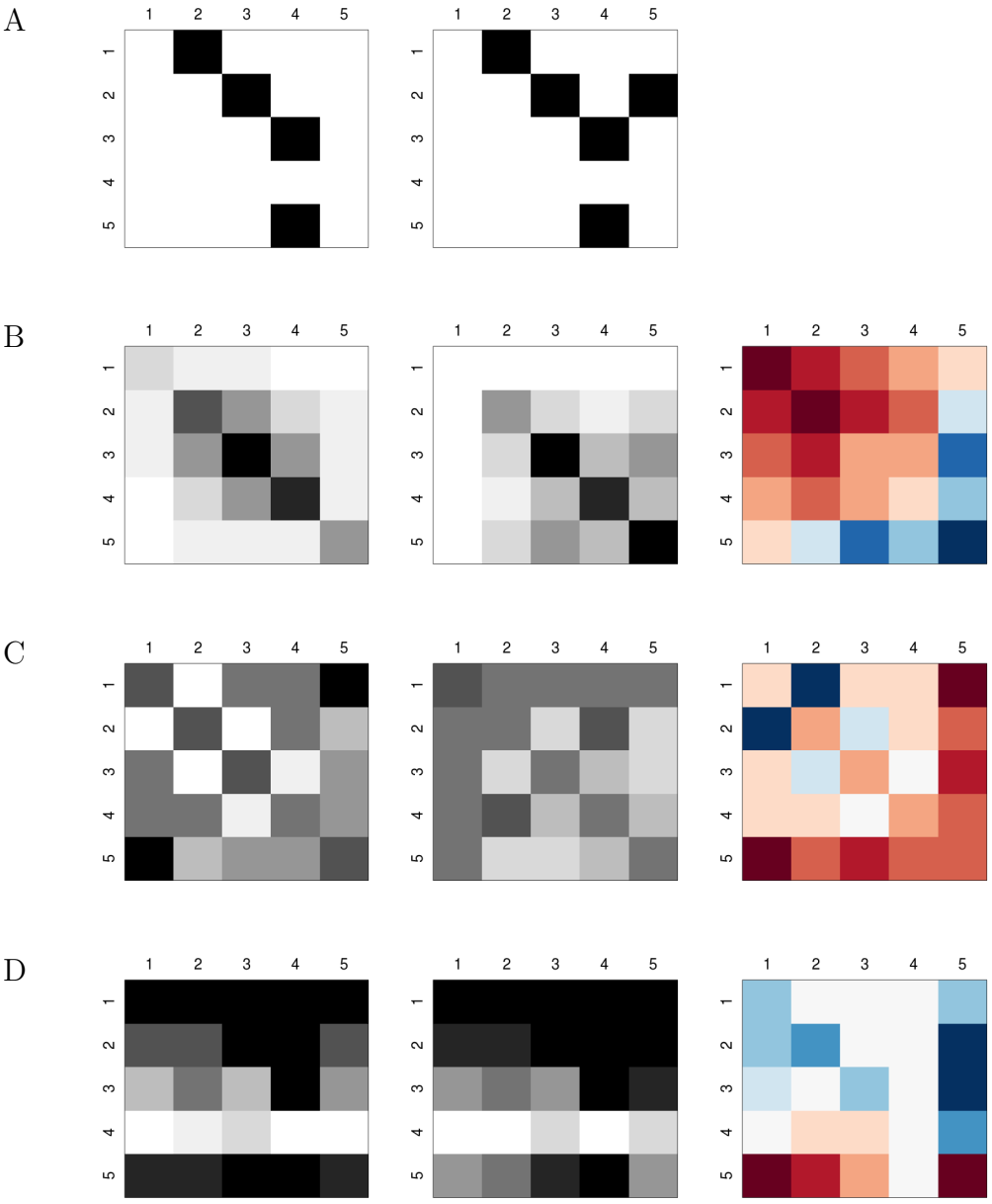

Figure A.2: Different summary statistics for graph pair 2 with $(N_d, N_P) = (250, 100N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{S}_1$ and $\bar{S}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: ΔS. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

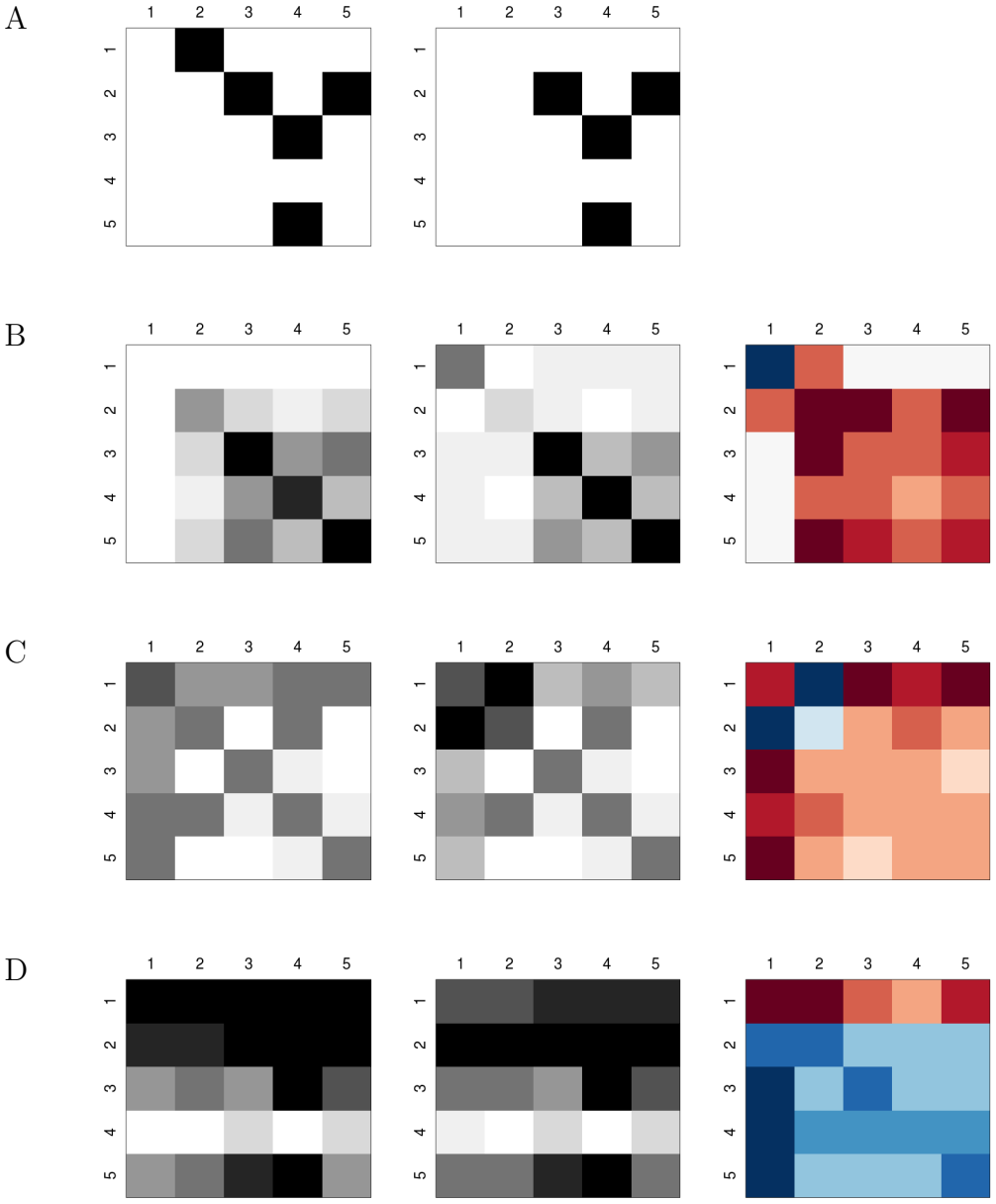

Figure A.3: Different summary statistics for graph pair 3 with $(N_d, N_P) = (250, 100N)$. (A) Adjacency matrices of the two graphs, left: graph 1, right: graph 2. Black fields correspond to a 1 in the matrix (an edge), white stands for a 0 (no edge). (B) Image plots of covariance matrices. Left and middle: $\bar{S}_1$ and $\bar{S}_2$. Images are scaled pairwise. Colors range from light grey (minimum value) to dark grey (maximum value). Right: ΔS. Colors range from dark blue (minimum value) to dark red (maximum value). Medium values are shown in white. (C) Image plots of precision matrices. (D) Image plots of subset structures.

A.4 Results for optimization in classification

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.356	0.367	0.365
Precision matrix	0.349	0.377	0.388
Subset structure	0.368	0.400	0.407

Table A.1: Average training errors for polynomial kernel of degree 2. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 100$ datasets were generated and the data was simulated with $N_P = 20$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.372	0.384	0.385
Precision matrix	0.369	0.429	0.453
Subset structure	0.368	0.401	0.412

Table A.2: Average training errors for polynomial kernel of degree 4. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 100$ datasets were generated and the data was simulated with $N_P = 20$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.101	0.143	0.148
Precision matrix	0.080	0.111	0.105
Subset structure	0.103	0.166	0.167

Table A.3: Average training errors for radial kernel. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 250$ datasets were generated and the data was simulated with $N_P = 100$ patients. Training was done with an SVM using a radial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.059	0.093	0.092
Precision matrix	0.089	0.106	0.097
Subset structure	0.081	0.108	0.094

Table A.4: Average training errors for linear kernel. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 250$ datasets were generated and the data was simulated with $N_P = 100$ patients. Training was done with an SVM using a linear kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.269	0.276	0.286
Precision matrix	0.263	0.267	0.269
Subset structure	0.306	0.332	0.335

Table A.5: Average training errors for polynomial kernel of degree 2. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 250$ datasets were generated and the data was simulated with $N_P = 100$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.117	0.164	0.169
Precision matrix	0.104	0.137	0.128
Subset structure	0.125	0.193	0.199

Table A.6: Average training errors for polynomial kernel of degree 3. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 250$ datasets were generated and the data was simulated with $N_P = 100$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

Summary statistics	nNodes 5	nNodes 8	nNodes 10
Covariance matrix	0.284	0.304	0.308
Precision matrix	0.288	0.317	0.336
Subset structure	0.306	0.344	0.345

Table A.7: Average training errors for polynomial kernel of degree 4. The table shows the average training errors for three different summary statistics and three different network sizes. From each graph $N_d = 250$ datasets were generated and the data was simulated with $N_P = 100$ patients. Training was done with an SVM using a polynomial kernel and the training error was determined via 10-fold cross validation. Values are averaged over 100 graph pairs.

A.5 Calculation times of the SVM based method and the MLE of the CBN model

Figures A.4 and A.5 show the calculation times of the SVM based method and the MLE of the CBN model for mean noise levels $\bar{\epsilon} \in \{0.05, 0.1\}$ used in data simulation.

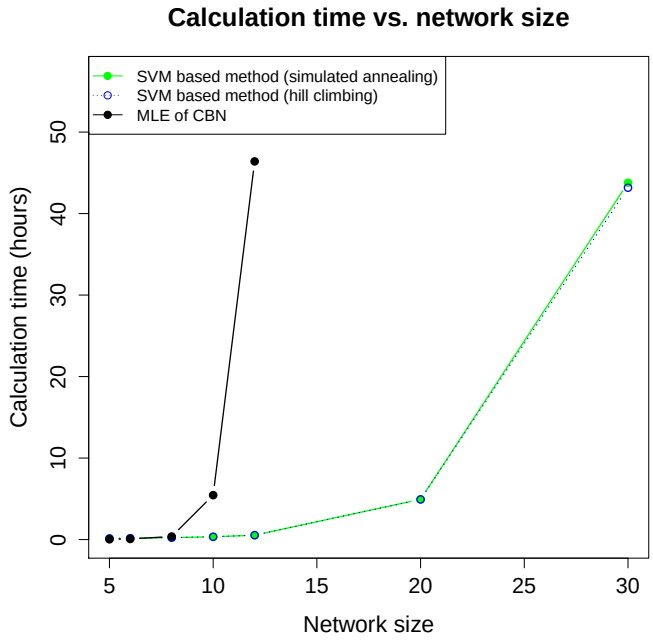

Figure A.4: Calculation times of the SVM based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.05$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model, green belongs to the SVM based method using simulated annealing and the calculation times of the machine learning based method using hill climbing are shown in blue.

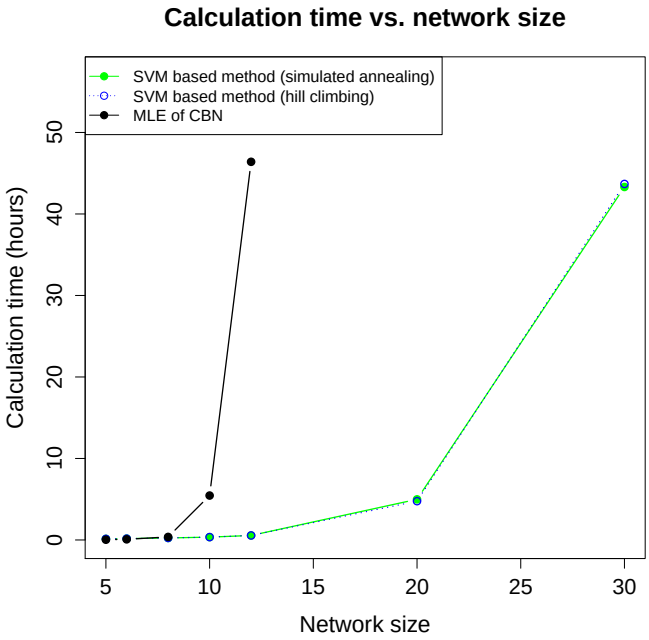

Figure A.5: Calculation times of the SVM based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.1$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model, green belongs to the SVM based method using simulated annealing and the calculation times of the machine learning based method using hill climbing are shown in blue.

A.6 Calculation times of the ABC based method and the MLE of the CBN model

Figures A.6 and A.7 show the calculation times of the ABC based method and the MLE of the CBN model for mean noise levels $\bar{\epsilon} \in \{0.05, 0.1\}$ used in data simulation.

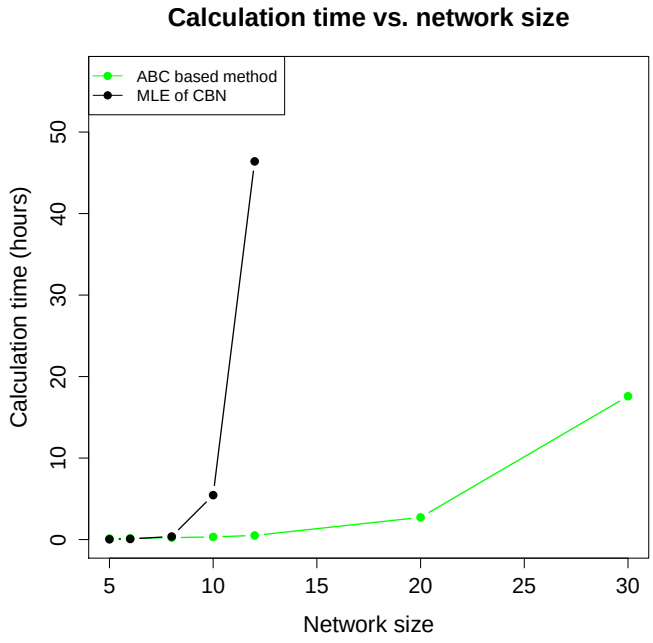

Figure A.6: Calculation times of the ABC based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.05$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model and the green line belongs to the ABC based method.

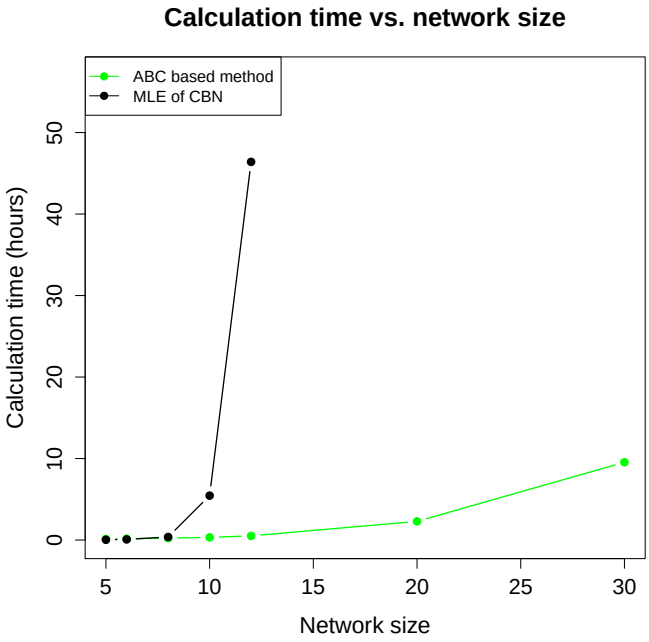

Figure A.7: Calculation times of the ABC based method and the MLE of the CBN model when using a mean noise level of $\bar{\epsilon} = 0.1$ in data simulation. The x-axis shows the network size, whereas on the y-axis the calculation times in hours are plotted. The dots represent average values over $N_r = 20$ runs. The black line corresponds to the MLE of the CBN model and the green line belongs to the ABC based method.

A.7 Hardware specifications

The simulations were performed on the high performance computing (HPC) cluster at the University of Regensburg. A detailed description can be found in [2]. Each node was equipped as follows:

CPU: 2 × AMD K10 Quad-Core, 2.2 GHz (Opteron 2354, Barcelona B3)
RAM: 16 GiB

Acronyms

APC adenomatous polyposis coli.

DCC deleted in colorectal cancer.

DPC4 deleted in pancreatic cancer locus 4.

pRb retinoblastoma protein.

ABC approximate Bayesian computation.

C-SVC C-Support Vector Classification.

CBN conjunctive Bayesian network.

CGH comparative genomic hybridization.

DAG directed acyclic graph.

DNA deoxyribonucleic acid.

EM expectation–maximization.

HPC high performance computing.

LOH loss of heterozygosity.

MCMC Markov chain Monte Carlo.

MLE maximum likelihood estimation.

SA simulated annealing.

SVM support vector machine.

TSG tumor suppressor gene.

Bibliography

[1] Surveillance, Epidemiology, and End Results (SEER) Program (www.seer.cancer.gov) SEER 13 Stat Database: Incidence - SEER 13 Regs Research Data, Nov 2011, National Cancer Institute, DCCPS, Surveillance Research Program, Surveillance Systems Branch, released April 2013, based on the November 2012 submission.

[2] Athene hpc cluster universität regensburg, 2014. URL `http://www.uni-regensburg.de/EDV/kurs_info/brf09510/hpc/hpc08.pdf`. [Online; accessed 27-March-2014].

[3] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[4] A. Aizerman, E. M. Braverman, and L. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[5] M. Alimandi, L.-M. Wang, D. Bottaro, C.-C. Lee, A. Kuo, M. Frankel, P. Fedi, C. Tang, M. Lippman, and J. H. Pierce. Epidermal growth factor and betacellulin mediate signal transduction through co-expressed erbb2 and erbb3 receptors. *The EMBO journal*, 16(18):5608–5617, 1997.

[6] P. Anand, A. B. Kunnumakara, C. Sundaram, K. B. Harikumar, S. T. Tharakan, O. S. Lai, B. Sung, and B. B. Aggarwal. Cancer is a preventable disease that requires major lifestyle changes. *Pharmaceutical research*, 25(9):2097–2116, 2008.

[7] M. Baudis. Genomic imbalances in 5918 malignant epithelial tumors: an explorative meta-analysis of chromosomal CGH data. *BMC cancer*, 7(1):226, 2007.

[8] M. Baudis and M. Cleary. Progenetix. net: an online repository for molecular cytogenetic aberration data. *Bioinformatics*, 17(12):1228, 2001. ISSN 1367-4803.

[9] M. A. Beaumont. Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41:379–406, 2010.

[10] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[11] N. Beerenwinkel and S. Sullivant. Markov models for accumulating mutations. *Biometrika*, 96(3):645–661, 2009.

[12] N. Beerenwinkel, N. Eriksson, and B. Sturmfels. Evolution on distributive lattices. *Journal of theoretical biology*, 242(2):409–420, 2006.

[13] N. Beerenwinkel, N. Eriksson, and B. Sturmfels. Conjunctive bayesian networks. *Bernoulli*, 13(4):893–909, 2007.

[14] G. Bergers, D. Hanahan, and L. Coussens. Angiogenesis and apoptosis are cellular parameters of neoplastic progression in transgenic mouse models of tumorigenesis. *The International journal of developmental biology*, 42(7):995, 1998.

[15] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[16] N. Bouck, V. Stellmach, and S. C. Hsu. How tumors become angiogenic. *Advances in cancer research*, 69:135–174, 1996.

[17] G. M. Brodeur, A. A. Tsiatis, D. L. Williams, F. W. Luthardt, and A. A. Green. Statistical analysis of cytogenetic abnormalities in human cancer cells. *Cancer genetics and cytogenetics*, 7(2):137–152, 1982.

[18] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[19] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[20] F. Colotta, P. Allavena, A. Sica, C. Garlanda, and A. Mantovani. Cancer-related inflammation, the seventh hallmark of cancer: links to genetic instability. *Carcinogenesis*, 30(7):1073–1081, 2009.

[21] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[22] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[23] D. G. DeNardo, P. Andreu, and L. M. Coussens. Interactions between lymphocytes and myeloid cells regulate pro-versus anti-tumor immunity. *Cancer and Metastasis Reviews*, 29(2):309–316, 2010.

[24] R. Desper, F. Jiang, O.-P. Kallioniemi, H. Moch, C. H. Papadimitriou, and A. A. Schäffer. Inferring tree models for oncogenesis from comparative genome hybridization data. *Journal of Computational Biology*, 6(1):37–51, 1999.

[25] R. Desper, F. Jiang, O. Kallioniemi, H. Moch, C. Papadimitriou, and A. Schäffer. Distance-based reconstruction of tree models for oncogenesis. *Journal of Computational Biology*, 7(6):789–803, 2000.

[26] R. Diestel. *Graph theory*, volume 4th electronic edition 2010, corrected reprint 2012. Springer Verlag, Heidelberg, 2010. URL `http://www.flooved.com/reader/3447`.

[27] S. du Manoir, M. R. Speicher, S. Joos, E. Schröck, S. Popp, H. Döhner, G. Kovacs, M. Robert-Nicoud, P. Lichter, and T. Cremer. Detection of complete and partial chromosome gains and losses by comparative genomic in situ hybridization. *Human genetics*, 90(6):590–610, 1993.

[28] H. F. Dvorak. Tumors: wounds that do not heal: similarities between tumor stroma generation and wound healing. *The New England journal of medicine*, 315(26):1650–1659, 1986.

[29] E. Fearon and B. Vogelstein. A genetic model for colorectal tumorigenesis. *Cell*, 61(5):759, 1990.

[30] E. R. Fearon, K. R. Cho, J. M. Nigro, S. E. Kern, J. W. Simons, J. M. Ruppert, A. Preisinger, G. Thomas, K. Kinzler, et al. Identification of a chromosome 18q gene that is altered in colorectal cancers. *Science*, 247(4938):49–56, 1990.

[31] J. Ferlay, I. Soerjomataram, M. Ervik, R. Dikshit, S. Eser, C. Mathers, M. Rebelo, D. Parkin, D. Forman, and F. Bray. GLOBOCAN 2012 v1.0, Cancer Incidence and Mortality Worldwide: IARC CancerBase No. 11. Lyon, France:

International Agency for Research on Cancer; 2013. URL `http://globocan.iarc.fr`. accessed on 17/01/2014.

[32] L. Foulds. The experimental study of tumor progression: a review. *Cancer research*, 14(5):327–339, 1954.

[33] Y.-X. Fu and W.-H. Li. Estimating the age of the common ancestor of a sample of DNA sequences. *Molecular biology and evolution*, 14(2):195–199, 1997.

[34] M. Gerstung, M. Baudis, H. Moch, and N. Beerenwinkel. Quantifying cancer progression with conjunctive Bayesian networks. *Bioinformatics*, 25(21):2809, 2009. ISSN 1367-4803.

[35] C. Godsil and G. Royle. *Algebraic graph theory*. Springer Verlag, Heidelberg, 2013.

[36] A. S. Goustin, E. B. Leof, G. D. Shipley, and H. L. Moses. Growth factors and cancer. *Cancer Research*, 46(3):1015–1029, 1986.

[37] S. I. Grivennikov, F. R. Greten, and M. Karin. Immunity, inflammation, and cancer. *Cell*, 140(6):883–899, 2010.

[38] D. Hanahan and J. Folkman. Patterns and emerging mechanisms of the angiogenic switch during tumorigenesis. *Cell*, 86(3):353 – 364, 1996.

[39] D. Hanahan and R. A. Weinberg. The hallmarks of cancer. *Cell*, 100(1):57–70, 2000.

[40] D. Hanahan and R. A. Weinberg. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–674, 2011.

[41] A. Hauser and P. Bühlmann. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13:2409–2464, 2012. URL `http://jmlr.org/papers/v13/hauser12a.html`.

[42] L. Hayflick. Mortality and immortality at the cellular level. A review. *Biochemistry-New York-English Translation of Biokhimiya*, 62(11):1180–1190, 1997.

[43] S. P. Jackson and J. Bartek. The DNA-damage response in human biology and disease. *Nature*, 461(7267):1071–1078, 2009.

[44] J. Jacob, M. Jentsch, D. Kostka, S. Bentink, and R. Spang. Detecting hierarchical structure in molecular characteristics of disease using transitive approximations of directed graphs. *Bioinformatics*, 24(7):995–1001, 2008.

[45] A. Jemal, F. Bray, M. M. Center, J. Ferlay, E. Ward, and D. Forman. Global cancer statistics. *CA: A cancer journal for clinicians*, 61(2):69–90, 2011.

[46] F. Jiang, R. Desper, C. Papadimitriou, A. Schäffer, O. Kallioniemi, J. Richter, P. Schraml, G. Sauter, M. Mihatsch, and H. Moch. Construction of evolutionary tree models for renal cell carcinoma from comparative genomic hybridization data. *Cancer research*, 60(22):6503, 2000. ISSN 0008-5472.

[47] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012. URL `http://www.jstatsoft.org/v47/i11/`.

[48] A. Kallioniemi, O. Kallioniemi, D. Sudar, D. Rutovitz, J. Gray, F. Waldman, and D. Pinkel. Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors. *Science*, 258(5083):818, 1992.

[49] M. B. Kastan. DNA Damage Responses: Mechanisms and Roles in Human Disease, 2007 GHA Clowes Memorial Award Lecture. *Molecular Cancer Research*, 6(4):517–524, 2008.

[50] R. Kim, M. Emi, and K. Tanabe. Cancer immunoediting from immune surveillance to immune escape. *Immunology*, 121(1):1–14, 2007.

[51] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[52] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26): 15324–15328, 2003.

[53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087, 1953.

[54] D. Meyer and T. U. Wien. Support vector machines. The interface to libsvm in package e1071. Online-documentation of the package e1071 for R, 2001.

[55] S. Negrini, V. G. Gorgoulis, and T. D. Halazonetis. Genomic instability—an evolving hallmark of cancer. *Nature Reviews Molecular Cell Biology*, 11(3):220–228, 2010.

[56] P. C. Nowell. Mechanisms of tumor progression. *Cancer research*, 46(5):2203–2207, 1986.

[57] V. Potter. The biochemical approach to the cancer problem. *Federation Proceedings*, 17:691–697, 1958.

[58] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, 1999.

[59] B.-Z. Qian and J. W. Pollard. Macrophage diversity enhances tumor progression and metastasis. *Cell*, 141(1):39–51, 2010.

[60] P. W. Rigby, M. Dieckmann, C. Rhodes, and P. Berg. Labeling deoxyribonucleic acid to high specific activity *in vitro* by nick translation with DNA polymerase I. *Journal of molecular biology*, 113(1):237–251, 1977.

[61] J. J. Salk, E. J. Fox, and L. A. Loeb. Mutational heterogeneity in human cancers: Origin and consequences. *Annual Review of Pathology: Mechanisms of Disease*, 5:51–75, 2010.

[62] D. Shibata, M. A. Reale, P. Lavin, M. Silverman, E. R. Fearon, G. Steele Jr, J. M. Jessup, M. Loda, and I. C. Summerhayes. The DCC protein and prognosis in colorectal cancer. *New England Journal of Medicine*, 335(23):1727–1732, 1996.

[63] M.-A. Shibata, I. G. Maroulakou, C. L. Jorcyk, L. G. Gold, J. M. Ward, and J. E. Green. p53-independent apoptosis during mammary tumor progression in C3 (1)/SV40 large T antigen transgenic mice: suppression of apoptosis during the transition from preneoplasia to carcinoma. *Cancer research*, 56(13):2998–3003, 1996.

[64] A. Sigal and V. Rotter. Oncogenic mutations of the p53 tumor suppressor: the demons of the guardian of the genome. *Cancer research*, 60(24):6788–6793, 2000.

[65] M. B. Sporn. The war on cancer: A review. *Annals of the New York Academy of Sciences*, 833(1):137–146, 1997.

[66] S. Stjernqvist, T. Rydén, M. Sköld, and J. Staaf. Continuous-index hidden Markov modelling of array CGH copy number data. *Bioinformatics*, 23(8):1006–1014, 2007.

[67] H. Symonds, L. Krall, L. Remington, M. Saenz-Robles, S. Lowe, T. Jacks, and T. Van Dyke. p53-dependent apoptosis suppresses tumor growth and progression in vivo. *Cell*, 78(4):703–711, 1994.

[68] S. Tavaré, D. J. Balding, R. Griffiths, and P. Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518, 1997.

[69] M. W. Teng, J. B. Swann, C. M. Koebel, R. D. Schreiber, and M. J. Smyth. Immune-mediated dormancy: an equilibrium with cancer. *Journal of leukocyte biology*, 84(4):988–993, 2008.

[70] S. Thiagalingam, C. Lengauer, F. S. Leach, M. Schutte, S. A. Hahn, J. Overhauser, J. K. Willson, S. Markowitz, S. R. Hamilton, S. E. Kern, et al. Evaluation of candidate tumour suppressor genes on chromosome 18 in colorectal cancers. *Nature genetics*, 13(3):343–346, 1996.

[71] M. G. Vander Heiden, L. C. Cantley, and C. B. Thompson. Understanding the warburg effect: the metabolic requirements of cell proliferation. *Science*, 324 (5930):1029–1033, 2009.

[72] V. Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[73] B. Vogelstein and K. W. Kinzler. Cancer genes and the pathways they control. *Nature medicine*, 10(8):789–799, 2004.

[74] B. Vogelstein, E. Fearon, S. Hamilton, S. Kern, A. Preisinger, M. Leppert, Y. Nakamura, R. White, A. Smits, and J. Bos. Genetic alterations during colorectal-tumor development. *The New England journal of medicine*, 319(9): 525–532, 1988.

[75] A. Von Heydebreck, B. Gunawan, and L. Füzesi. Maximum likelihood estimation of oncogenetic tree models. *Biostatistics*, 5(4):545–556, 2004.

[76] R. A. Weinberg. *The Biology of Cancer*. Garland Science, Taylor & Francis Group, 2007.

[77] M. Weiss, M. Hermsen, G. Meijer, N. Van Grieken, J. Baak, E. Kuipers, and P. Van Diest. Comparative genomic hybridisation. *Molecular pathology*, 52(5): 243–251, 1999.

[78] L. Wood, D. Parsons, S. Jones, J. Lin, T. Sjöblom, R. Leary, D. Shen, S. Boca, T. Barber, J. Ptak, et al. The genomic landscapes of human breast and colorectal cancers. *Science*, 318(5853):1108, 2007.

Acknowledgements

This work was carried out in the department of *Statistical Bioinformatics* of the Institute of Functional Genomics at the University of Regensburg. I thank all past and present colleagues for the good working atmosphere and fruitful scientific and non-scientific discussions.

I am very grateful to my supervisors Rainer Spang, Elmar Lang and Giusi Moffa for their constant advice. Especially I want to thank Rainer Spang for giving me the opportuny to carry out this thesis in his group. Special thanks also go to Giusi without whose advice this work wouldn't have been possible.

While working on this thesis I met many people who made contributions either direct or indirect. In particular I want to thank Katharina Meyer - my desk neighbor, Inka Appel, Julia Engelmann, Franziska Taruttis, Eva Beilschmidt, Christian Hundsrucker and Peter Butzhammer. I also want to thank my "climbing-group" for the many climbing sessions we had after work.

I am very grateful to my sister Liane, her boyfriend Ananda and to Regina Hampel who read drafts of this thesis and greatly improved it by their comments.

Especially I want to thank Matthias for his scientific advice and for supporting me throughout this thesis. I also thank my family for encouraging me throughout the time it took to finish the presented work.

Eidesstattliche Erklärung

1. Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet.
2. Weitere Personen waren an der inhaltlich-materiellen Herstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe eines Promotionsberaters oder anderer Personen in Anspruch genommen. Niemand hat von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.
3. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Regensburg, 10.06.2014 Daniela Herold