Institut für Informatik
Wissensverarbeitung und Informationssysteme

Institut de Recherche en Informatique
et Systèmes Aléatoires

Analyzing biological expression data based on decision tree induction

Dissertation
zur Erlangung des akademischen Grades
“Doctor rerum naturalium”
(Dr.rer.nat.)
in der Wissenschaftsdisziplin Bioinformatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
André Flöter

Potsdam, den 27.05.2005

Acknowledgements

Beside my compassionate fiancée Sophie I would like to thank the following people for their support,

- Torsten Schaub and Jacques Nicolas, as the tutors of this thesis, for their invaluable advice and vast help in coping with all kinds of technical, formal, and financial problems,
- Joachim Selbig for the financial aid and his kind assistance in writing publications,
- Joachim Kopka and Georg Leggewie for the biological data and their inexhaustible patience with my repetitive enquiries,
- Philippe Besnard for his advice and the revision of all French text I had to produce,
- Wolfgang Severin for his substantiated help whenever my JAVA interpreter was of a different opinion than myself,
- Matthias Möhlig for his cooperation and scientific enthusiasm which motivated me to the end of my thesis,
- and all the rest of the staff who has helped me during the last years.

Additionally, I would like to thank Peter-Uwe Zettiér, Matthias Scholz, Gert Zöller and Carsten Haustein for comforting me in times of fatigue and doubt.

Contents

Résumé

Les techniques modernes d'analyse biologique fournissent aux scientifiques diverses formes de données. Une catégorie parmi d'autres est appelée " données d'expression ". Ces données indiquent les quantités des composés biochimiques présents dans les tissus, échantillon par échantillon.

Depuis peu, les " données d'expression " peuvent être générées très rapidement. Ce qui aboutit à des masses de données qui ne sont plus analysables par les techniques statistiques classiques. Le " Systems biology " est un nouveau domaine dédié à la modélisation de ces informations.

Actuellement, il y a diverses approches qui sont appliquées à cet effet. L'une d'elles est l'apprentissage automatique. Les méthodes de ce type ont, jusque récemment, été surtout employées pour des tâches de classification et de prédiction, négligeant un avantage secondaire important : la capacité d'induire des modèles interprétables.

L'obtention de tels modèles est devenu un sujet crucial en " Systems biology ". De nombreuses approches ont été proposées et ont suscité d'intenses débats. Cette thèse s'attache à examiner ainsi qu'à exploiter une approche de base : les arbres de décisions.

La notion de comparaison d'ensembles d'arbres de décision est introduite afin de permettre l'identification de seuils pour certains attributs (à domaine continu ou discret). Déterminer des seuils significatifs constitue un moyen d'identifier des états pour des organismes vivants. Cette connaissance relative aux états fournit des indices extrêmement précieux pour la compréhension des processus dynamiques en jeu. Appliquée aux " metabolite concentration data ", la méthode proposée ici a permis d'identifier des états qui n'avaient pu être mis en évidence par les techniques conventionnelles d'extraction de seuils.

Une seconde approche exploite la structure des ensembles d'arbres de décision dans une perspective de découverte de dépendances combinatoires entre attributs. Les travaux antérieurs sur la question se sont limités soit à des méthodes coûteuses en calcul soit à l'interprétation d'arbres de décision simples – correspondant à une sous-exploitation drastique des données. Ceci a débouché sur des résultats incomplets voire instables. C'est pourquoi est introduite ici une nouvelle méthode qui a recours aux ensembles d'arbres de décision pour surmonter ces limitations.

Chacune des deux méthodes introduites a donné lieu à des logiciels, d'ores et déjà disponibles, qui peuvent être appliqués indépendamment ou l'un après l'autre. Le tout forme un package d'outils analytiques qui se présente comme un complément profitable aux méthodes existantes.

Par le biais de ces outils, ces nouvelles méthodes ont permis de confirmer certains points connus et surtout de suggérer de nouvelles relations très intéressantes entre " metabolites ".

Zusammenfassung

Neuere biologische Analysetechniken liefern Forschern verschiedenste Arten von Daten. Eine Art dieser Daten sind die sogenannten “Expressionsdaten”. Sie geben die Konzentrationen biochemischer Inhaltsstoffe in Gewebeproben an.

Neuerdings können Expressionsdaten sehr schnell erzeugt werden. Das führt wiederum zu so großen Datenmengen, dass sie nicht mehr mit klassischen statistischen Verfahren analysiert werden können. “System biology” ist eine neue Disziplin, die sich mit der Modellierung solcher Information befasst.

Zur Zeit werden dazu verschiedenste Methoden benutzt. Eine Superklasse dieser Methoden ist das maschinelle Lernen. Dieses wurde bis vor kurzem ausschließlich zum Klassifizieren und zum Vorhersagen genutzt. Dabei wurde eine wichtige zweite Eigenschaft vernachlässigt, nämlich die Möglichkeit zum Erlernen von interpretierbaren Modellen.

Die Erstellung solcher Modelle hat mittlerweile eine Schlüsselrolle in der “Systems biology” erlangt. Es sind bereits zahlreiche Methoden dazu vorgeschlagen und diskutiert worden. Die vorliegende Arbeit befasst sich mit der Untersuchung und Nutzung einer ganz grundlegenden Technik: den Entscheidungsbäumen.

Zunächst wird ein Konzept zum Vergleich von Baummengen entwickelt, welches das Erkennen bedeutsamer Schwellwerte in reellwertigen Daten anhand ihrer zugehörigen Entscheidungswälder ermöglicht. Das Erkennen solcher Schwellwerte dient dem Verständnis von dynamischen Abläufen in lebenden Organismen. Bei der Anwendung dieser Technik auf metabolische Konzentrationsdaten wurden bereits Zustände erkannt, die nicht mit herkömmlichen Techniken entdeckt werden konnten.

Ein zweiter Ansatz befasst sich mit der Auswertung der Struktur von Entscheidungswäldern zur Entdeckung von kombinatorischen Abhängigkeiten zwischen Attributen. Bisherige Arbeiten hierzu befassten sich vornehmlich mit rechenintensiven Verfahren oder mit einzelnen Entscheidungsbäumen, eine sehr eingeschränkte Ausbeutung der Daten. Das führte dann entweder zu unvollständigen oder instabilen Ergebnissen. Darum wird hier eine Methode entwickelt, die Mengen von Entscheidungsbäumen nutzt, um diese Beschränkungen zu überwinden.

Beide vorgestellten Verfahren gibt es als Werkzeuge für den Computer, die entweder hintereinander oder einzeln verwendet werden können. Auf diese Weise stellen sie eine sinnvolle Ergänzung zu vorhandenen Analyswerkzeugen dar.

Mit Hilfe der bereitgestellten Software war es möglich, bekanntes Wissen zu bestätigen und interessante neue Zusammenhänge im Stoffwechsel von Pflanzen aufzuzeigen.

Abstract

Modern biological analysis techniques supply scientists with various forms of data. One category of such data are the so called “expression data”. These data indicate the quantities of biochemical compounds present in tissue samples.

Recently, expression data can be generated at a high speed. This leads in turn to amounts of data no longer analysable by classical statistical techniques. Systems biology is the new field that focuses on the modelling of this information.

At present, various methods are used for this purpose. One superordinate class of these methods is machine learning. Methods of this kind had, until recently, predominantly been used for classification and prediction tasks. This neglected a powerful secondary benefit: the ability to induce interpretable models.

Obtaining such models from data has become a key issue within Systems biology. Numerous approaches have been proposed and intensively discussed. This thesis focuses on the examination and exploitation of one basic technique: decision trees.

The concept of comparing sets of decision trees is developed. This method offers the possibility of identifying significant thresholds in continuous or discrete valued attributes through their corresponding set of decision trees. Finding significant thresholds in attributes is a means of identifying states in living organisms. Knowing about states is an invaluable clue to the understanding of dynamic processes in organisms. Applied to metabolite concentration data, the proposed method was able to identify states which were not found with conventional techniques for threshold extraction.

A second approach exploits the structure of sets of decision trees for the discovery of combinatorial dependencies between attributes. Previous work on this issue has focused either on expensive computational methods or the interpretation of single decision trees - a very limited exploitation of the data. This has led to incomplete or unstable results. That is why a new method is developed that uses sets of decision trees to overcome these limitations.

Both the introduced methods are available as software tools. They can be applied consecutively or separately. That way they make up a package of analytical tools that usefully supplement existing methods.

By means of these tools, the newly introduced methods were able to confirm existing knowledge and to suggest interesting and new relationships between metabolites.

Introduction

Contributions

The thesis at hand is a contribution to the communities of Machine learning, Knowledge discovery and Systems biology. It presents the following novelties:

- The genuine new concept of establishing distance measures between decision forests:
  This can be regarded as a loose extension to the works of several authors concerned with finding and evaluating thresholds in continuous data (abstracted by [48] and [96]). The new concept has been presented and published in [60] and [61].

- The concept of using conditional mutual information for the detection of combinatorial dependencies in continuous biological data:
  This idea is an extension to the works of several authors who use partial correlation for the reconstruction of biological networks [101, 179, 142].

- The estimation of high conditional mutual information through decision forests and the reconstruction of dependency networks with it:
  This idea was inspired by the work of Breiman [24] and has also been picked up in a more limited way by Soinov [170]. The new approach has been presented and published in [59] and [62].

- The application of the introduced new techniques on metabolite concentration data and the derivation of new insight into metabolism:
  So far, very few authors have tackled with this problem. These results will partially be published in Leggewie et al. [99] and in an article resulting from a poster of Möhlig et al. [123].

Additionally, two documented and ready-to-use computer programs with a graphical user interface for applying the introduced techniques are supplied to the biological community.

Overview of the thesis

The thesis has been partitioned into four chapters:

1. Observing life at the molecular level: Systems biology

In Chapter 1, an introduction is given to the background of the data used in Chapter 4. Systems biology is introduced as the discipline which is concerned with the construction of models for molecular biological systems. The major kinds of data used in this discipline are described.

Further, some problems and inconveniences occurring with the realisation of the physiological experiments are outlined. In the second half of this chapter, conventional methods for the processing and analysis of the generated data are outlined. Also, a brief introduction to graphs, a prime choice for visualising and reasoning on network models, is given.

2. A tool for the identification of structure in data: Decision trees

In Chapter 2, the basics of decision tree learning are introduced. These are the underlying techniques for all of the newly introduced methods of this thesis. A large number of issues concerning the construction of trees is addressed and it is indicated if and how these could affect the methods introduced in Chapter 3.

3. From raw data to biological networks: a contribution to the analysis of dependencies among sparse and noisy continuous data

In Chapter 3, the new and original methods of this thesis are presented. An algorithm is given for the growing of heterogeneous decision forests. This algorithm is then integrated in an approach to compare forests for the evaluation of discretisation thresholds. The same algorithm is further used for the estimation of high conditional mutual information, a concept whose usage is motivated in another section of this chapter.

4. An experiment in the analysis of metabolite concentration data for potatoes

In Chapter 4, the previously introduced techniques are tested on metabolite concentration data. In a first step, the implementations of the new techniques are described. Then, the actual usage of the supplied tools is introduced in a user scenario. Subsequently, the tools are applied to the data in a meaningful order. Finally, the results on the real data are presented and discussed from a biochemical perspective.

Preliminaries

Most of the notation used in this thesis is introduced on demand. The following conventions are valid for all the text unless explicitly invalidated for a specific section.

- Structural units (chapters, sections, figures, tables etc.) are written with an upper case if they are given with a number and thereby refer to a specific unit. They are written with a lower case in all other cases. Example: All subsequent chapters are based on Chapter 1.
- Double quotes are used for words that do not have a clear definition in the given context. These words are to be understood in a more intuitive manner. Usually, the meaning of those words should become understandable from the related context. Example: Many suggestions have been made for the determination of which test is “best” for the problem.
- Single quotes are used for newly introduced terms that are not explicitly introduced in a formal definition. The particular meaning should become clear after the first occurrence of the term. Example: The test with the highest information gain is the ’best test’.
- Italics are used for emphasising statements that either pose a contrast to a previously made statement or indicate an important conclusion not to be overlooked. Example: The test is invalid on attribute A. However, it *is* valid on all other attributes.

Chapter 1

Observing life at the molecular level: Systems biology

Research in molecular biology has undergone several major changes in the last decade. A trigger for this development was the ability to produce large amounts of molecular biological data with new, so called high throughput techniques[1]. This led in turn to the need for computational assistance in the analysis of the data. The field of research addressing this subject is called 'Bioinformatics' or 'Computational Biology'.

Recently, Anglo-Saxon literature tends to distinguish between the two terms in the way that

- Bioinformatics is predominantly addressing data management, e.g. the development, allocation, and maintenance of data bases holding molecular biological data, and

- Computational Biology is addressing the development and application of elaborate algorithms for the analysis of molecular biological data.

Such a distinction is not always clear which sometimes leads to a synonymous or inconsistent usage of the terms.

Computational Biology, as in the stricter Anglo-Saxon definition, has mainly focused on the analysis of data produced without considering potential analysis methods [158]. However, most methods work only effectively if the input data meets specific requirements. Furthermore, the retrieval of complex structures from data (e.g. networks) strongly relies on customised experiments. That is why recently, the requirements of computational analyses have been considered in the setup of new physiological experiments (illustrated in Figure 1.1). This new way of designing experiments according to algorithmic needs and thereby facilitating the retrieval of complex structures has been included in a newly labelled field of research called 'Systems biology'. Systems biology is the integration of multiple data sources and systematic use of computational aid in order to be able to predict, control and design living systems [5].

The thesis at hand is a contribution to the computational part of the field of Systems biology. This chapter will give an overview of the molecular biological data sources used and some of the classical analytical methods referred to in the subsequent chapters. Most techniques will be introduced in a simplified manner in order to allow for a quicker and more intuitive understanding of the needed basics.

[1] Experimental techniques that can produce at a high rate comprehensive measurements from biological samples.

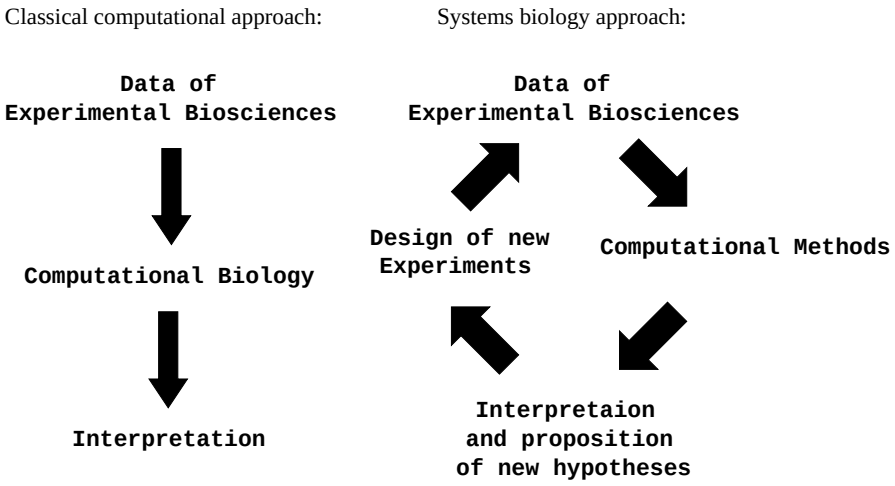

Figure 1.1: The step from classical computational biology to Systems biology.

1.1 Available data sources of Systems biology

A major aspect of Systems biology is the integration of multiple data sources and their exploitation through computational methods. In this section, two sources of molecular biological data will be introduced in more detail. New computational methods to analyse these data will be presented in the subsequent chapters.

In addition, other forms of data sources available in Systems biology will be mentioned in order to give a more comprehensive overview.

1.1.1 Metabolomics

Metabolites are the small molecules of a living system. Metabolism is the chemistry taking place in any living organism. This includes, in particular, the turning of an organism's food into more organism. The chemical steps of a metabolite being transfered into another metabolite including all involved intermediate products, catalysts and kinetics is called a 'metabolic pathway' [73].

The Metabolome is the totality of all metabolites and all active metabolism in a cell at a given point in time. It is a pattern of molecules and metabolic pathways that reflects the cell's status [8].

The Metabolome gives a direct picture of the cells activity in its environment. It presents a powerful portrait, reflecting health, disease, ageing and the effects of drugs and the environment. Metabolomics[2] is the field of research that deals with analysing, modelling and predicting the metabolome.

In this subsection, an introduction is given on the key technology that is used in Metabolomics and that will be used later in this thesis.

Metabolic Profiling

A metabolic profile is the entirety of all metabolite concentrations in (parts of) a living organism at a given point in time. It is thereby not the metabolome because it does not include direct information on active metabolism. Strictly, it is usually not even *part* of the metabolome because, for technical reasons, most profiles are taken from a mixture of tissue and not from a single cell.

[2]Metabolomics focuses on the analysis of plants. There is another term "Metabonomics" that refers to the same analytical processes but for data of animals and humans.

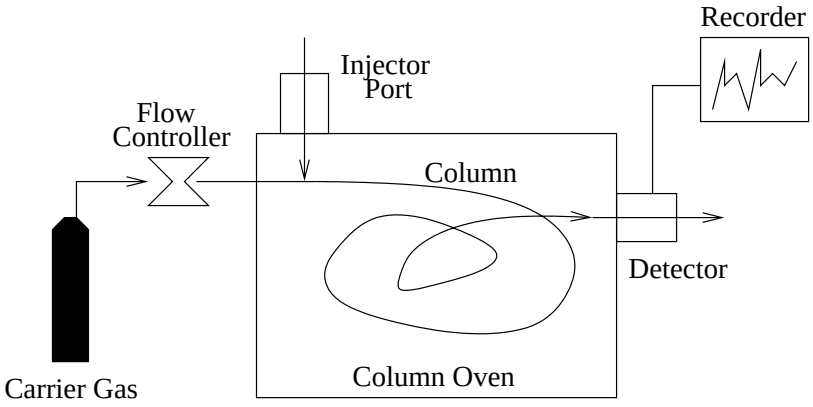

Figure 1.2: A schematic diagram of a gas chromatograph.

But the tissue used for establishing a profile can be confined to a set of very similar cells (e.g. certain leaf cells). That has been done in the experiments used in this thesis, and that is why metabolic profiles will subsequently be regarded as part of the metabolome.

Metabolic profiles are thus a valuable source of information for the understanding of an organism's current activity or status, a property that will be exploited in Section 3.1. Such profiles can be gained from a procedure called Gas Chromatography and Mass Spectrometry (GC/MS).

Gas chromatography

Chromatography is a process used to separate chemical compounds based on their differing adsorption to a fixed matrix. It is particularly useful for the separation of proteins, amino acids and nucleotides, hence of metabolites [25]. There are several different procedures of chromatography [8], one of which is gas chromatography.

Gas chromatography -specifically gas-liquid chromatography- involves a sample being vapourised and injected onto the head of the chromatographic column [88]. The sample is transported through the column by the flow of inert, gaseous mobile phase (see diagram in Figure 1.2). The column itself contains a liquid stationary phase which is absorbed onto the surface of an inert solid. The individual compounds of the sample move at different speeds through the column. At the end of the column, a detector will measure the discharge of the individual compounds and record the speed they have needed to traverse the column. This record is called a chromatogram. It records the amount of each chemical compound present in the sample. To a certain extent, it also allows for the identification of the compounds according to their traversal time.

However, there are numerous parameters that can influence the quality of the chromatogram (e.g. oven temperature, properties of the column, size of the sample etc.). Thus, the interpretation of the chromatograms is not entirely reliable. At best, some of the chemical compounds of the sample can be identified at the end of this process. Yet, the compounds are now well separated and can be analysed in the subsequently explained process.

Mass Spectrometry

For a reliable identification of the detected compounds, a second process called Mass Spectrometry is needed [88, 149]. It uses a mass spectrometer to measure the exact molecular mass of a molecule. This is done by tracking its flight path through a set of magnetic and electric fields.

To be more precise, the mass/charge ratio is measured. However, the operating chamber of the

spectrometer contains a vacuum. In vacuum, nearly all biological molecules have no charge, but the method relies of them being charged to make them susceptible to magnetic and electric fields. Thus, mass spectrometers use a method to bombard the target molecules with radiation to charge them up.

The bombardment can in turn fragment the molecules. This is a benefit because then you can measure the mass/charge ratio of bits of the molecules as well as the whole thing in one measurement [8]. Piecing together these data allows for the reconstruction of the original molecule.

GC/MS

GC/MS is the abbreviation for the combination of gas chromatography and mass spectrometry. With modern machines, this combination facilitates a fast way of measuring and identifying metabolites. Because of this, it is labelled as a high throughput technique.

The output of the GC/MS procedure are values (real numbers) indicating the quantities of the identified metabolites in the sample. Usually, the values are given as relative changes of the metabolite's concentrations as compared to those of a reference sample [149]. To some degree, this normalisation ensures comparability between several experiments. The final output is generally delivered as a data vector containing the values of concentrations (see Section 4.2.1 for more details).

It has to be mentioned that GC/MS encounters some technical limitations (for more details see [49]). Most of them can be suppressed to a certain extent by a meticulous use of the equipment.Yet, the possibility of erroneously generated pieces of data always remains [19, 93]. One such possibility lies in the insecure attribution of metabolites to peaks of the chromatogram [154]. Typically, this leads to a few unidentified fragments. This effect can be seen later in this thesis. But for all of the difficulties, GC/MS is presently one of the best analyses to gain information on the metabolome.

1.1.2 Transcriptomics

Genes are the factors that control heredity. They are pieces of information that determine properties of living organisms. The entirety of all genes of an organism is called the organism's genome.

Genes are coded onto chromosomes. A chromosome is composed of proteins and desoxyribonucleic acids (DNA). The DNA can be regarded as a code that uses an alphabet of four symbols: G (guanine), C (cytosine), T (thymine), and A (adenine). These acids are the basic components of all DNA.

In any organism, genes are coded statically[3] into the four acids G, C, T, A. However, similar to a computer program, this static information still allows for dynamic reactions to environmental or internal stimuli onto the organism. That is, particular genes will only cause an effect if a specific stimulus exists. This effect is a process leading to the production of ribonucleic acid (RNA) and subsequently, in many cases, of a protein. Simplified, proteins are the one key for almost all dynamical processes in a living organism [25]. Thus, knowledge about the types and amount of produced proteins is very valuable information on the understanding of an organism's dynamics.

Transcription is the first step in the production of a protein [73]. It refers to the production of RNA[4] from DNA. The thereby produced RNA is called RNA-transcript because its composition is established according to the transcribed code of a DNA (see Figure 1.3). In a second step, the code of the RNA is utilised for the production of a specific protein. In life sciences, this process

[3] For abstraction, I disregard spontaneous mutations and single nucleotide polymorphisms here. In fact, they usually have little effect on a fully-grown organism [25].

[4] In literature, this RNA is more specifically called messenger-RNA (mRNA). For simplicity and because this distinction is not important for this thesis, those acids will just be called RNA here.

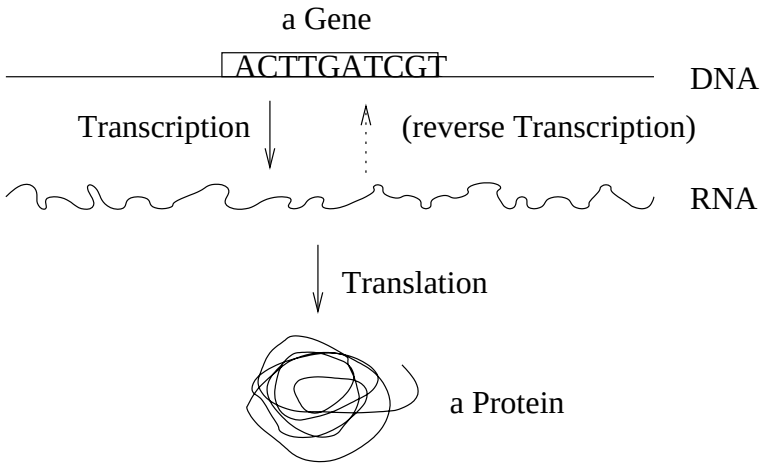

Figure 1.3: The two main steps of gene expression, referred to as the “Central Dogma”, are called “Transcription” and “Translation”. The additional step of “reverse transcription” is indicated by the dotted line.

is defined as ’translation’ [141]. Hence, the amount and type of transcript present in a cell gives information[5] on the proteins produced in it.

The totality of all RNA-transcript in a cell at a given point in time is called the cell’s transcriptome. The field of research dealing with analysing, modelling and interpreting the transcriptome is called Transcriptomics.

In this subsection, the key technology for gaining data in Transcriptomics is introduced. In later chapters, new techniques will be introduced which also allow for an analysis of the data used in Transcriptomics.

Analytically exploitable properties of DNA and RNA

DNA and RNA both carry the code of genes. This code is a sequence of four acids. In DNA, those acids are guanine, cytosine, thymine, and adenine. In RNA, thymine is replaced by uracile, but the code remains equivalent.

This sequence of acids is pieced together as a strand. In DNA, such a strand has the shape of a helix. DNA is generally made up of two strands in the shape of a double helix. That is, each acid on one strand has a neighbour on the second strand.

In stable double helices, neighbours are always[6] ’complementary’. That means, one strand of the double helix determines the second strand by the following rules:

- Adenine must have cytosine as neighbour, and
- guanine must have thymine (uracile in RNA respectively) as neighbour.

These rules derive from numerous physio-chemical constraints (for more details see [25]).

The two strands of DNA can be separated by applying high temperature. That process is called melting. After cooling down, single strands can[7] again recombine with a complementary

[5] Protein synthesis is a very complex process. The sole knowledge about the amount of present RNA-transcript is just a rough indicator for the amount of proteins produced. Nonetheless, it is a key factor in the process and gives valuable indications for protein production.

[6] Most stable double helices contain small regions of non-complementary neighbours. For simplicity, this is ignored here.

[7] under specific environmental conditions

strand to a double helix. This process is called reassociation or hybridisation. Note that the two complementary strands need not be the same as they have been before the melting, they only have to be complementary. This last property is essential for the success of the technique introduced in the following subsubsection.

Obtaining information on the transcriptome: Microarray technology

Microarray technology is a means to determine the activity of genes (usually referred to as gene expression) in a target sample. It is based upon the ability of DNA to hybridise.

The basic principle of the technology can be described in the following simplified manner: A set of known single strand DNA is attached to specific spots onto a support medium (e.g. glass). Then, radioactively[8] marked DNA is produced[9] from the RNA of the sample. This DNA is given onto the support medium. The medium is heated and all DNA melts. After this, the medium is cooled down and the single DNA strands hybridise. At this point, some of the radioactive marked DNA strands will hybridise with complementary DNA attached to the support medium. Subsequently, the non-attached DNA is washed from the medium. Now, it is possible to measure the amount of radioactivity at each spot of the medium. The radioactivity indicates the amount of specific RNA that has been in the original sample.

This process allows for the quantification and identification of RNA in a sample. Because the produced amount of each RNA indicates the activity of a particular gene it is thereby possible to roughly quantify a degree of expression of that gene. Due to the many spots that can be located on a medium (up to 1 million) it is possible to measure the activity of complete genomes.

Using microarray technology includes numerous technical parameters and difficulties which will not be discussed in this thesis (for more details see [73]). The above description is very brief and serves only for getting a feel for the complex way of generating the data. At the end of the process, a vector is obtained that holds real values indicating the degree of expression of each particular gene in a sample. At present, these data are still quite error-prone due to the many process-related difficulties in producing it.

1.1.3 Other types of data

There are other kinds of data used in the field of Systems biology. For the matter of a more complete overview, two important areas of research are mentioned in this subsection. Genomics is a superordinate concept of the previously introduced data sources. Proteomics is another important part of Systems biology, but its data is not used for the approach of this thesis.

Genomics

Genomics is a generic term for all studies involving the genome of living organisms [78]. Typically, it addresses the branch of genetics that deals with identifying all DNA sequences of an organism, also referred to as sequencing. Thus, the classical data produced in genomics is four-lettered code (see also Subsection 1.1.2).

Knowledge about complete DNA sequences is a prerequisite for the understanding and mapping of genes to proteins. Transcriptomics, Metabolomics, and Proteomics make use of this information. That is why they are also referred to as techniques of the *post-genomic* era.

Today, genomics is often subdivided into functional genomics and structural genomics [21]. Functional genomics focuses on the determination of the biological functions of the genes and

[8] or fluorescently or colourescently

[9] in a process called reverse transcription

	leaves	height	fruit	colour
plant 1	6	2.2	3	g
plant 2	4	5.8	8	b
plant 3	7	0.7	25	g
...	⋮	⋮	⋮	⋮

Figure 1.4: A data matrix containing plants as samples with the attributes: leaves, height, fruit, and colour.

their products. By definition, Transcriptomics, Metabolomics, and Proteomics overlap with it [83]. Structural genomics deals with the determination of three-dimensional structures of proteins. This field is not discussed in this thesis.

Proteomics

Proteomics is the study of the full set of proteins encoded by a genome [106]. It deals with gaining knowledge on protein biochemistry using the same philosophy of high throughput analysis that has been applied in Transcriptomics and Metabolomics [25].

The proteome is the protein complement to a given genome (see also Subsection 1.1.2). However, it is much more complex than the corresponding genome. In humans, for example, about half a million proteins are generated from some 25.000 genes. Moreover, many proteins are modified after their synthesis, and their expression levels are differentially regulated in space and time or in healthy and diseased states.

Proteomics seeks to identify and characterise the many functional dependencies existing between proteins and their necessary requisites.

1.2 Cleaning biological data: data preprocessing

A major problem in the evaluation of physiological experiments in the domain of Systems biology is the technical heterogeneity of the data. Many different experimental setups deliver numerous types of data. Apart from the plain problem of handling proprietary file formats there are also contentual problems. Two major ones of them are the comparison of data from several and/or different experiments and the trimming of data in order to be able to apply a specific method of analysis to it. In this section, a few of the standard methods for coping with these problems are introduced. These methods are partially used in the experiment presented in Chapter 4.

For simplicity (and for the rest of this thesis[10]), it is assumed that the concerning data is available in form of a data matrix that always contains attributes[11] in its columns and experiments (samples respectively) in its rows (see Figure 1.4). In statistical literature, this is usually referred to as the 'spreadsheet data representation' [189]. It is also the representation commonly used for expression data in the biological literature.

[10] A formal introduction will be given in Chapter 2

[11] A formal definition of the term attribute is given in Definition 3 in Chapter 2. So far, it can be regarded as just a variable.

1.2.1 Describing empirical data: Statistical standard measures

For an objective description and characterisation of data sets, statistics has produced numerous measures and methods [53, 6] that are also used in Systems biology [58, 149, 158]. This thesis does not focus on the classical statistical analysis of data. The focus is on the adaptation, enhancement and application of decision tree techniques to Systems biology data (see Chapters 2, 3, and 4). Nonetheless, a few statistical measures and terms are needed in the subsequent sections and chapters.
The following standard terms are used for a given data vector $\vec{a} = (a_1, a_2, ..., a_n)$:

$$\text{the 'mean' is } \bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

$$\text{the 'standard deviation'is } \sigma_a = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2}$$

The square of the standard deviation is called the 'variance'.
Given a second vector $\vec{b} = (b_1, b_2, ..., b_n)$,

$$\text{the 'covariance' is } \tilde{\sigma}_{ab} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}).$$

Given a data set with m attributes $X_1 ... X_m$, the 'covariance matrix' is an $m \times m$ matrix. The elements of that matrix are given as $\tilde{\sigma}_{X_i X_j}$, where i denotes the ith element in the jth row.

1.2.2 Normalisation

Normalisation refers to the process of scaling the attributes of a data matrix into the same range. Typically, this characteristic is required to suppress the predominance of attributes with disproportionately large scales (see Subsection 1.3.2 for more detail). Often, the target range of normalisations is limited and small, like -1 to 1, or 0 to 1 [70]. There are several commonly used methods to achieve this property:

Min-max normalisation

Min-max normalisation [70] performs a linear transformation on the original data. Given that A_{min} and A_{max} are the minimum and the maximum value of an attribute A and x is a value of A,

$$minmax(x) = \frac{x - A_{min}}{A_{max} - A_{min}}$$

. This will map any value into the range 0 to 1.

This is the basic normalisation available in most statistic toolboxes (e.g. S-Plus [38]). It produces problems if the data contains outliers because that would vigorously shrink the significant share of the range. For that reason, the following more elaborate normalisation techniques have been developed.

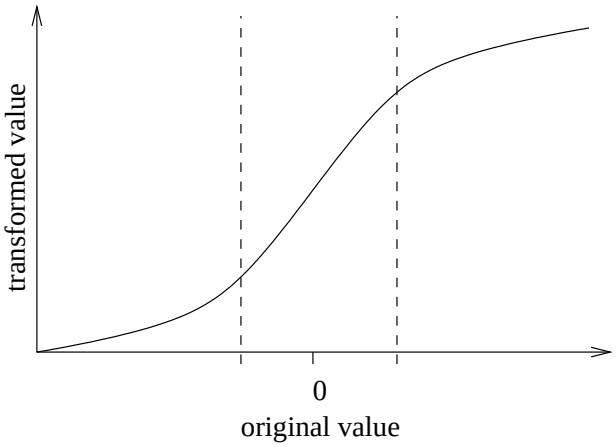

Figure 1.5: Softmax scaling maps values of an attribute into a nearly linear part of the range in the middle and two 'softened' parts at the edges.

Z-score normalisation

In z-score normalisation [70] (also known as zero-mean normalisation or z-transformation), the values for an attribute A are normalised based on the mean $\bar{A}$ and standard deviation σ_A of A. Given a value x of A,

$$zscore(x) = \frac{x - \bar{A}}{\sigma_A}$$

This normalisation is useful when the range of attribute A is unknown and/or outliers are expected. However, it does not preserve the metric properties of the data.

Softmax scaling

Another way to compensate for outliers is softmax scaling [131]. The name refers to the characteristic of this normalisation to 'soften' the effect of outliers, values close to the minimum or maximum of the range. There are two versions of softmax scaling, a continuous and a non-continuous one. The non-continuous one divides the range into three intervals, each of them using a different mapping function. Here, only the more common continuous version will be explained.

Given a parameter r and the standard deviation σ_A and mean $\bar{A}$ of an attribute A and a value x of A,

$$softmax(x) = \frac{1}{1 + e^{-y}} \quad \text{, where} \quad y = \frac{x - \bar{A}}{r \cdot \sigma_A}$$

The parameter r controls the portion of A's range that will be mapped nearly linearly (in Figure 1.5 approximately between the dashed lines).

Softmax scaling is useful when the values of an attribute are to retain most of their linear behaviour but outliers are expected. To a certain extent, it preserves the original metric of the data.

Rank Ordering

Rank ordering is a form of normalisation that maps values from a metric scale into an ordinal non-metric scale [6]. Non-metric means that the real distances between values have no significance; only the order of the values is important.

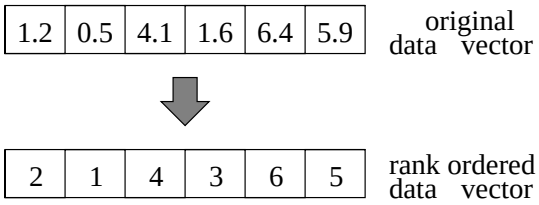

Figure 1.6: Rank ordering: The numbers of a data vector are mapped into their ranks.

		Attributes						
		g1	g2	g3	...	g1352	g1353	g1354
Experiments	sample 1	5.7	2.8	5.9	...	0.8	4.3	4.3
	sample 2	7.7	7.2	8.4	...	6.8	6.5	8.0
	sample 3	0.7	3.1	3.2	...	1.0	5.9	1.2

Figure 1.7: The property of biological data matrices containing a lot more attributes than samples is referred to as the 'curse of dimensionality'.

Rank ordering is achieved by sorting [89] all values of an attribute and replacing the values of the original vector by their position in the sorted domain (this position is often referred to as the 'rank'). See Figure 1.6 for an example of rank-ordering a numerical vector.

This normalisation introduces a strong bias to the data. It is usually only applied if the significance of the original scale is unclear and/or the data contains a lot of noise.

1.2.3 Dimension reduction

Modern high throughput techniques can measure many attributes of a given biological tissue sample. However, in Transcriptomics and in Metabolomics the application of such an analysis is still rather costly. That is why most laboratories can only afford to perform a limited set of high throughput analyses on their samples. Thus, the obtained data matrices are typically very asymmetric, containing a lot more attributes than samples. This characteristic is sometimes referred to as the "curse of dimensionality" (see Figure 1.7).

Furthermore, values of some of these attributes depend on values of other attributes. That is, the values of one attribute allow for the prediction of the values of another attribute. In Metabolite concentration data, for instance, this is the case when two metabolites are the product of only the same metabolic pathways, or, in transcript data, when two genes are coregulated.

When searching for complex dependencies between attributes, such dependant attributes do not carry any additional information but they sometimes hamper with analytical algorithms. Thus, it would be helpful to remove such redundant attributes. Moreover, many statistical analysis techniques (e.g. cluster analysis [172]) generally exhibit problems when dealing with more attributes than samples in a data matrix. For these reasons, methods are applied for reducing the number of attributes.

The most commonly used technique for doing this with Systems biology data [194] and an improvement thereof are introduced in this subsection. Another technique, dimension reduction through decision trees, is mentioned in Section 2.2.

Principal Component Analysis

The concept of detecting dependant attributes and merging them into new and preferably independent attributes is known as feature extraction [146]. A widely available method to perform this is Principal Component Analysis (PCA) [81]. There exist several ways to calculate principal components. The most commonly used method is roughly outlined in the following paragraph. For brevity, the calculation of Eigenvectors and Eigenvalues is not explained here but taken from *"Linear Algebra"* by Klaus Jänich [82]. For a more detailed introduction on PCA as a whole, see Smith [169] or Burges [146]. The following steps are necessary:

1. From all attribute values of the given data set, the mean of the respective attribute is subtracted.

2. The covariance matrix is built on the new attribute values.

3. The Eigenvectors of the covariance matrix are calculated and normed to length 1.

4. The original attribute values are expressed through linear combinations of the Eigenvectors.

Up to this point, no information has been lost. The data has only been expressed in an orthogonal basis. The directions of the Eigenvectors are called the components. They serve as attributes in the new space. Now, the Eigenvalues of the Eigenvectors can be calculated. The Eigenvalues indicate the contribution of the component toward the explanation of variance in the data. The components with the highest associated Eigenvalues are called the principal components.

Intuitively, PCA generates a new data matrix with uncorrelated new attributes containing the same information as the old data. These attributes can then be ranked according to their significance in explaining the variance of the original data. It is now possible to drop the least significant components and thereby decrease the dimension of the feature space. This way, it is hoped to obtain a smaller new data matrix retaining most of the information of the old matrix while deleting the noise in the data.

Independent Component Analysis

Another method for dimension reduction becoming more popular in Systems biology is Independent Component Analysis (ICA) [37]. ICA tries to generate new and, as far as possible, statistically independent attributes. Statistical independence of attributes means that the attributes do not carry any Mutual information (see Subsection 1.3.2 and Subsection 3.2.2 for details on Mutual information). Note that statistical independence is a stronger criterion than the non-correlatedness of PCA.

Another key characteristic of ICA as compared to PCA is that the attributes generated by ICA do not have to be orthogonal. This gives ICA a more flexible manner of generating the new attributes.

ICA has several particularities requiring some expertise in its application. It is therefore not possible to simply replace PCA with ICA. For more details on ICA and its application in Systems biology see Scholz et al. [156]. ICA has not been applied comprehensively to the data of this thesis.

1.2.4 Discretisation

The data of Metabolomics and Transcriptomics is usually given with a precision of several decimal digits. This lets the numbers appear quite precise. However, biological experiments mostly

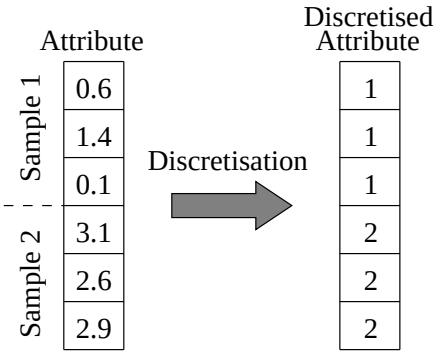

Figure 1.8: An original data set and a discretized version of it.

contain a large amount of noise. That is why recently, some scientists argue that the continuous representation is often misleading. They say, a discrete representation can display the same information in a clearer manner [64, 65, 102].

For instance, in Figure 1.8 we have measured three times an attribute of two samples. Theoretically, they should exhibit the same value for all three measurements of the same sample. But the values contain a noticeable variance. This may mislead the experimenter to believe that there is a biological process to be observed within the measurements of one sample. In reality, the variance is a mere artifact of the delicate processes within the measuring device. If the data is discretized, as in the righthand part of Figure 1.8, the variance is filtered and the data displays its veritable information.

Finding such a 'revealing' discretisation is not a trivial task. Therefore, some methods of discretisation fit for specific needs have been proposed in the past. An older but comprehensive synopsis of existing discretisation techniques has been given by Dougherty et al. [48]. To my knowledge, they were the first to introduce a systematic categorisation of techniques. The three proposed axes were *global* vs. *local*, *supervised* vs. *unsupervised*, and *static* vs. *dynamic*. The choice of names for the latter axis seems a little ambiguous and inapt as those terms are usually used in a different manner. That is why Kwedlo & Kretowski [96] introduced it again by the name of: *univariate* vs. *multivariate*. A recent overview of discretisation techniques with the goal of constructing better Bayes classifiers can be found in Yang et al. [192]. Strengths and weaknesses of new techniques belonging to particular categories have been discussed for many discretisation problems [74, 66, 188, 96, 13].

Supervised techniques make use of a class label attributed to every sample in the data set. Generally, supervised methods are said to deliver more useful results than unsupervised techniques [48]. However, they strictly require the presence of such a preclassified variable, which is usually not given with metabolite concentration data.

Global discretisation performs the discretisation of all continuous values in one step, while local discretisation processes only subsets of the data at a time. Ho and Scott [74] argue about advantages and disadvantages of *global* vs. *local* discretisation. They state that local discretisation can lead to more accurate results at the cost of higher computation time. But they also note that local discretisation might deliver ambiguous results which are harder to interpret.

Discretisation is often considered just as a data preprocessing aimed at eliminating noise. In practice, only the most basic discretisation methods are applied, if any. The above mentioned categorisation of discretisation techniques is a mere theoretical problem. For most categories, no implemented technique is generally available. However, as illustrated in Figure 1.8 and argued by some scientists, feasible discretisation can be regarded as a valuable stand-alone analysis [96].

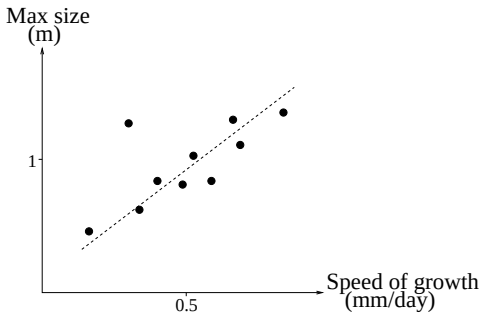

Figure 1.9: A scatter plot displaying two attributes with a linear correlation.

That is why it makes sense to actually develop techniques with certain properties. In Section 3.1, I will introduce a new discretisation technique to analyse biological expression data. It tries to keep the advantages of supervised discretisation in an unsupervised context by conducting an exhaustive search through possible class labellings.

1.3 Finding interdependencies between attributes: correlation analysis

Correlation is a concept quantifying the interrelation of metric attributes [53]. If, for instance, the values of one attribute tend to rise whenever the values of another attribute rise these attributes are noted to correlate. An example for two correlating attributes can be seen in Figure 1.9. In molecular biology, correlation is used to find elements that have a similar biological function [52, 149]. There are several ways to detect and/or quantify the correlation between attributes.

1.3.1 Pearson's correlation coefficients

A similarity measure to identify vectors (objects) whose scalars exhibit a correlated progression of values is Pearson's correlation coefficient [155]. The strength of correlation between two vectors can be calculated in the following way:

Definition 1 (Pearson's Correlation Coefficient) *Given two vectors $\vec{a}$ and $\vec{b}$ (both with n scalars) with their means $\bar{a}$ and $\bar{b}$, the Pearson's correlation coefficient is given as*

$$r = \frac{\sum_{i=1}^{n}(a_i - \bar{a}) \cdot (b_i - \bar{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2 \cdot \sum_{i=1}^{n}(b_i - \bar{b})^2}}$$

where a_i (b_i respectively) denotes the value of the i-th scalar of $\vec{a}$ ($\vec{b}$ respectively).

In the denominator, the coefficient uses the standard deviations of $\vec{a}$ and $\vec{b}$ to scale its value. Thereby, the coefficient is always between -1 and 1. A value of the coefficient close to 0 means that there is little or no linear correlation between the two vectors. A value close to 1 indicates a positive correlation and a value close to -1 indicates a negative correlation between the vectors.

In Systems biology, the coefficient is often used in cluster analysis to group attributes that exhibit a similar behaviour over the sample data [149, 179]. Certainly, the coefficient can also be applied without the use of cluster algorithms. In this thesis, however, it is only used as a distance measure in cluster analysis.

Spearman's correlation coefficient

There are other correlation coefficients available. One of them is Spearman's correlation coefficient. It is calculated by rank ordering the data vectors before applying Pearson's correlation coefficient. It does not measure the linear correlation between the vectors but their *monotonic* correlation. Theoretically, this means that it can also detect certain non-linear correlations. In practise, it is more often used for simply suppressing the effects of noise in the data.

Mutual information

A distance measure having recently become more popular in Systems biology is the Mutual information (MI) [166]. It is an entropy measure taken from information theory [40] (for more details on entropy see Subsection 2.2.4). Here, it can be regarded as an extension to Pearson's correlation coefficient. Unlike the coefficients, MI can generally also detect non-linear correlation. This is the reason why some scientists favour it over Pearson's correlation coefficient [41].

Given two random variables A and B that can take values $\{a_1..a_{M_A}\}$ and $\{b_1..b_{M_B}\}$, the mutual information is

$$MI(A,B) = \sum_{i=1}^{M_A}\sum_{j=1}^{M_B} p(a_i,b_j) log \frac{p(a_i,b_j)}{p(a_i)p(b_j)}$$

where p denotes the probability (joint or marginal, respectively) of the occurence of certain values. A detailed description of how Mutual information is calculated is given in Subsection 3.2.2.

1.3.2 Cluster analysis

Correlation measures are commonly used in a framework called 'cluster analysis'. It refers to numerous techniques aimed at automatically detecting similar objects within a given set [68]. In Systems biology, this analysis is often used as a first step for disclosing structure in new data [52, 58, 149].

In statistics, cluster analysis is also known as 'numerical taxonomy', 'automatic classification' or 'typology'. In machine learning, the techniques belong to the category of unsupervised learning techniques. That is, they can be applied without any further knowledge about a given and complete data set; they try to deliver results fully automatically.

These techniques are not focus of this thesis. For more information see [114].

1.4 Network induction

The goal of all methods used in Systems biology is the modelling of interactions between molecular biological elements. Modelled interactions between several components constitutes a network [11]. One such network commonly known to biologists is displayed in the Böhringer-Mannheim Chart of Metabolic Pathways (see Figure 1.10) [109]. It tries to integrate knowledge about all known metabolic pathways.

Systems biology seeks to integrate even more data. Thus, the resulting network is potentially very complex. In the thesis at hand, I will only introduce techniques for the reproduction of a very limited and abstract part of that network. In this section, an introduction is given on general issues concerning network composition as referred to later in this thesis.

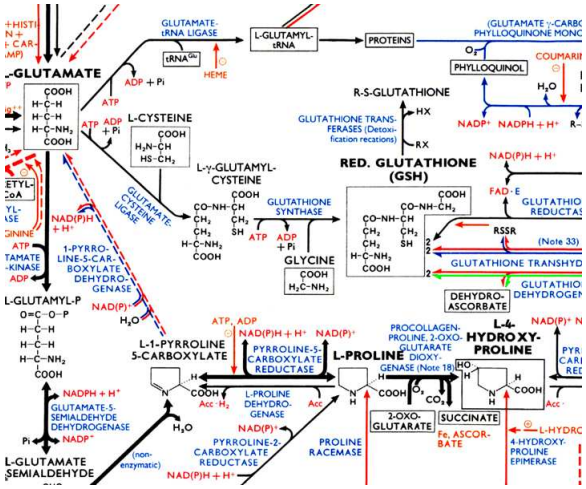

Figure 1.10: A small section of the Böhringer-Mannheim Chart of Metabolic Pathways.

1.4.1 Graphs

Graphs are a basic concept of discrete mathematics [16]. A 'vertex' is the basic element of a graph. It is often denoted as an identifier (e.g. A or B). An 'edge' is a pair of vertices (e.g. AB). A 'directed edge' is a pair of vertices with a given direction (e.g. $\vec{AB}$, the notation is that of a vector).

Definition 2 (Graph and Directed Graph) *A Graph $\mathcal{G}$ is a set of vertices and a set of edges between vertices (e.g. $\mathcal{G}$=$\{A,B\}$,$\{AB\}$). A directed graph is a graph with directed edges (e.g. $\mathcal{G}$=$\{A,B\}$,$\{\vec{AB}\}$).*

Figure 1.11 illustrates a simple directed graph made up of the set of vertices $\{A, B, C\}$ and the set of directed edges $\{\vec{AB}, \vec{AC}, \vec{CB}\}$. The direction of edges is indicated by arrows.

1.4.2 Putting information in Graphical models

Graphical modelling is a form of multivariate analysis that uses graphs to represent models [50].

Directed graphs are suited for the depiction of networks in general [50] and microbiological networks in particular [187]. The Böhringer-Mannheim Chart, as illustrated in Figure 1.10, is such a graph. Typically, the vertices depict elements (in the Chart: metabolites) and the directed edges indicate an influence of one element on the regulation or production of another element. In addition, the edges can hold extra information (coded, for example, as colourings [103], width of the arrows [as in Figure 1.10], or as textual annotations).

Graphs have several favourable properties for analyses [50]. One of them is their fitness for computational processing [162]. Yet, there are numerous well-evolved algorithms for working with graphs (e.g. learning [27], finding paths [162], determining properties [50] etc.). Another favourable property of graphs is their interpretability by the human eye. In subsequent chapters, I will make use of both of these properties for the analysis of Systems biology data.

1.4.3 Causal networks

Causal networks are directed graphs with additional information and additional semantics to it [167]. The additional semantics are:

- Each vertex represents a quantifiable physical entity.

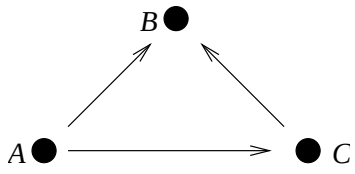

Figure 1.11: A graph consisting of 3 vertices and 3 directed edges.

- Each edge represents a direct causal relationship between two entities. That is, the quantity of the entity represented by the starting point influences the quantity of the entity represented by the end point.

The additional information states how the quantities of the entities of an edge influence each other. This information can be given in several ways (e.g. joint probability distributions or other functions).

Causal networks model the true causal relationships between its components (vertices) comprehensively. However, it is virtually not possible to derive true causal relationships from present Systems biology data [95, 174]. To distinguish between causal models and models that can only be a rough approximation of real world processes (due to incomplete or erroneous data), another term 'observational model' is used.

In the subsequent subsection and in Chapter 3, techniques for inducing observational models are introduced.

1.4.4 Bayesean approach

A far-reaching approach for obtaining observational models from Systems biology data has been founded in the work of Pe'er et al. [65, 129]. This work has later been continued and expanded by Ong et al. [128].

The approach deals with inferring Bayesean networks from transcription data. Simplified, a Bayesean network is a graphical model that provides probability distributions for each impact (directed edge) that is modelled in it. Through this characteristic, the negative effects of "hard" choices are softened. That is in the learning process, a hypothesis based only on weak evidence (few samples) cannot mask out an almost similarly likely hypothesis. Instead, all possible hypotheses remain in the Bayesean model with assigned probabilities of their correctness. With this additional information, Bayesean models can carry more information than simple graphical models (e.g. decision trees), but they are also much more complex and thereby harder to learn and interpret.

Pe'er et al. seek to learn a simple Bayesean net from perturbed transcription data. Numerous samples with well-defined genetic perturbations are necessary for this approach because only that way the effects of distinct genes can be traced. This principle is also discussed in another context and in more detail by Steuer et al. [174].

Pe'er further introduces the constituents "mediator", "activator" and "inhibitor" in order to be able to model causal relationships. These constituents are determined through the evaluation of a statistical significance measure. By this, the learned Bayesean nets can be interpreted as small causal networks.

Chu et al. [36] and others [95, 45, 79, 193] argue that with present Systems biology data there are not enough objects to support statistical significance of the models. Thus, it would not be possible to obtain realistic causal structures. Based on a similar argumentation, Ong et al. introduce

the concept of utilising known components[12] which are only *fitted* into a larger model. That way, Peer's statistical approach is a mere helping factor for placing the known structural components. Small-sized causal structure is then only inferred statistically at the transitions between operons, if any. This reduces the likelihood of wrongly inferred causal relations that would be derived from the poor statistics.

Despite a fundamental discussion about the validity of inferred Bayesean (or other) models in Systems biology, this approach is to date the most noted regarding automatic construction of biological (sub-)networks[13]. In a continuing work of the group of Friedman et al. [164], the approach adopts some of the criticism and is further narrowed down to only identifying conditions for regulatory dependencies. In the thesis at hand, a simpler approach is developed that also yields conditional dependencies but in a less complex representation. This makes it less susceptible to noisy data and thereby more robust. Additionally, it remains simpler to interpret and easier to calculate.

1.5 Public databases related to Systems biology

Apart from single handedly performing experiments there is yet another way to gather data for specific questions: public data bases. Due to the high costs of physiological experiments many research institutions have decided to pool their data in publicly accessible data bases. Two of the most used data bases for the investigation of metabolic networks are introduced in this subsection. Additionally, one of the best access points for further information on Systems biology data is described.

1.5.1 KEGG

The 'Kyoto Encyclopedia of Genes and Genomes' (KEGG) is an initiative of the Kanehisa Laboratory of the Kyoto University Bioinformatics Center [84]. It is aimed at providing a complete computer representation of the cell and the organism, which will enable computational prediction of higher-level complexity of cellular processes and organism behaviours from genomic information. KEGG is a suite of databases and associated software, integrating the current knowledge on molecular interaction networks in biological processes (PATHWAY database), the information about the universe of genes and proteins (GENES/SSDB/KO databases), and the information about the universe of chemical compounds and reactions (COMPOUND/GLYCAN/REACTION databases). In scientific practise, KEGG is the prime source for obtaining existing information on biochemical processes in the cell and the related genetic backgrounds.

1.5.2 BRENDA

BRENDA is the main collection of enzyme functional data available to the scientific community [157]. It is a data base maintained by the Institute of Biochemistry at the University of Cologne. Its focus is on the providing of access to functional data for gene products; those are the proteins and in particular enzymes. This data collection is being developed into a metabolic network information system with links to Enzyme expression and regulation information. An additional objective of this initiative is the identification of synonymous notations in an attempt to unify the

[12] Ong uses operons as known components (see [128] for further details.

[13] Other approaches for inferring biological networks are published (e.g. [3, 77, 180, 187]) but implementations are not publicly available.

nomenclature. Inconsistent nomenclature is one of the major problems in the evaluation of available experimental data. For the time being, BRENDA provides an extensive thesaurus of existing terms for this purpose.

1.5.3 ExPASy

The 'Expert Protein Analysis System Proteomics Server' (ExPASy) is dedicated to the analysis of protein sequences and structures as well as 2-D PAGE [9]. It comprises several different data bases (e.g. Swiss-Prot, PROSITE, ENZYME) and many analytical software tools for the identification of proteins, the analysis of their sequence and the prediction of their tertiary structure. It also offers many documents relevant to these fields of research. Links to most relevant sources of information across the Web are indicated. Through this vast cross-linking, ExPASy features one of the best access points for analytical purposes in the field of Systems biology.

1.6 Summary and conclusions

In this chapter, the most common sources of Systems biology data have been introduced. The sources referred to later in this thesis were described in more detail. Further, standard statistical techniques for analysing these data were presented. For the majority of publications in the domain of molecular biology, these techniques have been sufficient for an effective data analysis.

The last two subsections have dealt with more elaborate analysis techniques. These techniques help for the construction of networks that display causes and effects in the interaction of molecular biological components. Note that the construction of a comprehensive network displaying *all* causes and effects in molecular biology is the long-term goal of Systems biology.

It has been pointed out that this long-term goal is more of a vision at the moment. This is mostly due to the current ways of generating Systems biology data. The fundamental problem is that, even when disregarding noise and combining all available data types, this data does not at all constitute a comprehensive snapshot of the underlying system. Thus, a model derived from this data has to be incomplete.

On account of this, available techniques focus on the construction of small sub-networks, modelling only a very delimited part of the microbiological universe. After all, some interrelations can be derived also from incomplete data. But even for this purpose, the available analysis methods encounter technical problems because statistics are bad or the computational demand is too high. That is why it is feasible to develop new techniques that focus on small problems and circumvent a few of the existing difficulties. In the following chapters, novel techniques will be introduced that cope with existing difficulties and still deliver valuable knowledge about the biological system.

Chapter 2

A tool for the identification of structure in data: Decision trees

An important field of research for the processing and analysis of given data is called 'machine learning' [115, 10, 90, 110]. It addresses the question of how to construct computer programs that can learn from data and thereby improve their effectiveness. The topic is usually divided into the categories 'supervised' and 'usupervised' learning. The latter category is largely referring to cluster analysis[172], techniques which are not used in this thesis. Supervised learning is called supervised because an expert has first to evaluate a set of training data before the learning algorithms can start their learning process.

Machine learning techniques are used in a wide range of applications where ordinary programs fail to work effectively. We find such systems, for instance, for the recognition of handwritten zip codes in postal relay stations [145] or for the classification of customers in banks [115]. But a key application area of machine learning remains the discovery of structure in data sets. According to Wrobel and others[190, 55, 70], this is sometimes also referred to as 'Data Mining'[1].

Machine learning encompasses various techniques and approaches. 'Classification' is one superordinate category of such techniques. It deals with the automatic assignment of class labels to data objects. That is, given an object with a set of known attribute values, a classifier assigns one of several previously defined classes to that object.

An intuitive real-world example of a classification system can be found on chicken farms: In Figure 2.1, an egg is assigned to a quality class by a classification machine. The classification machine uses the egg's attributes for its decision. In this case, the attributes are weight, colour and size of the egg. Based on the attribute values of the sample egg, it obtains the label 'quality class I' here. There are many other practical examples for classification [24]. For a more comprehensive introduction into machine learning and more examples see [115].

[1]Note that some authors refer to the complete process of Knowledge Discovery as Data Mining[124]. This is not the definition used in this thesis.

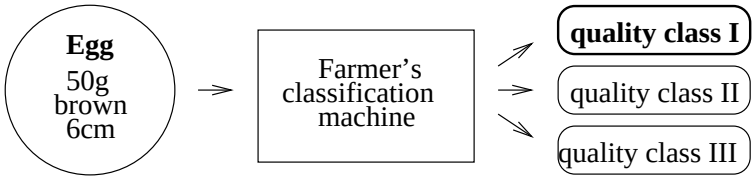

Figure 2.1: An egg with certain attributes is classified as belonging to quality class I.

	$\vec{a}$	$\vec{b}$	$\vec{c}$
obs1	4.6	6.4	yes
obs2	3.1	7.3	no
obs3	2.8	8.2	yes

Figure 2.2: A training matrix composed of three column vectors $\vec{a}, \vec{b}$ and $\vec{c}$ where $\vec{a}$ and $\vec{b}$ contain values for attributes a and b and $\vec{c}$ contains classifications labels "yes" or "no" for each of the three data objects "obs 1", "obs 2", and "obs 3".

In many[2] machine learning techniques, the learned systems are given as a formal description fixed before the first predictive task is performed. Systems of this kind are therefore referred to as 'predictive models'. Decision trees are one form of a predictive model. They can be "learned" automatically from given data [71]. Furthermore, they are simple to understand and can be processed effectively on computers. So, the trees can be used for both describing and predicting data[3].

In the subsequent chapters, decision tree learning is used and adapted for the analysis of molecular biological data. In this chapter, a motivation is given for the choice of the decision tree techniques as well as an introduction to the related basic algorithms, their characteristics and problems.

2.1 Machine learning on attributes

Most machine learning techniques follow the same rough principle:

- A set of data objects with known attribute values and target values, also referred to as 'training data', is used as input for a learning algorithm.
- The algorithm then delivers a function that can be used to predict target values for new objects with unknown target values.

To give a formal definition of a machine learning algorithm and a classification function, the following objects are needed:

Definition 3 (Attribute) *An attribute is a variable whose value describes a characteristic.*

Definition 4 (Data Object) *A data object is a vector $\vec{x}$ with scalars holding attribute values.*

Definition 5 (Training Matrix) *A training matrix is a data matrix D composed of data objects as rows plus an additional column vector $\vec{c}$ holding values, the classification labels, for each data object.*

Training data, as used in this thesis, is always given as a training matrix. In Figure 2.2, there is a simple example of a training matrix.
A machine learning algorithm is then defined as a function:

Definition 6 (Machine Learning Algorithm) *Let $\mathcal{C}$ be the set of all possible predictive functions and D^* be the set of all possible training data matrices. Then, a machine learning algorithm is a function*

$$\Lambda : D^* \rightarrow \mathcal{C}$$

[2] Case-based learning is an exception to this.

[3] Two well-founded theoretical justifications for the use of decision trees are given by Fiat *et al.* [57] and Karakos *et al.* [85].

The output of such an algorithm is a classifier:

Definition 7 (Classifier) *Let $\mathbb{X}$ be the set of all possible data objects and $\mathbb{S}$ be the set of all possible values of the target attribute. Then, any classifier $\Gamma \in \mathcal{C}$ is a function*

$$\Gamma : \mathbb{X} \rightarrow \mathbb{S}$$

The basic differences of machine learning techniques lie within the nature of the demanded training data and the learning principle and representation of the classifiers. Not all classifiers can handle all types of input data. Hence, the formal differences between machine learning algorithms are the possible types of the elements of D^* and $\mathbb{S}$ (and thus the functionality of Λ and Γ), the algorithm to actually calculate Λ, and the representation of Γ.

2.1.1 Data types

As for the type of training data, two major categories are distinguished: discrete (nominal) and continuous (real) data [67].

Discrete data contains only variables of discrete domains. A discrete domain spans over a finite or infinite set of nominal values (which are sometimes also referred to as symbols) [53, 17]. If the set is finite this data is more specifically called qualitative or categorical. Discrete domains can have at maximum an ordinal order. In particular, this means that no metrical distances can be determined between the objects.

Continuous data contains variables of real valued domains. Real valued domains are characterised by either an interval or an indefinite range. Variables of that domain can take any value within that range. Continuous domains always have a quantitative order.

Note that discrete variables can be mapped into continuous domains. However, in doing so, one artificially introduces a quantitative order for the previously discrete variable, and this order might not have existed in the original domain. This modification is nonetheless applied to data by many researchers in the biological domain. As indicated later, it does not necessarily invalidate a thereby obtained result.

2.1.2 Types of predictive models

Three main categories of predictive models are distinguished.

Classifiers are representations of discrete-valued functions. They map a (discrete or continuous) input vector into a discrete value of a finite set of possible values. Or in terms of machine learning: they assign a class label to any observed data object. Rule learning, decision trees and inductive logic programming are probably the best known among these techniques [135, 117, 136, 71].

Regression models are representations of real-valued functions. They map input vectors into a real value. Neural networks and support vector machines are actually the most commonly used techniques for regression [34, 115, 145, 1].

Probabilistic models are representations of probabilistic functions. They map input vectors into a vector of probabilities assigning one probability value to each of the predefined target classes. These models are sometimes regarded as an extension to classifiers. Bayesean Nets are the most utilised technique of this kind [69].

Various methods exist to learn and represent models of these categories. They have been tested and applied in numerous ways [87, 115]. To improve classification quality, it is also possible to combine techniques of different categories into hybrid methods in order to exploit advantages of both [67]. This approach will be briefly discussed in Subsection 2.3.9.

2.1.3 Graphical and rule-based representations of classifiers

The focus of this thesis is not on optimising classification accuracy. It is on the utilisation of the representations of classifiers and their interpretability in a molecular biological domain. For this purpose, simple and small representations must be favoured over complex and sized ones [12].

Graphs are a rich, flexible and easy to understand way of representing classifiers [50, 18]. To my knowledge, some form of graphical representation exists for all types of classifiers [27, 20, 145, 115, 1]. Some of those representations are neither easy to interpret nor simple to process for algorithms [26]. For obtaining biological hypothesis, however, it is essential to generate interpretable output [176]. Furthermore, for developing subsequent algorithms, it is beneficial to work on simple classifier structures. Classifiers which can be mapped into rule-based representations tend to be both, easy to interpret and simple enough for an easy processing by algorithms [17].

Decision trees can be learned efficiently with established methods (see Subsection 2.2.3). They have a simple graphical representation and can easily be mapped into rulesets [115] (see Subsection 2.2.2). If necessary, they can also be pruned into even less complex representations by numerous approved pruning strategies [136] (see Subsection 2.3.2). Thus, decision trees are a tradeoff between interpretability, efficiency and flexibility. No other type of classifier offers this mix of favourable features. Those are the best regarding the goal of interpreting *and* utilising the classifiers in extended algorithms.

2.2 Basics of Decision Trees

The primary purpose of decision trees is to provide a means for classifying data objects into discrete target classes [115]. The straightforward illustration of dependencies between attributes and the possibility of interpreting them is an additional feature of the trees [24]. Furthermore, decision trees can also be used for filtering attributes in view of dimension reduction [189].

Classification through decision trees is based upon a set of selected attributes. Each node in a tree represents a test on the value of an attribute, an edge corresponds to a possible value of the attribute, and a leaf specifies a possible target class. Finally, a decision tree represents a hierarchically organised set of tests which allows for classifying new observations. A simple tree can be seen in Figure 2.3. It is explained in more detail in the subsequent subsection. The following definitions[4] will introduce binary[5] decision trees formally as classification functions with a set of constraints regarding the way they can compute their value.

Definition 8 (Ordained scalar) *Let $\vec{\mathbb{V}}$ be a set of vectors, and let each vector $\vec{x} \in \vec{\mathbb{V}}$ have n scalars. Further, let $i \in [1, 2, ..., n]$, then the ordained scalar $\vec{x}^i$ is the i-th position of scalars in $\vec{\mathbb{V}}$.*

Note that a data matrix is a set of vectors. An ordained scalar can thereby be regarded as an *attribute* of a data matrix. It does not refer to the value of a specific scalar in a vector but to all values of scalars at specific positions in a set of vectors.

Definition 9 (Domain of an ordained scalar) *Given an ordained scalar $\vec{x}^i$, the domain $\langle \vec{x}^i \rangle$ of that scalar is the set of values $\vec{x}^i$ can take.*

[4] For the more precise referencing needed later in this thesis, more notions will be defined than in other introductions to decision trees.

[5] Since the biological questions addressed in this thesis are accomodated to a binary nature binary trees are the model of choice.

Definition 10 (Decision test) *Let n be the number of scalars of any vector $\vec{x}$ of a given data matrix, let $\vec{X}^* = \{\vec{x}^i, ..., \vec{x}^n\}$ be the set of ordained scalars of $\vec{x}$, and for any $i \in \{1, ..., n\}$ let a_i be an element of $\langle \vec{x}^i \rangle$, then a decision test is a boolean expression either of the form*

$$\vec{x}^i > a_i \text{ or } \vec{x}^i \geq a_i \text{ or } \vec{x}^i = a_i$$

Definition 11 (Leaf) *Let $\mathbb{S}$ be a set (the discrete classification labels). A leaf is an element of $\mathbb{S}$.*

The following two definitions are indirectly recursive:

Definition 12 (Decision tree edge) *Let $\mathcal{D}$ be a set of decision nodes, and let $\mathbb{S}$ be a set of leaves. A decision edge is an element of $\mathcal{D} \times \{\mathcal{D} \cup \mathbb{S}\}$.*

Definition 13 (Decision node) *Let t be a decision test, and let Δ_1 and Δ_2 be decision edges. Then, a node is a function*

$$\Delta(t) := \begin{cases} \Delta_1 & \text{if } t = \text{true} \\ \Delta_2 & \text{if } t = \text{false} \end{cases}$$

Intuitively, a Δ represents a test on one attribute that leads either to a target class or to another test. If $\Delta_x \in \mathcal{D}$ maps into a $\Delta_y \in \mathcal{D}$ then Δ_x is called a 'predecessor' of Δ_y, and Δ_y a 'successor' of Δ_x. A decision node without a predecessor is called 'root node'. Eventually, the evaluation of a Δ has to lead to a value of $\mathbb{S}$ in order to be a decision tree.

Definition 14 (Decision tree) *A decision tree Θ is a decision node whose recursion always terminates.*

Note that Θ is classifier $\mathbb{X} \to \mathbb{S}$ (as defined in Definition 7). Θ needs only a subset of ordained scalars from the vectors in $\mathbb{X}$ to calculate its value $s \in \mathbb{S}$. This subset is given through the subset of $\mathbb{T}$ that is used in all the nodes of Θ.

There are several well established algorithms for learning decision trees from data [24, 136, 184]. This section introduces the properties and basic techniques related to decision trees. In the subsequent chapter, these techniques will be extended and adapted to molecular biological problems.

2.2.1 Graphical representation of trees

The common way of representing a decision tree is by a directed acyclic graph [89]. In decision trees, every node, which is not a leaf, represents a test on an attribute. In Figure 2.3 there are two such nodes, the root node (A) and another node (C). In order to come to a decision about an object's target class, the decision tree tests for the values of the indicated attributes. Suppose we had the following vector representing an object:

$$(A = a1, B = b2, C = c2)$$

For deciding the object's target class, the tree starts with the root and first tests for attribute A. As its value is $a1$ it follows the edge "a1" to the next node (C). Now, it tests for attribute C and follows the appropriate edge "c2" to the leaf with the label "I". The object is thereby classified as belonging to target class "I".

Note that the tree came to its decision regardless of the value of attribute B. Obviously, only attribute A and C are relevant for the decision of an object belonging to either target class "I" or "II". Further, as one leaf can be reached just by knowing the value of attribute A (the rightmost leaf), we can note that higher nodes have a stronger significance for the decision problem. Given any (huge) amount of attributes, a good classifier can come to its decision with a hierarchically ordered set of tests on only a few attributes. Thus, it is often possible to understand the importance of attributes for the decision problem just by looking at the tree's graphical representation.

Minimum Description Length Principle

The quality of decision trees using only *necessary* tests is considered to be advantageous according to the 'Minimum Description Length Principle' (MDL). This principle has its origin in a discussion started by William of Occam in the year 1320 who stated: "Prefer the simplest hypothesis that fits the data." [115]. This statement is still valid because it is believed that shorter hypotheses constitute better generalisations for models of real world problems [112]. However, apart from empirical evidence there is no hard proof for this statement.
Occam's statement has meanwhile been developed into the MDL [44]. The MDL can be described formally through the following definitions:

Definition 15 (Code) *[178] A code is a set of unambiguous rules specifying the manner in which data may be represented in a discrete form.*

Definition 16 (Minimum Description Length Principle) *Let D be a training matrix (as defined in Definition 5) and $\vec{c}$ be the column vector of D holding the classification labels. Then, the Minimum Description Length principle is a criterion seeking a model which permits the shortest encoding of the vector $\vec{c}$ given the matrix $D \setminus \vec{c}$.*

Applied to decision trees, MDL can be described in a more specific manner. The encoding of a decision tree Θ are the nodes[6] attributed to Θ (see Definition 14). For any given training matrix, the set of all possible nodes $\mathcal{D}$ is finite[7]. In information theoretic terms, this set can be regarded as the 'symbols' of an 'alphabet' [76].

Definition 17 (Description length of a tree) *The description length of a tree Θ is the cardinal number of tests in Θ. It is denoted as $|\Theta|$.*

The MDL criterion for decision trees is to minimise $|\Theta|$. Or in other word, MDL is to minimize the number of nodes in a decision tree while retaining the reproducibility of all class labels of the training data.

More recent works on MDL handle the problem in a more "flexible" manner, allowing the vector $\vec{c}$ to be only approximated by the model. The thereby inherited errors in $\vec{c}$ are encoded separately [143, 139, 107]. To obtain the shortest description, all subsets of $\mathcal{D}$ receive a score according to their probability of being significant for the reproduction of preferably many scalars of $\vec{c}$ (classification labels). The ones with the lowest scores are dropped for the benefit of fewer models (subsets of $\mathcal{D}$) to be considered. For each remaining model, the misclassified scalars[8] of $\vec{c}$ are encoded into a vector $\vec{c^*}$. The criterion is then to find the model that minimises $|\mathcal{D}| + ||\vec{c^*}||$ where $||\vec{c^*}||$ denotes the number of components of $\vec{c^*}$.

The application of MDL is particularly feasible if the training data contains noise. That topic will be tackled in subsections 2.3.1 and 2.3.2.

2.2.2 Propositional rules

Another way to represent decision trees is by propositional rules [161, 115]. Intuitively, these rules can be read as if-then-statements. That is, if a rule's "preconditions" are met then the rule's "consequence" applies. In case of decision rules, such a consequence is always the attribution of a target class to an object. For instance, the rule

$$(A = a1 \wedge C = c2) \rightarrow I$$

[6] Thus, the basic components of that code are functions.

[7] For continuous data, we attribute equivalence classes of functions to nodes. Functions that process the same training samples and lead to the same result are equivalent.

[8] often referred to as 'exceptions'

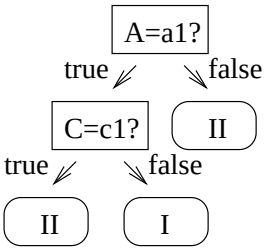

Figure 2.3: A simple decision tree consisting of the two tests A and B and three leaves for the target classes 'I' and 'II'.

means that given an object with attribute A holding value $a1$ and attribute C holding value $c2$ then the object's target class is I.

Decision trees can easily be mapped into propositional rules [115]. One just has to extract every path from the root to a leaf and transfer it into a rule. For the tree in Figure 2.3, this leads to the following three rules:

$$(A = a1 \wedge C = c1) \rightarrow II$$

$$(A = a1 \wedge C = c2) \rightarrow I$$

$$(A = a2) \rightarrow II$$

Rules referring to the same target class can be combined into disjunctions in order to obtain a more compact representation. In our example, this applies to the two rules leading to target class II. We thus obtain the following ruleset for the example tree:

$$(A = a1 \wedge C = c2) \rightarrow I$$

$$((A = a1 \wedge C = c1) \vee A = a2) \rightarrow II$$

There are several advantages of the representation through rulesets. First, the rules can be read by humans as plain sentences, thereby making it somehow intuitive to understand them [71]. Second, there are efficient methods to delete redundant information from rules which are not applicable to trees [136]. And last but not least, some pruning strategies work on the rule representation only [115].

A drawback of rules is that learning them straightforwardly is less efficient than learning trees. That is why, in this thesis, the rules have only been used indirectly[9]. The subsequently introduced methods are all based on the tree representation.

2.2.3 Learning decision trees

There are several well known techniques for learning (inducing) decision trees [24, 132, 136, 122, 4]. These techniques mostly come from statistics [72], graph theory [50], and information theory [40].

The presumed founder of the current decision tree community is said to be Leo Breiman. He joined knowledge of the three mentioned disciplines (in particular statistics) and combined it with a greedy algorithmic framework which he called *CART* (Classification and Regression Trees) [24]. CART is a family of algorithms specifying four key characteristics that have been followed and/or enhanced by all subsequent tree learning programs:

[9] Rules have been used by the C5.0 programs for pruning trees.

1. a split-criterion
2. criteria for class assignment to leaves
3. stop-critera for the induction process
4. pruning strategies

All programs of this family fall back onto greedy principles to cut computational complexity because an exhaustive search through the hypothesis space would be an NP-complete problem [2].

The original CART was furnished with numerous alternatives for the above demanded criteria. It also suggested strategies for tree evaluation and pruning (see also Subsection 2.3.2). Many of those ideas have been picked up and expanded in subsequent publications and programs. One such technically mature and widely available collection of programs is introduced with the programs ID3 and C4.5 in the following subsection. Pruning strategies and alternative stop-criteria are discussed in Section 2.3.

2.2.4 ID3/C4.5

Probably the most commonly used and studied programs for decision tree induction are ID3 and particularly its successor C4.5 (C5.0 respectively) [136]. Formally, they are a specialisation of CART. The basic principle of them is described in Table 2.1. As shown below, the difference between ID3 and C4.5 is hidden in the 'best test' on an attribute (line 4 in Table 2.1). This is the formerly mentioned split criterion.

Finding this "best test" is a complex problem itself. It is a key characteristic of decision tree induction algorithms. As seen later in this section, the method for finding the test also purports the type of data which can be handled by the algorithm.

Determining the best test

As "best test", Quinlan [132] proposes to choose the test on an attribute that yields most information gain regarding the classification problem. This is the test which solely allows for the most accurate classification possible. Intuitively, this test splits the training data into subsets with "least disorder" regarding the target classes. 'Least disorder' means that in the subsets, objects of one target class have to outbalance objects of the other target class(es) in the clearest manner possible.

In Figure 2.4, we see a set of eight circles in a feature space of the continuous attributes A and B. Each circle represents an object which can be characterised through its value of attribute A and B. The solid circles belong to target class I and the others to target class II. The objective is now to find the single attribute that can classify the objects into their target classes most accurately, thus leaving the subsets with least possible disorder regarding to the target classes.

In this example, this would be a test on attribute A with threshold $a1$. This test can split the data into one "tidy" subset with only solid circles and a second subset with only little disorder containing predominantly transparent circles. The best test on attribute B (the one with threshold $b1$) would only have led to more disordered subsets (see Figure 2.4). The emerging question is how to compute disorder.

Quantifying the worth of a split

For the quantification of order or disorder[10] within a subset, Quinlan suggests to use an entropy measure [166]. For the quantification of disorder of *several* subsets, he uses the concept of mutual

[10] Quinlan and Breiman call it 'impurity', but here, the information theoretic term 'disorder' will be used.

The following description outlines the basic greedy and recursive algorithm for decision tree learning. It is given in a pseudo-computer-program-code. All terms in italics (e.g. *root node*) denote instances of data structures (e.g. values of variables). The selection criterion for *best test* and its typisation is a key difference of all common decision tree learning programs. In the subsequent generic description, *best test* is given as an abstract structure that can take values which specify subsets of the training data.

Input:

- *training data* in the form of a training matrix
- *target attribute*, a discrete-valued attribute whose value is known for all data objects of the training data
- *attribute set*, the set of attributes of *training data* without the target attribute

Output: a *decision tree*.

Method `grow`::(*training data*, *target attribute*, *attribute set*) $\rightarrow$ *decision tree*

1. Create a *root node*.
2. If the *target attribute* has the same value for all vectors in *training data* then return a *decision tree* with just the *root node* and label it as leaf with the value of the *target attribute*.
3. If *attribute set* is empty then return a *decision tree* with just the *root node* and label it as leaf with the *target attribute value* that is most common in *training data*.
4. Select the *best test* on the *set of attributes*.
5. Label the *root node* with the *best test*.
6. For each possible value c of *best test*
   - let t_c be the subset of *training data* that is specified by c.
   - if t_c is not empty
     - grow a branch b_c from *root node* and label it with value c.
     - attach the tree `grow`(t_c, *target attribute*, *attribute set* without attribute used in c) to b_c.
   - if t_c is empty
     - then attach a node to b_c and label it as leaf with the value of *target attribute* that is most common in *training data*.

Table 2.1: Top down induction of decision trees.

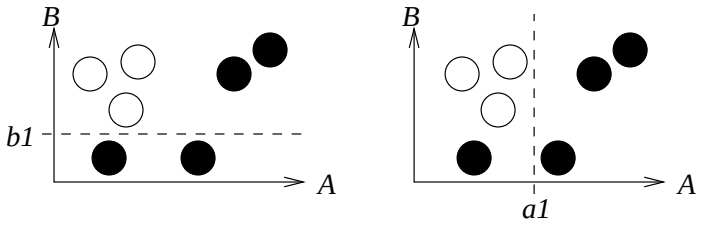

Figure 2.4: A set of objects divided by tests on either attribute A or B.

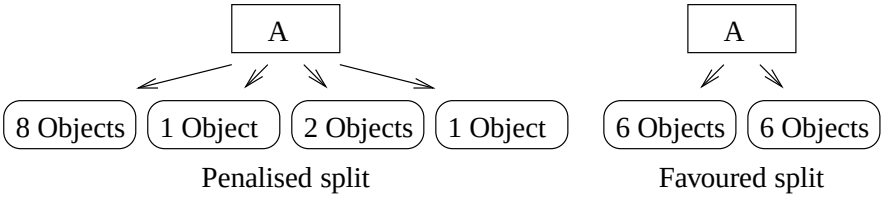

Figure 2.5: The weighed evaluation of *gain ratio*.

information [40]. In ID3, he then proposes to calculate the gain of (mutual) information obtained in subsets when applying the split, as compared to the original unsplit set. This measure is called 'information gain' [132] (see Definition 19).

Definition 18 (Entropy of a training matrix) *Let D be a training matrix containing c different classification labels, and let p_i be the frequency of occurrence of the i-th label in D. Then*

$$Entropy(D) := \sum_{i=1}^{c} -p_i \cdot log_2\, p_i$$

Definition 19 (Information Gain) *Let D be a training matrix with discrete attributes, and let $|D|$ denote the number of data objects in D. Let A be one attribute of D, let $dom(A)$ be the domain of A and let $|dom(A)|$ be the cardinality of $dom(A)$. Further, for any $i \in \{1, ..., |dom(A)|\}$ let α_i denote the i-th element of $dom(A)$. Let D_v denote the subset of data objects of D that carry value v for attribute A. Then*

$$Gain(D, A) := Entropy(D) - \sum_{i=1}^{|dom(A)|} \frac{|D_{\alpha_i}|}{|D|} \cdot Entropy(D_{\alpha_i})$$

Since information gain has a strong bias toward attributes with many possible values it has been replaced in C4.5 by the so-called 'gain ratio' [115, 136] (see Definition 21).

Definition 20 (Split Information) *Let D be a training matrix, and let A be one attribute of D with c different labels. Then*

$$SplitInformation(D, A) := -\sum_{i=1}^{c} \frac{|D_i|}{|D|} \cdot log_2 \frac{|D_i|}{|D|}$$

Definition 21 (Gain Ratio) *Let D be a training matrix and let A be one attribute of D, then*

$$GainRatio(D, A) := \frac{Gain(D, A)}{SplitInformation(D, A)}$$

The gain ratio measure favours balanced equilibrated trees. Those are trees that primarily contain splits that cause a uniform number of data objects in few branches. Splits with asymmetric allocations and many branches are penalised by the criterion (see Figure 2.5). But this measure runs into numerical problems when attributes have the same value for nearly all objects [46].

All measures for finding a (nearly) optimal split in decision trees have some drawback for specific properties of the data. That is why there have been numerous proposals for alternative measures [24, 136, 115]. However, empirical studies suggest that the choice of the measure is really not that crucial [113]. In this thesis, the choice of the measure is even less critical as all data is handled in a binary manner (see Subsection 2.3.5 and Chapter 3).

2.2.5 Alternative developments of decision tree learners

Apart from the CART familiy there has been another strand of decision tree learners: the AID (Automatic Interaction Detection) family [116]. It was designed to detect complex relationships between attributes. The AID family originally comprised techniques aimed at only a certain objective. That was, the detection of complex statistical relationships (e.g. combinatorial relationships). However, research of the AID family has meanwhile integrated into the rest of the decision tree community. It is thus of little practical value to distinguish between them. Concepts and ideas originating from the AID family will be used in this thesis as they are needed.

2.3 Advanced issues

The basic decision tree induction algorithm of the CART family is fit for many problems which supply discrete valued data sets. For the application in the molecular biological domain, they still need to be adapted to continuous data and other specific properties. Many of the problems appearing in this context have already been addressed in the machine learning domain. Those will be described in this section. Genuine new techniques for the adaptation to molecular biological data will be described in the next chapter.

2.3.1 Overfitting

One of the major problems of all machine learning techniques is overfitting. Overfitting means that learned classifiers tend to classify (nearly) perfectly the objects of the given training data but perform poorly on other data objects. The classifiers are thereby overfit to the training data.

The common way to detect overfitting is by reserving parts of the training data as validation data. The decision tree is induced only on the non-reserved parts of the training data. Then, the tree's classification accuracy on the validation data is measured. This is in turn compared against the accuracy of a simpler[11] version of the tree. If the simplified version performs better on the validation data but the original tree performs better on the training data then the original tree is *overfit* to the training data.

A formal specification of the term 'overfitting' can be given with the following definitions:

[11] See subsection 2.3.2 for simplification strategies.

Definition 22 (Correctness) *Let Θ be a decision tree and let D be a training matrix with column vector $\vec{c}$ holding the classification labels for each data object of D. Then, $correct(\Theta, D)$ is the number of correctly reproduced labels of $\vec{c}$ by Θ.*

Or, in other words, $correct(\Theta, D)$ gives the number of training samples that can be classified correctly through Θ.

Definition 23 (Classification Accuracy) *Let Θ be a decision tree and let D be a training matrix with column vector $\vec{c}$ holding the classification labels for each data object of D. Then,*

$$accuracy(\Theta, D) := \frac{correct(\Theta, D)}{||\vec{c}||}$$

where $||\vec{c}||$ denotes the number of scalars of $\vec{c}$.

Definition 24 (Reduced Decision Tree) *Let Θ be a decision tree with k nodes Δ, and let $1 \leq n \leq k$. Then, $\Theta(n)$ is the tree Θ with only n of the k functions Δ where*

- *the number of Δ is reduced by successively deleting those Δ that map solely into leaves*[12]*,*
- *the predecessor Δ_{pre} of such a deleted Δ is replaced with a Δ^* that is similar to Δ_{pre}, but instead of mapping into the deleted Δ it maps into a leaf that is assigned with the value $s \in \mathbb{S}$ that was most significant*[13] *for the deleted Δ.*

Definition 25 (Overfitting) *Let D be a training matrix. Let VS (validation set) be a random selection of data objects of D with $\vec{vs}$ holding the classification labels for VS, and let TS (training set) be D without the data objects of VS and with $\vec{ts}$ holding the classification labels for TS. Then, a decision tree $\Theta(k)$ with $k \in \{\mathbb{N} \setminus 1\}$ is overfit to the training data D if*

$$accuracy(\Theta(k), TS, \vec{ts}) > accuracy(\Theta(k-1), TS, \vec{ts}) \quad \textit{AND}$$
$$accuracy(\Theta(k), VS, \vec{vs}) < accuracy(\Theta(k-1), VS, \vec{vs}).$$

An illustration of overfitting is given in Figure 2.6. After a while, trees with higher accuracy on the training data perform worse on validation data. As *general* rules are tried to be derived from decision trees in the molecular biological domain, it is important to avoid overfitting. Strategies to avoid overfitting of decision trees can be categorised into

- approaches that hold before perfect matching of the training data
- approaches that apply a pruning step after the end of tree construction

The latter approach will be discussed in subsection 2.3.2. Methods of both categories will be applied in the subsequent chapters.

2.3.2 Pruning

When trying to further process or interpret decision trees it is important to obtain simple and accurate classifiers in the first place. Besides choosing a favourable strategy to induce simple trees there are several methods available for simplifying existing trees. These methods are called 'pruning' techniques. Generally, they cut inefficient parts out of trees and prune the remaining parts into a new and less complex structure.

There are many pruning strategies available which are fit to specific types of data [24]. Here, two techniques are described which will be used later in this thesis.

[12] There are several strategies on how to choose the two Δ. A simple one is to first select those which are furthest away from the root function Δ_{root}. The distance of any target Δ to Δ_{root} is measured as the number of Δs which have to be called by Γ until the target Δ is called.

[13] Most significant is that value s that has occurred most frequently in the subset of the training data on which the deleted Δ have originally been built.

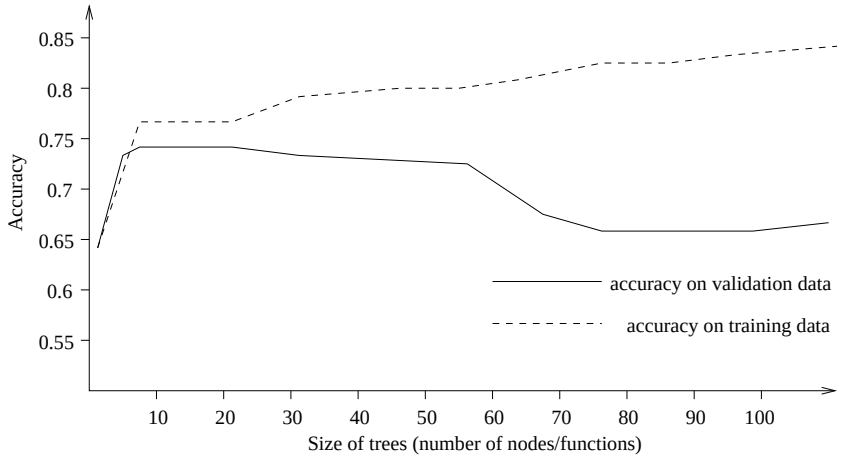

Figure 2.6: A typical progression of classification accuracy when decision trees become overfit (example taken from [115]). The accuracy on the training data rises with the number of nodes allowed in the tree while the accuracy on the validation data drops for larger sized trees.

Reduced error pruning

'Reduced error pruning' is a simple technique used for pruning decision trees [133]. Starting from the root node, it systematically replaces the subtrees by a leaf that is labelled with the most common classification of training samples associated with that branch. If the simplified tree performs more accurately on the validation data the leaf is accepted. Otherwise, the old subtree will be kept and the procedure carried out again on the branches of that subtree.

Rule post-pruning

A generally more effective method is 'rule post-pruning' [136]. It uses the rule representation of decision trees. Successively, for any rule, one prerequisite is deleted. If the abbreviated rule performs more accurately on the validation data than the initial rule, then the abbreviated version will replace the original rule. The procedure is applied to the rules until no more improvement can be achieved on the validation data.

2.3.3 Cross-validation, Jackknife, Bootstrapping

One of the frequent problems with current metabolite concentration and gene expression data is its sparsity in data samples. That is, many datasets provide only few samples with a disproportionate high number of attributes. With too few training samples, machine learning techniques perform poorly and lead to overfitted classifiers [20, 1, 115]. Taking out samples from the data set as validation data would further reduce the training set and thereby the classifier's quality. But knowledge about the accuracy of a classifier is required if the goal is to extract valid rules from it. Three closely related techniques for evaluating a given statistic in such a stringent environment are introduced below. Two related techniques aiming at the same problem, "Boosting" and "Bagging", are introduced in Subsection 2.3.7.

Bootstrapping

Bootstrapping is a simple way of detecting the generalisation error of a chosen statistic on a given data set [43, 51]. Given a data set and a statistic that has to be evaluated on it, bootstrapping

procedes as follows:

- Multiple resampling of the given data set.
- Calculation of the given statistic on the resampled sets.
- Evaluation the standard-deviations of the distributions of the calculated values of the statistic.

A resample is usually[14] established by randomly picking n objects out of the k ($k \geq n$) objects of the original sample. The mean of the standard-deviations of the distributions indicates the strength of the generalisation error.

Cross-validation

Cross-validation is a specific way of resampling that is commonly used to detect overfitting [24]. Most often, it is used for measuring the generalisation error of classifiers especially on smaller data sets.

Cross-validation needs an integer parameter f, the fold. An f-fold cross-validation divides the set of samples into f approximately equal sized subsets[15]. In turn, a decision tree is constructed from the samples of $f - 1$ subsets. The classification accuracy of this tree is then tested on the remaining subset. This procedure is repeated until each subset has been used once for testing the classification accuracy. The average accuracy of all f folds is used as an estimate for the accuracy of a tree grown on the complete set.

Jackknifing

Jackknifing is closely related to cross-validation, sometimes even referred to as 'leave-one-out-cross-validation' [191]. However, it is not only used to calculate the generalisation error of classifiers but to estimate the bias of any statistic. Applying the Jackknife, each training case is omitted in turn and the chosen statistic is calculated on the remaining subset. This is similar to f-fold-cross-validation when f is the number of samples in the complete training set and the statistic is the generalisation error of the classifier. The average of the statistics calculated on the subsets is then compared to the statistic on the entire training set. The difference can be used as an estimate for the overall bias.

Empirical studies emphasise the superiority of general f-fold-cross-validation over Jackknifing, especially on small data sets.

2.3.4 Missing values

Practically, all biological data sets contain missing values. This is due to the complex experimental setups which potentise the impact of technical imprecision through many levels. However, most algorithmic analyses require complete sets. Thus, in most cases missing values have to be erased from the data.

Some of the missing values can still be estimated through expert knowledge. But some values remain unknown for the computational analysis.

There are two basic strategies for coping with missing values [124]:

- deletion and

[14] More elaborate ways of resampling assign probabilities of being picked up to each object.

[15] If the number of samples cannot evenly be divided by f some subsets may contain an extra sample.

- imputation.

Statistically, deletion is the safe way. That is, no additional bias can be introduced into the data. All attributes and/or samples that contain a missing value are deleted from the data set. Unfortunately, this can erase desired structure and quickly leads to no remaining data at all. This is particularly disadvantageous in domains with more than 10% missing values in the set (e.g. biological data). Thus, imputation has to be applied in most cases [134].

Imputation of averages

The substitution of missing values with default values is called imputation. Using averages for the substitution is one of the most common approaches to do that. Some basic averages work on:

1. the concerned attribute,
2. the concerned object, or
3. the complete data set

While all of these can lead to feasible results on rather homogeneous types of data, they will often lead to poor results on heterogeneous data such as biological sets. A good aid is to identify an "environment" in which the missing value occurs. Such an environment defines a subset of samples with a certain common characteristic. It is then possible to calculate averages only from samples of the same environment.

These environments can be identified with pure statistical measures. They could be, for instance, a subset of samples

- with similar variances,
- with similar scales, or
- with similar distributions of their attributes.

When inducing decision trees, such a subset can also be defined at a certain node by the samples remaining under that node. The subset can be further specified by samples in the split with only a certain classification [113].

Another way of defining an environment is by means of characteristics of the biological domain. For instance:

- samples of the same genotype,
- samples taken from the same series of experiments, or
- samples measured by the same experimenter.

Ultimately, any of the above environments can be combined in order to obtain the most specific characterisation of a subset. Yet, too specific characterisation can decrease the number of samples in the subset to a number that is no longer statistically justifiable for further induction. The better the characterisation of the subset is, the more it is assumed that the substitution value is close to the unknown real value. However, no general rule can be given to accomplish this task. It remains mostly subject to the experience of the analyst.

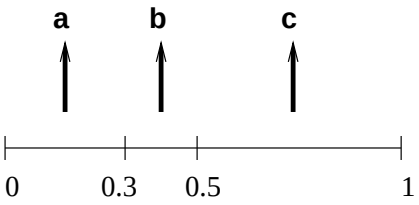

Figure 2.7: Mapping of intervals into the discrete values a,b and c.

Other strategies for handling missing values

There are numerous more complicated strategies for coping with missing values published in the literature (e.g. [140, 182, 126]). Most of them were designed to work under specific conditions with specific problems. Thus, they cannot be applied straightforwardly to other data. Furthermore, only very few publicly available implementations of those techniques are available at the moment. A good synopsis of more general techniques can be found in Little & Rubin [104]. In tests on synthetic and real data, differing strategies have shown to have only a low effect on the output of the subsequently introduced data and algorithms (see also Subsection 4.2.2). That is why more elaborate missing value strategies have not been examined in detail in this thesis.

2.3.5 Continuous data

Plain decision tree induction algorithms can handle discrete data only [24, 132]. The biological data used in this thesis, on the other hand, is exclusively continuous. Hence, the data has to be discretized à priori or the algorithms have to be adapted. Since all results of this thesis depend strongly on an appropriate handling of continuous attributes the used methods will be described in detail. The adaptation of algorithms will be discussed in this subsection and general discretisation in Section 3.1.4.

Basic handling of continuous attributes

When discussing the use of continuous attributes in decision tree induction two types of attributes have to be considered: the target attribute and the other attributes.

The range of the target attribute has to be divided into target classes by an expert. Once the target classes are assigned to intervals of the attribute's domain, the induction algorithm treats it as discrete valued. This process is critical because it strongly biases the information held by the target attribute. Thus, it should be performed very considerately. Note that the expert's attribution of target classes to the samples is the reason why this process is called *supervised* learning.

The rest of the continuous attributes can be handled automatically. The idea is to dynamically map intervals of the attribute's domains into discrete symbols[16]. In Figure 2.7 three intervals of the domain, $[0, 0.3), [0.3, 0.5)$, and $[0.5, 1]$, are mapped into the discrete values **a**,**b**, and **c**. The remaining question is how to find appropriate thresholds which mark the discretisation intervals.

C4.5 uses a boolean approach. Every continuous attribute considered for a split is divided into two intervals. Since there is a finite number of samples in the training data there can only be a finite number of possible binarisation[17] thresholds for each continuous attribute. Note that this approach can easily be expanded to derive non-binary discretisation by subsequently applying several different binary discretisations on the same attribute.

[16] This is also called *local* discretisation and will be discussed in more detail in subsection 3.1.4.

[17] Binarising means discretising into the domain $\{0,1\}$.

Sample	1	2	3	4	5	6
Attribute X	0.5	0.7	0.8	1.4	1.8	2.1
Target attribute	A	A	B	A	A	B

Table 2.2: Training data with 6 samples: a continuous attribute, and a binary target attribute.

In Table 2.2, there is an example with a small training set. For demonstration purposes, the samples are sorted according to the value of attribute X. With six different values of attribute X there are five feasible thresholds to binarise attribute X's domain (for instance 0.6, 0.75, 1.2, 1.6, 1.95). By mapping the values of attribute X into two classes (0 for values below a threshold and 1 for values above it) we obtain 5 possible binarisations for attribute X.

When the induction algorithm evaluates the worth of a split according to attribute X, it now calculates the worth for each of the possible binarisations: attribute X with threshold 1, attribute X with threshold 2 and so on. Hence, many more evaluations have to be performed than if attribute X were binary from the start.

However, Fayyad proved that not all binarisations have to be considered for the determination of the best split [54]. Only those thresholds that lie between samples of different target classifications are feasible. In Table 2.2, those are the ones between sample 2 and 3, between sample 3 and 4, and between sample 5 and 6. That way, the number of evaluations can usually be cut down considerably.

Advanced methods for handling continuous attributes

Some problems with the above method have been reported when a data set contains continuous and discrete attributes at the same time [4, 48]. Then, some splitting criteria (e.g. gain ratio) have a strong bias toward the use of continuous attributes as opposed to discrete ones [137]. Several authors have proposed alternative splitting criteria that try to prevent this bias [4, 137]. Since all of the biological data used in this thesis is purely continuous those problems do not apply to the studies below. That is why the basic discretisation procedure of C4.5 will be used in all cases.

2.3.6 Oblique hyperplanes

One limitation of conventional decision tree algorithms is that all splits are performed parallelly to an attribute axis. For decision trees, this means that at each node there can be a test on only one attribute.

In Figure 2.8 there is an example where this limitation would lead to a non-perfect split: It is not possible to split the circles correctly by any test considering only one attribute (an axis-parallel hyperplane respectively). The best axis-parallel split would be the one depicted in the lefthand figure of Figure 2.8 by the threshold $b1$ on the B-axis.

However, there would be a better hyperplane to split the data as indicated in the righthand figure of Figure 2.8. This is an *oblique* (non axis-parallel) hyperplane. It splits the circles perfectly into the subsets of filled and empty circles. To achieve a split like that at a node in a decision tree, the test at that node has to be a linear combination of the attributes A and B. Here, this is indicated by the sum $a1 * x + b1 * y$. The remaining issue is thus to determine the attributes (here A and B) and the factors (here x and y) for each split.

OC1 is an algorithm that can induce tests on linear combinations for each node [122]. In practical applications, OC1 delivered significantly shorter trees on problems of continuous domains [121]. The crucial drawback of OC1 is its computational complexity [76, 122].

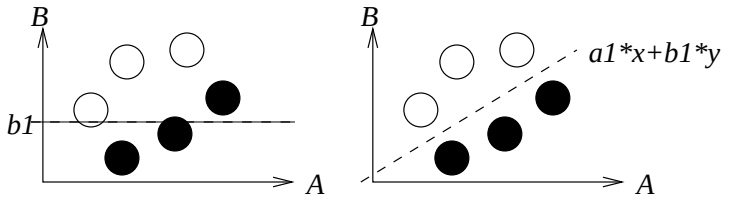

Figure 2.8: An axis-parallel and an oblique split.

Because of the otherwise favourable characteristics of OC1 there have been recent studies to cut down on the computational demand [31]. The idea is to use evolutionary algorithms [75, 115] for the approximation of good linear combinations. This idea is still under examination. Up to now there is no implementation of it publicly available. So, in subsequent chapters I will still use axis-parallel decision tree inducers, although this means clearly an interesting perspective.

2.3.7 Ensemble techniques

The main objective of all machine learning techniques is to construct classifiers that classify new observations correctly [115]. To achieve that goal, many different classification techniques have been developed. A basic paradigm of them is to construct single classifiers with high classification accuracy. Yet, single classifiers are bound to several constraints.

A recent trend to improve classification accuracy of existing techniques is the use of sets of classifiers rather than single ones. Boosting (or Arcing) [63] and Bagging [23] are the two basic approaches known to date. Both offer a strategy to induce sets of classifiers, also referred to as 'ensembles' or 'decision forests'. When a continuous value is to be predicted, the average prediction of all classifiers in the ensemble is used as outcome. For the prediction of discrete classes, a majority vote of the ensemble is used.

Theoretically, Boosting and Bagging can be used to improve any classification technique delivering single classifiers. But they perform with varying success on each of the individual methods. On decision trees, for instance, improvements are reported to be considerable [105]. On support vector machines, on the other hand, they are very weak, because the boosting and the induction principle are conflicting.

Bagging

The word "Bagging" is derived from "bootstrap aggregating" [23]. Bootstrapping has been introduced in subsection 2.3.3. The basic principle of Bagging is as follows: Several bootstrap replicates of the original data set are used as training sets. Subsequently, a classifier is constructed for each of the training sets. These classifiers are in turn aggregated to an ensemble. As described above, a classification is achieved by taking a majority vote of all classifiers in the ensemble (or, for continuous predictions, by computing the average, respectively).

For decision trees and neural networks, this technique generally improves classification accuracy [12]. Tests on several data sets drawn from the UCI data repository [120] have shown for decision trees that Bagging ensembles nearly always outperform single classifiers. Such ensembles are also relatively robust against noisy data [105].

Boosting

Boosting describes a technique to "boost" (improve) classification accuracy of weak learning algorithms [150]. This approach is based upon the so called "PAC learning theory" (Probability Approximately Correct) [185] which is not discussed in this thesis. There are several known boosting algorithms available [147]. Here, I describe briefly a version of AdaBoost [151].

The word "AdaBoost" is derived from "adaptive boosting", meaning that the algorithm is constantly adapting the training set during progression [152]. The principle is as follows: Initially, a classifier is learned on the original training set with any chosen learning algorithm (e.g. C4.5). Then, a new training set is produced by attributing weights to the samples of the original data set. That is, samples which are classified correctly by the learned classifier get a lower weight, and incorrectly classified samples get a higher weight. One illustrative way to achieve this is by simply duplicating misclassified samples in the new training set (see Figure 2.9). Based on the new training set, a new classifier is learned. Finally, all n learned classifiers (n can be arbitrary or determined by some criterion [153]) are combined in an ensemble. Classifications are again obtained by majority voting as described in subsubsection "bagging".

Tests on UCI data (see above) have shown that boosted decision tree ensembles often classify better than Bagging ensembles [105]. However, boosted ensembles are prone to overfit training data and they are somewhat susceptible to noise. Consequently, on more data sets than with Bagging, the boosted forests performed worse than single trees [12].

Characteristics of ensemble techniques

For improving classification accuracy, one could draw the conclusion that carefully applied boosting yields better results than Bagging. On the other hand, Bagging is less prone to noise and often generalises the data better. Bagging could thus be the method of choice for less experienced users [105].

However, we note that all ensemble techniques have in common that they improve classification accuracy at the sacrifice of simplicity. To be more specific, rules (or criteria) for a classification are multiplied by the number of classifiers in the ensemble. That is why ensemble techniques are generally unfavourable for interpretation and further processing of the classifiers [125]. In the thesis at hand, the focus *is* on interpretation and further processing. Thus, the known ensembles techniques cannot be used directly here. In section 3.2, a new ensemble technique is introduced with the desired behaviour.

2.3.8 Decision lists

Decision lists are a concept closely related to decision trees [144]. They are rule learning classifiers. A few scientists consider them as an independent machine learning technique [119]. Though, they really are a superclass of other techniques (e.g. decision trees [29]).

The lists serve for finding decisions[18]. They consist of a linearly ordered set of boolean functions, each consisting of k clauses. Such a function can either lead to a decision or refer to a subsequent function. If k is set to 1 the lists are a specialisation of decision trees.

The lists obtained their name from their listlike appearance (see Figure 2.10). Due to their linearity the lists have a very clear structure. Hence, they can easily be understood and interpreted.

For higher k, there is no generally approved learning strategy for the lists. Often they are learned with a greedy hill climbing algorithm [144] or by methods based on PAC learning [185, 33]. But more "exotic" learning strategies, e.g. through genetic logic programing [177], are also in discussion.

[18] In this context, decisions are the same as predictions or classifications.

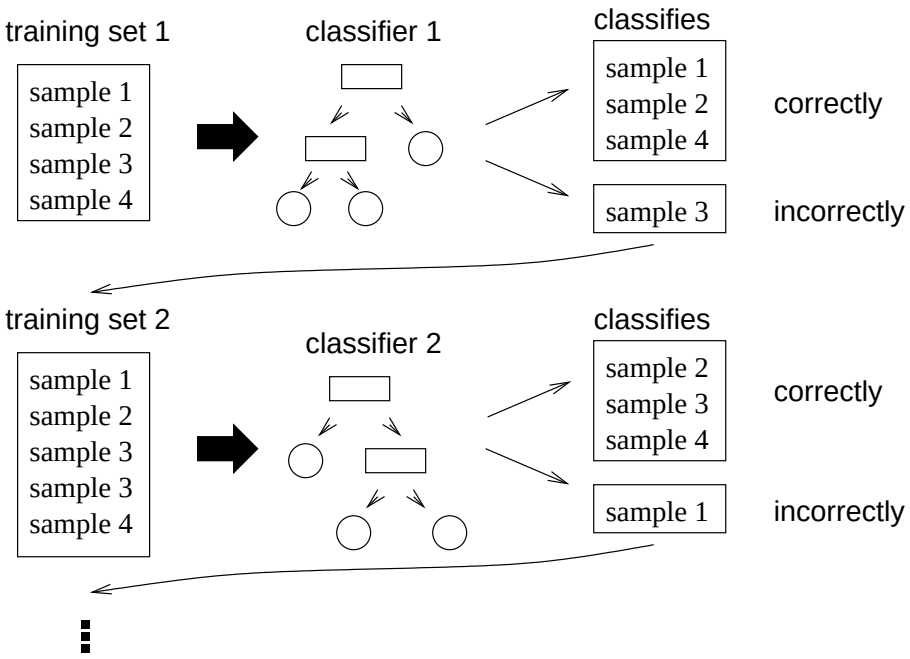

Figure 2.9: Boosting a weak learning algorithm.

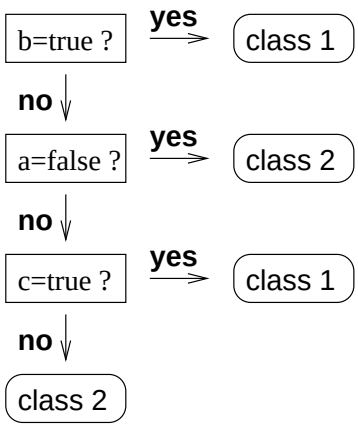

Figure 2.10: A decision list testing binary attributes a,b, or c to predict class 1 or 2.

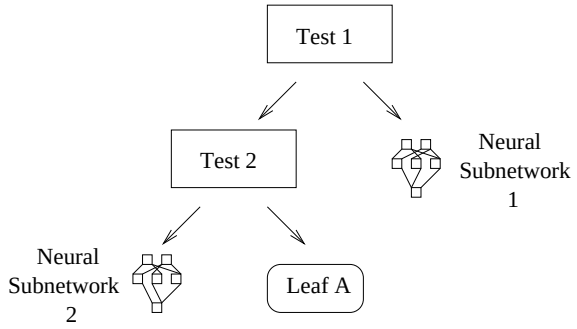

Figure 2.11: A hybrid decision tree with neural networks as sub-classifiers.

Since decision tree learning is more straightforward and the trees are still well interpretable it was opted for using the trees instead of lists in subsequent chapters.

2.3.9 Hybrid decision tree approaches

An alternative way to develop improved classifiers is to combine existing methods into a *hybrid* technique. The challenge is to exploit the advantages of different methods while avoiding their drawbacks.

In machine learning, I distinguish two different classes of hybridisations:

1. techniques utilising one method to induce a better classifier of another method

2. techniques combining two different principles into a hybrid classifier.

The first class has already been mentioned in Subsection 2.3.6 where genetic algorithms have been proposed for inducing oblique decision trees [31]. Genetic algorithms have also been used to approximate other types of rule learners such as decision trees or first order logic [183]. In these cases, one learning method is used to circumvent a specific problem which is hard to solve within the framework of the other method. In the end, a pure "one-method" classifier is obtained that can be handled by conventional computational tools. This approach will not change the characteristics of the used type of the target classifier. It is therefore not necessary to consider techniques of this kind when studying interpretability and handability of classifiers (as will be shown in subsequent chapters).

The second class of hybridisations leads to newly structured classifiers. The idea is to replace inefficient parts of a classifier with more efficient structures of another type of classifier [163] (see Figure 2.11). For instance, decision trees have known weaknesses due to their greedy split criterion at each node [32]. If such a split leads to a remarkable loss in classification accuracy the corresponding node (or leaf) can be replaced by a better suited (sub-)classifier [196]. The advantage of this approach is that it can preserve desired features of decision trees, in particular interpretability within the top nodes, while facilitating a higher classification accuracy through another technique. This approach is strictly aimed at improving classification accuracy. It does not improve interpretability and handability which is needed in the subsequent chapters of this thesis. That is why I have not considered it any further.

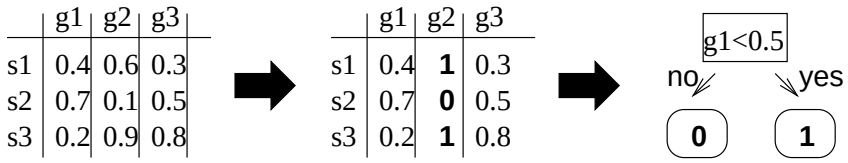

Figure 2.12: The production of a decision tree on the discretized gene `g2`.

2.4 Previous applications in biological data analysis

2.4.1 Classification of biological tissue samples

In most investigations where decision trees have been applied to molecular biological data, they have been used for classification tasks [165, 195, 47, 15, 181]. In this field, decision trees compete directly against all other classification techniques. Particularly Support Vector Machines [186, 35, 160], Neural Networks [145] and Bayesean Networks [34, 80] have been used for classifications in the life sciences in numerous publications [159, 130, 28, 108, 14, 97]. As the classification accuracy of basic decision trees tends to be lower than that of the above mentioned techniques [67] this method is rarely used as a prime choice for straightforward classification tasks. Instead, it often appears as a statistically different approach to reinforce classification results obtained from other techniques (e.g. in Beerenwinkel et. al. [15]). But the classification accuracy of decision trees can be enhanced by the means of boosting and bagging (see subsection 2.3.7). With these improvements, the trees match the accuracy of other methods on particular data sets.

The subsequently introduced methods do not rely on ultimate classification accuracy but on interpretability. That is why issues on accuracy are not discussed in more detail here.

2.4.2 Reconstruction of gene networks

For this thesis, the more important area where decision trees have been applied to molecular biological data is the reconstruction of gene networks [170].

Gene networks are a representation of interactions between genes. Empirically, behaviour of genes is determined through the behaviour of the corresponding gene expression levels [86]. In the last years, it has become possible to measure gene expressions on a large scale by high-throughput methods [197]. One consequence is that a lot of effort is now invested in the induction of gene networks from gene expression data [3, 47, 77, 129, 180, 128, 164].

The approach of Soinov et al. [170] discusses the interpretation of decision trees for the reconstruction of gene networks. In that work, decision trees are induced for the prediction of gene expression levels through knowledge about the levels of other genes. More precisely, it is assumed that expression levels can generally be mapped into an `active` and an `inactive` state. For these two target classes (`active` [or **1**] and `inactive` [or **0**]), a decision tree is constructed that predicts the states through the expression levels of other genes. The complete process is illustrated in Figure 2.12.

The resulting decision tree is then transformed into a ruleset (similar to the description in Subsection 2.2.2). These rules are established on a subset of the original set of genes. The genes in this subset are called *explaining genes* because they are sufficient to explain the target classes (of the *predicted gene*).

Soinov et. al. argue that the characteristics of the explaining genes justify the construction of a (sub-)network [170]. They leave it unclear, though, how this network is to be constructed from the ruleset. Furthermore, such networks can be expected to be highly irreproducible from

the given data. This is due to the fact that the used decision tree induction algorithms produce unstable results when applied to noisy data [24]. However, gene expression data usually contains a lot of noise which is too high in this context. It is thus likely that decision trees constructed on similar experiments produce highly different rules and thereby highly different networks.

A solution to these difficulties is offered in Subsection 3.2. There, the idea of Soinov et. al. is enhanced by the introduction of decision forests. Forests are a good means to counterbalance the effects of instability in decision tree learning (as previously mentioned in Subsections 2.3.3 and 2.3.7).

2.5 Summary and conclusions

In this chapter, a brief introduction to machine learning has been given. Within this field, decision tree learning is one method among others. Compared to the other methods, the most favourable characteristic of decision trees is their simple representation that makes them easy to interpret and process. Another favourable property is the existence of well established heuristics to quickly learn the trees. These properties are advantageous, yet needed for the goal of extracting general knowledge from large amounts of data.

Further, an extensive introduction to issues linked to decision tree learning has been given. Some of these issues are often discussed in the literature but do not affect the techniques introduced in the next chapter. It has been described which techniques are of this kind and why. Other issues do affect the subsequently introduced techniques. For them, it has been motivated why certain choices were taken.

Finally, some examples have been given that demonstrate previous applications of the trees in the domain of molecular biology. It has been indicated that the trees are not a prime choice for pure classification tasks. But these examples show that the graph structure can be exploited to draw conclusions about an underlying network of the given data. The interpretation of trees can thus be a valuable source of knowledge in Systems biology.

In view of the given goal of extracting genuine new knowledge from Systems biology data, the trees offer best premises. That is why they were chosen for the following development of new techniques.

Chapter 3

From raw data to biological networks: a contribution to the analysis of dependencies among sparse and noisy continuous data

As described in chapter 1, the objective of systems biology is to model dynamic processes between biological elements such as cells, metabolites or genes. These models are best described in the form of networks [50, 22]. Depending on the diversity of the used biological elements such networks can become very complex. Up to now, there is no universal technique for deriving complex networks from the available sources of biological data (see chapter 1). Thus, most research projects focus on the reconstruction of small networks from very specific data types [3, 77, 129, 128, 45, 142].

In this chapter, two methods will be introduced that focus on the analysis of metabolite concentration data [58, 149]. The first method allows for automatically modelling stable states through interdependencies in the concentrations. The second method can derive dependency networks around pivotal metabolites. Although the focus is on metabolite concentration data, these methods are generally capable of handling different types of data. Thereby, they are a contribution towards a more universal way of network reconstruction.

3.1 Revealing stable states of an organism

The main goal of the examination of metabolite concentrations is to be able to reconstruct the dynamics of interaction between the metabolites. The following method proposes a contribution towards this goal, trying to detect significant thresholds for some concentration variables based on the global analysis of the complete data set. The basic assumption is that, as for any dynamical system, one can observe a finite set of "stable" states between which the system evolves. A state is considered to be a reasonably stable condition of any measurable variable[1], observed directly at the level of concentrations, in a (sub-)set of samples.

It is assumed that a change of such state indicates a reaction to external (environmental) or internal stimulus on the examined organism. A simple example for the impact of an external stimulus is reflected in the distribution of Figure 3.1. In this textbook example [30], the concentration of the metabolite $NADPH_2$ has been measured in the leaves of a plant[2] at daytime and at night-

[1] or a subset of variables respectively

[2] Actually, they were several phenotypes of the same plant grown under identical conditions.

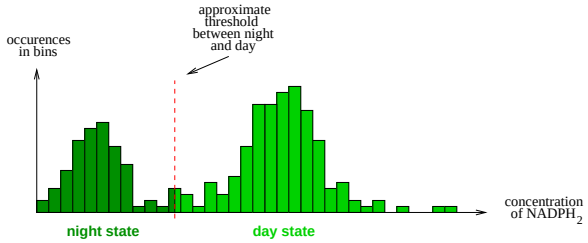

Figure 3.1: The bimodal distribution of $NADPH_2$.

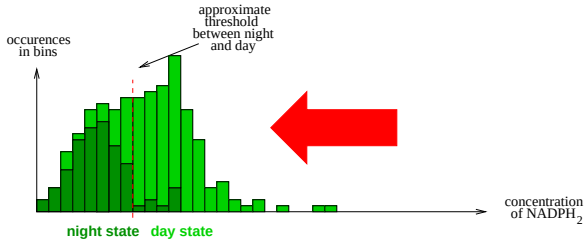

Figure 3.2: A shift of the second mode to the left makes it visually disappear.

time. $NADPH_2$ is known to be involved in photosynthesis. In the distribution of the corresponding concentrations, one observes a subset of samples with an increased level of $NADPH_2$ and a second subset with a decreased level of it. In fact, the samples with the decreased level were the ones measured at nighttime and the others the ones measured at daytime. Thus, the plant can be considered as having two distinct states; we could label them as 'night state' and 'day state'.

There are two modes[3] in the distribution of Figure 3.1 indicating each of the two states. Here, it is known that $NADPH_2$ increases with the amount of light the leaf is exposed to. It is usually not that easy to relate the states of an organism to a variable.

Often the distributions of variables appear to be uniform, Gaussian or indeterminable as in Figure 3.3. Hence, several distinct states (or modes) cannot be read off or found with conventional statistical methods (such as [168]). Nonetheless, there can still be several states which are just hidden in the sum of several modes (see Figure 3.2) or in the noise of the data. After all, despite substantial advances in analytical techniques, biological data has considerable variances.

We address this problem by developing a tool for identifying some of these hidden states in variables. Since functional dependencies (including states) cannot be derived reliably from single variables with few data points we use a global approach to increase robustness. It considers for any given target variable a set of thresholds and compares them in *effectiveness* and *stability* through sets of decision trees. With this approach, it is possible to find robust and explainable states in variables. Once the states are identified, a direct examination can lead to further understanding of

[3]modes = peaks in the distribution; for more details see [53]

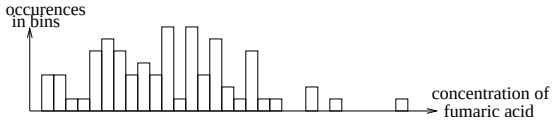

Figure 3.3: There are no clear modes in the distribution of fumaric acid.

the organism's dynamics.

3.1.1 Established Methods considered in this approach

The subsequent work on finding significant thresholds is based upon several well established methods which will be outlined below.

Discretisation

The problem of finding significant thresholds in continuous data is largely equivalent to the problem of discretisation. This field has been intensely investigated in the past (see Subsection 1.2.4). And though, discretisation is often considered just a pre-processing for further examination, it is also accepted as a stand-alone analysis [91].

There is no hard evidence of whether the one or the other category of discretisation techniques is better fitted to discretize metabolite concentration data. That is why the characteristics of data and methods can only be surveyed in a fairly general (and later empirical) manner.

The main difficulty in discretising metabolic data stems from the conjunction of a high amount of noise with a relatively low number of available samples. It is thus of utmost importance to make the most out of the available information and dependency structure in the data. Kohavi et al. [91] and Bay [13] indicate that *multivariate* discretisation is best fitted to satisfy this need. Multivariate techniques consider interdependencies of all variables in the feature space simultaneously. Further, a global approach is preferred because interpretability might still be of interest in the analysis. Note that local methods can produce ambiguous results (as described in Subsection 1.2.4) and are thus hard to interpret. To my knowledge, no such discretisation technique is currently available that could be applied to current forms of metabolic data.

Decision Forests

For the modelling and evaluation of discretisation thresholds, inductive learners taken from machine learning are the prime choice because they can automatically construct models for a given threshold (see Chapter 2). The alternative would be the construction of models through experts which is out of question due to its cost (time and money). There are several types of models which can be learned by inductive learners (see Chapter 2).

Decision tree methods comprise effective induction algorithms and interpretability of the models (see Subsections 2.2.3 and 2.1.3). The data structure of a tree can also easily be handled by subsequent algorithms. However, trees develop an unfavourable property in the biological domain. That is, the induced models are unstable in noisy environments (e.g. metabolite concentration data). This means, a little variation of the data can lead to a substantial difference of the induced tree.

This effect can be counterbalanced by the use of sets of decision trees [181] or by pruning strategies (see also Subsection 2.3.2). Methods of both categories can be applied either separately or together.

Sets of trees are called ensembles (see Subsection 2.3.7) or decision forests. In the following subsections, a strategy is developed that constructs and uses decision forests while preserving the preferable characteristics of trees.

Starting from this background, a new discretisation technique is now introduced which is, in terms of prior work, *global*, *unsupervised*, and *multivariate*, but tries also to make biologically plausible discretisation choices.

3.1.2 Modelling states of an organism

In order to identify possible states of an organism, significantly stable conditions of concentration variables are tried to be detected. These can be identified by the help of decision trees.

First, let us assume we already knew about two states and we could attribute one to each sample. Then, these states could be modelled straightforward by decision trees by means of supervised learning: The possible states are considered as target classes and the metabolite concentrations are used as explaining attributes. Therewith, any decision tree induction algorithm can grow a model for explaining the two states (e.g. C4.5 [136]).

Further, if such a state was expressed in a given variable this variable can be dichotomised into the classes "state 1" and "state 2". Largely, this dichotomisation can be performed by finding the concentration threshold dividing the two states. Literature refers to such a threshold as a *cut point* [56]. With the obtained two classes, again, a decision tree can be induced as a model for explaining these states.

The remaining issue is to find an appropriate threshold for the discretisation of the target variable. In the example of Figure 3.1, the samples can easily be classified into "night state" and "day state" according to their $NADPH_2$ level. The discretisation threshold can visually or statistically [168] be determined between the two modes. However, as mentioned in the first paragraph of Section 3.1, most distributions do not allow for a clear distinction between two modes (respectively states). Thus, we have to find another way to pick an appropriate threshold out of the many possibilities.

3.1.3 Growing decision forests

We propose to grow sets of decision trees for each considered discretisation threshold and compare them. They provide a more stable means of classification than individual trees and thereby grant more reliable results(see Subsections 2.3.7 and 3.2).

To get candidate thresholds the domain of the target variable is divided into intervals. The intervals can be determined by any binning strategy (e.g. uniform binning, equal frequency binning, or exhaustive binning [131]). For example, in Figure 3.4 the simplest form of binning is applied to a variable's domain: uniform binning. As long as a reasonable small size of the data set permits effective computation, exhaustive binning should be preferred over the other strategies because it yields a comprehensive search of the hypothesis space.

For each possible threshold, a decision forest is grown with an embedded decision tree induction algorithm. We used C4.5, one of the most established algorithm for this task [136]. Initially, the set of available variables contains all measured variables minus the target variable. Then, the following procedure is used:

- While variables are present in the data set do
    1. Grow a decision tree with C4.5 on the discretized target variable and add it to the forest.
    2. Remove the variable occurring at the top of the tree from the set of available variables.
- Sort the trees of the forest according to their predictive accuracy and keep the k best trees in the forest ($k = 3$ in our experiments).

That way, we obtain a forest of varying trees with highest predictive accuracy for each target discretisation threshold. This algorithm is loosely inspired by the idea of 'variable deletion' by Breiman [24]. He recommends it for finding variables of equal entropy.

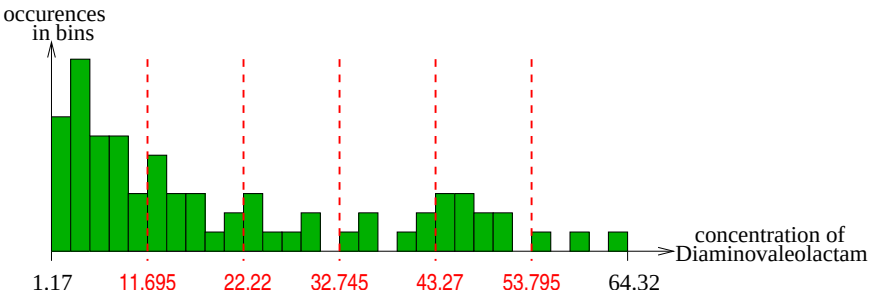

Figure 3.4: Determining candidate thresholds 1...n by uniform binning.

Through the complete procedure, we gain the possibility of using a supervised learning approach in an unsupervised process by systematically using *all* candidate thresholds and constructing models for them.

3.1.4 Threshold extraction

At this point, a particular decision forest has been produced for each of the considered discretisation thresholds.

Definition 26 (Neighbouring forest) *Two decision forests are neighbours if they have been grown on two subsequent candidate thresholds.*

Each forest is evaluated in turn through comparison with the two neighbouring forests as outlined below.

Definition 27 (Similarity of decision trees) *Given an arbitrary precision parameter $n \in \mathbb{N}$ and two decision trees Θ_1 and Θ_2,*

$$sim(\Theta_1, \Theta_2) = \begin{cases} 1 & \textit{if the two sets of ordained scalars used in decision tests of the upper } n \textit{ levels of both trees are identical.} \\ 0 & \textit{else} \end{cases}$$

Or more intuitively, we use a syntactical similarity criterion: Two trees are similar if the attributes used in the first n levels of both trees are the same (n was set to 1 in our experiments).

Definition 28 (Stability of forests) *Let Υ_a, Υ_b, and Υ_c be decision forests (where Υ_a and Υ_c are the neighbours of Υ_b), and let $\Theta_{k,\Upsilon}$ denote the* kth *tree of the forest Υ, then*

$$stability(\Upsilon_b) = \sum_{i=1}^{max} \sum_{j=i}^{max} sim(\Theta_{i,\Upsilon_a}, \Theta_{j,\Upsilon_b}) + \sum_{i=1}^{max} \sum_{j=i}^{max} sim(\Theta_{i,\Upsilon_b}, \Theta j, \Upsilon_c)$$

where max is the maximum number of trees in the forest.

In other words, the stability function compares all trees of the neighbouring forests and grants a score of 1 for each pair of trees that is similar. That way, $stability(\Upsilon)$ gives high scores to forests with similar neighbours.

With this "smoothening" process thresholds are found that promote environments of "stable" models of the data. That is, these models are robust against a slight shift of the discretisation threshold to either direction. If the scores are plotted into a curve we can identify regions of stable

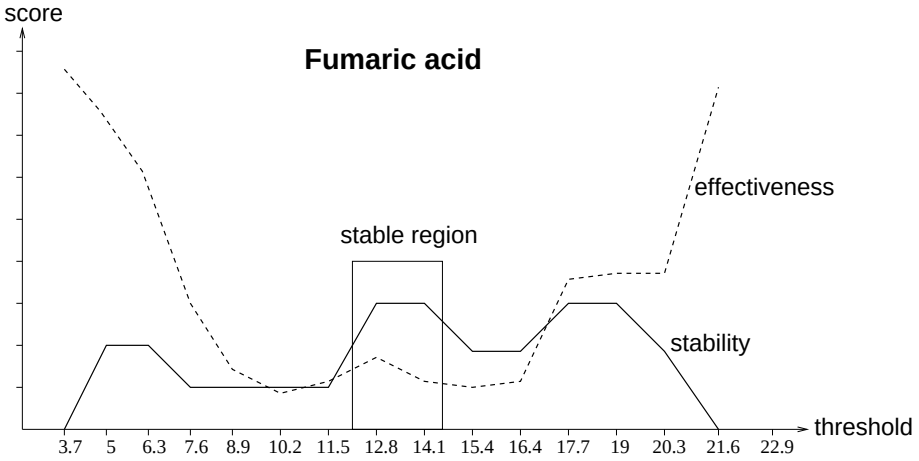

Figure 3.5: Peaks or elevated plains in the score functions indicate regions of stable models.

forests (see Figure 3.5). Stable forests indicate robust models for the explanation of the target variable. We can assume that robust models indicate a biologically feasible choice for the target classes and thus the discretisation threshold.

Another way to compare the forests is by their 'effectiveness'. To measure this quality, we propose the following function which has been inspired by the weighting criterion of Breiman's *cost-complexity pruning* [24] (see also Subsection 2.3.2):

Definition 29 (Effectiveness) *Let T be a binary decision tree of depth n and let D be a set of classified objects. For $1 \leq i \leq n$, let C_i be the set of objects from D, being correctly classified by T at depth i. Then, define the quality of T by means of the following function:*

$$effectiveness(T) := \sum_{i=1}^{n} |C_i| \cdot \frac{1}{2^i}$$

This function delivers high values for trees classifying the training samples with little error and few decisions. For comparing *forests* we use the arithmetic mean of effectivenesses of the trees in the forests and compare them.

As a matter of principle, this function produces peaks for discretisation thresholds close to the boundaries of the target variable's domain. This is due to the very asymmetric distribution of samples in the target classes when discretising is done with a marginal threshold. These peaks are called 'sparse data peaks', because one of the two target classes contains very few samples. These peaks are not considered for the determination of highly effective forests.

With the two quality measures 'stability' and 'effectiveness' it is possible to find discretisation thresholds for any given variable based on peak analysis. If the measures lack remarkable peaks in their values it is assumed that there are no inherent stable states in the examined variable.

3.1.5 Parameters of the threshold extraction technique

The impact of parameters of the threshold extraction technique has been studied empirically[4]. Effects that have occured with the change of parameters are discussed in this subsection in theory. It will be explained which data preprocessing techniques make sense and which do not.

[4]Those studies were performed with the programs given in Appendix A.

Possible binning methods

The method for extracting thresholds depends strongly on the choice of candidate thresholds. These are generated through a binning strategy. The effects of binning strategies on the effectiveness curve and the stability curve will be discussed here.

Uniform binning (or equal range binning) divides the range of given data points into intervals of the same size. The boundary of these intervals are used as the candidate thresholds. The number of bins is the only parameter of this method. This method gives candidates independently of the distribution of the data points. In this way, it can be regarded as producing unbiased candidate thresholds.

If uniform binning is used with a very high number of bins it leads to peculiar curves of the stability measure: Due to the similarity of subsets produced with very close cut points, neighbouring thresholds produce similar decision forests. Thereby, the stability score tends to be often maximal. Only when there is a change in the composition of the forests the score drops. However, rarely the composition changes drastically between two close neighbours. By comparing only close neighbours it is thus no longer possible to see the progression of stability between larger regions. But this is what makes up for the desired information in the stability curve. That is why too many candidate thresholds are unfavourable for the introduced method.

Equal frequency binning divides the range so that each bin contains the same number of data points. The boundary of the intervals (usually the arithmetic mean between the highest value in the lower bin and the lowest value in the higher bin) are used as the candidate thresholds. The number of data points per bin is the only parameter of this method. This method is dependant of the distribution of the data points. Parts of the range with few data points will be shrunk into one bin (or very few). This is unfavourable for the threshold extraction method because, that way, there might be too few candidate thresholds to detect ranges of the data that promote high scoring forests. This is particularly a problem for bimodal distributions with unequally sized modes. In Figure 3.6, equal frequency binning with 7 data points per bin (leading to four bins) is compared against uniform binning with 4 bins. Equal frequency binning also delivers strange results if missing values are replaced with single values (e.g. 0) because this can lead to identical intervals. Because of the above reasons, equal frequency binning with a large number of data points per bin is unfavourable for the threshold extraction method. With a small number, it exhibits the same problem as uniform binning with too many bins.

Exhaustive binning is a special case of equal frequency binning: It is equal frequency binning with one data point per bin. This method exhibits the problems of uniform binning with too many bins. For the given threshold extraction method, it could only be used to produce a very smooth progression of the effectiveness curve. Thus, exhaustive binning is not advisable for the use with our technique.

If sufficient computational power is available, we propose to use uniform binning with an average number of 5 data points per bin as a default.

Binning and normalisation

Common binning strategies each interact differently with normalisation procedures. For the threshold extraction method, is is only important to examine whether data points will fall into a different bin after the normalisation. For instance, if equal frequency binning is used bin memberships will not be influenced by any normalisation method. Uniform binning, on the other hand, is influenced by all normalisation methods. As uniform binning is the method of choice for the threshold extraction technique, data normalisation has an influence on the result. In pratice, we advise to use the threshold extraction technique on a normalised and a non-normalised data set.

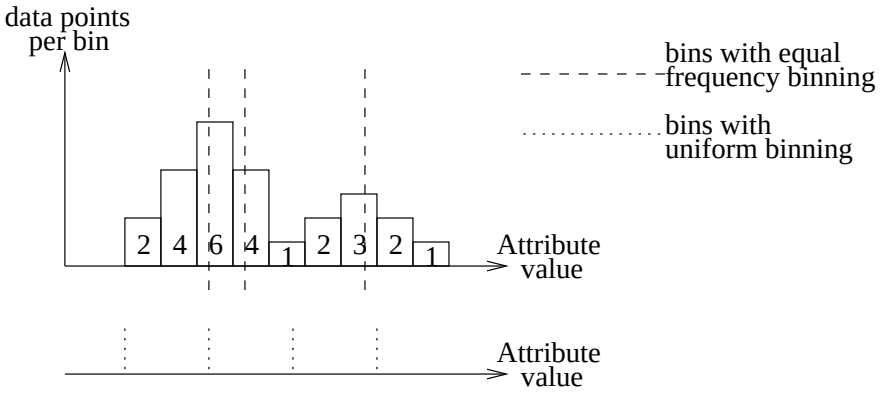

Figure 3.6: An problematic example case: In this multimodal distributions, equal frequency binning delivers candidate thresholds only in the middle of the modes.

Then, approximately the same number of detected thresholds in both runs is an indicator of a reliable result. In any case, the significance and meaning of thresholds has always to be subsequently examined by physiological experimentation.

Impact of missing value strategies

Normally, missing attribute values are not a problem with decision tree methods because they can simply be ignored by the induction criterion. Entropy measures can be calculated on the remaining valid values. However, the threshold extraction technique uses (a dichotomised version of) each attribute also as target attribute. And the target value has to be defined in order to be usable for induction. Samples with a missing value at the target attribute are not usable for an induction algorithm. Hence, there can be a loss of information. But this loss only poses a problem if the missing values mask out all samples that carry certain information which is not reflected in the rest of the data. That is why data preprocessing with missing value strategies is generally not needed for the threshold extraction method.

Strategies that map missing values to a single value over the complete data matrix are largely neutral towards decision tree techniques. There cannot be a split according to an attribute within samples that have the same value for that attribute. Even if the replacement value lies within the regular range of the attribute there will, at worst, appear a few extra splits (to cut out that value) in the decision tree. Replacing missing values with a single value (e.g. 0) poses thus no problem for the threshold extraction technique.

Strategies that replace missing values with various values do bias the introduced method. The impact of those strategies on results depends on the individual data matrix. It cannot be specified generally.

Classification accuracy and comparison of decision forests

There are numerous proposals for improving classification accuracy of decision trees. However, absolute classification accuracy is not an issue for the introduced threshold extraction method. The decision forests are not used for classification. They are only compared relatively in classification accuracy. The effectiveness measure does make use of classification accuracy. But the introduced method only aims at identifying forest that have a higher effectiveness than their neighbours, no matter what the absolute effectiveness is. That is why a simple decision tree learner (C4.5 in this thesis) can be used for this purpose.

Number of trees to be compared

The algorithm for evaluating decision forests offers a choice on the number of trees that are compared from each forest. The range of that number is from 1 to the number of attributes of the data matrix. The comparison is performed through comparing the top nodes of the trees. If there is a similar top node in the neighbouring forest a score is granted.

The more trees there are compared from the forests the more likely there will be similar trees in the neighbouring forest. Thus, the stability measure will rise with the number of compared trees. For the maximum value, the curve will become a line. The effectiveness measure will drop at the same time because only less effective trees are added with a rising number of comparisons. So, the two curves become smoother with a higher number of comparisons but they loose identifiable peaks. A low default value of 3 has proven to deliver feasible results.

3.2 Revealing combinatorial dependencies

Metabolite concentration data is a powerful source of information about metabolic activity in organisms. The interpretation of such data is often done by means of correlation coefficients (see Section 1.3). Such analysis has already led to some understanding of the connection between metabolite concentration levels and metabolic pathways [149]. However, this approach is strictly limited to pairwise and undirected relations. For the generation of more specific hypothesis, the knowledge about dependencies between more than two variables at a time is of great importance. To this end, we extend the correlation approach in this section.

3.2.1 Partial correlation

The basic idea of this approach is as follows: A correlation measure between two variables is systematically calculated under different conditions of a third variable. That way, correlation appearing only under certain conditions of other variables can be observed. This procedure is known from the literature as 'partial correlation' [171]. It calculates the average of Pearson's correlation coefficients [81] for subsamples restricted to certain assigned values.

An illustrative example for the gain of additional knowledge through partial correlation is given in Table 3.1. Here, three variables (number of defective life jackets, number of survivors, and boat size) are given for boat disasters. Then, the correlation coefficients are calculated between all variables. We obtain a strong positive correlation between the number of defective life jackets and the number of survivors. A straight interpretation would lead to the conclusion that fully functional life jackets are bad for surviving in water. Of course, exactly the opposite is correct.

Let the data now be split into three subsets: one with the three samples from large boats, one with the samples from medium sized boats, and one from small boats. If correlation is calculated again for each of the three subsets there is a strong negative correlation between defective life jackets and the number of survivors. This illustrates how straight application of correlation coefficients can lead to awkward interpretations.

The fixing of the third variable facilitates the recognition of the true correlation. This procedure is the above mentioned partial correlation.

3.2.2 Mutual information and conditional mutual information

An alternative measure to Pearsons's correlation coefficient is the 'mutual information' (MI) [166] used in this section. Mutual information considers both linear and non-linear dependencies. Generally, results obtained through Pearson's correlation are also detected with mutual information

defective life jackets	survivors	boat size
9	7	large
8	8	large
7	9	large
6	4	medium
5	5	medium
4	6	medium
3	1	small
2	2	small
1	3	small

Table 3.1: Example data on boat disasters.

[173]. If used only on selected subsets (as described above) literature refers to it as the 'conditional mutual information' (CMI) [40]. CMI can be considered as the information theoretic analogon to partial correlation.

Conditional mutual information has some drawbacks concerning its numerical estimation as will be elaborated later on. To date, scientists searching for combinatorial dependencies rely thus on linear partial correlation [179, 101]. Because of the drawbacks it is not clear if MI really is superior to linear correlation coefficients in the metabolic domain with its persistent lack of data samples. However, the existence of non-linear correlations in metabolic data has been indicated by recent data sets [99]. Given that there will be sufficient data samples in the near future, it is opted for the use and examination of the potentially more expressive mutual information in this thesis.

Numerical estimation of mutual information

As mentioned above, mutual information produces problems when it actually has to be calculated. The MI between two variables A and B is defined as follows [166]:

Definition 30 (Mutual Information)

$$MI(A,B) = \sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{B}|} p(a_i, b_j) \cdot log\left(\frac{p(a_i, b_j)}{p(a_i) \cdot p(b_j)}\right)$$

Here, $p(a_i, b_j)$ specifies the joint probability of variable A and B taking the values $a_i \in \mathcal{A} = \{a_1, ..., a_{|\mathcal{A}|}\}$ and $b_j \in \mathcal{B} = \{b_1, ..., b_{|\mathcal{B}|}\}$, respectively. The marginal probabilities are $p(a_i) = \sum_{j=1}^{|B|} p(a_i, b_j)$ and $p(b_j) = \sum_{i=1}^{|A|} p(a_i, b_j)$. $|\mathcal{A}|$ and $|\mathcal{B}|$ denote the number of assigned values (size of the sample) of the respective variable.

The issue is now the numerical determination of the probability distributions p. There are several strategies for estimating p from finite data [173]. All strategies exhibit problems when the given data set is too small[5]. Although these strategies follow fundamentally different approaches the results on empirical data seem to be rather similar [42]. For this reason and for simplicity, it is opted for the most straightforward scheme in this thesis.

That is, the needed probabilities are simply estimated by their relative frequencies of occurrence (in statistics usually called the 'Laplace probability' [53]). To be specific, the Laplace probability is calculated for intervals a_i and b_j as follows

[5]There are several opinions about what "too small" is [173].

Definition 31 (Laplace Probability)

$$p(a_i, b_j) = \sum_{k=1}^{N} \frac{\Omega_{i,j}(x_k, y_k, a_i, b_j)}{N}$$

where N is the number of samples in the given data set and

$$\Omega_{i,j}(x_k, y_k, a_i, b_j) := \begin{cases} 1 & \textit{if } x_k \in a_i \textit{ and } y_k \in b_j \\ 0 & \textit{otherwise.} \end{cases}$$

Ω is a function indicating if two given samples (x_k and y_k) lie in the range of specified intervals (a_i and b_j) or not. The intervals (also often referred to as 'bins') can be obtained through various binning strategies [131]. In analogy to the effects discussed in Subsection 3.1.5, any strategy other than uniform binning will strongly bias the numerical value of the probability distributions p. In order to avoid undesired bias, all the binning will thus be performed according to the uniform binning strategy in this chapter.

Estimation of conditional mutual information

Conditional mutual information (CMI) introduces a third variable. This variable is called the "conditional" variable, because it establishes a condition under which the MI of the other variables is evaluated. The CMI is defined as [40]

Definition 32 (Conditional Mutual Information)

$$CMI(A, B|C) := \sum_{i,j,k} p(a_i, b_j, c_k) \cdot log \frac{p(a_i, b_j|c_k)}{p(a_i|c_k) \cdot p(b_j|c_k)}$$

Note, that $CMI(A, B|C)$ is the *average* MI between A and B under all possible conditions of C. For a numerical estimation analogously to that of Subsection 3.2.2 significantly more data is needed to ensure a reasonable number of samples per interval. Further, the consideration of all possible conditions of C requires an exponentially high computational demand. To counterbalance this effect, we introduce a new and more restricted form of CMI: the *local* CMI.

Definition 33 (Local Conditional Mutual Information)

$$lCMI(A, B|c_k) := \sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{B}|} p(a_i, b_j|c_k) \cdot log \frac{p(a_i, b_j|c_k)}{p(a_i|c_k) \cdot p(b_j|c_k)}$$

Here, the MI between A and B is calculated for just one specific condition c_k. c_k can be chosen so that a sufficient number of samples satisfies this condition.

There are several strategies for the determination of such a c_k. Again, the binning strategies can be consulted to obtain proposals [131]. In an analogous approach with 'selective linear correlation'[6], A. Tiessen [179] proposed to use a derivate of equal frequency binning. In this, c_k is always chosen so that half the samples satisfy the boolean condition c_k. That way, there are only two conditions (c_k and `not` c_k) to be considered for each MI between two variables A and B.

In order to evaluate more possible conditions (and thereby possibilities of hidden correlations), another approach is used in this thesis: a derivate of uniform binning that excludes all conditions c_k which result in a subsample size of fewer than w instances. The number of considered c_k is chosen as high as possible. It must be adapted to the available computational power. The more power, the more c_k can be considered. w is chosen arbitrarily. As a reasonable minimum to detect real correlations, we propose an average of at least 5 instances per bin. A higher w should be chosen if sufficient data samples were available.

[6] That is lCMI but with mutual information replaced by Pearson's correlation coefficient.

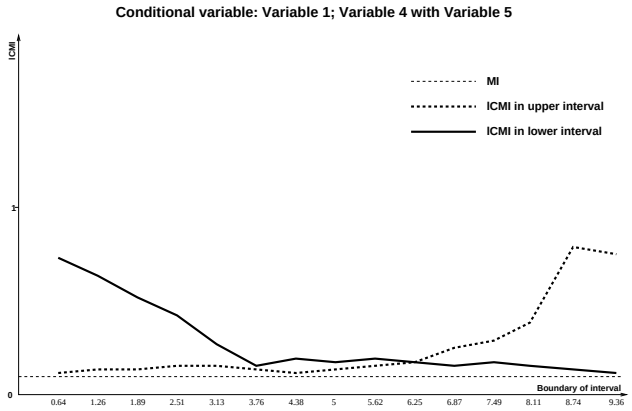

Figure 3.7: Local conditional mutual information on uncorrelated variables in synthetic data. The thin dashed line indicates the MI between the target variables 4 and 5. The thick lines indicate the MI in subsamples of objects with values of variable 1 being higher (lower) than the cut point.

3.2.3 Conditional mutual information on artificial data

To illustrate the effectiveness of lCMI, several synthetic data sets have been created[7]. Each set contains 200 data objects (observations) and between 5 and 10 variables[8]. This number of objects is a size to be expected for metabolite concentration data in the near future. For a demonstration, a typical set is described and analysed in detail. Results on the rest of the data sets are outlined later.

A demonstration on a sample set

In the demonstartion set with 6 variables, the first three variables exhibit a partial correlation. That is, variable 2 and 3 correlate positively if the (sub)sample is restricted to objects where variable 1 takes values of only a certain range. In this case, the two ranges of variable 1 which reveal the positive correlation between variable 2 and 3 are [0..5) and [5..10]. Variable 4 and 5 are linearly correlated but independent of the other variables. Variable 6 is an independent random variable. Variables 1-5 are uniformly distributed in the range of [0..10]. Variable 6 is uniformly distributed in the range of [10..20].

Then, the lCMI is calculated for all possible combinations of variables in the data. For this calculation, the number of evaluated cut points (called c_k in subsection 3.2.2) has been set to 18. This high number is chosen here for demonstration purpose only[9]. The number of bins (as described in subsection 3.2.2) is set to 6, a default value. The binning strategy is again uniform binning.

In Figure 3.7, we see the lCMI between two uncorrelated variables under irrelevant conditions. The overall lCMIs are low. We note that the value of lCMI rises when the size of the subsample shrinks. This is again the same statistical phenomenon referred to in subsection 3.1.4 as the 'sparse data peaks'. They cannot be interpreted as existent correlations.

[7] Such sets can be quickly created with the program given in Appendix A.

[8] The actual number of variables does not affect the progression of the curves because they only use three variables at a time.

[9] The number of cut points adds linearly to the computing time. This is a considerable factor on data sets with many variables where computation can take hours or even days.

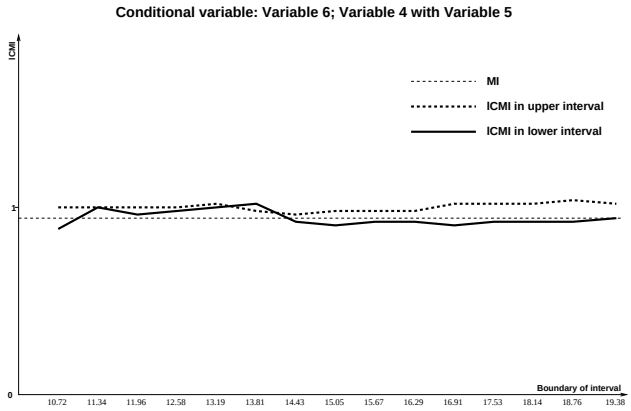

Figure 3.8: Local conditional mutual information on linearly correlated variables 4 and 5 in synthetic data. Typically, the condition of any third variable (here variable 6) does not affect the mutual information.

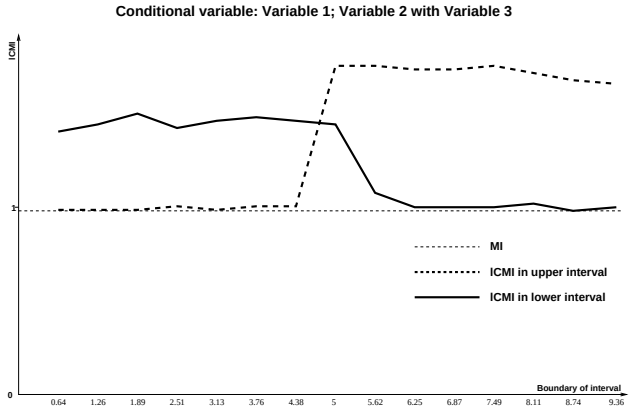

Figure 3.9: Local conditional mutual information on partially correlated variables 2 and 3 in synthetic data. The requirement of variable 1 being higher or lower than a threshold (here 5) causes a abrupt rise or fall of mutual information. On partially correlated variables, the MI curves exhibit this typical sigmoidal progression.

In Figure 3.9, we see the result on the nearly perfect[10] partial correlation. The distinctive feature here is the sigmoidal progression of the two curves. They suggest that there is a very delimited range where the choice of a cut point leads to an abrupt rise or fall in lCMI. This indicates in turn the existence of a higher mutual information (stronger correlation, respectively) in subsets with a well defined condition of a third variable. To detect such a kind of correlation is exactly the intent of partial correlation and conditional mutual information.

In Figure 3.8, we see the curves for two linearly correlated variables. Since the variables are correlated over all observations there is no change in MI if the subsample is restricted to observations with specific conditions.

[10]There is a small bias introduced through the actual random number generation and the limitation of the sample size.

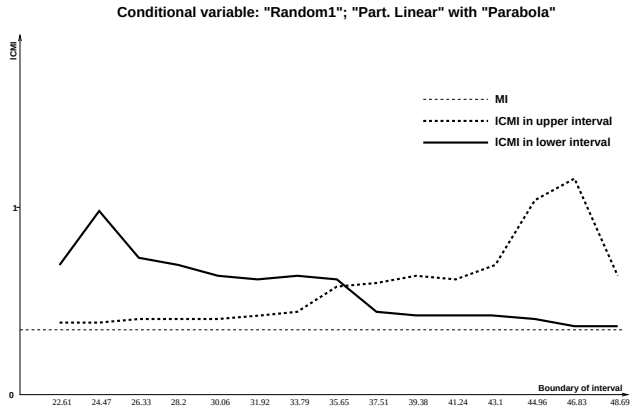

Figure 3.10: lCMI on partially correlated variables with 50% noise.

Results on more data sets

The same analysis has been performed on numerous replications of data sets with partial correlations in differing 3-way-combinations of the variables and with different parameters. The used parameters for the generation of data were:

- the level of noise,
- the distribution of random variables (uniform, Gaussian),
- use of non-random variables (time dependent quadratic or linear functions),
- regular or partial correlations between variables,
- linear or non-linear correlations between variables.

The typical results on these data are outlined below. The tests were performed on more than 20 data sets yielding more than 1000 curves. For brevity, only the most notable curves can be discussed here.

The level of noise showed a low impact on the curves. With increasing noise, the sigmoidal curves tend to flatten at their saddle point. Up to approximately 50% noise, the sigmoidal progression is still visible[11] (see Figure 3.10).

Artificial sets with uniformly distributed variables exhibited clearer results than Gaussian distributed sets if cut points are determined with uniform binning. Gaussian distributed variables have a high density of samples around their peak value. If the cut points are chosen through uniform binning each new cut around the peak value adds many samples to the subset while only few samples are added at the margin of the range (see also Subsection 3.1.5). This leads to more drastic changes in the composition of the evaluated subsets in the center of the range, which leads[12] in turn to more drastic changes in the mutual information. Thus, the curves tend to exhibit sooner a sigmoidal progression than with uniformly distributed variables[13]. This may lead to the identification of “false positives” (detected partial correlations that really are none). The determination of cut points through equal frequency binning resolves this effect.

[11] Note that these results are achieved on an underlying “perfect” partial correlation; they cannot be expected in this clearness on real data.

[12] unless the mutual information is the same in all sample subsets as with regular correlations

[13] On real data, we expect conditioning variables (those that indicate distinct states) not to be Gaussian distributed (see also the discussion at the beginning of Section 3.1). That is why this problem is more of a theoretical nature.

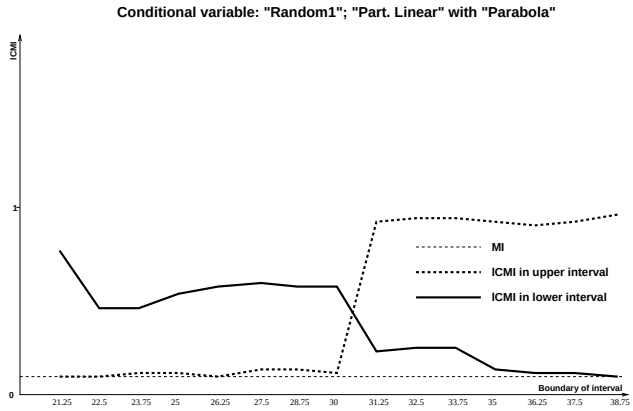

Figure 3.11: lCMI on non-linearly partially correlated variables with 15% noise.

The introduction of time-dependent linear variables had no detectable effect on the curves. Variables generated by a quadratic function led to a similar effect as with Gaussian distributed variables.

In all test data sets, regular correlations could be identified through their curves which were similar to Figure 3.8 (all MIs on the same level). Partial correlations always exhibited sigmoidal curves unless the noise level was beyond 50%.

Most notable was the ability to clearly detect non-linear correlations. Pearson's correlation coefficient is, as a matter of principle, not sensitive to non-linear correlations (see Subsection 1.3.1). However, on non-linear correlations in noiseless test data, the coefficient still indicated a moderate correlation (curves look similar to the "noisy" curve in Figure 3.10). On noisy data, the correlation coefficient fails to indicate any partial correlation. Here, local conditional Mutual information still provides a clear indication of non-linear partial correlations (see Figure 3.11).

We can thus note that the concept of lCMI is able to find high mutual information (correlation, respectively) which is not detectable with simple MI and/or Pearson's correlation coefficient. Such extra findings point to a dependence of the high MI on a specific condition of a third variable. In biological data, this would be an interesting observation as it can indicate interrelations between biological components which come into effect only under certain preconditions. Such results can, for instance, be used to verify hypotheses of *combinatorial* dependencies between components (as indicated by metabolic pathways, for instance).

3.2.4 Dependency network inference

In the past, correlation measures have been used to reconstruct interrelations between variables of metabolic data sets [58, 149]. These interrelations can be displayed in a graph (see Figure 3.13). A feasible way to do this is to chart those relations whose correlation coefficients are larger than an arbitrarily chosen threshold [94]. That way, the attention of an analyst is quickly drawn to the more significant correlations.

In complex biological data, it is likely that some correlations are only exhibited under certain conditions [179]. It is thus reasonable to test for them with an appropriate measure. For biological data, this task has to date been tackled with the measure of partial correlation [171, 101, 179]. A graphical visualisation method for such dependencies is given below.

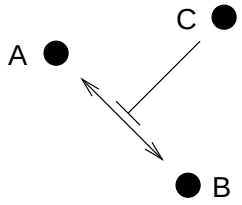

Figure 3.12: A conditional dependency triplet: Attribute A correlates with Attribute B under a condition on Attribute C.

Assembling a network

The information available for visualisation are several triplets of the following kind: One attribute's value depends on another attribute's value if a third attribute has a certain value. This can be graphically represented as shown in Figure 3.12. Each triplet is then represented by a mini-graph. The last step remaining is now to meaningfully integrate the mini-graphs into a single graph.

The method proposed for this is straightforward: All attributes are put as vertices into the graph. Then, triplets are selected that exceed a chosen significance threshold[14]. Finally, the edges of all selected mini-graphs are copied to the graph. The result is a graph that depicts all the significant conditional dependencies in the given data. Such graph can be regarded as a network as introduced in Subsection 1.4.3.

Meaning of the network

A graph depicting simple correlation, as given in Figure 3.13, is a means to get a simple overview of attribute interrelations [94]. All relations are undirected. No information can be given on the cause and the effect of an indicated correlation [167]. Figure 3.14 shows a graph constructed for the same data set but assembled from triplets. It now contains some additional and directed edges. Those are the influences of some variables (which constitute a condition) on the correlation between two other variables. Note that it is still not possible to recognise the cause or the effect of the correlated variables, but it *is* possible to notice the effect of a conditional variable on the correlation between two others. In real data, this direction is usually unambiguous[15]. The new graph is a means to get a quick overview of combinatorial attribute interrelations.

The emerging question is: What do these interrelations mean in biological terms? The answer depends much on the used data. For all biological expression data, the interrelations model some form of regulation. But the method cannot determine whether this regulation is direct or indirect or which physiological process is responsible for it. For metabolite concentration data, we assume that interrelated metabolites are closely connected through a metabolic pathway. In mixed gene expression and metabolite data, we suspect a gene-metabolite interrelation to indicate the production of a specific proteine that in turn stimulates the production of a metabolite [92].

In either case, interrelations between concentration levels point to putative physio-chemical connections. These can be examined more target-oriented if the options are reduced to a small and auspicious set of possibilities.

[14] In the simplest case, this is an arbitrarily chosen correlation value. Alternatively, in Subsection 3.3.1, a method is introduced that automatically yields significant triplets.

[15] When evaluating three attributes, the strength of the impact of a condition onto the correlation between two attributes is quantifiable. In real data, there usually exists a strongest impact among all possibilities.

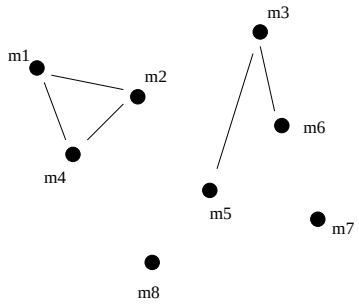

Figure 3.13: Correlations of metabolites m1-m8 visualised in a graph.

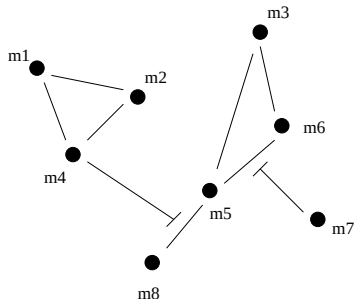

Figure 3.14: Visualisation of simple and partial correlations between metabolites in a graph. Metabolites m4 and m7 form a condition for two partial correlations.

This method is also advertised by Rice et al. In their recent publication "Reconstructing biological networks using conditional correlation analysis" [142], they use partial correlation to assemble complete networks. The key difference of their approach to ours is the subsequently introduced use and estimation of *conditional mutual information* instead of partial correlation.

Features of using conditional mutual information

Partial correlation can find linear correlations between variables. In this section, conditional mutual information has been introduced as a measure to detect also non-linear correlations. Except for the fact that it might indicate a few more correlations (presumably the non-linear ones) it will be applied in the same context as partial correlation. Hence, the graphs can be produced the same way as for partial correlation.

When using CMI, there are several ways to enhance the graphical output of the described process. One of them is the attribution of weights indicating the strengths of correlations to all edges. Additionally, a comparison of partial correlation and conditional mutual information allows for a discrimination between linear and non-linear correlations. These indices have been included in the graph of Figure 3.15. This last graph is an enhanced means to get a quick overview of combinatorial attribute interrelations and their properties.

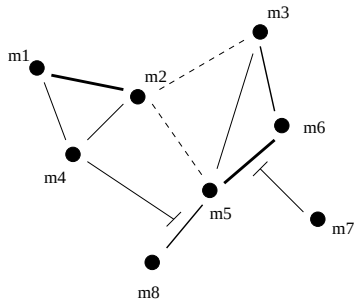

Figure 3.15: Visualisation of conditional mutual informations in a graph. Solid lines indicate linear correlations, dashed lines non-linear correlations. The strengths of the lines indicates the strength of the correlation.

3.3 A heuristic approach: Estimating conditional mutual information through decision forests

However, the use of conditional mutual information entails a few drawbacks. Straightforward calculation of conditional mutual information has a cubical[16] complexity in time [76]. A major problem arises when it has to be calculated for large data sets (e.g. unprocessed microarray data [see Chapter 1]). Thus, a faster way for its estimation would make it better applicable to biological data. A second problem of mutual information is its numerical estimation on small data sets. In the following subsection, a heuristic approach borrowed from decision tree learning is developed to circumvent both problems.

3.3.1 Exploiting decision tree heuristics

The primary purpose of decision trees is to provide a means for classifying data objects into discrete target classes (see Chapter 2). This classification is based upon a set of selected attributes. Each node in a tree represents a test on the value of an attribute, an edge corresponds to a possible value for such an attribute, and a leaf specifies a possible target class.

Decision tree induction algorithms split their training data at each node into subsets of samples with a specific range of values for one attribute. This range of values can also be regarded as a condition satisfied by the corresponding data. Thereby, for subsequent nodes, mutual information is calculated only for samples verifying the specified condition (see also Subsection 2.2.4). That is, again, local conditional mutual information (see Subsubsection 3.2.2).

A key characteristic of decision tree induction is that it computes mutual information for a greedily selected third variable (see Subsection 2.3.5). This heuristics thus saves a lot of computing time. Further, the C4.5 algorithm considers combinations of conditions of more than one conditional variable if classification is not possible with a single condition. In this case, again, computation is cut down due to the greedy heuristics. In the following sections, the heuristics of C4.5 is used to quickly estimate promising relationships of high conditional mutual information.

3.3.2 Making classifiers robust with decision forests

As already seen in Subsection 3.1.1, a known drawback of decision trees is their structural instability against noisy data [24]. In view of enhancing classification accuracy, this can be counterbalanced through the use of decision forests. Present techniques for generating forests tend

[16] See Appendix for details on the complexity.

to converge the trees into some "average" of an optimal classifier (see Subsection 2.3.7). Here, the focus is not on classification accuracy but on interpretability and *diversity* of the trees [7]. Preferably, each tree should present a new hypothesis of a conditional mutual information. This makes sense in the biological domain because often more than one biological cause (e.g. several pathways) has an effect on an examined unit (e.g. a metabolite) [175]. We thus need a new technique adapted for generating forests of this kind. The algorithm for growing forests described in Subsection 3.1.3 meets these requirements.

3.3.3 An illustrative example: Interpreting a decision forest

The algorithm for growing decision forests of Subsection 3.1.3 has been applied to the data used in Subsection 3.2.3. The subsequent illustration is based on the demonstartion set. In this process, all attributes have been used as starting attribute once. To avoid any unintentional bias, each of them was discretized with 5 candidate thresholds (obtained through uniform binning). Thereby, the forest growth algorithm has been applied 5 times for each of the 6 attributes leading to 30 runs.

The trees with best error rates could be generated on the target attributes 'Variable 2' and 'Variable 3' (both discretized with threshold 4.5). The first tree of the forest for the target 'Variable 3' can be seen in Figure 3.16. This example best illustrates the straightforwardness of interpretation for the grown forests.

In Figure 3.16, the tree is only able to classify all objects correctly with the attribute 'Variable 2' if attribute 'Variable 1' takes certain values. That is the conditional dependency which has been inserted into the synthetic data. After the algorithm has taken out the attribute 'Variable 1' the tree inducer (C5.0[17]) does not find any good classifier (see tree in Figure 3.17). This points to the fact that the dependency of tree 1 is truly conditional and that there is not any other dependency in the data. Note that these are exactly the properties inserted into the synthetic data.

To examine the robustness of this approach, the same evaluation has been made on data with noise: With a noise level of 10%, the trees remain the same[18] but exhibit a classification error of $\sim$5% (measured as mean of cross validation [see Subsection 2.3.3]). With 20% noise, the trees obtain additional nodes and exhibit a classification error of $\sim$10% (see an example in Figure 3.18). We observe that the additional node in Tree 3 is a first evidence of overfitting (see Subsection 2.3.1). With 30% noise, the trees become more complicated but still use (predominantly) the attributes of the inserted dependency (an example tree is given in Figure 3.19). The misclassification level here is about 15%. On data with 40% noise, eventually, C5.0 decides to back out on simple one-node-trees with the only attribute 'Variable 1'. This means that any more complicated decision tree would not improve classification accuracy, thus indicating that the conditional dependency can no longer be detected within the noise. Yet, we observe that the misclassification level of these one-node trees still remains below 20% and that the used attribute is one of the inserted dependency.

To this point, it has been demonstrated that the decision tree heuristic is able to find the same dependencies as conditional mutual information. The computational demand, however, is much lower than that of calculating CMI. Additionally, the rigorous discretisation performed by C5.0 led to simple and expected results up to a level of 30% noise in the data. It is therefore a feasible approach for cutting down computational demands when trying to identify conditional dependencies.

[17] C5.0 is the successor of C4.5. It offers several new features, e.g. boosting and misclassification costs, that have not been implemented in C4.5 and a higher computational efficiency [138]. In the presented studies, C5.0 has only been used with the same parameters that have been available in C4.5 already.

[18] except for insignificant deviations in the thresholds

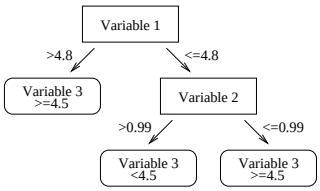

Figure 3.16: First tree of the forest grown on all attributes. Note that Variable 3 specifies the target class.

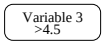

Figure 3.17: No good classifier can be induced on the same set without Variable 1. This one-node tree has an error rate of 43.5%.

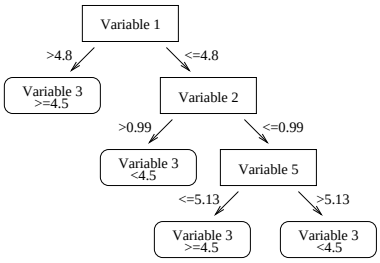

Figure 3.18: The trees still indicates the partial correlation on the complete data but with 20% noise.

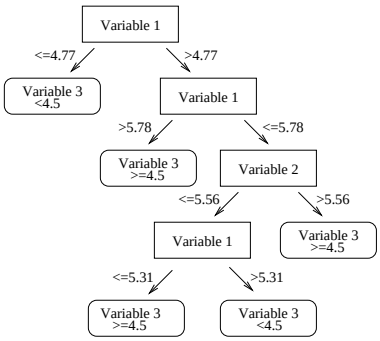

Figure 3.19: Even with 30% noise, the trees predominantly use the variables involved in the partial correlation.

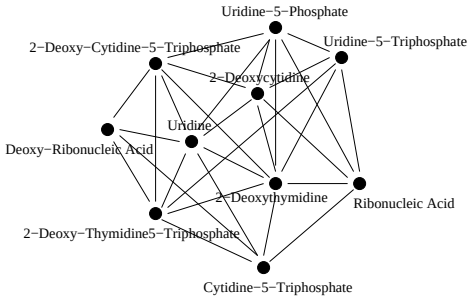

Figure 3.20: Correlations between metabolites involved in the reduction of Pyrimidine-nucleotide. The network appears heaviliy interconnected due to a low visualisation threshold.

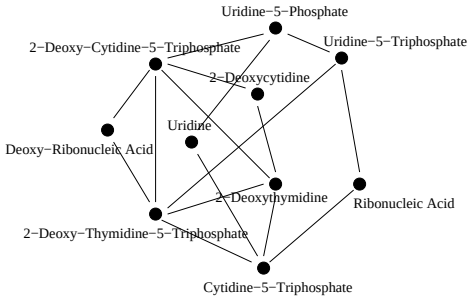

Figure 3.21: The network gets closer to the underlying biological network [109] if only those correlations are indicated whose value exceed a higher threshold.

3.3.4 Characteristics and discussion of the output structure

It has been mentioned in Section 1.4 and Subsections 2.4.2 and 3.2.4 that conditional and unconditional dependencies (e.g. correlation) are often visualised in graphs for further analysis. Graphs obtained by such a combination of individual correlations sometimes become complex and little interpretable (see Figure 3.20). The major factor for this is the number of correlations in the graph. This leads to the question of *which* correlations to include. The most straightforward approach to handle this question is to assign a threshold of a correlation value which selects the correlations to be depicted in the graph [94]. This way, insignificant correlations will be excluded. Such a simplified graph can be seen in Figure 3.21

When calculating partial correlations (or conditional mutual information respectively) the networks can be further reduced to only those correlations which are conditional. Again, the decision on which correlations are considered to be conditional is performed through a threshold.

Choosing significant triplets is similar: The trees in the forest can be ordered according to their accuracy or efficiency (see Definition 29). Each tree can then be transformed into up to two triplets. The most accurate or effective trees of a forest yield the most significant triplets. These are in turn included in the graph. The level of significance that is accepted for the graph is again established through a chosen threshold.

The transformation of trees into triplets is straightforward. As outlined in Subsection 3.3.1, the test of the top node of a tree is the condition under which the tests of the second level lead to a target class[19]. Thus, the top node gives the conditional variable and a subsequent branch the

[19]Trees with more than two levels have to be pruned to two levels. Trees with a leaf on the second level yield only one triplet. One-node-trees do not yield a triplet.

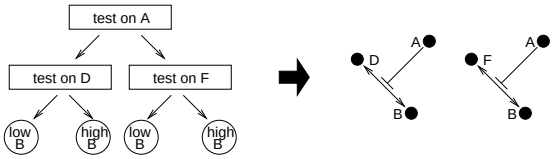

Figure 3.22: A two-level decision tree can be transformed into two triplets.

dependent correlation. Figure 3.22 illustrates this point. Note that the triplets are just an indication for a high conditional mutual information.

In the above approaches, a graph is constructed to visualise the correlations between examined concentration variables (e.g. metabolites in Subsection 3.2.4). This sometimes misleads to the assumption that they can be interpreted in a similar way to pathway charts [109]. However, these graphs depict something fundamentally different. If the data is not gathered as a time series the visualisations cannot represent a cause-and-effect relationship of the concerning variables [167]. And even if it were a time series it would still be hypothetical that visualised directions of interdependencies represent a physiological cause-and-effect (see Subsection 1.6). But despite these limitations it is still valuable for a researcher to know about dependencies between variables. That is because there are too many possible interrelations to be examined in costly physiological experiments. Correlations, and in particular partial correlations, point to a rather manageable number of interrelations to be examined. These will include more potentially interesting relations than a pure random selection of possible relationships [72, 149, 167].

3.4 Summary

In this chapter, the two main contributions of the thesis have been introduced in detail. Both are discussed thoroughly and tested on validation data.

The method for detecting stable states is based on classical decision tree induction. It adopts ideas from cluster analysis (for the comparison of sets) and general learning theory (the adaptation of Occam's razor[115] for the calculation of effectiveness). It serves for the finding of values in individual variables that indicate stable conditions in subsets of the rest of the variables.

The estimation of local conditional mutual information is an extension to a method taken from classical information theory. It serves for finding combinatorial dependencies between variables.

Both methods have proven to work effectively on validation data. They are applied to molecular biological data in the next chapter.

Chapter 4

An experiment in the analysis of metabolite concentration data for potatoes

The methods described in Chapter 3 have been implemented and applied to metabolite concentration data. The first section of this chapter will illustrate the handling of the original software. Meanwhile, more user-friendly implementations of the methods are available. Those are briefly introduced in the appendix. The second section describes the application on yet unpublished metabolite concentration data. The results are discussed in detail, and conclusions about the expressiveness of the method and the data are drawn.

4.1 A new tool: the provided software package

The software tools at hand need a Java^{TM} interpreter of version 1.1 or higher. Higher versions are recommended because the new JIT ('just in time' = compiling at run time) technology increases the performance of the given tools up to 800%. Additionally, a basic PERL©interpreter, a LaTeX compiler and a tool for displaying Postscript^{TM} (e.g. GhostView) are needed. Depending on the original format of the data, a spreadsheet tool (e.g. StarOffice^{TM} or EXCEL©) or a good text editor (e.g. EMACS) are helpful. All interactions between the user and the supplied applications are carried out in command line mode.

4.1.1 Implementation details on the state identifier

In this subsection, the state identifier will be briefly described from a technical point of view. The state identifier consists of 4 JAVA classes:

- 'schwellsucher.class' is the main class of the application. It interprets the command line parameters and reads the input data stream (STDIN) and an additional name file 'namen.buf'. It then uses the other three classes to process the input. Output is written to the output data stream (STDOUT), messages and errors are reported to the error data stream (STDERR).
- 'Entscheidungsbaum.class' implements the decision tree induction algorithm C4.5. The class supplies the object 'Entscheidungsbaum' on which the method 'c45' can be called. The class 'Knoten.class' is used by it.
- 'Knoten.class' supplies the object 'knoten' of which decision trees are made of. The class also implements the method 'bewerte' for calculating the effectiveness (see Subsection

3.1.4) of the (sub-)tree affixed to an object 'knoten'. Several trivial methods for setting and reading the attributes of a 'knoten' object are also supplied.

- 'Datenmatrix.class' implements all stream input used by the application. It supplies the object 'datenmatrix' and the methods 'getNames' and 'getFromStream' for importing data from a name file and the input stream (STDIN) and storing it into a 'datamatrix' object.

For automatically evaluating all attributes of a data matrix an additional main class is supplied that differs from 'schwellsucher.class' only in the type of output. This class is called 'ziehdurchalle.class'. It generates LaTeX output which can be used to graphically display the progression of effectiveness and stability of thresholds in attributes. The PERL-Script 'ziehdurchalle.pl' is provided to correctly access this class.

4.1.2 Implementation details on tools for the calculation of MI

The tools for calculating mutual information and/or local conditional mutual information are technically described in this subsection. These tools both make use of the JAVA class 'Datenmatrix.class' that has been introduced in the previous subsection. These are the additional classes:

- 'mutual.class' is a main class. It interprets command line parameters, reads the input data stream (STDIN) and an additional name file 'namen.buf'. It then computes the mutual information between two attributes of the input data matrix and writes the result to the output data stream (STDOUT).

- 'coregulation.class' is a main class. It interprets command line parameters, reads the input data stream (STDIN) and an additional name file 'namen.buf'. It then computes the local conditional mutual information between a given target attribute and all combinations of two other attributes. If demanded, a sorting algorithm will sort the results according to gain of local conditional mutual information against mutual information. The final result will be written to the output data stream (STDOUT).

A further class 'filtered.class' is supplied that behaves similar to 'coregulation.class' but additionally filters out insignificant cut points for attributes. Insignificant cut points are those that split the data into two shares of which one contains less than 5% of samples from the number of all samples.

For the graphical indication of conditional mutual information, another tool is supplied. The tool is implemented in the class 'production.class'. It again uses 'Datenmatrix.class' for reading input from the input data stream (STDIN). It reads command line parameters and the name file 'namen.buf'. LaTeX output is written to the output data stream (STDOUT).

4.1.3 Implementation details on the dependency inducer

In this subsection, the dependency inducer will be briefly described from a technical point of view. The dependency inducer consists of 4 JAVA classes:

- 'baumsucher.class' is the main class of this application. It interprets command line parameters, reads the input data stream (STDIN) and an additional name file 'namen.buf'. It then induces a decision forest based on the variable deletion method. The forest is in turn sorted according to its classification accuracy. Output of the forest is written to the data output stream (STDOUT).

- 'Entscheidungsbaum.class' (see Subsection 4.1.1)

- 'Datenmatrix.class' (see Subsection 4.1.1)
- 'Knoten.class' (see Subsection 4.1.1)

4.1.4 User scenario

The supplied software tools can be used together or separately on a given data matrix. A combined use would start with the application of the state identifier. It will search for thresholds than can be used as cut points for determining the target classes of the dependency inducer.

Preparing the data matrix

The most time-consuming step in the application of the given tools is usually the preparation of the data matrix. In this description, it is assumed that the data matrix is available in the spreadsheet data representation (see Section 1.2). Then an editor or a spreadsheet tool is used to cut out the names of the attributes (that is usually the first line of the data matrix). These names have to be put into a text file with the name 'namen.buf'. The contents must look similar to this:

```
'name1'
"name2"
name3
```

All symbols of the supplied character set on the available computer can be used. Each name is separated from the next name by a carriage return. This format is a standard output format of most spreadsheet tools. If the names are in a consecutive format (separated by white spaces or TAB-stops) they will be interpreted as a single name. Thus, they have to be 'rotated' in the editor or spreadsheet tool first. Empty lines will be interpreted as empty names. If the file 'namen.buf' is corrupt, too short or completely missing the supplied tools will substitute the unknown names with generic numbers.

In a second step, the values of the matrix have to be saved into a separate text file. This file must look similar to this:

```
0.4     0.346   3.12
5.789   4,32    -0.12
3.5e+02 0       6
```

All display formats (including scientific numbers) can be parsed. Commas and points will be equally interpreted as decimal separators. The numbers have to be separated by TAB-stops. The lines (representing the samples) have to be separated by carriage returns. This is the default text output format of most spreadsheet tools. Note that no textual annotations (including names of samples) must remain in this file. If the data matrix is not completely consistent the tools will give a warning and replace improper entries with 0s.

Applying the state identifier

For the application of the state identifier, the user best changes in command line mode to the directory where the name file and the data matrix file are. For the following explanations, it is assumed that the data matrix is in a file with the name 'datamatrix.dat' and the names are in a file 'namen.buf'. Then, the following line will start the examination of an attribute:

```
java schwellsucher v w x y z <datamatrix.dat
```

The five letters are the parameters for the tool.

- v is the number of the attribute to be examined. The first attribute is attribute 0.
- w is the number of generated threshold candidates.
- x is the number of trees that will be consulted to calculate the quality measures.
- y is the maximum number of trees that will be generated for each decision forest.
- z is the maximum depth of decision trees.

All parameters will be replaced by feasible default values if not specified on the command line. In most cases, the target attribute is the only parameter to be given by the user. The last two parameters (y and z) should not be changed unless the user possesses in-depth knowledge of the implementation.

After starting the tool, the output will be written to the screen (unless redirected to a different output medium). For each candidate threshold, the numerical values of the quality measures are now given.

For a graphical and more intuitive output, a PERL script is supplied. In it, the number of attributes of the data matrix and the file for the data matrix have to be specified by the user (see code of that script). The script can then be used to generate a LaTeX file that can in turn be compiled into a graphical output. An analysis run could be conducted like this:

```
ziehdurchalle.pl >output.tex
latex output
xdvi output
```

Depending on the size of the data matrix and the used computer, a complete run will, in most cases, last from seconds to several minutes. The resulting graphical output shows the curves of the quality measures for all attributes. They can then be interpreted by the user.

Directly calculating MI and lCMI

A direct calculation of mutual information and local conditional mutual information is possible with the supplied tool 'coregulation'. For the following example, the file of the data matrix 'data-matrix.dat' and the corresponding name file 'namen.buf' are assumed to be in the current directory. The following line will start the calculation of MI and lCMI:

```
java coregulation x y <datamatrix.dat
```

The two letters are the parameters of the tool.

- x toggles a sorting of the result [0=no sorting, 1=sorting].
- y specifies the target attribute. If none is specified all possible MI and lCMI are calculated.

The output is given as plain ASCII text. An output line is structured like this:

```
condition - no. of objects with condition - attribute 1 - attribute 2 - CMI -
```

An output line could look similar to the following example:

```
LDL<=78.8  27  R_STAT_N  ADI_NEU  0.2612  0.01172  0.2494
```

This line has to be interpreted in the following manner: For objects where the attribute value of 'LDL' is lower or equal than 78.8 the mutual information between the attribute 'R_STAT_N' and 'ADI_NEU' is 0.2612. The mutual information between the same 'R_STAT_N' and 'ADI_NEU' on *all* objects is 0.01172, so the gain of lCMI against regular MI is 0.2494. The number of objects that satisfy the given condition is 27.

If a sorting is desired the output lines will be sorted according to the gain of lCMI against MI. Thus, the first output line would indicate the dependency with the highest mutual information that can only be found for objects that satisfy a given condition.

At the time, the number of investigated conditions for the conditional attribute is fixed to 5. Higher numbers have resulted in premature termination of the tool on the available computers. The computational demand for the calculation of MI is so big that, even when considering only 5 conditions, available biological data sets could not be processed exhaustively.

Graphically displaying MI and CMI

A graphical display of the calculated mutual information and local conditional mutual information can be obtained with the tool 'production'. For the following example, it is assumed that the file of the data matrix 'datamatrix.dat' and the corresponding name file 'namen.buf' are in the current directory. A graphical output will be obtained with the following commands:

```
java production x y <datamatrix.dat >plots.tex
latex plots
xdvi plots
```

The two letters x and y are the parameters of the tool.

- x is the number of cut points used for the determination of conditions.
- y is the target attribute.

The graphical output is mostly self-explaining. The dotted curves (one made of little circles and one made of little squares) indicate the progression of lCMI for objects that have a conditional attribute value greater (or lower) than the cut point. The solid line indicates the regular MI.

As opposed to the previous tool 'coregulation', the number of evaluated conditions can now be specified with the parameter x. But again, depending on the available machines and biological data, numbers above 5 often result in crashes due to lack of memory.

Applying the dependency inducer

For the heuristic determination of high conditional mutual information, the tool 'baumsucher' is supplied. The following example requires the file of the data matrix 'datamatrix.dat' and the respective name file 'namen.buf' to be in the current directory. The subsequent command starts the calculation:

```
java baumsucher w x y z <zwischen.dat
```

The letters are the parameters of the tool.

- w specifies the target attribute.
- x indicates the cut point for the binarisation of the target attribute.
- y gives the maximum tree depth.

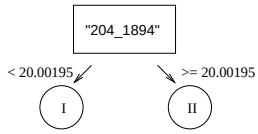

Figure 4.1: The graphical representation of the induced decision tree.

- z gives the maximum error accepted for the establishment of a leaf in a decision tree.

The cut point for the binarisation has to be chosen arbitrarily by the experimenter. A good choice for such a point is a significant threshold, either to be found visually or with the previously introduced tool for finding thresholds. The last two parameters y and z can be disregarded. They have only a minor impact on the result; the default values were sufficient in all test cases.

The output of the tool are decision trees in a plain text description. A simple example tree is given here:

```
Tree with depth 2 and 0 errors and a value of 1.0
*** LEVEL 1 ***
100 Objects at node with attribute "204_1894 " and threshold 20.00195
--> Branch SMALLER 20.00195 from level 1
    *** LEVEL 2 ***
    50 Objects in leaf with class 1
*** back to level 1 ***
--> Branch GREATEROREQUAL 20.00195 from level 1
    *** LEVEL 2 ***
    50 Objects in leaf with class 0
*** back to level 1 ***
```

The graphical representation of this tree can be seen in Figure 4.1. The trees are sorted according to their accuracy. If the accuracy is the same for two trees then the efficiency is a secondary sorting criterion. Through this sorting, the trees with most promising conditional mutual information are placed on top of the output list.

4.2 Application of the introduced techniques on metabolite concentration data

The software introduced in Section 4.1 has been used for the analysis of metabolite concentration data. The data has been produced from potato leaves with the GC/MS procedure introduced in Subsection 1.1.1 and explained in more detail in the following subsection. The tools facilitated the retrieval of hidden states and a partial reconstruction of a dependency network.

The recovered states were mostly consistent with the assumed structure of the data. But beyond that, a few very clear states revealed an until then unknown behaviour of some metabolites.

The dependency structure was again in major parts compliant with the expected results. However, a larger portion of the output could also be identified as artifacts of noise.

The obtained results will be discussed in detail in the subsequent subsections.

4.2.1 Metabolite concentration data of transgenic potato

In this subsection, a detailed description is given of how the used data has been generated. This description contains several specifications which are needed to reproduce the complete chain of

the experiment. It is not necessarily essential for the understanding of the introduced data analysis techniques. For that purpose, it would be sufficient to read Subsection 1.1.1 and 'the summary of the experimental background' in this subsection.

Biological source material

The seed of the potatoes (*Solanum tuberosum L cv. Desirée*) was obtained at the "Saatzucht Lange AG" (Bad Schwartau, Germany). Plants were maintained in tissue culture with a 16 hour light and 8 hour dark regime on a specified medium (MS-medium[1] defined by Murashige and Skoog [118]) with 2% sucrose. All plants were grown in the greenhouse under the same regime with a minimum of 250μmol photons m^{-2} s^{-1} at 22°C. All samples were taken under these conditions. The plants were divided into two groups and transfered from tissue culture to hydroponics or quartz sand cultures, respectively, 6 weeks after cutting.

Quartz sand cultures were used for optical evaluation of root development. Here, growth conditions and fertiliser treatment were done as described in Leggewie et al. [100]. Essentially, plants were grown in sealed pots filled with quartz sand and supplied with 0.5x Hoagland's medium (as defined in Röhm and Werner [148]). Conditions with a deprivation of Phosphate (-P conditions) were generated by flushing the pots with demineralised water. Subsequently, in this water, phosphate was replaced by KCI at the same molarity. The solution was then returned to the pots.

Plants in hydroponic cultures were grown in 1x Hoagland's medium per 2l. The medium was in light and tight plastic pots that were aerated. The solution was exchanged weekly. The plant shoot was fixed in small openings in the lid of the pots using foam. Growth was allowed for 3 weeks in 1x Hoagland's medium. Afterwards, the medium was changed according to the needs of various experiments. Root samples were harvested, blotted dry, frozen in liquid nitrogen and kept in a -70°C freezer for no longer than a month.

Tissue homogenisation

Only samples from hydroponic cultures were used for metabolite concentration measurements. Between 25g and 50g of tissue were ground in a mortar under liquid nitrogen into a fine powder and collected in 2 ml Eppendorf tubes. Extraction of metabolites was done according to Rößner et al. [149] with the following modifications:

- 420 l uptake buffer was mixed with the base material.
- The buffer was composed of
  - 330 μl methanol solution (100%) pre-cooled to -20°C,
  - 30 μl Ribotol (Sigma 0.2mg/ml stock in methanol [100%]), and
  - 30 μl d4 Alanin (Sigma: 1mg/ml stock in water).
- After incubation for 15 mins at 70°C under constant shaking, the samples were extracted with 200l chloroform followed by an incubation at 37°C for 5 mins.
- The extracted samples were mixed with 400l H_2O and recovered from the soluble phase by centrifugation.
- Finally, the tissue was dried in a centrivac without heating the sample.

[1] Basal salt mixture containing micro and macro elements with vitamins

Derivation was achieved by modifying the protocol of Rößner et al. [149] in the following manner: The dried sample was incubated for 90 mins at 30°C in 40ml of methoxyaminhydrochloride (20mg/ml in pyridine). Then, the solution was replaced by 70ml of MSTFA and 10ml of the following alkanemix: 3.7% (w) heptanoic acid, 3.7% (w) non-noic acid, 3.7% (w) undecanoic acid, 3.7% (w) tridecanoic acid, 3.7% (w) pentadecanoic acid, 7.4% (w) non-adecanoic acid, 7.4% (w) tricosanoic acid, 22.2% (w) heptacosanoic acid, and 44.5% (w) hentriacontanoic acid. All ingredients were redissolved in TFA at 10mg/ml total concentration. The subsequent GC/MS analysis was performed according to the procedure given by Rößner et al. [149].

Chromatogram evaluation

The evaluation of the chromatogram was done using the following three methods:

- Manual evaluation as described in Rößner et al. [149]
- The Automated Mass Spectral Deconvolution and Identification System [127] with the following settings:
  - resolution = medium
  - shape requirement = low
  - sensitivity = medium
  - adjacent peak = 2
- The MassLab software [39]

Summary of the experimental background

The experiment described above led to the production of 73 tissue samples in which 117 metabolite fragments could be detected. 37 of those samples came from plants that were treated to develop only low concentrations of phosphate. This is often referred to as 'imposing stress' by biologists. The other 36 were left unaffected, leading to regular phosphate levels. Thereby, two distinct states ("presence" and "absence" of phosphate) can be expected to exist in the data. Since phosphate is an important factor in cell metabolism it is clear that concentrations of other metabolites are affected by it. Additionally, it can be assumed that some of those concentration levels reflect the above two states in some manner. Further, it could be presumed that some other metabolite concentrations show an indirect response to the imposed stress. These assumptions are the initial point for the following analyses.

Compiling data matrices from the source data

In this section, three data matrices are used for demonstrating the introduced techniques of this thesis. Here, it will be described how they were compiled.

For each fragment that could be separated by the chromatography, the GCMS procedure generated numbers[2]. These numbers were combined into vectors, each vector containing the numbers of all measured metabolite concentrations of one sample. In doing so, it was made sure that the scalars of all the vectors had the same order regarding the fragments. Thereby, the vectors could be combined into a data matrix afterwards. The obtained data matrix contained 73 samples with concentration values (attributes) of 117 identified fragments. Subsequently, this matrix will be referred to as the 'raw data matrix'.

[2]These also include missing values.

The raw data matrix has later been replenished with 61 more samples, augmenting it to 134 samples. This bigger matrix has been treated with various methods. First, multiple fragments belonging to the same metabolite were identified by a biochemist. Through this, 58 metabolites could be determined that were represented by two fragments. These fragments were examined for their difference in information.

Hierarchical cluster analysis with Pearsons's correlation coefficient revealed that the fragments belonging to the same metabolite were always strongly correlated. Dependency analysis (as introduced in Subsection 3.2.4) showed that, for all fragment pairs, one fragment could be replaced by the other fragment without decreasing classification accuracy. That is why it is assumed that the two fragments of any pair carry the same information. The data matrix was thus reduced to 59 concentrations by deleting the concentration of one fragment from each pair. This data matrix will subsequently be referred to as the 'reduced data matrix'.

The third used data matrix was produced from the reduced data matrix. In it, 18 samples were deleted that exhibited minor experimental irregularities in the GCMS process. By this, it was tried to obtain data with the least possible portion of experimental artifacts. This data matrix will be referred to as the 'revised data matrix'.

Preparing the data matrices for numerical processing

Due to the complex experimental process, the generated data matrices contained some missing values (11.9% of all values). Where needed, these missing values have been replaced according to inputation strategies introduced in Subsection 2.3.4.

First, the deletion strategy has been tested. Statistically, it is the best strategy because it does not introduce any additional bias. Unfortunately, it resulted in too much loss of data leading to very unstable results with the applied analysis techniques. It will therefore not be discussed in the following.

The second strategy was inputation with averages of attributes. This strategy is a standard strategy for coping with missing values if no more specific information is given.

The third strategy was inputation with 0. Statistically, this strategy is rather crude because it maps all missing values independently of their context into one value. When applying decision trees, the mapping to a single value is not necessarily a bad choice. A single value for all missing values means that the induction algorithm cannot place a split point within the missing values. In this way, the values are predominantly neutral for the process. Even if the replacement value is within the regular range of values it can at worst lead to one extra split (for cutting out the replacement value) in a binary tree.

On the other hand, if the replacement value itself turns out to be the criterion for an important split (possibly in the root node) it can be assumed that the cause leading to missing values *is* itself important for the decision problem.

Other strategies (like several inputation strategies with averages of predefined subsets of the samples) have also been tried, but the results have not differed much from the above approaches.

4.2.2 Interpreting the discovered stable states

The method for finding stable states introduced in Section 3.1 has been applied to the given data matrices. If not specified otherwise, the following parameters were used:

- The number of evaluated candidate thresholds per attribute was set to 15.
- The strategy used for generating candidate thresholds was uniform binning.
- The used algorithm for inducing decision trees was C4.5.

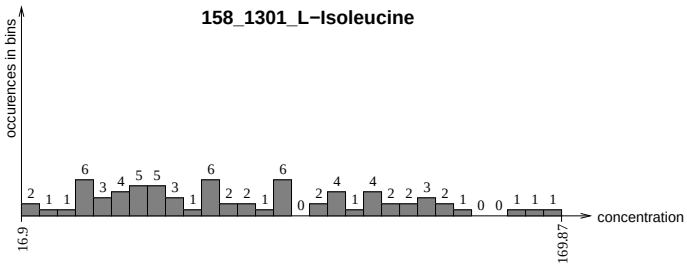

Figure 4.2: A typical unspecific distribution of an attribute in the raw data matrix.

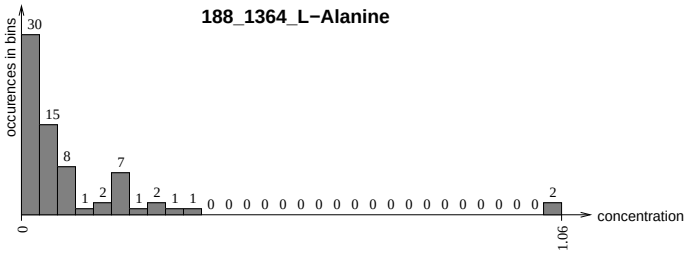

Figure 4.3: One of the many attributes with notable outliers in the raw data matrix.

- For comparing decision forests, the first three trees were taken into account.

These parameters are the default parameters of the implemented method. Variations of them did not lead to significantly different results. Some of those variations will be described later in this subsection. The results on the particular matrices will be explained in detail in the following.

Application on the raw data matrix

The raw data matrix is characterised by attributes whose values do not exhibit common distributions. For example, one such unclear distribution is shown in Figure 4.2. Here, it is virtually not possible to recognise one of the standard distributions (e.g. Gaussian or multi-modal). This is neither possible for most other attributes.

Further, some attributes contain remarkable outliers. An example is given in Figure 4.3. These outliers impede a straightforward and feasible binning of the scale. Thus, it is not possible to quickly recognise a familiar distribution visually. As will be explained below, outliers also impair the finding of thresholds. About 50% of the attributes contain such strong outliers.

Typically, the above characteristics lead to the application of normalisation strategies. This has been performed and described in the next subsubsection. Here, the raw data matrix has not been normalised in order to show empirically the effects on non-normalised data. The impact of normalisation on the technique for finding thresholds has been described in theory in Subsection 3.1.5.

The application of the method for finding thresholds has led to 117 plots, one for each attribute. Each plot contained two curves, one indicating the effectiveness and another one indicating the stability of the thresholds. A typical example plot is displayed for succinic acid in Figure 4.4. There, the two curves indicate bad thresholds in the first third of the range and good thresholds in the other two thirds. The reason for this can be read off succinic acid's distribution in Figure 4.5: Two outliers distort the range of that attribute. This has led to the evaluation of only four thresholds in that part of the range where 97% of the samples are. Four points (thresholds) are not

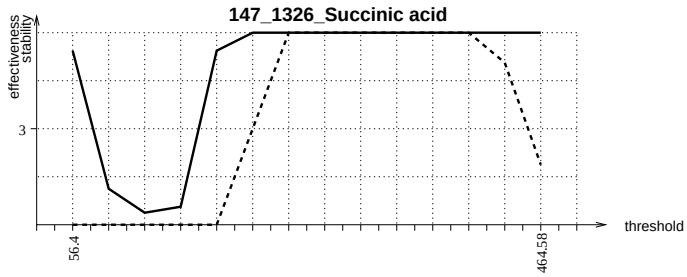

Figure 4.4: The plot for succinic acid on the raw data matrix.

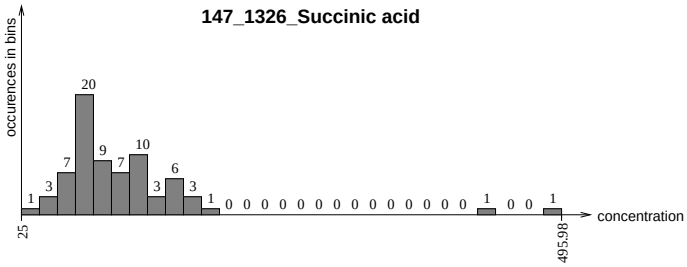

Figure 4.5: The distribution of succinic acid of the raw data matrix.

enough to undertake peak analysis feasibly. Hence, the candidate thresholds have to be chosen differently. As mentioned above, about half of the attributes exhibit this problem.

For obtaining candidate thresholds within that part of the range where the bulk of the samples is, two measures can be used:

- Utilising a different binning strategy (in particular equal frequency binning or exhaustive binning), or
- normalising the attribute to fit better into the range.

Both measures bias the quality and stability measure in a similar way (as explained in Subsection 3.1.5. So, a choice based on technical circumstances was made for normalising the data.

But despite the problems with outliers, it is still possible to notice some very interesting attributes. For instance, diaminovaleolactam develops clear peaks within the regular part of the range (see Figure 4.6). This indication of a good discretisation threshold was not expected. It cannot be found with univariate discretisation techniques.

Application on the reduced data matrix

The reduced data matrix contains only one vector of concentrations for each metabolite, 59 in total. This makes it clearer to determine the behaviour of a specific metabolite. Because of the problems with outliers in the raw data matrix, the reduced data matrix has additionally been normalised according to the z-transformation.

The resulting matrix is characterised by about[3] 45% fairly normally distributed attributes (see example in Figure 4.7). Further, about 10% of the attributes develop clear bimodal distributions. Aminobutyric acid is one example for this (see Figure 4.8). However, roughly 45% of the attributes

[3]The determination of the distributions has been estimated visually.

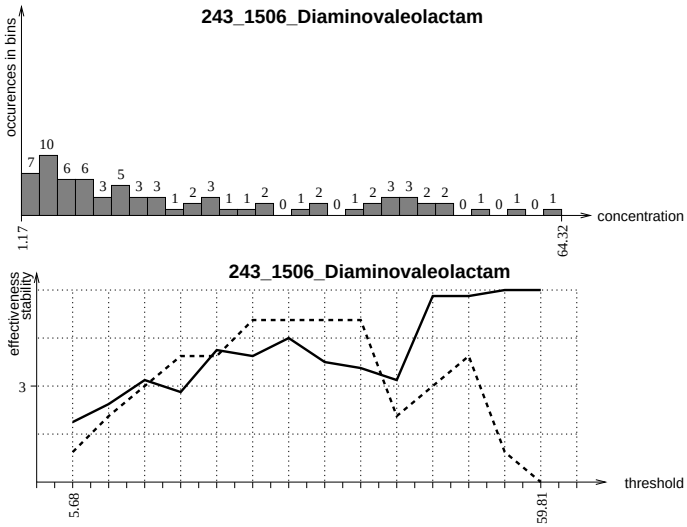

Figure 4.6: Diaminovaleolactam (here in the raw data matrix) develops clear peaks for thresholds between concentration levels 30 and 35.

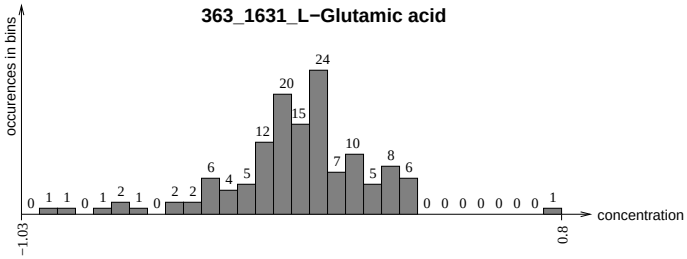

Figure 4.7: Glutamic acid exhibits a nearly normal distribution in the z-transformed reduced data matrix.

do not develop a recognizable distribution. For those, it would be most interesting to see if they yield a significant threshold with the new technique.

For the following calculations, missing values have been replaced with averages of attribute values. The introduced technique for finding thresholds has then been applied to all attributes. The results can be broken down into three categories:

- All attributes that exhibit a bimodal distribution develop peaks in the quality and stability function for thresholds of concentration levels between the modes. This result is intuitive because each mode can be assumed to represent a particular state of the plant. As example, the distribution and scores of phosphoric acid are given in Figure 4.9. The list of the 7 metabolites of this category are:

  1. Phosphoric acid
  2. myo-Inositolphosphate
  3. Aminobutyric acid
  4. Aspartic acid
  5. unidentified fragment 204_3141
  6. Diaminovaleolactam

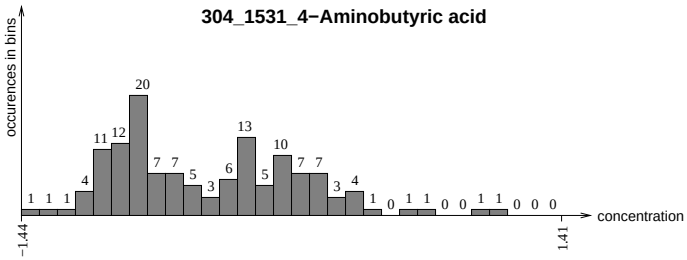

Figure 4.8: Aminobutyric acid shows a bimodal distribution in the z-transformed reduced data matrix.

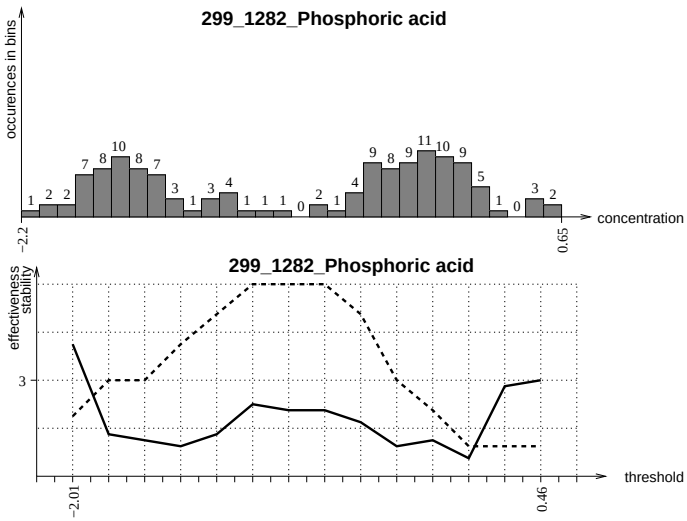

Figure 4.9: Phosphoric acid shows a bimodal distribution and clear peaks in the score functions.

7. unidentified fragment 144_2430

- 23 attributes exhibit a nearly normal distribution. None of them develops significant peaks in the score functions. As example, the distribution and scores of L-Homoserine are given in Figure 4.10.
- The remaining 29 attributes have non-specific distributions. Of those, most do not develop clear peaks in the score functions. These include some where only one of the scores has a peak while the other remains low. But the following metabolites exhibit explicit peaks in both scores at unexpected points:

  1. Citric acid
  2. Tyramine
  3. Glucose-6-phosphate
  4. unidentified fragment 141_1385

  As an example, citric acid is shown in Figure 4.11.

Out of the 59 attributes, 4 could be discovered that exhibit unexpected thresholds. For all clearly distributed attributes, a threshold was clearly found (not found respectively) as it could be expected from the distribution. Thus, we can note that the method is selective.

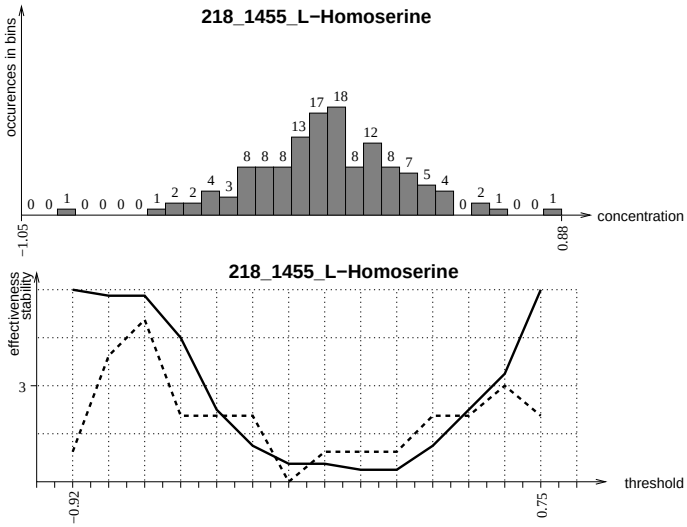

Figure 4.10: Homoserine has an unimodal distribution and no peaks in the score functions.

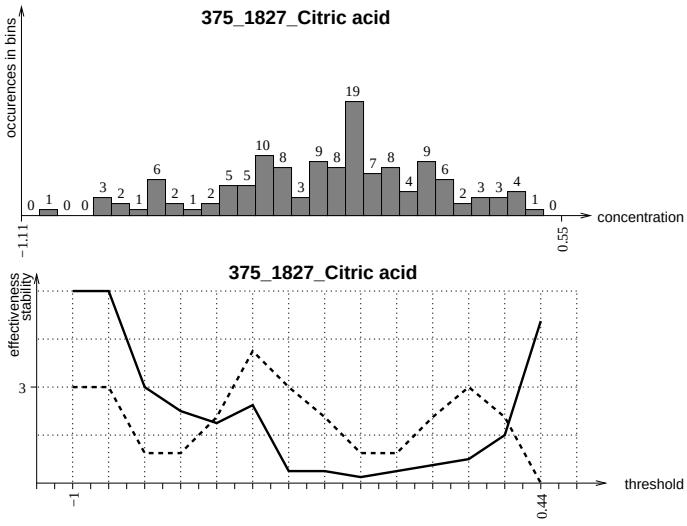

Figure 4.11: Citric acid has an unclear distribution but develops an identifiable peak in both score functions for thresholds at approximately -0.45.

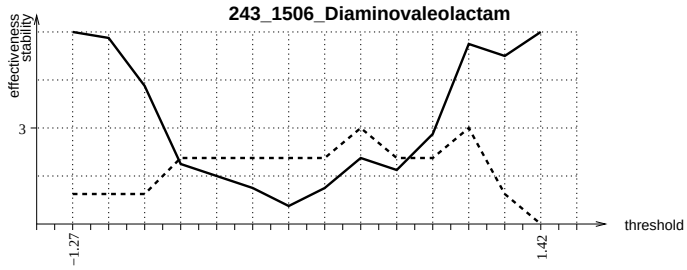

Figure 4.12: Diaminovaleolactam exhibits clearer peaks in the reduced data matrix where missing values have been replaced with 0 instead of attribute averages.

Application on differently preprocessed matrices

For testing the impact of another missing value strategy, the method has been applied to the reduced data matrix but with missing values replaced by 0. Statistically, replacing missing values with the single value 0 is a rigorous difference to replacing with averages. Still, the result was generally very similar to that of the matrix with missing values replaced by attribute averages. Only very few attributes showed noticeable differences. That is why more complicated strategies have not been consulted.

In another run, the revised data matrix has been used. The revised data matrix has the same number of attributes as the reduced data matrix but 18 samples fewer. The attribute values in the remaining samples are the same as in the reduced data matrix.

Application of the method on this data matrix revealed very little differences in the result. Obviously, the deleted samples have not carried any significant bias or the method is robust against that bias.

Tests with variations of the parameters

The introduced method for detecting thresholds has a range of parameters which can be set by the user. Their theoretical impact on results has been discussed in Subsection 3.1.5. In order to get a more complete picture of how the parameters affect results on empirical data, these parameters have been varied.

First, the number of evaluated candidate thresholds was lowered to 5, the absolute minimum for detecting a local peak. Because few data points can only describe simple curves, this resulted in a loss of numerous peaks in the score functions. Thus, we note that too few candidate thresholds worsen the expressiveness of the curves. Though, strong peaks of the preceding applications remained noticeable. For example, diaminovaleolactam (see Figure 4.13) still indicated a clear peak.

Then, exhaustive binning (leading to 133 candidate thresholds on the reduced data matrix) was tested. As a matter of principle, this resulted in very high stability scores. The quality curves became smooth but kept the same trend as in the preceding applications. A result on diaminovaleolactam can be seen in Figure 4.14.

Obviously, the stability curve can no longer be interpreted in the usual manner. It is arguable that, for such a curve, the negative peaks have a significance. Those occur whenever there is a change in the composition of the decision forests. This may indicate the change of a state. At this point, it was not possible to track this feature back to the experimental origin. It is thus not possible to give a satisfactory interpretation. Hence, binning strategies leading to very few or very many samples per bin will not be considered any further.

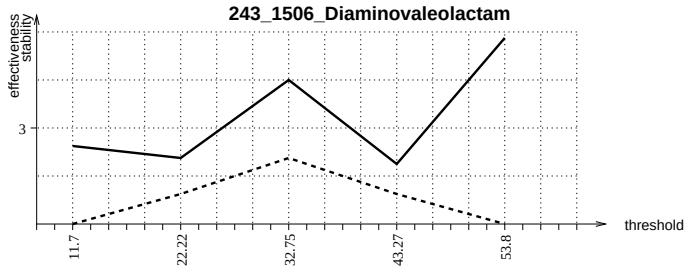

Figure 4.13: Diaminovaleolactam (in the raw data matrix) exhibits a clear peak even with very few candidate thresholds.

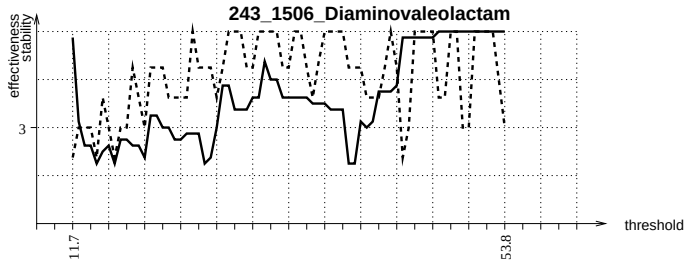

Figure 4.14: Diaminovaleolactam (in the raw data matrix) exhibits very fuzzy curves when exhaustively evaluating all possible candidate thresholds. However, the effectiveness curve still indicates regions of increased values.

Next, the number of trees to be compared against neighbouring forests was examined. Roughly, when rising this number up to half of the given attributes (here about 30 of the 59), the curves seemed to get more and more compressed into a smaller range while maintaining their general progression. Values of more than half the number of attributes have led to a compression so strong that the curves appeared linear. No peaks could be identified any longer. It could be observed that a value between 3 and 10 delivered the most discriminable peaks.

Summarised, the parameters have only a moderate effect on the peaks. If they are left in the default setting of the implementation the curves show usable results.

Evaluation of the results

The introduced method was able to find thresholds in metabolite concentrations where biologists had expected them. In particular, those where thresholds in concentrations of the metabolites related to the phosphate cell metabolism. Those thresholds are predominantly able to split the samples into the treated and untreated share. Further, the method indicated thresholds at points where they can be identified visually in the distributions of concentrations. That is, the distributions exhibit two modes and the thresholds are between them. Some of these thresholds have not been expected by biochemists and could not easily be related to biochemical knowledge. But they can also be detected with established discretisation techniques (e.g. Silverman's test for multimodality [168]). Additionally, the introduced method found thresholds in metabolite concentrations where neither a biochemist expected them nor available discretisation techniques can find a cut point. These findings point to putative new states. All such unexpected states may lead to interesting new insight into a plant's behaviour under the inflicted stress situation.

4.2.3 Interpreting the dependency structure

In this subsection, the method for finding combinatorial dependencies introduced in Section 3.2 has been applied to the data of Subsection 4.2. The output will be described in the following.

Calculating partial mutual information in metabolite concentration data

To understand the dependency structure between the attributes of a given data set, it would be desirable to survey all conditional dependencies. Even when restricting conditional dependencies to 1 condition for each possible pair of attributes there would still be, for n attributes, $\frac{n!}{(n-3)!}$ combinations. For the 59 attributes of the reduced data matrix, this would be 195054 possible conditional dependencies. It is not possible for a regular scientist to visually review and compare this number of curves. Furthermore, to obtain curves in the first place, a sufficient number of values for the conditions has to be considered (at minimum 5 to see a possible sigmoidal progression). For the reduced data matrix, this leads to a minimum of about 1 million calculations of mutual information. Since mutual information is complex to calculate the demanded computational power is extremely high. Because of these two reasons, it has not been possible to survey conditional dependencies on the available hardware with this approach.

Estimating high partial mutual information in the reduced data matrix

Instead of calculating all possible combinations of partial mutual information, the method of Subsection 3.3.1 has been applied. This heuristics reduces the computational demand to a fraction of that of an exhaustive calculation. In the following descriptions, the names of metabolites will be put into double quotes whenever they refer to the concentration level of that metabolite. The names are left without quotes whenever the metabolites as such are meant.

For the estimation of high mutual information between combinations of attributes, it is necessary to identify attributes that obviously express two states. These have to be dichotomised in order to work as target attributes for a tree induction. For the following demonstration, three of the attributes are used that have been detected in Subseection 4.2.2 to express two states. The choice of attributes was motivated by the desire to examine the impact of the treatment to metabolites known to be involved in the phosphate cell metabolism. The chosen attributes are:

- Phosphoric acid
- myo-Inositolphosphate
- Aminobutyric acid

These attributes show the clearest bimodal distributions (see again Figure 4.9 for the distribution of Phosphoric acid). Thereby, it is possible to dichotomise these attributes with the help of thresholds between the modes. We observe that these thresholds also happen to be the best thresholds detected by the state identifier. For the calculations, the maximum depth of decision trees has been set to 3. The induced decision trees indicate the predictability of the chosen target attributes with the remaining attributes.

For Phosphoric acid, the dichotomisation threshold has been set to -0.4, splitting the data matrix into its 67 treated and 67 untreated samples. The three best[4] trees of the forest for Phosphoric acid are shown in Figure 4.15. There, it can be seen that the attributes having most predictive power for Phosphoric acid are “unidentified fragment 133_1769”, “myo-Inositolphosphate” and “Glucose-6-Phosphate”. These carry the highest mutual information with Phosphoric acid. All

[4] regarding classification accuracy

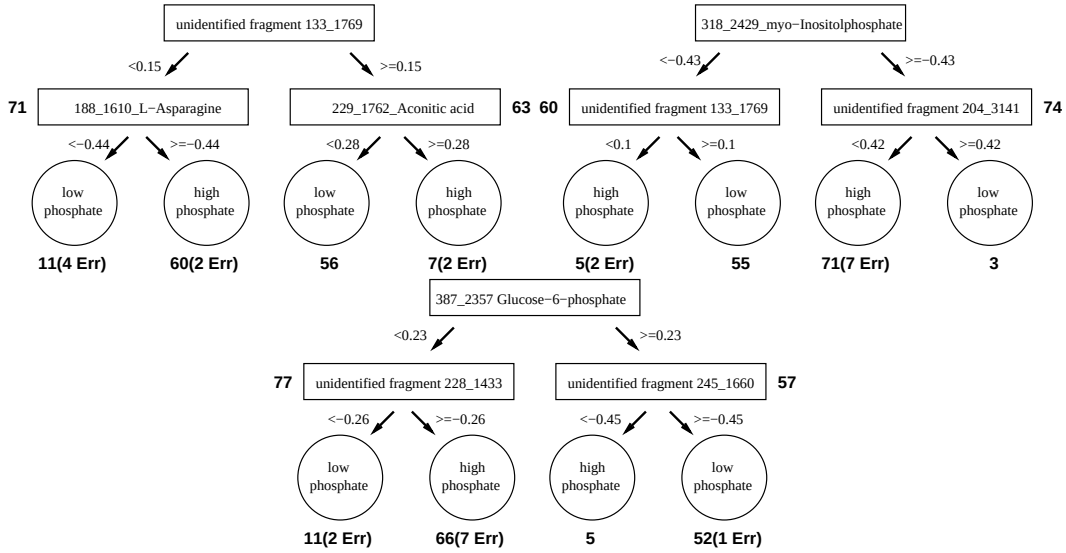

Figure 4.15: Three simple trees that can predict “Phosphoric acid” fairly accurately. The boldfaced numbers indicate the number of evaluated training objects at the respective node.

of them have clear bimodal distributions. Based on these trees, it can be assumed that the modes reflect the same states of the original plant in all mentioned attributes. From a biochemical point of view, this makes sense as all of these phosphates are involved in the same (and artificially perturbed) cell metabolism [175]. The “unidentified fragment 133_1769” is something unexpected. Obviously, it is closely related to the phosphate cell metabolism. This evidence can help to identify the corresponding metabolite.

Further, the trees indicate attributes that help to predict the target attribute but only under a given condition. In the first tree, these are “L-Asparagine” and “Aconitic acid”. These carry *conditional* mutual information with Phosphoric acid. The condition is a certain level of “unidentified fragment 133_1769”. This result suggest a link between Phosphoric acid and two metabolites that are not directly involved in the phosphate cell metabolism. A physiological examination of this coherence is yet planned to be undertaken by biochemists [98].

In the second and third tree, there are again attributes indicated to carry conditional mutual information with Phosphoric acid. Here, it has to be pointed to the low significance of those attributes. Three of them are responsible for only 5 or fewer samples to be predicted correctly. It is highly probable that these interrelations are mere artifacts of noise in the data. However, it is a delicate process to determine a clear significance threshold. At this point, it was decided to accept only attributes responsible for the correct classification of at least 10% of the samples. This leaves “unidentified fragment 228_1433” under a condition of “Glucose-6-phosphate” to be the only valid conditional mutual information of the last two trees.

More combinations carrying conditional mutual information with Phosphoric acid can be found in the rest of the decision forest. Valid ones are:

- “Mannose” under a condition of “unidentified fragment 392_1777”,
- “Mannose” under a condition of “L-Isoleucine”,
- “Mannose” under a condition of “L-Leucine”,
- “Glycerol-1-phosphate” under a condition of “unidentified fragment 292_1810”,

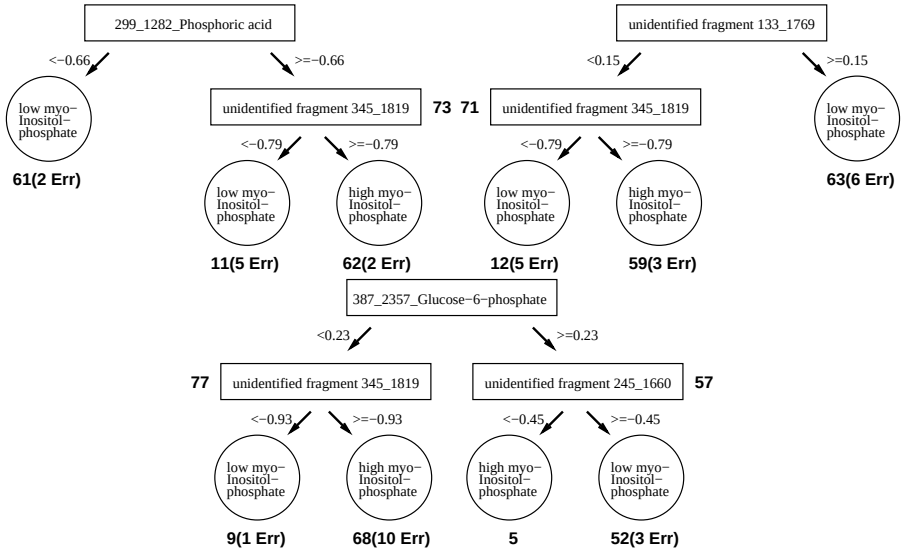

Figure 4.16: “Myo-Inositolphosphate” is clearly predictable with the same attributes as Phosphoric acid.

- “Glycerol-1-phosphate” under a condition of “unidentified fragment 144_2430”,
- “Glycerol-1-phosphate” under a condition of “unidentified fragment 284_2691”,
- “Glycerol-1-phosphate” under a condition of “unidentified fragment 141_1385”,
- “unidentified fragment 245_1660” under a condition of “Glycerol-1-phosphate”,
- “unidentified fragment 144_2430” under a condition of “L-Asparagnie”,
- “Allantoin” under a condition of “unidentified fragment 204_2844”,
- “Aconitic acid” under a condition of “4-Aminobutyric acid”,
- “Glucose” under a condition of “Aconitic acid”,
- “L-Leucine” under a condition of “unidentified fragment 243_1506”,
- “L-Tyrosine” under a condition of “L-Threonine”, and
- “Aminobutyric acid” under a condition of “L-Tyrosine”.

Two metabolites stand out in the above table. Most notable is the predictiveness of “Glycerol-1-phosphate” only under the conditions of several unidentified metabolites. It suggests a detachable link between “Phosphoric acid” and “Glycerol-1-phosphate”, possibly a metabolic pathway that is only activated by metabolism involving the unidentified metabolites. Furthermore, the predictiveness of “Mannose” under conditions of “Isoleucine”, “Leucine”, or an unidentified fragment hints to the relatedness of the “unidentified fragment 392_1777” to the leucines. These suggestions have yet to be examined by biochemists.

For myo-Inositolphosphate, the dichotomisation threshold has been set to -0.36, splitting the data into its treated and its untreated share. The three best trees for myo-Inositolphosphate are shown in Figure 4.16. As expected from the high mutual information between “myo-

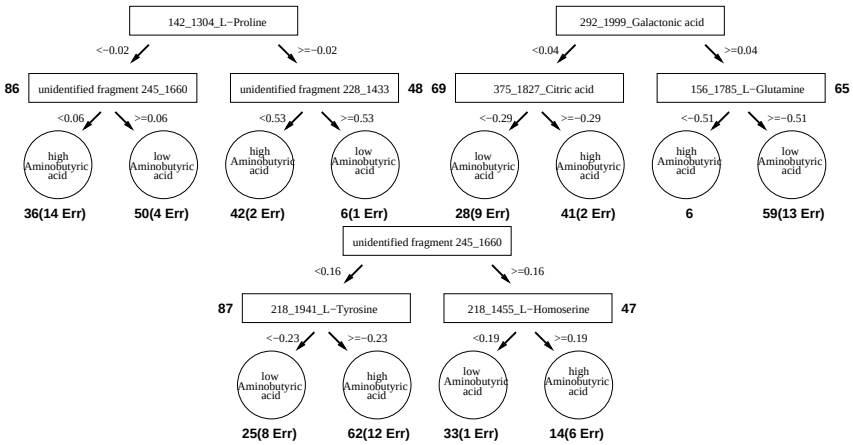

Figure 4.17: “Aminobutyric acid” shows to be less closely related to Phosphoric acid than expected.

Inositolphosphate” and “Phosphoric acid”, the attributes facilitating a prediction of “myo-Inositolphosphate” are very similar to those of “Phosphoric acid”. The reoccurring attribute “unidentified fragment 345_1819” will be exempt from interpretation according to the previously introduced 10% rule. It seems to predict only a small share of samples that contains a certain form of noise.

For Aminobutyric acid, the dichotomisation threshold has been set to -0.55, again splitting the samples into two equal shares. The three best trees for Aminobutyric acid are shown in Figure 4.17. The used dichotomisation threshold was indicated by the method for finding thresholds. It divides the samples into two equal sized shares. Yet, the induced decision forest uses completely different attributes for the prediction than the forests of “Phosphoric acid” and “myo-Inositolphosphate”. On account of the method, this indicates a shift (however slight) of the predicted states. Obviously, “Aminobutyric acid” does not directly reflect the ’absence’ and ’presence’ of Phosphoric acid. Instead, it points to an indirect connection to the stress inflicted on the plants. That connection seems to be associated to “L-Proline”, “Galactonic acid” and “L-Tyrosine”. This constitutes a new factor to be examined more closely in biochemical terms.

Assembling a dependency network

Besides directly interpreting the decision forests this information can also be displayed and analysed in a graph. This is particularly helpful if a quick overview of the conditional dependencies is desired. This has been done for “Phosphoric acid” as displayed in Figure 4.18. The used threshold for the significance was an absolute error number of 10 (that is an accuracy of 0.925). The information included in the graph is the same as it can be directly read off the trees. Only the accuracy is not included in the graph for the matter of a simpler visualisation. The graph is thereby a means for a scientist to quickly see the conditional dependency structure around one (or more) selected attribute(s). If including the dependency structure of several attributes the graph can be regarded as a dependency network. Note that this network is only reflecting high conditional mutual information. It does not directly propose any biochemical reaction[5]. It only proposes putative interesting dependencies to be examined in more detail.

Here, it can be observed that Phosphoric acid interrelates with myo-Inositolphosphate and

[5]Rice et al. also stress this limitation in their recent publication [142].

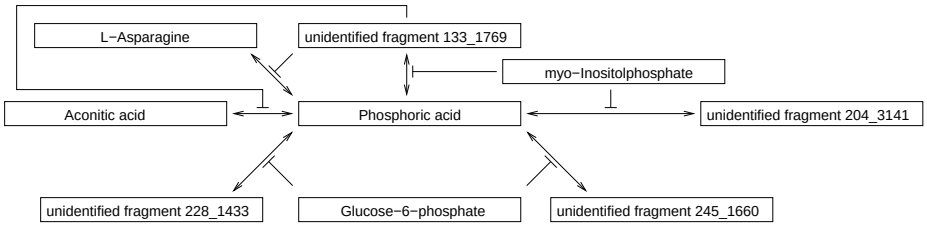

Figure 4.18: The conditional dependencies involving phosphoric acid. Edges with two arrows indicate high conditional mutual information, edges that end in a bar indicate a condition of the originating attribute for the high mutual information.

Glucose-6-phosphate. These interrelations correspond directly with known biochemical processes [175]. More conclusions have yet to be drawn by a life scientist.

4.2.4 Summary of the analysis

The previously introduced methods have been tested on data stemming from a GC/MS experiment. This data has been preprocessed in various ways, including some expert adjustments. Through this, several different data matrices have been obtained. The two methods have then been consecutively applied to these matrices.

The state identifier showed only moderate sensitivity to the used preprocessing. Indicated results remained fairly stable with the different matrices. The results itself disclosed expected and unexpected coherences. Among the unexpected results were the exhibition of unknown states in four metabolite concentrations. These findings point to putative new states to be examined in further experiments.

After the detection of states, some of the output thresholds have been used as input for the dependency inducer. Several interconnections between the examined metabolites and other metabolites were discovered. Most of them had been known from biochemical knowledge and can also be found through simple correlation analysis. But some metabolites clearly exhibit only a conditional impact to the experimental treatment. These results are new and cannot be detected with conventional analysis tools.

Summarised, the two introduced methods revealed properties that have not been known before. Thus, the given tools proof to be an interesting new approach in the analysis of biological expression data.

Concluding remarks

Summary

In this thesis, we have developed new methods for efficient data analysis based on decision tree induction. The methods have been validated on synthetic data and have then been applied to Systems biology data. Results on metabolite concentration data have led to the discovery of known structure and to the gaining of new biological insight.

At first a method has been introduced for detecting significant thresholds. This method is based upon a comparison of decision forests, a new concept in the scientific community. Applied to metabolite concentration data of potatoes, the resulting thresholds disclosed states in the examined organism. Some of these states reflect known properties bound to the physiological experiment, others point to putative new knowledge.

In a second step, conditional mutual information has been introduced for the search of combinatorial dependencies in biological data. The method was discussed and compared against comparative methods. While conditional mutual information has long been known, it has never before been applied to Systems biology data before. This is mostly due to the numerical problems that arise when calculating mutual information on large data sets.

That is why, in a third step, conditional mutual information has been estimated with the help of decision tree heuristics. In doing this, the method for extracting thresholds can be used to supply a required parameter: the cut point. Estimating conditional mutual information has been validated on synthetic data and then been applied to metabolite concentration data. The application resulted in new insight into the interrelations of metabolites in potatoes. Ultimately, a limited dependency network could be reconstructed which allowed for a quick overview of interdependencies around phosphoric acid in the examined organism.

Both the extracting thresholds method and the estimating conditional mutual information method have been implemented in software tools. With them, it is possible to perform the analysis on ordinary desktop computers.

The two approaches are based on new ideas. However, their goal of understanding interrelations between biological components in an organism is not new. It has been pursued by many other scientists with numerous different techniques. A direct and fair comparison of these techniques is not yet possible because of their highly diverse outputs. In the end, it is always up to a human scientist to interpret and evaluate the output of a given tool.

In this context, the methods proposed here can be regarded as yet another way to extract valuable information from given data. It is certainly not a comprehensive approach for reconstructing biological networks. But the methods have provided some favourable results for this purpose that could not have been found with other methods in this field. This makes them a valid choice for the analysis of Systems biology data.

Future work

The threshold extraction method introduces a completely new concept: the establishment of a distance measure between decision forests. To a certain extent this has been examined in this thesis. However, it was not possible to solve all the problems and survey all the parameters involved in-depth. One remaining problem, the sparse data peaks, has been mentioned. This must be solved before the technique can be completely automated.

Other aspects which could lead to clearer results also have to be examined in more detail. Above all, the question of which type of classifier is best for establishing a distance measure has to be sufficiently treated. The main reason decision trees were chosen for this purpose in this thesis is that the results are interpretable and can therefore be easily be followed by a human. It could be that regression models provide a better framework for developing a significant distance measure. In this respect, the thesis has to be seen as just a starting work.

Finally, it has to be noted that the practical value of the introduced techniques depends heavily on the availability of a convenient computer tool. That is why the original tools have recently been furnished with better user interfaces. This work is very important, as a wide circulation of the methods offers the only chance for a discussion amongst life scientists. This discussion is essential for the further development of the techniques. These tools are now publically available on the author's website.

Bibliography

[1] S. Abe. *Pattern Classification*. Springer, London, 1st edition, 2001.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison Wesley Publishing Company, Reading, Massachusetts, 1974.

[3] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for inferring qualitative models of biological networks. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 293–304, 2000.

[4] P. Auer, R. Holte, and W. Maaß. Theory and application of agnostic paclearning with small decision trees. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 21–29, San Francisco, 1995. Morgan Kaufmann.

[5] E. Aurell, J. Elf, and J. Jeppsson, editors. *ICSB 2002 conference proceedings - Preface*, Stockholm, 2002. Karolinska Institutet, Samuelson & Nordhaus.

[6] K. Backhaus, B. Erichson, W. Plinke, and R. Weiber. *Multivariate Analysemethoden*. Springer, Berlin, 9th edition, 2000.

[7] D. Baier and K.-D. Wernecke, editors. *Innovations in Classification, Data Science, and Information Systems, Proceedings of the 27th Annual GfKl Conference 2003*, University of Cottbus, March 12-14 2004. Springer-Verlag.

[8] W. Bains. *Biotechnology from A to Z*. Oxford University Press, Oxford, UK, 2nd edition, 1998.

[9] A. Bairoch and F. Lisacek. Expasy. http://www.expasy.org/, 2004.

[10] P. Baldi and S. Brunak. *Bioinformatics - The Machine Learning Approach*. MIT Press, London, England, 2nd edition, 2001.

[11] A. Barabasi and R. Albert. *Linked, the new science of networks*. Perseus publishing, Camebridge, Massachusetts, 2002.

[12] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1998.

[13] S. Bay. Multivariate discretisation of continuous variables for set mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 315–319, 2000.

[14] N. Beerenwinkel, T. Lengauer, M. Däumer, R. Kaiser, H. Walter, K. Korn, D. Hoffmann, and J. Selbig. Methods for optimizing antiviral combination therapies. *Bioinformatics*, 19(Suppl.1):i16–i25, 2003.

[15] N. Beerenwinkel, B. Schmidt, H. Walter, R. Kaiser, T. Lengauer, D. Hoffmann, K. Korn, and J. Selbig. Geno2pheno: Interpreting genotypic hiv drug resistance tests. *IEEE Intelligent Systems in Biology*, 16(6):35–41, 2001.

[16] C. Berge. *Graphs*. North-Holland mathematical library, Amsterdam, 3rd edition, 1991.

[17] M. Berthold. *Intelligent Data Analysis*. Springer-Verlag, Berlin, 1999.

[18] J. A. Bilmes and K. Kirchhoff. Directed graphical models of classifyer combination: Application to phone recognition. In *Proceedings of the international conference on spoken language processing*, Gareth Moore, October 2000.

[19] C. Birkenmeyer, A. Lüdemann, C. Wagner, A. Erban, and J. Kopka. Metabolome analysis: the potential of *in vivo* labeling with stable isotopes for metabolite profiling. *TRENDS in Biotechnology*, 23(1):28–33, 2004.

[20] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1st edition, 1995.

[21] M. Blot, editor. *Prokaryotic genomics*. Birkhäuser, Basel, 2003.

[22] J. M. Bower and H. Bolouri, editors. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, Cambridge, Massachusetts, 2001.

[23] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth International Group, Belmont CA, 1984.

[25] T. A. Brown. *Moderne Genetik*. Spektrum Akademischer Verlag, Heidelberg, 2nd edition, 1999.

[26] A. Buja and Y.-S. Lee. Data mining criteria for tree-based regression and classification. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pages 27–36, San Francisco, 2001.

[27] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[28] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In P. E. Bourne and M. Gribskov, editors, *Proceedings of the ISMB 2000*, pages 75–85, Menlo Park, CA, 2000. AAAI press.

[29] M. E. Califf and R. J. Mooney. Advantages of decision lists and implicit negatives in inductive logic programming. *New Generation Computing*, 16(3):263–281, 1998.

[30] N. A. Campbell and J. B. Reece. *Biologie*. Spektrum Lehrbuch. Spektrum Akademischer Verlag, Heidelberg, 6th edition, 2003. German edition by Jürgen Markl.

[31] E. Cantú-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computing*, 7(1):54–68, February 2003.

[32] D. Carvalho and A. Freitas. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. In *Proc. Genetic and Evolutionary Computation Conf (GECCO-2000)*, pages 1061–1068, Las Vegas, USA, July 2000. Morgan Kaufmann.

[33] J. Castro and J. L. Balcázar. Simple PAC learning of simple decision lists. In *Proceedings of the 6th International Workshop of Algorithmic Learning Theory*, volume 997, pages 239–248, Fukuoka, Japan, 1995. Springer.

[34] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of AAAI 1988*, pages 607–611, 1988.

[35] V. Cherkassky and F. Mulier. *Learning from data: Concepts, theory, and methods*. John Wiley & sons, New York, 1998.

[36] T. Chu, C. Glymour, R. Scheines, and P. Spirtes. A statistical problem for inference to regulatory structure from associations of gene expression measurements with microarrays. *Bioinformatics*, 19(9):1147–1152, April 2003.

[37] P. Comon. Independant component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[38] I. Corporation. S-plus, statistical and data mining software. http://www.insightful.com/, 1999.

[39] T. E. Corporation. The masslab software. http://www.thermoquest.com/.

[40] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley Interscience, New York, 1991.

[41] C. Daub. *Analysis of integrated transcriptomics and metabolomics data : a systems biology approach*. PhD thesis, University of Potsdam, 2004.

[42] C. Daub, S. Kloska, and J. Selbig. Metagenealyse: Analysis of integrated transcriptional and metabolite data. *Bioinformatics*, 19:2332–2333, 2003.

[43] A. Davison and D. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, Camebridge, 1997.

[44] L. D. Davison. Universal noiseless coding. *Transactions on Information Theory*, IT-19, Nov 1973.

[45] M. de Hoon, S. Imoto, and S. Miyano. Statistical analysis of a small set of time-ordered gene expression data using linear splines. *Bioinformatics*, 18(11):1477–1485, 2002.

[46] R. L. de Mantaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6(1):81–91, 1991.

[47] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[48] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 194–202, San Francisco USA, 1995. Morgan Kaufmann Publishers.

[49] F. Douglas. Gc/ms analysis. *Scientific Testimony - An online journal*, 2004.

[50] D. Edwards. *Introduction to Graphical Modelling*. Springer, New York, 2nd edition, 2000.

[51] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[52] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95(25):14863–14868, 1998.

[53] L. Fahrmeir, R. Künstler, I. Pigeot, and G. Tutz. *Statistik*. Springer, Berlin, 4th edition, 2003.

[54] U. M. Fayyad. *On the induction of decision trees for multiple concept learning*. PhD thesis, ECCS department, University of Michigan, 1991.

[55] U. M. Fayyad. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Camebridge, 1996.

[56] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the IJCAI'93*, volume 2, pages 1022–1027, Chambéry, France, 1993. Morgan Kaufmann Publishers.

[57] A. Fiat and D. Pechyony. Decision trees: More theoretical justification for practical algorithms. In *Proceedings of ALT'04*, 2004. to appear.

[58] O. Fiehn, J. Kopka, P. Dörrmann, T. Altmann, R. Trethewey, and L. Wilmitzer. Metabolite profiling for plant functional genomics. *Nature Biotechnology*, 18:1157–1161, Nov 2000.

[59] A. Flöter, J. Kopka, T. Schaub, J. Nicolas, and J. Selbig. Finding combinatorial causal relationships in metabolite concentration data using decision tree heuristics. In *ECCB 2003 Posters*, page 582, Paris, France, 2003.

[60] A. Flöter, J. Nicolas, T. Schaub, and J. Selbig. Threshold extraction in metabolite concentration data. In *Proceedings of the German Conference on Bioinformatics*, volume 1, pages 33–39, October 2003.

[61] A. Flöter, J. Nicolas, T. Schaub, and J. Selbig. Threshold extraction in metabolite concentration data. *Bioinformatics*, 20(10):1491–1494, July 2004.

[62] A. Flöter, T. Schaub, and J. Selbig. Finding metabolic pathways in decision forests. In *Proceedings of the 27th Annual Conference of the GfKl*, pages 199–206, Heidelberg, 2003. German Society of Classification, Springer-Verlag.

[63] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[64] N. Friedman and D. Koller. Being bayesian about network structure. In *Proc. Sixteenth Conf. on Uncertainty in Artificial Intelligence*, pages 201–210, 2000.

[65] N. Friedman, M. Linial, I. Nachman, and D. Peer. Using bayesian networks to analyze expression data. In *Proceedings of RECOMB 2000*, pages 127–135, Tokyo, Japan, April 2000. ACM.

[66] J. Gama, L. Torgo, and C. Soares. Dynamic discretization of continuous attributes. In *Proceedings of the Sixth Ibero-American Conference on AI*, pages 160–169, 1998.

[67] O. Gascuel, B. Bouchon-Meunier, G. Caraux, P. Gallinari, A. Guénoche, Y. Guermeur, Y. Lechevallier, C. Marsala, L. Miclet, J. Nicolas, R. Nock, M. Ramdani, M. Sebag, B. Tallur, G. Venturini, and P. Vitte. Twelve numerical, symbolic and hybrid supervised classification methods. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(5):517–571, 1998.

[68] U. Grimmer and H.-J. Mucha. *Data Mining*, chapter Datensegmentierung mittels Clusteranalyse, pages 109–141. Physica Verlag, Heidelberg, 1998.

[69] J. Haigh. *Probability Models*. Springer undergraduate mathematics series. Springer, London, 1st edition, 2002.

[70] J. Han and M. Kamber. *Data Mining - Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco, California, 1st edition, 2001.

[71] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, Massachusetts, 1st edition, 2001.

[72] J. Hartung and B. Elpelt. *Multivariate Statistik*. Oldenbourg, München, 6th edition, 1999.

[73] W. Hennig. *Genetik*. Springer, Berlin, 3rd edition, 2002.

[74] K. Ho and P. Scott. Zeta: A global method for discretization of continuous variables. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 191–194, 1997.

[75] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 3:297–314, 1962.

[76] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, Boston, Mass., 3rd edition, 1994.

[77] K. Horimoto and H. Toh. Automatic system for inferring a network from gene expression profiles. *Genome Informatics*, 12:270–271, 2001.

[78] S. P. Hunt, editor. *Functional genomics: a practical approach*. Oxfor University Press, Oxford, reprinted edition, 2002.

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

[80] F. Jensen. *An introduction to Bayesean networks*. Springer Verlag, New York, 1996.

[81] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, New Jersey, 4th edition, 1998.

[82] K. Jänich. *Lineare Algebra*. Springer, Berlin, 4th edition, 1991.

[83] G. Kahl. *The dictionary of gene technology : genomics, transcriptomics, proteomics*. Wiley, Weinheim, 2nd edition, 2001.

[84] M. Kanehisa. Kyoto encyclopedia of genes and genomes. http://www.genome.jp/kegg/, 2004.

[85] D. Karakos, S. Khudanpur, and J. Eisner. Unsupervised classification via decision trees: An information-theoretic perspective. In *Proceedings of ICASSP 2005*, Philadelphia PA, March 2005.

[86] S. Kauffman. Metabolic stability and epigenesis in randomly connected nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[87] R. King, C. Feng, and A. Sutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):259–287, May/June 1995.

[88] F. G. Kitson, C. N. McEwen, and B. S. Larsen. *Gas Chromatography and Mass Spectrometry: A Practical Guide*. Academic Press, 1996.

[89] D. E. Knuth. *The Art of Computer Programming: Fundamental Algorithms*, volume 1. Addison-Wesley, Reading, Mass., 3rd edition, 1997.

[90] Y. Kodratoff. *Introduction to machine learning*. Morgan Kaufmann Publishers, San Francisco, 1989.

[91] R. Kohavi and M. Sahami. Error-based and entropy-based discretisation of continuous features. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 114–119, 1996.

[92] J. Kopka. personal consultation, 2004.

[93] J. Kopka, A. Fernie, W. Weckwerth, Y. Gibon, and M. Stitt. Metabolite profiling in plant biology: platforms and destinations. *Genome Biology*, 5(6):Article 109, 2004.

[94] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn. Visualising plant metabolomic correlation networks using clique-metabolite matrices. *Bioinformatics*, 17:1198–1208, 2001.

[95] B. Krupa. On the number of experiments required to find the causal structure of complex systems. *Journal of Theoretical Biology*, 219:257–267, 2002.

[96] W. Kwedlo and M. Kretowski. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pages 392–397, 1999.

[97] K. Lee, D. Hwang, T. Yokoyama, G. Stephanopoulos, and M. L. Yarmush. Identification of optimal classification functions for biological sample and state discrimination from metabolic profiling data. *Bioinformatics*, 20(6):959–969, 2004.

[98] G. Leggewie. personal consultation. article in preparation.

[99] G. Leggewie, N. Gatzke, K. Piepenburg, B. Regierer, M. Udvardi, and J. Kopka. Metabolic adaptation to pi-deprivation in roots of wild type and transgenic potato. *submitted*, 2004.

[100] G. Leggewie, L. Wilmitzer, and J. Riesmeier. Two cdnas from potato are able to complement a phosphate uptake-deficient yeast mutant: identification of phosphate transporters from higher plants. *Plant Cell*, 9:626–629, 1997.

[101] K.-C. Li. Genome-wide coexpression dynamicas: Theory and application. *PNAS*, 99(26):16875–16880, December 2002.

[102] P. Lincoln and A. Tiwari. Symbolic systems biology: Hybrid modeling and analysis of biological networks. In *Proceedings of the 7th International Conference on Hybrid Systems: Computation and Control*, pages 660–672, Philadelphia, Pennsylvania, USA, March 2004. Springer.

[103] T. Linke. Graph theoretical characterization and computation of answer sets. In B. Nebel, editor, *IJCAI*, pages 641–645. Morgan Kaufman Publishers, 2001.

[104] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, 2nd edition, 2002.

[105] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551, 1997.

[106] Max-Planck-Gesellschaft. Definition proteomics. http://www.proteomics.em.mpg.de/, 2004.

[107] M. Mehta, J. Rissanen, and R. Agrawal. Mdl-based decision tree pruning. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, 1995. AAAI Press.

[108] R. Merkl and S. Waak. *Bioinformatik Interaktiv*. Wiley-VCH Verlag, Weinheim, Germany, 2003.

[109] G. Michal, editor. *Biochemical Pathways*. Spektrum Akademischer Verlag, Heidelberg, 1999.

[110] R. Michalski, J. Carbonell, and T. Mitchell, editors. *Machine Learning - an artificial intelligence approach*. Springer, Heidelberg, 1984.

[111] M. Middendorf, A. Kundaje, C. Wiggins, Y. Freund, and C. Leslie. Predicting genetic regulatory response using classification. In *Proceedings of the First Annual RECOMB Regulation Workshop*, June 2004.

[112] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 3(4):227–243, 1989.

[113] J. Mingers. An empirical comparison of selecting measures for decision-tree induction. *Machine Learning*, 3(4):319–342, 1989.

[114] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[115] T. Mitchell. *Machine Learning*. McGraw-Hill, Boston USA, 1st edition, 1997.

[116] Morgan and Sonquist. Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, 58:415–434, 1963.

[117] S. Muggleton, editor. *Inductive logic programming*. Academic Press, London, 1992.

[118] T. Murashige and F. Skoog. Murashige and skoog plant salts. *Physiol. Plant.*, 15:473–493, 1962.

[119] M. Murata, Q. Ma, and H. Isahara. Comparison of three machine-learning methods for thai part-of-speech tagging. *ACM Transactions on Asian Language Information Processing*, 1(2):145–158, June 2002.

[120] P. Murthy and D. Aha. Uci repository of machine learning databases (machine-readable data repository). University of California-Irvine, Department of Information and Computer Science, 1994.

[121] S. K. Murthy. *On growing better decision trees from data*. PhD thesis, University of Maryland, 1997.

[122] S. K. Murthy, S. Salzberg, and S. Kasif. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[123] M. Möhlig, A. Flöter, J. Spranger, M. Ristow, T. Schill, H. Schlösser, G. Brabant, A. Pfeiffer, J. Selbig, and C. Schöfl. Vorhersage eines gestörten glukosestoffwechsels bei frauen mit polyzystischem ovarsyndrom. In *43. DDG-Tagungsband*, page poster, Dresden, 2005. Deutsche Dermatologische Gesellschaft.

[124] G. Nakhaeizadeh, editor. *Data Mining - Theoretische Aspekte und Anwendungen*. Physica-Verlag, Heidelberg, 1998.

[125] R. Nock. Inducing interpretable voting classifiers without trading accuracy for simplicity: Theoretical results, approximation algorithms, and experiments. *Journal of Artificial Intelligence Research*, 17:137–170, August 2002.

[126] S. Oba, M. Sato, I. Takemasa, M. Monden, K. Matsubara, and S. Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19:2088–2096, 2003.

[127] N. I. of Standards and Technology. Amdis. http://www.amdis.net.

[128] I. Ong, J. Glaser, and D. Page. Modelling regulatory pathways in e.coli from time series expression profiles. *Bioinformatics*, 18(1):241–248, 2002.

[129] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl.1):215–224, 2001.

[130] P. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. The MIT press, Cambridge MA, 2000.

[131] D. Pyle. *Data preparation for Data Mining*. Morgan Kaufmann publishers, San Francisco, CA, 1999.

[132] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[133] J. R. Quinlan. Rule induction with statistical data - a comparison with multiple regression. *Journal of the Operational Research Society*, 38:347–352, 1987.

[134] J. R. Quinlan. Unknown attribute values in induction. In *Proceedings of the Sixth International Machine Learning Workshop*, pages 164–168, San Mateo, CA, 1989. Morgan Kaufmann.

[135] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[136] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo USA, 1993.

[137] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 3:77–90, 1996.

[138] J. R. Quinlan. C5.0. http://www.rulequest.com/, 2001.

[139] J. R. Quinlan and R. L. Rivest. Inferring decision trees using minimum description length principle. *Information and Computation*, 80(3):227–248, 1989.

[140] T. Raiko and H. Valpola. Missing values in nonlinear factor analysis. In *Proceedings of the 8th Int. Conf. on Neural Information Processing (ICONIP'01)*, pages 822–827, Shanghai, 2001.

[141] M. Regenass-Klotz. *Gentechnik*. Birkhäuser, Basel, CH, 1998.

[142] J. J. Rice, Y. Tu, and G. Stolovitzky. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics Advance Access*, October 14, 2004.

[143] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[144] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[145] R. Rojas. *Neural Networks: a systematic introduction*. Springer, Berlin, 4th edition, 1996.

[146] L. Rokach and O. Maimon, editors. *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, chapter Geometric Methods for Feature Extraction and Dimensional Reduction: A Guided Tour. Kluwer Academic Publishers, 2004. to appear.

[147] G. Rätsch and T. Onada. Boosting.org. http://www.boosting.org/, 2004.

[148] M. Röhm and D. Werner. Isolation of root hairs from seedlings of pisum sativum: identification of root hair specific proteins by in situ labelling. *Physiol. Plant.*, 69:129–136, 1987.

[149] U. Rößner, A. Lüdemann, D. Brust, O. Fiehn, T. Linke, L. Willmitzer, and A. Fernie. Metabolite profiling allows comprehensive phenotyping of genetically or environmentally modified plant systems. *Plant Cell*, 13:11–29, 2001.

[150] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[151] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the IJCAI'99*, pages 1401–1406, 1999.

[152] R. E. Schapire. A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[153] R. E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory*, volume 1720, pages 13–25, Tokyo, Japan, December 1999. Springer.

[154] N. Schauer, D. Steinhauser, S. Strekov, D. Schomburg, G. Allison, T. Moritz, K. Lundgren, U. Rößner-Tunali, M. G. Forbes, L. Willmitzer, A. Fernie, and J. Kopka. Gc-ms libraries for the rapid identification of metabolites in complex biological samples. *Elsevier FEBS*, Letters 579:1332–1337, 2005.

[155] H. Scheid, editor. *Duden - Rechnen und Mathematik*. Dudenverlag, Mannheim, 5th edition, 1994.

[156] M. Scholz, S. Gatzek, A. Sterling, O. Fiehn, and J. Selbig. Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics*, April 15th 2004. Advance Access.

[157] D. Schomburg. Brenda. http://www.brenda.uni-koeln.de/, 2004.

[158] D. Schomburg. Cubic. Invited Talk at Max-Planck-Institue for Molecular Plant Physiology, March 2004.

[159] S. Schulze-Kremer. *Molecular Bioinformatics*. deGruyter, Berlin, 1995.

[160] B. Schölkopf, C. Burges, and A. Smola. *Advances in kernel methods: Support vector learning*. MIT press, Cambridge, MA, 1999.

[161] U. Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, Heidelberg, 4th edition, 1995.

[162] R. Sedgewick. *Algorithmen*. Addison-Wesley, Bonn, 2nd edition, 1995.

[163] A. K. Seewald, J. Petrak, and G. Widmer. Hybrid decision tree learners with alternative leaf classifiers: An empirical study. In *Proceedings of the 14th International FLAIRS Conference 2001*, Menlo Park, CA, 2000. AAAI Press.

[164] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, 2003.

[165] A. D. Sevin, V. DeGruttola, M. Nijhuis, J. M. Schapiro, A. S. Foulkes, M. F. Para, and C. A. Boucher. Methods for investigation of the relationship between drug-susceptibility phenotype and human immunodeficiency virus type 1 genotype with applications to aids clinical trial group 333. *The Journal of Infectious Deseases*, 182:59–67, 2000.

[166] C. E. Shannon. A mathematical theory of communication. *The Bell System Technology Journal*, 27:379–423,623–656, 1948.

[167] B. Shipley. *Cause and Correlation in Biology*. Cambridge University Press, Cambridge UK, paperback edition, 2002.

[168] B. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society*, 43(Series B):97–99, 1981.

[169] L. I. Smith. A tutorial on principal component analysis. Technical report, University of Otago, New Zealand, 2002.

[170] L. Soinov, M. Krestyaninova, and A. Brazma. Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biology*, 4(R6), 2003.

[171] Electronic textbook statsoft. http://www.statsoftinc.com/textbook/stathome.html, 2003.

[172] D. Steinhausen and K. Langer. *Clusteranalyse*. Walter de Gruyter, Berlin, 1977.

[173] R. Steuer, J. Kurths, C. Daub, J. Weise, and J. Selbig. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl. 2):231–240, Oct 2002.

[174] R. Steuer, J. Kurths, O. Fiehn, and W. Weckwerth. Observing and interpreting correlations in metabolomic networks. *Bioinformatics*, 19(8):1019–1026, May 2003.

[175] L. Stryer. *Biochemistry*. W.H. Freeman and Company, New York, 4th edition, 1995.

[176] S. Suhai, editor. *Theoretical and computational Methods in Genome Research*. Plenum Press, New York, 1997.

[177] L. R. Tang, M. E. Califf, and R. J. Mooney. An experimental comparison of genetic programming and inductive logic programming on learning recursive list functions. Technical report, University of Texas, Austin, TX 78712-1188, March 3rd 1998.

[178] S. C. T. Telecommunications. American national standard t1.523-2001. http://www.atis.org/tg2k, 2001.

[179] A. Tiessen. *Regulatory Network of Starch Synthesis in Plants*. PhD thesis, University of Heidelberg, Germany, 2003.

[180] H. Toh and K. Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical gaussian modeling. *Bioinformatics*, 18(2):287–297, 2002.

[181] W. Tong, Q. Xie, H. Hong, H. Fang, L. Shi, R. Perkins, and E. F. Petricoin. Using decision forest to classify prostate cancer samples on the basis of seldi-tof ms data: Assessing chance correlation and prediction confidence. *Environmental Health Perspectives*, 112(16):1622–1627, 2004.

[182] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altmann. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 2001.

[183] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[184] P. Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[185] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[186] V. Vapnik. *Statistical learning theory*. John Wiley & sons, New York, 1998.

[187] J. Wang, O. Myklebost, and E. Hovig. Mgraph: graphical models for microarray data analysis. *Bioinformatics*, 19(17):2210–2211, 2003.

[188] K. Wang and B. Liu. Concurrent discretization of multiple attributes. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 250–259, 1998.

[189] S. M. Weiss and N. Indurkhya. *Predictive Data Mining*. Morgan Kaufman Publishers, San Francisco, CA, 1998.

[190] S. Wrobel. Data mining und wissensentdeckung in datenbanken. *KI*, 1:6–10, 1998.

[191] M. Yang and D. H. Robinson. *Understanding and Learning Statistics by Computer*. World Scientific, Singapore, 1986.

[192] Y. Yang and G. Webb. A comparative study of discretisation methods for naive-bayes classifiers. In *Proceedings of Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, Tokyo, Japan, 2002.

[193] C. Yeang and T. Jaakkola. Physical network models and multi-source data integration. In *RECOMB'03 Proceedings*, pages 312–321, Berlin, April 2003.

[194] K. Yeung and W. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[195] H. Zhang and B. Singer. *Recursive Partitioning in the Health Sciences*. Springer, New York, 1st edition, 1999.

[196] Z.-H. Zhou and Z.-Q. Chen. Hybrid decision tree. *Knowledge-Based Systems @Elsevier*, 15(8):515–528, 2002.

[197] G. Zweiger. Knowledge discovery in gene-expression-microarray data: mining the information output of the genome. *Trends in Biotechnology*, 17:429–436, 1999.

Chapter 5

Appendix

5.1 Novel tools for the application of the introduced techniques

This section briefly introduces two user-friendly tools that have been implemented after the completion of the previously presented studies. Both tools are written in JAVA^{TM} and are thus platform-independent. The screen shots are taken from a Windows based system.

Combinatorix

This computer tool allows for the calculation of Conditional Mutual Information as explained by T.M. Cover and J.A. Thomas [40]. That is the systematic calculation of mutual information under different conditions of a third variable. The tool needs a data file on which the different combinations of conditions and correlating variables are evaluated. It produces a result file that can either be viewed graphically within the tool or textually with a text editor. A once produced result file can be viewed independently of the data file.
In Figure 5.1, we see the initial screen of *Combinatorix* which allows for the setting of five parameters. The parameters are:

- Number of cut points: The number of cut points determines the number of conditions under which the Mutual Information between the correlating variables is calculated. For any conditioning variable and any cut point, the data set is split into a subset in which the value of the conditioning variable is below the cut point, and a subset in which the value is above the cut point. The Mutual Information is then calculated for both subsets.

- Target variable 1: This fixes the first correlating variable of any Mutual Information. If it is set to -1 all variables will be used. On larger data sets, a comprehensive calculation quickly exhausts the memory.

- Target variable 2: This fixes the second correlating variable of any Mutual Information. If it is set to -1 (or “all”) all variables will be used. For very large data sets, a comprehensive calculation may not be possible due to lack of computer memory. So, Target 2 can be limited to one variable. For exploratory analyses, Target 2 should tried to be set to “all”.

- Fix condition: This fixes the conditioning variable. If the checkbox is activated the conditioning variable is specified through the adjacent item. This allows for a further reduction of memory demand or a more target-oriented analysis.

- Sorted Output: This toggle initiates a sorting of all calculated Mutual Informations according to the gain of Mutual Information under a certain condition as compared to the Mutual

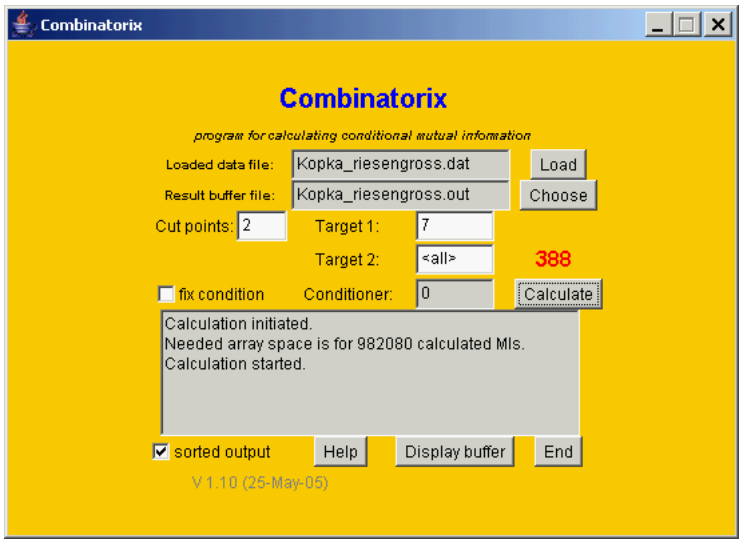

Figure 5.1: The initial screen of *Combinatorix*.

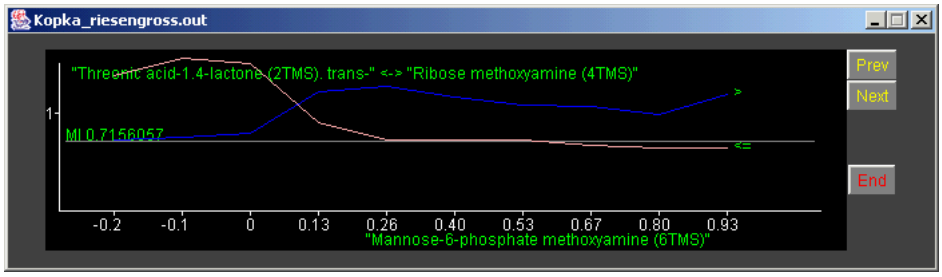

Figure 5.2: A sigmoidal progression of mutual information along several conditions.

Information on the complete data set. This feature is only feasible if the result buffer is viewed with a text editor. It has no effect on the graphical display. For exploratory analyses, the sorting allows for a quick finding of significant Conditional Mutual Informations.

Figure 5.2 depicts the graphical output window. It is possible to browse through all results with the ”Prev” and ”Next” buttons. Here, we can see that the mutual information between “Threonic acid” and “Ribose methoxyamine” drops (rises respectively) drastically when the data set is split at a threshold of approximately 0.134 for “Mannose-6-phosphate methoxyamine”. This is a very good example for a truly *conditional* mutual information.

With a click into the graphical display another window pops up that displays the scatter plots of the current variables. The first plot contains those objects where the conditioning variable is above the threshold closest to the click, the other plot contains the remaining objects. In Figure 5.3, we see the plots for the above mentioned constellation.

Identifix

This computer tool allows for the identification of significant thresholds as introduced by A.Flöter, J.Nicolas, T.Schaub and J.Selbig [61]. The tool needs a data file on which several thresholds are evaluated through two quality measures. It produces a result file that can be viewed graphically. A once produced result file can be viewed independently of the data file.

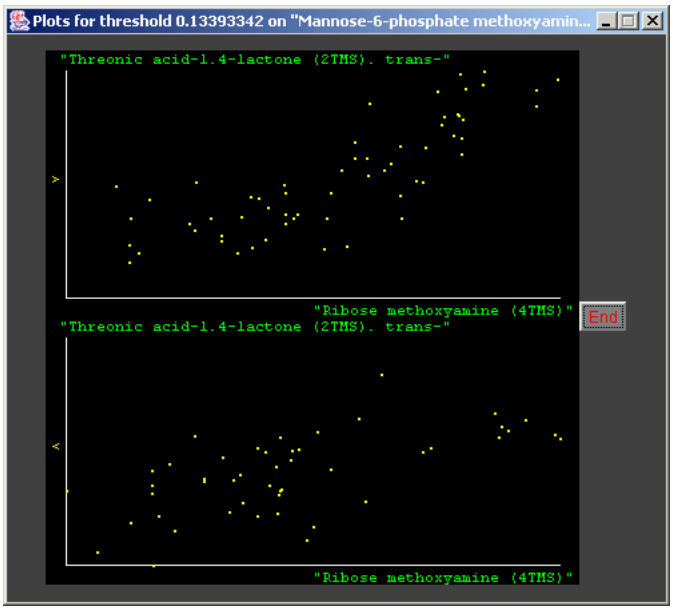

Figure 5.3: The scatter plots of the correlating variables for the particular threshold where MIs rise and drop.

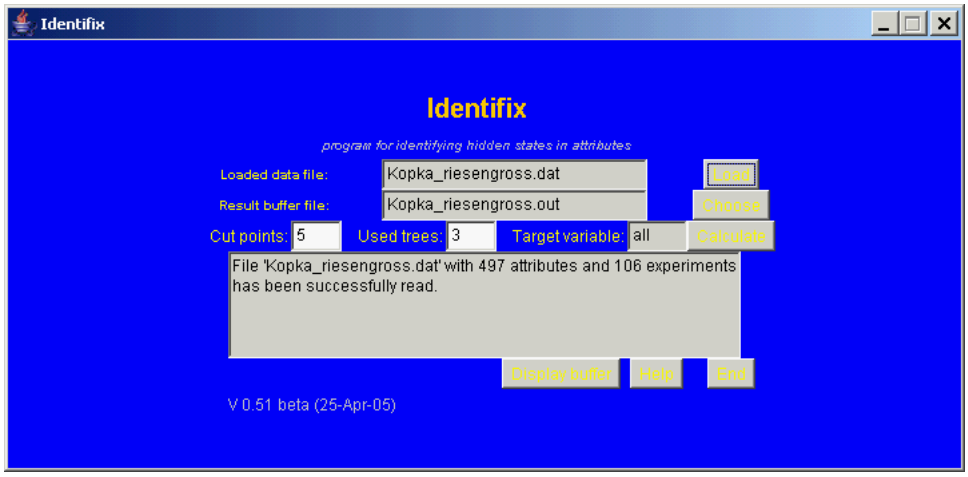

Figure 5.4: The initial screen of *Identifix*.

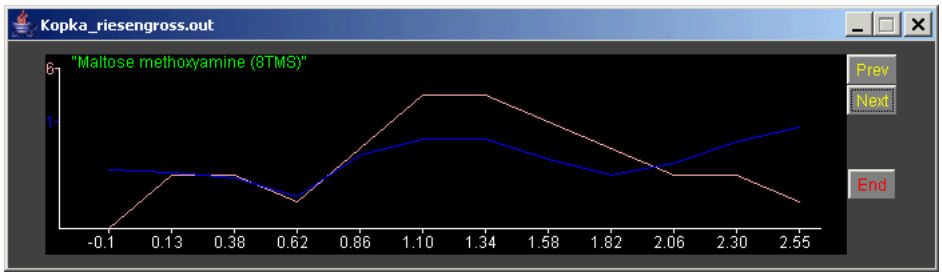

Figure 5.5: Progression of the quality measures along several thresholds. Thresholds 1.1 and 1.34 are identified as significant.

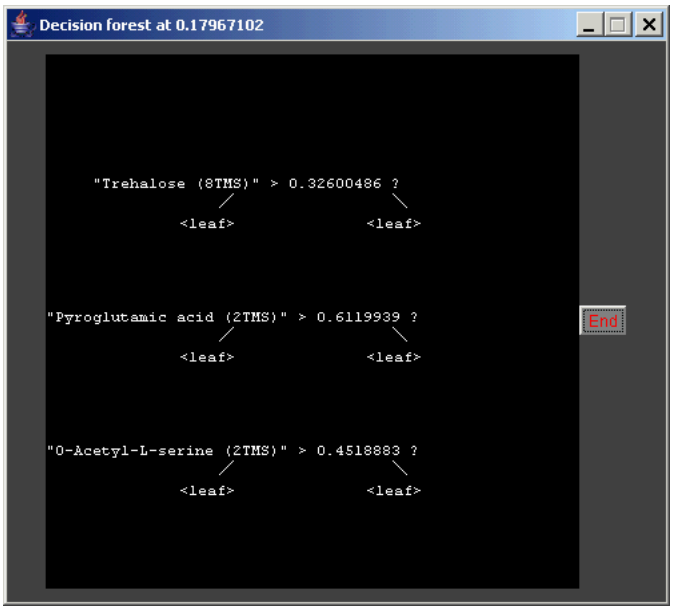

Figure 5.6: An example forest for a specific threshold containing 1-leaf-trees only.

Figure 5.4 displays the initial screen of *Identifix*. Both needed parameters are set here. These are the number of desired cut points and the number of trees accounted for when calculating the quality measures.

The number of cut points determines the number of thresholds that will be evaluated for each variable. Note that the computing time will linearly increase with the number of desired cut points. A minimum of 5 cut points is needed for feasible results. A number of 15 and higher is proposed for more convenient analyses.

The desired number of trees determines the number of trees that will be used in comparisons for the quality measures. It also determines the number of induced trees for each threshold (it has been fixed to the double of the number of used trees). That way, this parameter adds linearly to the computing time.

Figure 5.5 shows the graphical output window. In this display, it is possible to browse through all results with the ”Prev” and ”Next” buttons. The pink curve indicates the progression of the stability criterion, the blue curve indicates the progression of the effectiveness (quality) criterion.

A click at a specific threshold will pop up a new window that displays the actual decision forest that has led to the quality scores. In Figure 5.6, such a forest is displayed.

5.2 Complexity of the calculations

For a attributes, s objects in the sample and c conditions (= thresholds or cut points), the follwing table gives the time complexities of the three used methods (as implemented in this thesis):

Method	Time complexity
calculating lCMI	$a^3 * c^2 * s^2$
estimating high lCMI	$a * c * s * log(s)$
evaluating thresholds	$a^2 * c * s * log(s)$

5.3 Code for generating artificial data

This section contains the generic code for the production of partially correlated data. The dependencies can be altered by changing the assignments of values to variables.

```
// Generates data with specific dependencies between attributes
// File: generate.java
// Invokation: java generate
// Output to StdOut
// André Flöter 020205 (130203)

import java.io.*;
import java.math.*;
import java.util.*;

public class generate{

    static int samples = 200;
    static float noise = 60;  // percentage of noise

    public static void main(String[] args) {

        float b,c,d,e,f,g;
        Random r = new Random();
        float noise1=0f,noise2=0f;

        for (int a=0; a<samples; a++) {
            b = ((float)a*10/(float)samples-4.5f)*((float)a*10/(float)samples-4.5f);
            c = 20f-a*10f/(float)samples;            // linearly decreasing number
            d = 20f*r.nextFloat();                   // number between 0-20
            e = 20+20f*r.nextFloat();                // number between 20-40
            f = 20f*r.nextFloat();                   // number between 0-20
            g = 20f*(float)r.nextGaussian();         // Gaussian distributed number
            noise1 = noise*20f*r.nextFloat()/100f;   // noise variable
            noise2 = noise*20f*r.nextFloat()/100f;   // noise variable
            if (e>30f) { System.out.print(20f-d+noise1+"\t"); } // partial dependence
            else { System.out.print(40f-d+noise1+"\t"); }
            System.out.print(e+noise2+"\t");         // condition (uniform random)
            System.out.print(d+"\t");                // independent (uniform random)
            System.out.print(b+"\t");                // time-dependent (parabola)
            System.out.print(f+"\t");                // independent (uniform random)
            System.out.print(c+"\t");                // time-dependent (linear)
            System.out.println(g);                   // independent (gaussian random)
        }
    }
}
```