# DISTANCE-BASED LEARNING FROM STRUCTURED DATA

Valentin Gjorgjioski

**Doctoral Dissertation**
**Jožef Stefan International Postgraduate School**
**Ljubljana, Slovenia**

**MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA**
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Valentin Gjorgjioski

# DISTANCE-BASED LEARNING FROM STRUCTURED DATA

**Doctoral Dissertation**

# STROJNO UČENJE Z RAZDALJAMI ZA STRUKTURIRANE PODATKE

**Doktorska disertacija**

**Supervisor:** Prof. Dr. Sašo Džeroski

Ljubljana, Slovenia, March 2015

*To my parents and in memory of my grandmother Marija*

# Acknowledgments

First of all, I am very thankful to my supervisor Professor Sašo Džeroski for giving me the opportunity to work with him and to be part of his research group. He was guiding my research since my introduction to the area of machine learning and until the completion of this thesis. He was always feeding me with ideas, supporting and motivating me with feedback and keeping the big picture in front of me at any time it was needed.

I would also like to thank the members of the PhD Committee, Prof. Dr. Marko Bohanec, Asst. Prof. Bernard Ženko and Prof. Dr. Ljupčo Todorovski, for their engagement and careful reading, as well as, for their numerous comments and feedback given during my final work on the thesis. Many of their comments helped me to simplify the things and to make this thesis much easier to read and understand.

Several funding bodies financially supported the research presented here. I wish to thank all of them for their support through the years: the Slovenian Research Agency through the "young researcher" program, the European Commission through the research project PHAGOSYS (Systems biology of phagosome formation and maturation - modulation by intracellular pathogens). Also, the research presented here was performed in the context of two projects funded by the European Commission. First, at the start of the PhD, the IQ project (Inductive Queries for Mining Patterns and Models), which offered me a lot of experience and feedback needed for this research. Second, at the end of the PhD, the MAESTRA project (Learning from Massive, Incompletely annotated, and Structured Data), which helped placing the research in a wider context.

I would like to thank all of my co-authors of the papers that we wrote together, especially the co-authors of the papers included in this thesis: Jan Struyf, Ivica Slavkov, Sašo Džeroski, Dragi Kocev, Marko Debeljak, Andrej Bončina, Daniela Stojanova, Pance Panov and Andrej Kobler. Jan, thank you for introducing me to CLUS. Many thanks also to Alexandros Kalousis and Adam Woznica for their open communication, support and feedback during my visit in Geneva.

I wish to thank all members of the Department of Knowledge Technologies at the Jožef Stefan Institute. Especially thanks to my office mates Marko Debeljak, Aneta Trajanov, Dragana Miljković and Darko Čerepnalkoski for making every day at JSI different and fun, to Katerina Taškova for the interesting discussions and her always-willing-to-help attitude, and to Mili Bauer for her support and encouragement. I would like to say thank you to the members of the "group-of-six" that joined the Department of Knowledge technologies at the same time and walked the road-to-a-PhD together: Panče, Dragi, Aleksandar, Aneta and Ivica. Aleksandar thank you for all the night-long discussions, ideas and working on the topics related to machine learning.

I am grateful to many of my teachers during my education who influenced my life greatly, and certainly contributed in one way or another to the creation of this thesis. I would like to mention just some of them here in chronological order: Kuzman Kuzmanoski, Svetlana Gjorgoska, Kosta Mijoski, Risto Malčeski, Marjan Gušev, Nikola Pandeski, Dimitra Karčicka, Margita Kon-Popovska, Katerina Zdravkova and Sašo Džeroski.

My friends Kiril, Biljana, Lazar and Kiro were supporting me and were giving me the needed pushes when it was the hardest. Thank you for listening, understanding and for your words that were always very helpful. Big thanks also to Hristijan, Marko, Vladimir and Antonio for making the time spent outside of JSI that much more fun which was very important for recharging the batteries.

I want to thank all of the colleagues in SAS Adriatic Region, especially to the colleagues in the Ljubljana office. Their constant support and questions about the progress of my PhD were definitely an additional motivation for finishing this thesis. Thanks to Dare and Polona for the understanding and support. The biggest and special thank you to my office mate, Aleksander Pivk, not only for the understanding, but for the constant support, pushes and motivational speeches. You helped me a lot, thank you Sandi!

Before I say my thanks to my family, I would like to thank my friend Dragi Kocev. He moved to Ljubljana just a month later than me, and we spent a lot of time together watching football games and movies, discussing research, science, philosophy, and the inevitable topics: cars and girls. We helped each other during the writing of the thesis, but his help in the last year of writing of this thesis was enormously huge. His proof-reading of the thesis, suggestions for reorganization and insightful comments were irreplaceable. I will never forget that and words cannot express how thankful I am. Dragi, thank you!

I would like to express my deepest gratitude to my parents and my sister. To my father for his encouragement and his never-giving-up strategy. To my mother for her understanding and faith in me. To my sister for her don't-worry-it's-gonna-happen attitude, and to her two little angels for their honest smile and words that made my time spent with them unforgettable. Thank you for your love, understanding and unconditional support.

# Abstract

While the majority of methods in statistics, machine learning and data mining deal with data that is represented as tuples of scalar values, in this thesis, we focus on distance based learning algorithms for structured data. Besides tuples of discrete or real values, structured data include sets and sequences thereof, as well as, recursively defined, tuples, sets and sequences of structured data. Specifically, we consider distances on structured data and develop methods for learning from such data. We address both the tasks of predictive modeling and clustering in this context.

We explore two paradigms for using distances when learning from structured data, namely model-free and model-based learning. As representatives of the first, we consider instance-based nearest-neighbor algorithms for prediction and algorithms for distance-based clustering, for which no models are built. As representative of the second, we consider predictive clustering, where tree-based models are built. We propose several approaches for predictive modeling and clustering within each of the two paradigms.

Within the model-free learning paradigm, we propose and develop generic versions of the $k$-medoids and hierarchical agglomerative clustering algorithms for clustering structured data, and of the $k$-nearest neighbour algorithm for predictive modeling for structured data. In the latter case, the algorithm is able to handle structured input and structured output. Within the predictive clustering paradigm, we extend the framework for predictive clustering trees to be able to handle arbitrarily structured data on the target or output side.

We explore several distances for each of the type constructors, i.e., set, tuple and sequence. In addition to the implementation of known distances for sets, we propose and implement a new distance called Greedy Matching. The framework enables users to use these distances out of the box and supports various possibilities for combining distances at various levels in the structure of the data types.

The predictive clustering tree approaches for different types of structured outputs were evaluated and their utility demonstrated on a number of practically relevant problems. For the basic case of tuples of real values, the application consisted of predicting the state of the forests in Slovenia (and in particular forest stand height and canopy cover) from remotely sensed data, i.e., satellite images. For the case of predicting time series, the application considered was finding explained groups of genes with similar time course profiles of gene expression under different stressful conditions. Finally, for the case of tuples of time series, the application of clustering profiles of forest growth stock in Slovenian forests was considered.

We outline several possible directions for extending this work. One is the further evaluation of the instance-based algorithms on new real-life datasets. Another is the use of the framework for metric learning (i.e., learning distances) either by selecting the most appropriate distance from a space of possible distances, or by combining various distances using modifications of well known metric learning algorithms.

# Povzetek

Večina metod za analizo podatkov v statistiki, strojnem učenju in rudarjenju podatkov obravnava podatke v obliki n-teric skalarnih vrednosti (števliskih ali nominalnih), medtem ko se v tej disertaciji osredotočimo na obravnavo strukturiranih podatkov. Strukturirani podatki lahko poleg n-teric števliskih ali nominalnih vrednosti vključujejo tudi njihove množice in zaporedja, kot tudi rekurzivno definirane n-terice, množice ali zaporedja strukturiranih podatkov. V disertaciji obravnavamo razdalje med strukturiranimi podatki in razvijamo metode za strojno učenje iz takih podatkov. Ukvarjamo se tako s problemom napovednega modeliranja kot tudi s problemom razvrščanja v skupine, tj. z nadzorovanim in nenadzorovanim strojnim učenjem.

Raziščemo dve paradigmi za uporabo razdalj pri učenju iz strukturiranih podatkov. To sta učenje na osnovi primerkov (angl. instance-based learning), ki shranjuje in razvršča primerke ter ne uporablja modelov (angl. model-free), in napovedno razvrščanje (angl. predictive clustering), ki gradi modele v obliki dreves (angl. model-based). Predlagamo več pristopov k napovedovanju in razvrščanju znotraj obeh paradigm.

Znotraj paradigme učenja na osnovi primerkov predlagamo in razvijemo razširitve dveh pristopov k razvrščanju za delo s poljubno strukturiranimi podatki. Gre za metodi k-medoid in hierarhično razvrščanje z združevanjem (angl. hierarchical agglomerative clustering). Podobno razširitev predlagamo tudi za algoritem k-NN za napovedno modeliranje, ki lahko v razširjeni različici obravnava poljubno strukturirane podatke, tako na vhodu kot na izhodu. Znotraj paradigme napovednega razvrščanja razširimo pristop k učenju dreves za napovedno razvrščanje v smeri obravnave poljubno strukturiranih podatkov na izhodu.

Raziščemo več različnih razdalj za vsako vrsto strukturiranih podatkov, tj. za n-terice, množice in zaporedja. Poleg implementacij že znanih razdalj vpeljemo in implementiramo novo razdaljo med množicami, ki jo imenujemo razdalja požrešnega ujemanja. Naša metoda omogoča uporabnikom, da uporabljajo vnaprej definirane razdalje kot tudi da te razdalje kombinirajo na različne načine in na različnih nivojih v podatkovnih strukturah.

Ovrednotili smo razširjene algoritme za gradnjo dreves za napovedno razvrščanje za napovedovanje različnih vrst izhodov in pokazali njihovo uporabnost na več praktično pomembnih problemih. Drevesa za napovedovanje več zveznih izhodnih spremenljivk (n-teric realnih števil) smo uporabili za ocenjevanje stanja (višine in gostote) slovenskih gozdov iz daljinsko zaznanih podatkov oz. satelitskih posnetkov. Drevesa za napovedovanje časovnih vrst (zaporedij realnih števil) smo uporabili pri iskanju skupin genov glive kvasovke s podobnim časovnim odzivom izraženosti ter konsistentnimi funkcionalnimi lastnostmi. Drevesa za hkratno napovedovanje več časovnih vrst smo uporabili za razvrščanje profilov rasti lesne zaloge različnih razredov dreves v slovenskih gozdovih.

Podamo tudi več možnih smernic za razširitev in nadgradnjo pričujočega dela. Te vključujejo nadaljnjo evalvacijo predlaganih algoritmov za učenje na osnovi primerkov na novih množicah podatkov. Vljučujejo tudi uporabo naše metode za učenje ustreznih razdalj za podane podatke, pri čemer lahko razdaljo izberemo iz množice predefiniranih razdalj ali pa jo iz znanih razdalj sestavimo z uporabo obstoječih algoritmov za učenje razdalj (angl. metric learning).

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

In the area of data mining, machine learning and more generally data analytics we are increasingly often faced with increasingly complex data (Bakir et al., 2007). One major source of complexity of data is its internal structure. The classical composition of a data item is a record or a tuple of scalar values of certain variables, attributes or properties of the object described. However, we can also encounter sequences of scalar values or sets of scalar values or even tuples, sequences or sets of structured data values. When defining the space of data to be analyzed it is necessary on the one hand to specify the possible types of values that data points can take. On the other hand, a crucial part of the definition of the data space is the definition of a distance between the points in that space (Džeroski, 2006). Distances play a crucial role in both statistical and machine learning methods for data analysis.

Distances are important in both predictive modeling, where tasks such as classification and regression are addressed, and also in unsupervised learning or clustering (Langley, 1996). In fact, a very large portion of the research on clustering is concerned with distance-based clustering, where we try to group objects that are similar according to a given distance, while objects from different groups are as dissimilar as possible (Kaufman & Rousseeuw, 1990). Distances also play a key role in statistical methods such as nearest neighbor classification (Cover & Hart, 1967).

The majority of methods in statistics, machine learning and data mining deal with data that has the form of tuples of scalar values of variables. Therefore they consider distances in Euclidean spaces or spaces which are Cartesian products of sets of scalar values, be it continuous or discrete. In this thesis, we consider distances on structured data and develop methods for learning from such data. We address both the tasks of prediction and clustering in this context.

## 1.1 Learning from structured data

### 1.1.1 Structured data

When we say structured data we mean data that are composed of subparts organized or bound together by certain organizational structures or type constructors (Džeroski, 2006). The classical data structure considered in machine learning is the vector of attribute values or tuple or record which contains scalar values of either discrete or continuous variables. In the area of computer science, however, many different data structures have been considered which are composed from primitive datatypes and type constructors (Bakir et al., 2007).

The primitive datatypes considered are numerical continuous real values, which is the datatype primarily used in statistics, and also discrete, nominal, or categorical variables. Type constructors, which are used to put primitive values in more complex structures, include the tuple, the most commonly used one, but also sequences and sets. While other type constructors may be considered in practice, we will primarily focus here on the use of three type constructors: tuple, sequence and set.

### 1.1.2 Data mining tasks

Within the area of machine learning and data mining, many different tasks of data analysis have been considered. These can be roughly grouped into two major categories (Langley, 1996). The first category and the most commonly addressed one is the task of predictive modeling, where we want to predict the value of a certain property of an object from other properties of that object or in general a description of that object. The second category is the task of clustering, where we want to group objects that are similar to each other, however, no specific property is designed as the target that we want to predict. The first task is known under the name of supervised learning, where the values of the target property that we want to predict come from a human supervisor. The second category of problems is known under the name of unsupervised learning, because no such supervision is provided. More recently, a new paradigm has emerged in machine learning which combines properties of the predictive modeling and clustering paradigms and is known under the name predictive clustering (Blockeel, 1998; Kocev, 2011; Ženko, 2007; Kocev et al., 2013; Struyf & Džeroski, 2006).

### 1.1.3 Structured input data (relational learning)

Most commonly, an individual data item within a set of data is a tuple or a vector of scalar attribute values. However, to deal with structured data, we need to consider data which are composed from substructures and sub-parts. In the area of predictive modeling, significant attention has been devoted over the last two decades to the task of learning from structured input data (Bakir et al., 2007). An example task might concern predicting mutagenicity of chemical compounds, where the structure of the compounds is described by graphs of bonds that connect atoms within the compound (Debnath et al., 1991).

In the context of relational learning, the task of relational classification and relational regression have been considered (Džeroski & Lavrač, 2001). The task of relational clustering has received much less attention. An approach worth mentioning in this context is inductive logic programming, where structured descriptions in the form of logic programs are considered (Lavrač & Džeroski, 1994).

### 1.1.4 Structured output prediction

More recently, there has been an increased interest in the task of predictive modeling where the target that we want to predict is a structured value composed of a type constructor and its sub-components. Tasks like multi-target classification, multi-target regression, multi-label classification and hierarchical multi-label classification fall in this context (Kocev et al., 2013). A variety of approaches have been developed for these tasks, where the input is still a vector of attribute values, but the outputs are structured values as described above.

## 1.2 Distance-based learning

A large number of methods from statistics and machine learning take the notion of a distance as a central notion. These methods are called distance-based learning methods (Wettschereck, 1994). Methods for distance-based learning exist both for predictive modeling or supervised learning, on the one hand, and for unsupervised learning or clustering, on the other hand.

There are two paradigms for distance-based learning, namely model-free and model-based learning. As representatives of the first, we consider instance-based nearest-neighbor algorithms for prediction and algorithms for distance-based clustering, for which no models are built. As representative of the second, we consider predictive clustering, where tree-based models are built.

Purely distance-based methods for prediction are known under the name of instance-based learning, lazy learning, memmory-based learning or nearest neighbor methods. In essence, to make a prediction for a new data item, the data item is compared to previously seen data items: Distances to them are first calculated and then the most similar object or objects are taken to produce a prediction for the new example at hand. It is therefore clear that the notion of a distance plays a critical role in this context. Besides the basic nearest neighbor method, the natural extension, namely the k-nearest neighbors method, is often used. In addition, methods that filter the instances and do not store all of the instances can be used. Instance-based learning methods have recently also been considered in the online learning or streaming context (Shaker & Hüllermeier, 2012).

In unsupervised learning or clustering, while many different approaches exist, distance-based clustering is very common. In distance-based clustering, we are given a set of data points that need to be clustered or partitioned into subsets or clusters of objects that are similar to each other. Objects from different clusters should be as dissimilar from each other as possible. Clearly the notion of a distance plays a critical role in this context as well.

Nearest neighbor methods and distance-based clustering methods share a common approach. They do not build models, i.e., are model-free. They primarily store and partition instances. We will thus refer to both of them as instance-based approaches.

Predictive clustering is a recent model-based paradigm that combines the paradigms of predictive modeling on the one hand and clustering on the other hand. Predictive clustering methods produce clusters just like clustering and are able to make predictions just like predictive modeling methods. In addition, predictive clustering methods produce a symbolic description for each partition that they produce, using rules to describe individual clusters or trees to describe a hierarchical clustering (Blockeel, 1998).

## 1.3 Motivation, hypothesis and goals

In current machine learning methods that deal with structured data, we can identify two large groups of methods. The first one deals with structured input data as input to the task of predictive modeling. This is the case in relational learning where we consider the task of relational classification, for example, predicting whether a certain molecule is mutagenic or not, or relational regression where, for example, we could predict the biodegradability time of certain compound in water.

Different types of methods have been developed in this context that follow methods developed for learning from a single table of data. This includes methods for learning rules, methods for learning trees, but also distance-based methods. Relational distance-based methods typically define one specific distance measure on a relational data representation mostly taken from relational databases. One of the first approaches developed in this context was by Emde and Wettschereck (1996), a more recent approach is the one of Woznica (2008). While clustering has been mentioned in this context there has been very little work on clustering of structured data represented relationally. Work on clustering structured data is mostly performed by first calculating a distance matrix using a specific distance measure for the data structure considered and then using methods that do not explicitly refer to the distance measure used. For example, this includes methods like hierarchical agglomerative clustering or k-medoids clustering, which take as input a matrix of distances between pairs of objects in a given dataset. There has also been little work on considering explicitly different distance measures. As mentioned above, approaches such as those by Woznica (2008) and Emde and Wettschereck (1996) consider fixed distance measures and do not exploit a variety of distance measures.

A second class of methods are methods for predicting structured outputs. Here, the input is a vector of attribute values of continuous or discrete variables, therefore primitive datatypes, but the output can be a data structure consisting of type constructor and sub-components. The task of structured output prediction has been recently receiving increased attention: A reference for this subject is the book by Bakir et al. (2007). Depending on the data structure of the target, we distinguish different tasks of structured output prediction. For example, when we have a vector of discrete or continuous variables, we are dealing with the task of multi-target classification or multi-target regression. If we are dealing with sets of discrete values, we are dealing with the task of multi-label classification. If we are dealing with sets of discrete values organized in a hierarchy, we are dealing with the task of hierarchical multi-label classification.

One of the recently developed approaches for structured output prediction is situated within the predictive clustering paradigm. Methods for learning predictive clustering trees and ensembles of such trees have been developed which can handle many of the different structured output tasks. These include multi-target classification and regression, multi-label classification and hierarchical multi-label classification (Kocev et al., 2013; Struyf & Džeroski, 2006).

The main hypothesis of the thesis is that it is possible to develop distance-based learning methods for arbitrarily structured data. This can be achieved by extending existing or developing new methods from the areas of instance-based learning, clustering and predictive clustering. More specifically, we hypothesize that it is possible to develop methods for distance-based learning which can handle:

- arbitrary structures on the input side as well as the output side of instance-based learning
- clustering methods for arbitrarily structured types of data in the context of distance-based learning
- methods for handling arbitrarily structured target datatypes in the context of predictive clustering.

The goals of the thesis are closely related to the hypothesis of the thesis. More specifically, we aim to extend the methods for distance-based learning both in the instance-based learning and the clustering paradigm to deal with arbitrarily structured datatypes. This also includes the possibility to use different distances on structured data. The goals will be thus to:

a. Develop a framework for distance-based learning including instance-based learning and clustering for arbitrarily structured datatypes; and

b. Extend predictive clustering methods to deal with arbitrarily structured datatypes as targets.

Besides, we will also aim to demonstrate the utility of the developed methods on practically relevant problems from the areas of environmental and life sciences.

## 1.4 Scientific contributions of the thesis

In this thesis, we explore the two paradigms of instance-based learning and predictive clustering and propose several approaches within each of the two paradigms. We implement these approaches in appropriate software environments and illustrate their use on practically relevant problems. The scientific contributions of the thesis can be summarized as follows:

- *A generic framework and a software environment for instance-based learning from structured data.* The framework implements generic algorithms for both instance-based prediction and clustering, with arbitrarily structured datatypes on the input and the output side and arbitrary distances on such datatypes. The generalized $k$-NN and $k$-medoids algorithm implemented here work for arbitrary types of structured data.

- *Predictive clustering trees for arbitrary types of structured targets.* We have implemented an extension of the predictive clustering framework which generalizes the basic predictive clustering approach to use different distances and different structured datatypes as outputs. These include types with type constructors other than tuple (i.e., set and sequence) applied to primitive datatypes, e.g., sets of discrete/nominal values or sequences/time series of real values. They also include multi-layer datatypes, where type constructors are recursively applied to structured datatypes (e.g., sets of tuples, tuples of time series).

- *Applications of predictive clustering trees to practically relevant problems.* The predictive clustering tree approaches for different types of structured outputs are applied to a number of practically relevant problems. Two of these concern environmental sciences and in particular Slovenian forests, i.e., estimating the state of the forests from remotely sensed data and clustering profiles of forest growth stock. One comes from the life sciences and deals with finding explained groups of genes in yeast with similar time course profiles of gene expression under different stressful conditions.

## 1.5 Organization of the thesis

This chapter gives an introduction to distance-based learning and the context of the thesis. It outlines the motivation and the working hypothesis, and briefly explains the scientific contributions of the thesis. The remainder of the thesis is organized as follows.

Chapter 2 describes structured data and distances on structured data. Structured datatypes are composed of primitive datatypes, such as numeric and discrete, by the recursive applications of type constructors such as tuple, set and sequence. Distances on structured data are composed by aggregating the distances on the component datatypes. The sub-components of the two data items need to be matched in an appropriate manner which depends on the type constructor used.

Chapter 3 describes the first contribution of this thesis, namely, a framework and a software environment for instance-based learning for structured data. The approaches proposed and implemented here rely exclusively on the use of distances: They fall into the category of lazy learning approaches and do not create an explicit understandable model. Rather, they store the training instances or partition them into clusters.

A distinguishing feature of the proposed approaches is that they can deal with arbitrary structured datatypes. These can be defined as discussed in the previous chapter. For both nearest neighbor prediction and clustering using hierarchical agglomerative clustering or k-means/k-medoids, distances on the structured datatypes, as described in the previous chapter, can be used.

Chapter 4 describes the evaluation of the framework and distances introduced in Chapter 3 and Chapter 2 respectively. First, we evaluate the proposed framework on several benchmark datasets for the task of classification of structured data. We then present applications to several prediction tasks, starting with classification problems and continuing with problems that have structured data on the output side. At the end, we illustrate the use of our framework for clustering multi-layer structured data.

Chapter 5 describes the second main contribution of this thesis, namely, approaches for structured output prediction via predictive clustering. Predictive clustering, by design, is capable of predicting structured outputs. However, up to the point where our work started, predictive clustering had dealt mostly with tuples of discrete and continuous variables as targets. We extend predictive clustering approach to deal with targets composed by the use of the other type constructor such as sequence and set. First, we consider sets of discrete values and sequences of real values (i.e., time series) as targets. Next, we consider so-called multi-layer datatypes, where a type constructor is applied to structured datatypes, e.g., a tuple of time series.

Chapters 6 to 9 contain five papers, four already published and one in the process of publication, where predictive clustering methods for different types of structured outputs and their applications to practically relevant problems are described.

Chapter 6 describes the use of predictive clustering trees and ensembles thereof to the problem of estimating the state of the forest in Slovenia from remotely sensed data, i.e., satellite images. In particular, we addressed the problem of estimating vegetation height and canopy cover from satellite images. The structured output prediction task at hand is predicting a tuple of real valued variables.

Chapter 7 treats the task of predicting a set of discrete values also known as multi-label classification. It performs a comparison of the use of different distance measures on the target space. The distance measures on sets considered in this work include: the Euclidean distance, the Hamming distance, the Jaccard distance and the Matching distance.

Chapter 8 addresses the tasks of predicting time series with predictive clustering trees. The datatype of the target is composed by applying the type constructor sequence to real values. The chapter contains two papers. The first paper describes the implementation of the algorithm for prediction of time series within the CLUS system and the modifications of CLUS required for this. In particular, the use of sum of squared pairwise distances to calculate variance is described here for the first time. The proposed approach is evaluated on the task on analyzing gene expression time series and is compared with other approaches such as hierarchical agglomerative clustering.

The second paper in Chapter 8 focuses on the use of predictive clustering trees for predicting short time series to the problem of finding explained groups of time course gene expression profiles. The case study used is finding groups of genes that have a coherent response to different stress conditions as observed in yeast. The predictive clustering trees learned from the data are examined in more detail and their biological plausibility is discussed.

Chapter 9 describes the extension of predictive clustering to multi-layered datatypes as targets. The particular type of structured data treated here is multi-dimensional time series, which can be viewed as a tuple of time series. The proposed approach is applied to the task of modelling forest growing stock in Slovenian publicly owned forests.

Finally, Chapter 10 concludes this thesis. It first summarizes the scientific contributions of the thesis. It then outlines a number of directions for further work.

# Chapter 2

# Distances for Structured Data

In the previous chapter, we established that distances are a crucial part of distance-based methods. Furthermore, the variety of possible applications of distance-based methods requires different definitions of distance functions, so they can perform optimally. To this end, the distances need to be tailored specifically for each application separately. Moreover, the distance-based methods should be flexible enough to facilitate inclusion of background knowledge in the definition of the distance, as provided by domain experts.

The definition of a specific distance for a given application strongly requires a proper data representation and datatype definition. Adjusting the data representation for a given application can lead to learning a better and more accurate model. Considering this, in this chapter, we first discuss different datatypes and then provide definitions of different distances.

In general, we can distinguish between two datatypes: primitive and structured datatypes. A primitive datatype has value space defined either axiomatically or by enumeration (Panov, 2012), e.g., numeric or discrete datatype. Structured datatypes are composed of primitive datatypes by recursive application of type constructors such as tuple, set and sequence.

Consequently, we consider distances on primitive datatypes and distances on structured datatypes. Distances on primitive datatypes can be calculated relatively easy, since there exists a plethora of different distances (e.g., Euclidean distance for numeric datatypes). Distances on a structured datatype are composed by aggregating the distances on the component datatypes. The sub-components of the two data items compared need to be matched in an appropriate manner, which depends on the type constructor used.

## 2.1 Datatypes

In this section, we present the data representation and datatypes, which are the main ingredients for the definition of a distance. First, we discuss the primitive datatypes. Next, we introduce the type constructors that can be used together with the primitive datatypes to construct structured datatypes. Finally, we present representations of real-life examples with structured data.

A datatype is a data representational model denoting a type of data (Panov, 2012). The type of data is defined with its set of distinct values that it can take, the properties of those values, and the operations that can be performed on those values. In other words, a datatype (in computing terms) is a set of data values having predefined characteristics. When it comes to statistics and data mining, the selection of algorithms to be applied on specific data is constrained by the type of data. A basic understanding about the datatypes is helpful for selecting the optimal algorithms.

As stated earlier, in this dissertation, we distinguish between two different kinds of datatypes:

- Primitive datatypes and
- Structured datatypes

### 2.1.1 Primitive datatypes

Primitive datatypes are the basic building blocks of any data representation framework. In this sense, a data representation framework can be a part of a programming language, or it can be a specific standalone framework. Each framework and each programming language define what the primitive datatypes are within the given context. The definitions of the primitive datatypes depend on how they are used, which operations are defined on them and so on.

From a machine learning and data mining point of view, there are, generally speaking, two major types of data:

- Qualitative (i.e., nominal) data, and
- Quantitative (i.e., numeric) data.

The term *qualitative data* is often used interchangeably with the terms *categorical* and *nominal data*. Qualitative data is a categorical measurement that is not expressed with numbers, but instead with a natural language description: It takes a value from a predifined set of discrete values. While qualitative values like *low*, *medium*, and *high* can be ordered, we will treat such cases as quantitative/ordinal (see below), and use the term qualitative data in the sense of nominal data. For such data, there is no natural sense of ordering, thus qualitative data can be referred to as non-ordinal data.

Examples of qualitative data are the color of a ball, or the gender of a person. The qualitative data could be represented with numbers, and will appear numeric, but in this case the numbers are meaningless and comparison or mathematical operations are meaningless as well. For example, gender could be coded as `male=1` and `female=2` and performing any kind of mathematical operation on the value `1` and `2` is meaningless.

*Quantitative data* is often used interchangeably with *numeric data* in the literature. Quantitive data are data that are represented (i.e., ordered) on a numeric scale. Quantitative datatypes could be continuous or discrete. There are three different scales used for this datatype: ordinal, interval and ratio scale (Stevens, 1946). For the interval and ratio scale, mathematical operations are defined, while for the ordinal scale only the ordering (greater/less than) is defined. Because of this, ordinal data are sometimes referred to as quantitative data and sometimes as a mix between qualitative and quantitative data, having properties of both types.

All in all, in this dissertation, we define primitive datatypes as datatypes that have no structure and cannot contain another datatype within themselves (i.e., a primitive datatype is conceptually atomic). The treatment of the nominal and the numeric depends on the specific distance measure that is selected for the corresponding datatype.

### 2.1.2 Structured datatypes

Structured datatypes are composed of primitive datatypes (also called component datatypes in this context) by the recursive application of type constructors. Structured datatypes differ from one another by the relationships among the component datatypes, the relation between each component and the structured datatype; and the sets of characterizing operations. The properties specific to a structured datatype are related to the properties of the type constructor and the properties of the component datatypes. In this dissertation, we define three type constructors used to construct structured datatypes:

- Tuple,
- Set, and
- Sequence.

The type constructors listed above are widely known and well defined in mathematics and computer science. Nonetheless, we briefly describe each of these and emphasize their characteristics that are relevant for the framework proposed in this dissertation.

*Tuple* is an ordered list of elements. Its main properties include:

- Elements are ordered (like sequences, but unlike sets),
- It can contain elements of different types (unlike sequences and sets).

*Set* is a collection of distinct, i.e., unique elements. In this dissertation, we will use the term set also for multi-sets: sets that can contain same elements multiple times.

*Sequence* is an ordered list of elements, where the elements are from the same type. Unlike the *tuple* datatype, in *sequence* all elements must be of the same type. When observing the sequence only as a data representation (the syntactic part of it), it could be treated as a special case of a *Tuple*. However, when using the sequences in machine learning and data mining, the focus is not on the data representation only, but also on many other aspects of the sequence, such as the dynamics of the sequence and the order of the elements in the sequence. Especially important in this context is the latter: the ordering of the elements in sequences can not be changed, while the ordering of the elements in tuples can be changed (i.e., in sequences there can be causal relationships between the elements, while in tuples there are no such relationships). Consequently, different distances for tuples and sequences should be used.

We also discuss the datatype *Timeseries*, since data of this type are more and more abundant. *Timeseries* is not defined as a separate type constructor in our framework, but it is rather defined as a special case of the *sequence* datatype. It represents data that are typically measured at successive time points. The time points can be spaced at uniform time intervals or at varying (arbitrary) time intervals. Time series data have a natural temporal ordering, and this makes time series analysis distinct from other data analysis problems. While the *Timeseries* datatype can be treated as a sequence of reals, because of its wide-spread usage, we implemented it as a special datatype and it can be seen as a built-in structured datatype.

We would like to point out that the component datatypes (the 'inner' datatypes of a structured datatype) can be also structured. In this way, we are able to construct even more complex structured datatypes called multi-layered structured datatypes. For example, the elements of a *tuple* can be of the datatype *sequence*.

Since we have defined primitive datatypes and type constructors, we can now define structured datatypes. To illustrate the structured datatype construction, we give the following example. Consider the data from the yeast genes stress response experiments presented by Gasch et al. (2000). They contain time series of expression levels of yeast (*Saccharomyces cerevisiae*) genes under 12 diverse environmental stresses. The gene expression levels of around 5000 genes are measured at different time points using microarrays. Each gene can then be described with 12 time series – one time series per environmental stress, and the functions the gene has. The functions of a gene are chosen from the hierarchically organized Gene Ontology (GO) catalogue (Ashburner et al., 2000) and are thus sub-hierarchies of the GO.

Let us translate the data representation of the domain outlined above using the formalism outlined here. First, the different time series form the (multi-layered) structured type $Tuple[TimeSeries_1, TimeSeries_2, ..., TimeSeries_{12}]$. Second, the gene functions are organized hierarchically and can be represented as a directed acyclic graph (i.e., a gene function can have more than one parent function). The Gene Ontology catalogue of functions consists of three hierarchies: biological process, molecular function and cellular component.

Each of the hierarchies can be represented as a tuple of binary variables, which correspond to individual gene functions (i.e., either a gene has a given function or it does not), if we traverse the hierarchy and visit each function in a depth-first manner. These variables have some additional properties (e.g., child-parent relationships due to the hierarchy) that can be considered when designing distances. Consequently, the gene function annotation part can be represented as a structured (multi-layerd) datatype $Tuple[Tuple[binary], Tuple[binary], Tuple[binary]]$. Finally, each data example (i.e., each gene) can be represented as the following multi-layered structured datatype:

$$Tuple[Tuple[binary], Tuple[binary], Tuple[binary], TSeries_1, TSeries_2, ...TSeries_{12}] \quad (2.1)$$

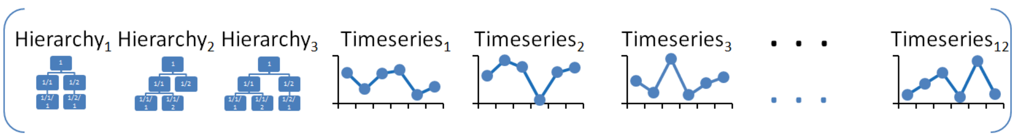


Figure 2.1: Graphical representation of an example of the structured datatype (tuple of hierarchies, timeseries).

Using the multi-layered structured datatype constructors, we have the ability to define very complex data structures, and thus more accurately represent many real-life problems from biology, medicine, enterprise data, and so on. More examples of structured datatypes, adequate distances for them and corresponding data mining method extensions will be presented in details in the following chapters.

## 2.2 Distances

In this section, we first review the basic distances on primitive datatypes. Next, we present distances for the structured datatype constructors (set, tuple and sequence). Finally, we give an algorithm for calculating the distance between two structured objects.

### 2.2.1 Distances on primitive datatypes

In this section, we present the distances for the primitive datatypes. More specifically, we first define the basic distance for the *numeric* datatype and then for the *nominal* datatype.

Let $x$, $y$ be objects of numeric datatype. The distance between objects of numeric datatype we used in this dissertation is defined as follows.

**Definition 2.1 (Distance between numerical values).** The distance measure between two numerical values $x$, $y$ is defined as the absolute value of their difference

$$d(x, y) = |x - y| \tag{2.2}$$

The distance can be standardized by dividing it with $|b-a|$, where $(a, b)$ is the range of possible values for $x$ and $y$.

Next, we give the distance for the nominal datatype. Let $x$, $y$ be now objects of nominal datatype. The distance between objects of nominal datatype we used in this dissertation is defined as follows.

**Definition 2.2 (Distance between nominal values).** The distance measure between two nominal values $x$, $y$ is defined as

$$d(x, y) = \delta(x, y) = \begin{cases} 0, & \text{if } x = y \\ 1, & \text{otherwise} \end{cases} \tag{2.3}$$

### 2.2.2 Distances on structured datatypes

In the previous section, we defined the distances for the primitive datatypes: numeric and nominal. In this section, we first define distances for the various type constructors: tuples, sets and sequences. For the sequences type constructors, in addition to the general definitions for the type constructor, we also define specific distances for time series. We then discuss the construction of distances for multi-layered datatypes.

#### 2.2.2.1 Distances on tuples

*Tuple* of objects is the most widely used type constructor, hence, distances over a tuple of objects are well studied. The most famous distance on tuples is the *Euclidean distance*, which is a special case of the *Minkowski distance*. We therefore first define the Minkowski distance, and we then discuss several well studied instantiations of the main parameter in this distance. These instantiations yield different behaviours of the Minkowski distance and are known under different names.

**Definition 2.3 (Minkowski distance between tuples).** Let $X$ and $Y$ be tuples represented as $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_n]$, respectively. We denote then the distances defined for each of the components of the tuples as $d_1, d_2, \ldots, d_n$. The Minkowski distance measure between two tuples $X$ and $Y$ is defined as

$$d(X,Y) = \left( \sum_{i=1}^{n} d_i(x_i, y_i)^p \right)^{\frac{1}{p}} \tag{2.4}$$

We next discuss the instantiations of the Minkowski distance for various values of the parameter $p$. If the value of $p$ is set to 1, then the Minkowski distance is known as *Manhattan distance*, or sometimes also called *city* or *taxi* distance. It is calculated as the sum of distances between the tuples' components:

$$d(X,Y) = \sum_{i=1}^{n} d_i(x_i, y_i) \tag{2.5}$$

If the value of $p$ is set to 2, then the Minkowski distance becomes the well-known *Euclidean distance*. The Euclidean distance is calculated as follows:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} d_i(x_i, y_i)^2} \tag{2.6}$$

In the limiting case of $p$ going towards positive infinity, we obtain the *Chebyshev distance*. The Chebyshev distance is calculated as follows:

$$d(X,Y) = \lim_{p \to \infty} \left( \sum_{i=1}^{n} d_i(x_i, y_i)^p \right)^{\frac{1}{p}} = \max_{i=1}^{n} d_i(x_i, y_i) \tag{2.7}$$

Similarly to Chebyshev distance, we can also instantiate the distance with the value of the parameter $p$ going towards negative infinity. The distance in that case is calculated as follows:

$$d(X,Y) = \lim_{p \to -\infty} \left( \sum_{i=1}^{n} d_i(x_i, y_i)^p \right)^{\frac{1}{p}} = \min_{i=1}^{n} d_i(x_i, y_i) \tag{2.8}$$

#### 2.2.2.2 Distances on sets

Distances on sets have been extensively studied in the past (Hastie et al., 2001). Consequently, a number of different measures have been proposed in the literature for defining distances between sets of objects. In this dissertation, we focus on the following distances: single linkage, complete linkage, average linkage, Hausdorff distance, Jaccard distance, matching distance and greedy matching.

Let $X$ and $Y$ be sets represented as $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$. We then define the variuos distances between the two sets $d(X,Y)$ as follows. The elements of $X$ and $Y$ are of type $T$ on which a distance $d_T$ is defined.

**Definition 2.4 (Single Linkage).** The *Single Linkage* distance is defined as the minimum of all pairwise distances (Hastie et al., 2001) between pairs of elements from the two sets. It can be calculated using the following equation:

$$d(X,Y) = min_{(x,y)\in X\times Y}\{d_T(x,y)\} \tag{2.9}$$

**Definition 2.5 (Complete Linkage).** The *Complete Linkage* distance is defined as the maximum distance of all pairwise distances (Hastie et al., 2001) between pairs of elements from the two sets. It can be calculated using the following equation:

$$d(X,Y) = max_{(x,y)\in X\times Y}\{d_T(x,y)\} \tag{2.10}$$

**Definition 2.6 (Average Linkage).** The *Average Linkage* distance is defined as the sum of all pairwise distances (Hastie et al., 2001) between pairs of elements from the two sets. It can be calculated using the following equation:

$$d(X,Y) = \sum_{(x,y)\in X\times Y} d_T(x,y) \tag{2.11}$$

The average linkage can be standardized by dividing the distance with the product $|X| \cdot |Y|$, where $|X|$ is the number of elements in $X$ and $|Y|$ is the number of elements in $Y$.

**Definition 2.7 (Hausdorff distance).** The *Hausdorff* distance is defined as follows:

$$d(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d_T(x,y), \sup_{y\in Y}\inf_{x\in X} d_T(x,y)\} \tag{2.12}$$

When working with finite sets, as in this dissertation, *s*upremum (sup) is equal to the *m*aximum (max) and *i*nfimum (inf) is equal to the *m*inimum (min).

The meaning and the calculation of the *Hausdorff* distance can be explained in plain words as follows. The two sets are close according to the *Hausdorff* distance as much as each point of either of the two sets is close to a point of the other set. In algorithmic terminology, the Hausdorff distance is calculated as follows: For each object from the first set, find the closest object from the second. Next, store the object that has the largest distance to its closest neighbor (i.e., the object that has 'the most distant nearest neighbor'). The same procedure is then repeated for the other set. Finally, the larger distance of these two distances is the *Hausdorff* distance.

In the following definition, we define the Jaccard distance which is based on the Jaccard index (Jaccard, 1901). In literature, it is sometimes also called the Tanimoto distance. It is defined on sets of discrete values.

**Definition 2.8 (Jaccard distance).**

$$d(X,Y) = 1 - J(X,Y) = \frac{|X\cup Y| - |X\cap Y|}{|X\cup Y|} \tag{2.13}$$

where the Jaccard index, also known as the Jaccard similarity coefficient, is defined as:

$$J(X,Y) = \frac{|X\cap Y|}{|X\cup Y|} \tag{2.14}$$

To be able to define the *Matching* distance, first we need to define the notion of *matching* between two sets.

**Definition 2.9 (Matching).** A relation $r \subseteq X \times Y$ between two finite sets $X$ and $Y$ is a **matching** if and only if:

$$\forall (a,b),(c,d) \in r : (a = c \Leftrightarrow b = d) \tag{2.15}$$

In other words, in matching each element of $X$ is associated with at most one element of $Y$ and vice versa. A matching $m$ between $X$ and $Y$ is *maximal*, if there is no matching $m'$ between $X$ and $Y$ such that $m \nsubseteq m'$. A *perfect* matching is a *maximal* matching between two sets of equal cardinality (Ramon & Bruynooghe, 2001).

Next, for each possible matching $m(X,Y)$, we define the following distance:

$$d(m,X,Y) = \sum_{(x,y)\in m} d_T(x,y) + \frac{||X| - |Y||}{2} \cdot M, \tag{2.16}$$

where $|X|$ and $|Y|$ are the cardinalities (number of elements) of $X$ and $Y$ respectively, and $M$ is a constant, defined as $M \geq max_{(x,y)\in X\times Y} d(x,y)$.

**Definition 2.10 (Matching distance).** Finally, the *Matching* distance is defined as follows:

$$d(X,Y) = min_{(r\in m(X,Y))} d(r,X,Y). \tag{2.17}$$

An algorithm for computing the matching distance is given by Ramon and Bruynooghe (2001) and is based on flow networks and the minimum weight maximum flow problem. In this dissertation, we have implemented the Matching distance following the instructions by Ramon and Bruynooghe (2001).

Finally, for sets, we propose a new distance which is a greedy approximation of the *Matching* distance described above. We call it *Greedy sum of Minimums* or *Greedy Matching*.

**Definition 2.11 (GM distance).** We define the *Greedy Matching* distance as follows:

$$d(X,Y) = min_{(x,y)\in X\times Y} d_T(x,y) + d(X \setminus \{x'\}, Y \setminus \{y'\}) + \frac{||X| - |Y||}{2} \cdot M, \tag{2.18}$$

where

$$(x',y') = argmin_{(x,y)\in X\times Y} d_T(x,y), \tag{2.19}$$

and $M$ is a constant, defined as $M \geq max_{(x,y)\in X\times Y} d(x,y)$

The computational complexity of the *GM* distance is $O(N^2 \cdot \log N)$, where $N$ is the number of elements of each set, which is lower than the computational complexity of the *Matching* distance, calculated by solving a minimum cost problem (Ramon & Bruynooghe, 2001). There are several known algorithms to solve this problem with complexities $O(N^4\sqrt{N})$, $O(N^4 \log N)$, $O(N^3 \log N)$ and the state of the art algorithm called double-scaling has the lowest complexity of $O(N^3)$ (Ahuja et al., 1992). The *Greedy Matching* distance can be considered as a very good and inexpensive, easy to implement, approximation of the *Matching* distance.

#### 2.2.2.3 Distances on sequences and time series

In this section, we review several distances on sequences. We start by describing the Levenshtein distance for sequences (Levenshtein, 1966) and then present several distances defined for the special type of sequences – time series. Originally this distance was defined as the edit distance between strings, i.e., sequences of characters, measuring the minimum number of operations (such as insert/remove a character or replace a character with another) needed to transform one string into the other.

Let $X$ and $Y$ be sequences defined as: $X = [x_1, x_2, \ldots, x_m]$ and $Y = [y_1, y_2, \ldots, y_n]$, where $x_i$ and $y_i$ are all from the same datatype $T$. The Levenshtein distance between these two sequences is then calculated as follows:

**Definition 2.12 (Levenshtein distance).**

$$d(X,Y) = lev_{X,Y}(|X|, |Y|) \tag{2.20}$$

where $lev_{X,Y}$ is defined as follows:

$$lev_{x,y}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0, \\ min \begin{cases} lev_{x,y}(i-1,j)+1 \\ lev_{x,y}(i,j-1)+1 \\ lev_{x,y}(i-1,j-1)+d_T(x_i,y_j) \end{cases} & \text{otherwise.} \end{cases} \tag{2.21}$$

The implementation of the Levenshtein distance in this dissertation uses dynamic programming (Bellman, 1954). It can be applied to sequences of any type of objects, whether primitive or not. For example, one could calculate the Levenstein distance between sequences of nominal values, sequences of real values, as well as sequences of hierarchies or sequences of sets.

In this dissertation, we also consider distances on time series. A time series is a sequence of real numbers, representing the measurements of a real variable at given time intervals. Time series are a specific type of sequences and various distances have been specially developed for solving problems that involve time series. We discuss the following distances for time series: Dynamic Time Warping (DTW) (Sakoe & Chiba, 1978), Qualitative distance (QD) (Todorovski et al., 2002) and Pearson distance (based on the Pearson correlation coefficient).

In addition to these, we have implemented the following distances: *Spearman* distance (based on the Spearman's rank correlation coefficient) and *HSim* distance, based on the *HSim* coefficient (mutual information). Combinations of Spearman and Pearson distances with *HSim* distance, called *SpearmanHsim* and *PearsonHsim* are also implemented. These distances measures are presented in details and evaluated for gene expression data in the work by Li and Z.-Z. Wang (2009). For a detailed explanation of these distances please refer to Chapter 8.

The Dynamic Time Warping (DTW) distance (Sakoe & Chiba, 1978) can capture a non-linear distortion along the time axis. It accomplishes this by assigning multiple values of one of the time series to a single value of the other. As a result, DTW is suitable if the time series are not properly synchronized, e.g., if one is delayed, or if the two time series are not of the same length (for details and illustration see Chapter 8).

The DTW distance between two time series $X$ and $Y$, $X = [\alpha_1, \alpha_2, \ldots, \alpha_I]$, $Y = [\beta_1, \beta_2, \ldots, \beta_J]$ is based on the notion of warping path between $X$ and $Y$. A warping path is a sequence of grid points $F = f_1, f_2, \ldots, f_K$ on the $I \times J$ plane. The DTW distance can be calculated as follows.

**Definition 2.13 (Dynamic Time Warping distance).** Let the distance between two values $\alpha_{i_k}$ and $\beta_{j_k}$ be

$$d(f_k) = |\alpha_{i_k} - \beta_{j_k}| \tag{2.22}$$

then an evaluation function $\Delta(F)$ is given by

$$\Delta(F) = 1/(I+J) \sum_{k=1}^{K} d(f_k) w_k \tag{2.23}$$

The weights $w_k$ are as follows

$$w_k = (i_k - i_{k-1}) + (j_k - j_{k-1}), i_0 = j_0 = 0 \tag{2.24}$$

The smaller the value of $\Delta(F)$, the more similar $X$ and $Y$ are. In order to prevent excessive distortion, we assume an adjustment window ($|i_k - j_k| \leq r$). $d_{\mathrm{DTW}}(X, Y)$ is the minimum of $\Delta(F)$. $d_{\mathrm{DTW}}$ can be computed with dynamic programming in time $O(IJ)$.

The qualitative distance (QD) (Todorovski et al., 2002) is based on a qualitative comparison of the shape of the time series. Consider two time series $X$ and $Y$, $X = [\alpha_1, \alpha_2, \ldots, \alpha_I]$, $Y = [\beta_1, \beta_2, \ldots, \beta_J]$. Then choose a pair of time points $i$ and $j$ and observe the qualitative change of the value of $X$ and $Y$ at these points. There are three possibilities, for $X$ i.e., increase ($\alpha_i > \alpha_j$), no-change ($\alpha_i \approx \alpha_j$), and decrease ($\alpha_i < \alpha_j$), as well as three possibilities for $Y$. $d_{\mathrm{qual}}$ is obtained by summing the difference in qualitative change observed for $X$ and $Y$ for all pairs of time points.

**Definition 2.14 (Qualitative distance).**

$$QD(X, Y) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \mathit{Diff}(q(X_i, X_j), q(Y_i, Y_j))}{N \cdot (N-1)} \tag{2.25}$$

where $\mathit{Diff}(q_1, q_2)$ is a function that defines the difference between different qualitative changes (see Table 2.1). Roughly speaking, $QD$ counts the number of disagreements in change of $X$ and $Y$.

Table 2.1: The definition of $\mathit{Diff}(q_1, q_2)$ for the Qualitative Distance (QD).

| $\mathit{Diff}(q_1, q_2)$ | increase | no-change | decrease |
|---|---|---|---|
| increase | 0 | 0.5 | 1 |
| no-change | 0.5 | 0 | 0.5 |
| decrease | 1 | 0.5 | 0 |

The Pearson correlation coefficient $r(X, Y)$ between two time series $X$ and $Y$ is calculated as follows:

**Definition 2.15 (Pearson correlation coefficient).**

$$r(X, Y) = \frac{E[(X - E[X]) \cdot (Y - E[Y])]}{E[(X - E[X])^2] \cdot E[(Y - E[Y])^2]} \tag{2.26}$$

where $E[V]$ denotes the expectation (i.e., mean value) of $V$.

$r(X, Y)$ measures the degree of linear dependence between $X$ and $Y$. It has the following intuitive meaning in terms of the shapes of $X$ and $Y$: $r$ close to 1 means that the shapes are similar. If there is a linear relation between $X$ and $Y$ then the time series are identical, but might have a different scale or baseline. $r$ close to -1 means that $X$ and $Y$ have "mirrored" shapes, and $r$ close to 0 means that the shapes are unrelated (and consequently dissimilar). Based on this intuitive interpretation, we can define the distance between two time series as $d_r(X, Y) = \sqrt{0.5 \cdot (1 - r(X, Y))}$. Note that $QD(X, Y)$ and $r(X, Y)$ require that the two time series $X$ and $Y$ are of equal length. DTW allows for sequences of different length to be compared.

### 2.2.3 Distances for multi-layered datatypes

Let us first summarize the discussion on structured datatypes from this chapter. We distinguish between two kinds of datatypes: *primitive* and *structured* datatypes. Primitive datatypes have no structure and do not contain another datatype within. Examples of primitive datatypes include *nominal* and *real*. Structured datatypes, on the other hand, are built from primitive datatypes. To build structured datatypes, we first need to define type constructors. Here, we consider the following type constructors: *tuple* ($Tuple\,(T_1, T_2, ...T_n)$), *set* ($Set\,\{T\}$), and *sequence* ($Sequence\,[T]$).

*Tuple* is a type constructor that can contain any pre-defined number of objects, each from an arbitrary, but fixed datatype (the objects at different positions in the tuple may be of different datatypes). *Set* is a type constructor that can contain any number (not fixed) of objects, all of the same (fixed) type. The type of objects in the set can be either primitive or structured. *Sequence* is a type constructor that is similar to the *Set* type constructor as it can contain an arbitrary number of objects of the same type. However, while the elements of a set are unordered, in a sequence the ordering of the elements is important. *TimeSeries* is a datatype constructed by using the sequence constructor from elements of the underlying type real ($TimeSeries = Sequence[real]$), where the ordering of the elements in the sequence is along the time dimension.

Using the notions on primitive and structured datatypes, we can define multi-layered datatypes, i.e., structured datatypes than can consist of other structured datatypes. For example, we can define multi-layered objects (or datatypes) as complex as $Tuple\,[TimeSeries, Sequence\,[Nominal]\,, Set\,\{TimeSeries\}]$. This object is then a tuple consisting of three elements: *TimeSeries*, *Sequence* and *Set*. Furthermore, the sequence is defined as a sequence of *nominal* and the set is defined as a set of *TimeSeries*.

Defining distances has been studied extensively and is reasonably well understood for primitive datatypes (Džeroski, 2006). The basic idea of a unified approach to distance-based learning from structured data is to derive the key components of data mining algorithms for a complex datatype (built through using type constructors) from information on the structure of that type (what constructors on what simpler datatypes) and the key components for the simpler datatypes. For example, a distance function $d$ on tuples of type $Tuple(T_1, \ldots, T_n)$ can be composed from distance functions $d_i$ on types $T_i$ by adding up the distances for each tuple component, i.e., $d(x, y) = d((x_1, ..., x_n), (y_1, ..., y_n)) = \sum_{i=1}^{n} d_i(x_i, y_i)$.

In general, a distance over a structured datatype can be defined as follows. Recall that an object of a structured datatype is obtained by applying a type constructor (tuple, set, sequence) to some components (arguments of the tuple, elements of the set, ...). Given a pair of such structured objects, we first decompose them into their components. We then construct pairs of components, matching components of the first object with components of the second object. How the components are matched depends on the type constructor: For tuples (as illustrated above) the arguments at the same position are paired, while for sets and sequences, the matching can be more complicated and alternative matching procedures are possible. For the matched component pairs, the distances on the components are calculated (recursively, if the component objects are structured). These distances are finally aggregated into a single value of the overall distance between the two structured objects.

The pseudo-code given in the Algorithm 2.1 illustrates the recursive calculation of the distance. The *calculateDistance* function takes as input two data objects ($O_1$ and $O_2$), the definition of the datatype of the data objects ($DT$), and the distance definition ($DI$). The distance definition in the case of primitive datatypes is the name *distanceID* of the distance, e.g., $\delta$ for the nominal datatype. For structured datatypes, it contains the definitions of the distances on the underlying component datatypes and the definition of the aggregation function ($DI = (Agg, DI_s)$).

The distance function calculation starts by checking whether the input objects are from a primitive datatype. If that holds, then the distance between $O_1$ and $O_2$ can be calculated instantly using the appropriate formula. For example, if the objects are from the datatype *nominal*, then the distance between them can be calculated by using the $\delta$ distance. If the objects $O_1$ and $O_2$ are from a structured or multi-layered datatype, then they are decomposed (using the *Decompose* function) into sets/lists of components $C_1$ and $C_2$ using the datatype definition $DT$. After the decomposition of the two objects, the *Matching* procedure matches the components from the objects and produces pairs of objects $(c_1, c_2)$ of the same type $d_t$, associated with a distance $d_i$ on $d_t$. In the next step, the *calculateDistance* function is called recursively for each of the component pairs $(c_1, c_2)$, given also as arguments the datatype $d_t$ and the associated distance $d_i$. Finally, the distances of the underlying components of the data objects are aggregated bottom-up by using the aggregation functions specified by $Agg$.

The above definition covers all structured data types, i.e., both single-layer and multi-layer structured datatypes. For sets, the three commonly used distances in clustering (single, complete and average linkage) easily fit the given template. Matching returns the Cartesian product of the two sets and the aggregate function is minimum, maximum and average, respectively.

To illustrate the work of the distance calculation algorithm on a multi-layered structured datatype, we present an example concerning the definition and calculation of the distance between two multi-dimensional time series. First, we define the structured datatype as *Tuple* $(T_1, T_2, ...T_n)$, where $T_1, T_2, ...T_n$ are time series. Then, we define the distances that will be used for the components. For example, for the *Tuple* we can use the aggregation function employed within the Euclidean distance ($SQRT(SumSQDist)$), and for the component *TimeSeries* we can use *Euclidean* and *QD* distance measures. We would like to note that our framework allows the use of a different distance for each component time series in the tuple. We will, however, use the same distance for all of the time series in the tuple.

Distances on structured objects define both the matching of the component objects and the aggregation function. For example, the matching function for the

**Algorithm 2.1:** The generic recursive algorithm for calculating the distance between two structured objects

```
**Data**: $O_1, O_2$ - data objects; $DT$ - datatype definition; $DI$ - distance definition
**Result**: d - distance value
1  if $DT$ is a primitive datatype then
2      if $DT = real$ and $DI = abs$ then
3          return $|O_1 - O_2|$;
4      end
5      if $DT = nominal$ and $DI = \delta$ then
6          return $\delta(O_1, O_2)$;
7      end
8  end
9  $DT_s \leftarrow$ Decompose($DT$), $(Agg, DI_s) \leftarrow DI$;
10 $C_1 \leftarrow$ Decompose($O_1, DT$);
11 $C_2 \leftarrow$ Decompose($O_2, DT$);
12 $M \leftarrow$ Matching($C_1, C_2, DT_s, DI_s$),
   $M = \{(c_1, c_2, d_t, d_i) | c_1 \in C_1, c_2 \in C_2, d_t \in DT_s, d_i \in DI_s\}$;
13 for each $(c_1, c_2, d_t, d_i) \in M$ do
14     $d_k \leftarrow$ calculateDistance($c_1, c_2, d_t, d_i$);
15 end
16 $d \leftarrow$ Agg($d_1, d_2, ..., d_{|M|}$);
17 return $d$;
```

Euclidean distance on time series defines that the respective elements of the equal-length time series are matched. The aggregation function for the Euclidean distance is defined as the squared root of the sum of squared distances between the matched pairs of elements. For the QD, the matching function and the aggregation function are explained in detail in Section 2.2.2.3.

Using the pseudo-code from Algorithm 2.1, we illustrate the calculation of distances over tuples of three time series (i.e., three-dimensional time series). We illustrate this on two examples. We first outline the calculation of the Euclidean distance for the multi-dimensional time series datatype. We then illustrate the calculation of the distance between multi-dimensional time series that uses the qualitative distance for the individual time series.

For both cases, the underlying datatype is $DT = Tuple(TimeSeries, TimeSeries, TimeSeries)$ and the distance between two objects (three dimensional tuple of time series) is calculated by calling $d = calculateDistance(O_1, O_2, DT, DI)$. For the first case, the distance is defined as follows: $DI = (SQRT(SumSQDist), [EUC, EUC, EUC])$. For the second case, the distance is defined as $DI = (SQRT(SumSQDist), [QD, QD, QD])$. A step-by-step explanation of the distance calculation is given bellow.

We consider two specific data examples with the datatype definition as given above:

$$O_1 = ([73, 66, 54, 41, 54], [225, 231, 193, 159, 147], [65, 122, 120, 185, 223]), \text{and}$$
$$O_2 = ([76, 97, 81, 73, 66], [165, 208, 143, 158, 162], [53, 102, 87, 117, 181])$$

In the first step, the algorithm will decompose the structured tuples into their underlying components: time series. The algorithm will then match the corresponding time series from the two tuples: the first time series from $O_1$ to the first time series from $O_2$, etc. Next, it continues by calculating the distance between the corresponding time series as follows:

$$\begin{aligned} d_1 &= QD([73, 66, 54, 41, 54], [76, 97, 81, 73, 66]) = 0.35, \\ d_2 &= QD([225, 231, 193, 159, 147], [65, 122, 120, 185, 223]) = 0.85, \\ d_3 &= QD([65, 122, 120, 185, 223], [53, 102, 87, 117, 181]) = 0.05. \end{aligned}$$

In the final step, the algorithm reads the aggregation function from the distance definition ($DI$): in this case, it is the sqare root of the sum of the squared distances ($SQRT(SumSQDist)$). The aggregation function is then applied to the distances on the underlying time series as follows:

$$d = \sqrt{d_1^2 + d_2^2 + d_3^2} = \sqrt{0.35^2 + 0.85^2 + 0.05^2} = 0.9206$$

At the end, the *calculateDistance* will return the value 0.9206 as the distance between the two data examples $O_1$ and $O_2$ given above.

This example of distance calculation emphasizes the flexibility and modularity of our framework. More specifically, it shows that the user can design the distance function based on the problem at hand. In other words, the user can decide to use different distances for the different parts/components of the data objects. Furthermore, the user can add specific distances and then use them to design more complex distance definitions.

To summarize, the calculation of the distance between two structured data objects consists of three major parts: matching function, aggregation function and distances on primitive and structured component datatypes. In the most general case, the matching function can be a simple Cartesian product between the set of components from the first data object and the set of the components from the second data object. However, depending on the specific datatype, one can define an appropriate matching function. For example, if the datatype considered is a tuple then the matching pairs the elements at the same position within the tuples: the $i$-th element of the first tuple is paired with the $i$-th element of the second tuple. Next, there are various possibilities for defining the aggregate function. These variants include simple functions such as *min*, *max*, *avg*, *sum*, as well as complex aggregation function such as constructing a graph or network from the distance matrix between the components and then calculating the network flow. Finally, the recursive definition of the distance, as we define it here, depends on the calculation of the distances (and type of distance applied) between the components of the data objects. For example, consider the data object presented in Fig. 2.1. It consists of tuples of binary variables and time series. The distance functions on these different structured datatypes would return values on different scales, thus performing simple average would prefer a given part of the data object over the other parts. To alleviate this issue, we propose to use standardization of the values when calculating the distances on component datatypes. This sets all of the values to the same range and thus makes the aggregation more meaningful. All in all, the calculation of the distance as proposed in this dissertation is very flexible, modular and enables the users to design and apply distances tailored for the specific domain.

# Chapter 3

# Instance-Based Learning for Structured Data

In this chapter, we discuss the use of the distance functions from the previous chapter in distance-based algorithms for structured data. We begin by outlining the general concepts behind distance-based algorithms in data mining. We then present the basic distance-based predictive algorithms: the nearest and $k$-nearest neighbors algorithms. Next, we give the basic distance-based clustering algorithms: hierarchical clustering and $k$-means/medoids clustering. Furthermore, we discuss some examples of applications of the distance-based predictive and clustering methods for structured data. Finally, we outline the main concepts and principles used in the implementation of these methods within a unifying framework.

## 3.1 General distance-based learning algorithms for structured data

Mining various structured types of data is one of the most challenging tasks in data mining (Dietterich et al., 2008; Kocev, 2011; Kocev et al., 2013). Moreover, solving the task in a unified way is a significant step towards the creation of a general data mining framework (Džeroski, 2006). In the previous two chapters, we described the representation of various structured datatypes and we defined various distances for structured datatypes. Having defined that, we next explain the unifying environment for distance-based learning on structured data.

The main component of distance-based algorithms, as suggested by their naming, are the distances. Distance-based algorithms are typically divided into two groups of methods: supervised and unsupervised (Langley, 1996; Hastie et al., 2001). First, we focus on the former group of methods.

Supervised distance-based algorithms in general can be defined as follows: Given is a data-set of examples $D = \{(x_i, y_i)|1 \leq i \leq N\}$, where $x_i$ are the inputs of datatype $T_I$, and the $y_i$ are the outputs of the datatype $T_O$. $T_I$ and $T_O$ can be primitive or structured datatypes. For a given unseen example $u$, we calculate the distances $d(u, x_i)$ from the unseen example $u$ to all of the known examples $x_i$.

Now, the output $y$ for $u$ can be calculated as a function of the distances between $u$ and all the known examples $x_i$ and their output values $y_i$: $output(u) = F(d(u, x_1), d(u, x_2), \ldots, d(u, x_n), y_1, y_2, \ldots, y_n)$, where $F$ is defined by the learning algorithm. For the nearest neighbor (NN) algorithm, the smallest $d(u, x_i)$ is found and the corresponding $y_i$ is the output. Other algorithms, like the $k$NN algorithm, take more than one $x_i$ into account, as well as the distances $d(u, x_i)$ when calculating the value of $y$, as described in more detail below.

Given that we defined different distances for different types of data in the previous chapter, we have thus defined all of the inputs needed for the learning algorithm. We address the handling of the specific distances and structured datatypes in the next sections.

Next, we focus on the unsupervised distance-based algorithms. A clustering is a set of clusters, while each cluster is a set of elements/units. In the most general case, clusters are not necessarily pairwise disjoint (i.e., non-overlapping): However, it is a common practice to assume that they are. In other words, given a dataset, i.e., a set of elements/examples/units, each partitioning of this dataset is called a clustering. In clustering, the goal is to minimize some criterion function that leads to more compact clusters, i.e., puts the objects that are more similar according to the criterion function in the same cluster. Hence, a clustering algorithm is an algorithm that optimizes/minimizes the value of such a criterion function. The criterion function, in a very simple way, can be defined as some aggregate (e.g., *sum*, *max*, or *weighted sum*) of the per-cluster errors. The per-cluster error is very often some measure of cluster impurity (e.g., average of pairwise distances, diameter of the cluster, etc), and it is calculated from the distances between the elements of the cluster, or between the elements and the representative of the cluster. The representative of a cluster can be either an element which is closest to the center, or the center of gravity of the cluster. From this, it follows that once we have distances on structured data defined, the distance-based algorithms can be easily adapted.

Finally, we enumerate the distances implemented within the proposed framework.

- *Tuples*: Minkowski distance (includes Euclidean, Manahattan) and Chebyshev distance
- *Sets*: Single linkage, Complete linkage, Average linkage, Hausdorff distance, Jaccard distance, Matching distance, and greedy mathcing (GM) distance
- *Sequences*: Levenshtein distance (edit distance)
- *Time series*: Dynamic time warping distance, Qualitative distance, Pearson distance, Spearman distance, SpearmanHsim distance, HSim distance and PearsonHsim distance

## 3.2 Nearest neighbor prediction with NN and $k$-NN

The nearest neighbor algorithm (Altman, 1992) is one of the simplest, best known, and most widely used machine learning algorithms. The nearest neighbor algorithm is also known as memory-based reasoning algorithm, or instance-based learning algorithm, or lazy learning algorithm. All of these names come from the inner mechanisms of the algorithm. When using the nearest neighbor algorithm, no model is built, but the entire training set is stored in the memory. Both for classification and regression, to predict the correct value of the target, the distances are computed between the example and each element of the training set. The target of the new example is then set to be equal to the target value of the closest example (i.e., the most similar example) from the training set.

More generally, the $k$-nearest neighbors can be computed, and then the target value of the new, unseen, query example is calculated from the target values of the neighbors by averaging in case of regression, or by majority voting in the case of classification. Moreover, instead of simple majority voting for classification and averaging for regression, weighed voting/averaging can be used. The votes of the $k$-nearest neighbors are typically weighted inversely proportionally to their distances to the query/unseen example.

In this section, we discuss the extension of the nearest neighbor and the $k$-nearest neighbor algorithms for handling structured data. The $k$-nearest neighbor ($k$-NN) algorithm is outlined in Algorithm 3.1. The nearest neighbor ($NN$) algorithm is obtained as a special case when $k = 1$. The dataset can be represented as a set of instances, where each instance has two structured objects: input and output. We denote the datatype of the input object as $T_I$ and the datatype of the output object as $T_O$. Having defined distances on the input space (type $T_I$, discussed in the previous chapter), the algorithm is able to calculate the distance between two samples from the dataset. Given an unseen sample, it will thus be able to find its $k$-nearest neighbors in the training set.

The next step is to make a prediction for the value of the target variable for the unseen example. To this end, the prototype of the output values of the $k$ nearest neighbors is calculated. In case we have a primitive datatype as output, for classification majority voting can be used, and for regression, averaging can be used, as discussed earlier in this section. Otherwise, when the output is of a structured datatype, majority voting can be used as well, by representing each different value of the output datatype as a separate class. However, in many cases this would be impossible since each of the objects from the output space can be different from the other output objects. Consequently, this will lead to very unstable and incorrect predictions, since it is trying to match the complete output structure and all of the objects will have a vote of 1, therefore, tie-breaking will need to be used very often. To alleviate this problem, in the case of structured data, we calculate the prototype from the $k$ nearest neighbors as follows. We consider the target values of the $k$ nearest neighbors as a set of $k$ data objects of type $T_O$. The prototype can be thus defined as an element from the set with a minimal sum of distances to the other elements. Note that the majority voting in classification is a special case of calculating the prototype of the set as described here, i.e., the prototype of the set of elements where the distance is the $\delta$ or 0/1 distance (defined by Equation 2.3).

The prototype is equal to the average, or the center of the set of the $k$ nearest neighbor targets in the special case of using the $k$-nearest neighbor algorithm for multi-target regression problems. We can define the predicted value of an unseen example as the average of the neighbors, in the case, where arithmetical and mathematical operations of sum and multiplying by scalar are defined for the type $T_O$ (i.e., $T_O$ is a vector space). Given these definitions, if the $T_O$ is a primitive datatype – a real number, then this corresponds to the special case of regression. Furthermore, in both scenarios, i.e., when using a prototype or using a centroid of the set (average of the elements), we can apply the weighing schemes discussed earlier in this section.

## 3.3 Distance-based clustering

Clustering is concerned with grouping objects into groups of similar objects (Kaufman & Rousseeuw, 1990). It has strong roots in the statistical community, but is

**Algorithm 3.1:** Pseudo-code for the $k$-NN algorithm

```
Data: $D$ - data set, $e$ - sample (instance), $DT$ - datatype and distance definition,
      $Agg$ - aggregation function definition, $k$ - number of nearest neighbors
Result: $knn$ - (array/list of) the $k$ nearest neighbors of $E$ from $D$
1 for each instance $I \in D$ do
2     $d_i \leftarrow$ calculateDistance($I, e, DT, Agg$);
3     $knn \leftarrow$ insertIntoSorted($knn, k, (I, d_i)$);
4 end
5 $target \leftarrow prototype(knn)$;
6 return $target$;
```

also commonly encountered in data mining. It is used in many fields including image analysis, information retrieval, and bioinformatics. The two most widely used types of clustering are distance-based and density-based clustering. In this dissertation, we focus only on the former type of clustering – distance-based clustering.

In distance-based clustering, the algorithms use the distances between the objects that have to be clustered and optimize usually two criteria: they minimize the distance between objects within a cluster (also called intra-cluster distance or intra-cluster similarity), and, maximize the distance between objects from different clusters (also called inter-cluster distance or inter-cluster similarity). In other words, the goal is to minimize the intra-cluster distance and maximize the inter-cluster distance. Hence, the most important component is the distance measure between two objects. Using the definition of a distance between arbitrarily structured objects from the previous chapter (Algorithm 2.1), we can adapt any distance-based clustering algorithm for the task of mining/clustering structured data. In the following sections, we will show the adaptation for the task of mining structured data for the two most widely used distance-based clustering algorithms: hierarchical agglomerative clustering and $k$-means/medoids clustering.

### 3.3.1 Hierarchical agglomerative clustering

Hierarchical agglomerative clustering (HAC) is based on joining together the most similar objects or, in a more general case, sets of objects (clusters). Given a dataset with $N$ objects, it starts by constructing the distance matrix ($N \times N$) between the objects. It then assigns each object to its own cluster. Next, it merges the two closest clusters into one cluster. The process continues until some constraint is reached (like the desired number of clusters, or the maximum distances at which two clusters are allowed to be merged) or it continues until all of the clusters are merged, and then the user explores the hierarchy of clusters and manually selects the appropriate number of clusters. Algorithm 3.2 presents the pseudo-code for the HAC algorithm.

Recall that the algorithm searches for the two closest clusters. When clusters are sets with one object (as it is at the start of the algorithm), it is very clear that the distance between two clusters is actually the distance between the objects comprising the respective clusters. However, once the clusters become sets of several objects, there is a possibility of defining various distances between two clusters: single linkage, complete linkage and average linkage. These three definitions of distances between clusters use the pair-wise distances between objects in the two clusters (the first object of the pair is an object from the first cluster, while the second object in the pair is an object from the second cluster). Single linkage considers the minimal pair-wise distance as the distance between two clusters, while complete linkage considers the maximal pair-wise distance as the distance between two clusters. Average linkage (also known as Unweighted Pair Group Method with Arithmetic Mean) considers the average pair-wise distance as the distance between two clusters. These definitions for distances between clusters are well-known and widely used in hierarchical clustering (Székely & Rizzo, 2005).

**Algorithm 3.2:** Pseudo-code for the HAC algorithm

```
Data: $D$ - data set, $DT$ - datatype and distance definition, $Agg$ - aggregation
      function definition, $k$ - number of clusters
Result: $clusters$ - (array/list of) the clusters
1  for each instance $I_i \in D$ do
2      assign $I_i$ to the cluster $clusters[i]$
3  end
4  for each cluster $c_i \in clusters$ do
5      for each cluster $c_j \in clusters$ and $c_j \neq c_i$ do
6          $d_j \leftarrow$calculateDistance$(c_i, c_j, DT, Agg)$;
7          if $d_j < mind$ then
8              $mind \leftarrow d_j$;
9              $bestMerge \leftarrow$ createBestMerge$(c_i, c_j, d_j)$;
10         end
11     end
12     push$(bestMerges, bestMerge)$;
13 end
14 $numberOfClusters \leftarrow$ size$(D)$;
15 while $numberOfClusters > k$ do
16     $(c_1, c_2) \leftarrow$ pop$(bestMerges)$;
17     $c \leftarrow$ mergeClusters$(c_1, c_2)$;
18     remove$(clusters, c_1)$;
19     remove$(clusters, c_2)$;
20     add$(clusters, c)$;
21     updateBestMerges$(bestMerges, c, c_1, c_2, DT, Agg)$;
22     $numberOfClusters = numberOfClusters - 1$;
23 end
24 return $clusters$;
```

Hierarchical clustering relies only on the distances between the objects and the distances between sets of such objects. Therefore, it is quite easy to extend hierarchical clustering for structured data: we need to calculate the distance matrix between the structured objects. Moreover, in the proposed framework for distance-based learning, we consider the three types of linkages between the clusters as a special case of distances between sets of objects. The distance between the clusters can be any of the distances that are defined for set of objects (see Section 2.2.2.2). Consequently, we define and implement a generic hierarchical clustering algorithm that needs to be parametrized with both the distance between the objects and the distance between the clusters.

### 3.3.2 *k*-means and *k*-medoids clustering

The $k$-means algorithm (Steinhaus, 1956; MacQueen, 1967) is one of the most widely used unsupervised clustering algorithms in machine learning. It is also known as a centroid-based clustering algorithm. The algorithm finds centers of a user-defined number of clusters. Each cluster center ('mean') is initialized to one random input example. The algorithm iteratively finds the nearest cluster center for each example, assigns each example to the nearest cluster and then computes the new center for each cluster as the average of all of the objects belonging to the cluster. The iterative process continues until no object changes its cluster assignment. Algorithm 3.3 outlines the pseudo-code for $k$-means clustering.

**Algorithm 3.3:** Pseudo-code for the $k$-means/medoids algorithm

```
Data: $D$ - data set, $DT$ - datatype and distance definition, $Agg$ - aggregation
      functions definition, $k$ - number of clusters
Result: $clusters$ - (array/list of) the clusters
centroids ← getRandomCentroids($k$);
clusters ← initializeEmptyClusters($k$);
converged ← false;
while not converged do
    oldcentroids ← centroids;
    for each instance $I_i \in D$ do
        $min \leftarrow +\infty$;
        for each centroid $c_j \in centroids$ do
            $d \leftarrow$ calculateDistance($I_i, c_j, DT, Agg$);
            if $d < min$ then
                $min \leftarrow d$;
                $min\_index \leftarrow j$;
            end
        end
        assign $I_i$ to $cluster_{min\_index}$;
    end
    for each $cluster_i \in clusters$ do
        $centroids_i \leftarrow$ calculateCentroid($cluster_i, DT, Agg$);
    end
    if oldcentroids==centroids then
        converged ← true;
    end
end
return clusters;
```

To be able to extend $k$-means clustering for structured objects, it is clear that we need two things: a definition of a distance between the center and each of the objects, and a cluster center calculation function for the case of structured objects. The first part was discussed in the previous chapter and can be readily applied here, so we focus on the second part. When we use the Euclidean distance, the centroid is defined as the average of the vectors that form the cluster. However, in the task of mining/clustering structured data, other distances are typically used, hence, calculating the average is not always possible. To overcome this issue, instead of the $k$-means algorithm, we propose to use the $k$-medoids algorithm for clustering arbitrarly structured data.

The $k$-medoids algorithm was introduced by Kaufman and Rousseeuw (1987) for clustering when using the $L1$ norm instead of $L2$. It is strongly related to the $k$-means algorithm, and its pseudo-code is the same as given in Algorithm 3.3. Just as $k$-means, it calculates the centroid/prototype/representative of the cluster, and then attempts to minimize the distance between the points assigned to the cluster and the centroid of the cluster. However, $k$-medoids clustering differs from $k$-means clustering in the calculation of the center of the cluster. While in $k$-means clustering, the center is the average of all of the vectors, in $k$-medoids the center of the cluster is an element from the cluster that minimizes the sum of distances between that element and all of the other elements of the cluster. Minimizing the sum of pairwise distances (dissimilarities) makes this algorithm usable for clustering arbitrarily structured data, as long as a distance/dissimilarity function is defined.

### 3.3.3 Quality of clusters and clustering

When using clustering analysis, besides the expert evaluation of the clusters, it is important to have an internal evaluation of the quality of the obtained clusters. As we discuss at the beginning of this section, the two most important measures for evaluation are the intra-cluster variance, denoted with $Var(C)$, and the inter-cluster variance.

First, we discuss the evaluation of the quality of a single cluster. The variance of a cluster $C$ can be defined based on a given distance $d$ as $Var(C) = \frac{1}{|C|}\sum_{x\in C} d(X,P)^2$, with $P$ the prototype of $C$, and $d$ the distance defined on the elements of $C$. This requires a definition of the prototype of the cluster. As discussed earlier, the prototype of the cluster $C$ can be defined as an element from the set with a minimal sum of distances to the other elements: $P = \operatorname{argmin}_Q \sum_{X\in C} d(X,Q)^2$. In this case the prototype can be computed with $|C|^2$ distance computations by trying for $Q$ all elements in the cluster.

An alternative way to define intra-cluster variance is based on the sum of the squared pairwise distances (SSPD) between the cluster elements, i.e., $Var(C) = \frac{1}{|C|^2}\sum_{X\in C}\sum_{Y\in C} d(X,Y)^2$. The advantage of this approach is that no prototype is required. It also requires $|C|^2$ distance computations.

Next, we define evaluation measure for estimating the quality of a clustering. The intra-cluster variation ($ICV$) can be calculated as a weighted sum of intra-cluster variances for each cluster, according to the following formula:

$$ICV(\mathcal{C}) = \sum_{C_i\in\mathcal{C}} \frac{|C_i|}{|C|} Var(C_i), \tag{3.1}$$

where $\mathcal{C}$ is the set of clusters (clustering), $|C|$ is the dataset size, and $Var(C_i)$ is the variance of cluster $C_i$.

The evaluation measures defined above measures only the minimization of the intra-cluster variance. To also consider the inter-cluster variance, the Dunn index (Dunn, 1973) can be used. It is defined with the following formula:

$$D = \min_{1\le i\le n}\left\{\min_{1\le j\le n, i\ne j}\left\{\frac{d(C_i,C_j)}{\max_{1\le k\le n} Var(C_k)}\right\}\right\}. \tag{3.2}$$

## 3.4 Implementation

In this section, we briefly describe the implementation of the framework for distance-based learning for structured data. We outline the algorithms and the data representation and distances described earlier. First of all, the framework is implemented in the Java programming language. The architecture of the framework follows the principles of object oriented programming. It uses encapsulation in order to create self-contained modules, and to increase the modularity of the framework, as well as its re-usability value. In addition, inheritance and polymorphism are two principles that are heavily used within the framework. We focused on implementing a framework that will be as generic as possible, and therefore easily extensible. The framework is implemented in several packages which are described in more detail in the following subsections.

### 3.4.1 Datatypes: the *types* package

The *types* package is one of the basic building blocks of the framework. It contains classes that define types and type constructors that are supported within the framework. The main class in this package is the *Type* class. *Type* is an abstract class that defines the basic properties of a datatype. All of the classes for the specific datatypes need to extend this class.

In addition, we define the AMType subclass, representing datatypes on which arithmetical and mathematical operations, such as sum, and multiplying with a real number, are defined. This class is an abstract class, and every datatype that exhibits these properties (for example, vectors/tuples of reals), should extend this class and implement the methods for addition and multiplication. Having defined the roots of the hierarchy, other classes in this package implement all of the datatypes and type constructors discussed and explained in Chapter 2.

Primitive datatypes are implemented through the classes $Real$ and $Nominal$, while the structured type constructors are implemented with the classes $Set$, $Tuple$, $Sequence$, and $TimeSeries$. Furthermore, some special datatypes are placed within this package, including $Instance$, $InstanceItem$, $Dataset$, $Cluster$, all of them being classes that facilitate working with the structured data. We present the package with its classes and their hierarchy in Figure 3.1.

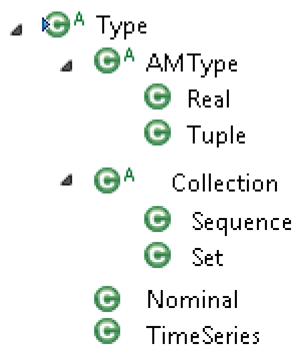


Figure 3.1: The *types* package and the hierarchical organization of classes / datatypes defined within the framework. The white letter C in a green circle denotes a class, while the superscript A denotes an abstract class.

### 3.4.2 Distances: the *distances* package

Within the framework, each distance is implemented in its own class, and all of the classes are stored in the *distances* package. They all extend one basic abstract class, and they need to implement the method *calculateDistance*. *calculateDistance* is a generic method that accepts two generic objects. Each distance will calculate the distance between the objects using its specific distance definition and formula.

If the objects are from a structured datatype then the method will calculate the distance recursively as outlined in Algorithm 2.1. The information about the specific datatypes and the distance information are provided by the *DistanceManager* class and the *DistanceStructure* class. The *DistanceManager* keeps a record of all the distances that are available within the framework, while the *DistanceStructure* class analyzes the structure of the objects, and according to the user preferences, constructs a structure of distances. Hence, these apply the mapping and matching between the various datatypes and distances.

### 3.4.3 Algorithms: the *dmalgo* package

In Sections 3.2 and 3.3, we discussed the algorithms for learning from structured data and their implementation. The package *dmalgo* contains four classes divided into two packages: *dmalgo.classify* and *dmalgo.clust*. The $k$-NN classifier is implemented with the class *kNNClassifier* (in the *dmalgo.classify* package), while the hierarchical agglomerative clustering algorithm, $k$-means and $k$-medoids algorithms are implemented with the *HierarchicalClustering*, *KMeansClustering* and *KMedoidsClustering* classes (in the *dmalgo.clust* package), respectively.

### 3.4.4 Utilities packages

The utility functions that are necessary to enable smoother execution of the code are coded in two packages: *io* and *util*. The *io* package contains classes responsible for handling the input (e.g., reading datasets, reading algorithm-specific settings etc) and the output of the framework (e.g., results from the application of the algorithms, clusters resulting from $k$-means etc.). The *util* package consists of various classes that handle various mathematical operations, such as working with graphs (constructing and calculating flows) and calculating evaluation measures (e.g., precision and recall curves, confusion matrices etc). The *launch* package contains useful classes for designing and organizing experiments, with the most important class being Experimenter, which enables the user to easily set up and experiment using various datatype representation or various combinations of distances for a given dataset/problem.

### 3.4.5 Getting started

In this section, we present a real-life example of usage of the framework for learning from structured data. Let us consider a data set of time series of gene responses to different stress conditions from micro array data (i.e., the yeast dataset from Section 4.2). A data instance can be represented as a tuple of time series and binary datatypes with hierarchical constraints (the datatype specification is given in Figure 2.1).

After defining the data representation, we can define the distances specific for the datatype and the domain considered in this example. To this end, we provide a distance for each part of the structured object in the *distance register* as shown in Table 3.1. It is possible for the same datatype at various positions in the structure to use different distances. This cannot be handled from the distance register, instead it requires several lines of Java code. For example, for the first time series in the tuple we can use the $DTW$, for the second time series we can use the $QD$ and for the third we can use the *Euclidean* distance. The framework creates a list of all possible combinations of distances that can be generated for a given structured datatype and using the distances defined in the *distance register*. The user can then select the desired distance combination to be used, or alternatively to use all of the distance combinations, iterating one by one.

Table 3.1: Part of the distance register file.

```
#primitive data types
IQReal, RealsDistanceMeasure
IQNominal, NominalDistanceMeasure

#tuples
IQTuple, ManhattanDistance
IQTuple, EuclideanDistance
IQTuple, ChebyshevDistance
IQTuple, MinkowskiDistanceMeasure

#sets
IQSet, GreedySumOfMinsDistance
#IQSet, MatchingDistance
IQSet, MaximalSetDissimilarity
IQSet, MinimalSetDissimilarity
IQSet, AverageSetDissimilarty
#IQSet, TanimotoDistanceMeasure
IQSet, HausdorffDistance

#time series
TimeSeries, DTWDistanceMeasure
#TimeSeries, QDMDistanceMeasure

#sequences
#IQSequence, EditDistance
```

We illustrate a simple user scenario of the framework with the code snippet given in Table 3.2. This scenario concerns the selection of the best distance structure and/or optimizing the $k$-parameter in the $k$-NN algorithm. In the first line of the code snippet, we read the dataset given in the file `dataFile`, or open a stream for reading a dataset. Next, we initialize the distance register from the input file `drFileName`. This file enumerates the distances that should be explored for the datatypes present in the given dataset. After the distance register is initialized, we use it to generate possible distance structures (combinations of distances) for the con-

Table 3.2: A code snippet from the instance-based learning framework for two user scenarios concerning the selection of the best distance structure and/or optimizing the $k$-parameter in the $k$-NN algorithm for a given dataset.

```
dataset = mdr.readDataset(dataFile);
dr = new DistanceRegister(drFileName);
distanceStructure = dr.generateDistanceStructure(dataset.getStructure());
e = new Experimenter(dataset);
e.doExperimentBestDistance(distanceStructure);
e.doExperimentOptimizeK(distanceStructure);
e.showResults();
```

sidered datatype. Next, we initialize the experimenter environment for distance/$k$ selection. It uses internal cross-validation to obtain the optimal parameters. The execution of these two scenarios is made with the last two lines of simple commands in the code snippet. Finally, with the last command, the results of the scenarios are displayed.

# Chapter 4

# Evaluation of Instance-Based Learning from Structured Data

In this chapter, we present the experimental evaluation of the instance-based class of distance-based methods for mining structured data, presented in the previous chapter. We first evaluate our nearest-neighbor approach on benchmark datasets of classification of structured data. We then consider practical applications of two types: predictive modelling applications and clustering applications. For each of these groups of applications, we briefly describe the underlying datasets and we give the results of the proposed methods' application.

## 4.1 Evaluation on benchmark classification datasets

In this section, we consider several benchmark datasets for the task of classification of structured data, which are composed by using the set type constructor. To these, we apply the nearest neighbor (and $k$NN) approach described in Chapter 2. In the first section, we compare the performance of our approach to the closest related work by Woznica (2008). In the second, we compare the performance of the matching distance for sets, found to perform very well, to a greedy approximation of this distance that we proposed in the previous chapter.

### 4.1.1 Comparing instance-based approaches to classification of structured data

The work of Woznica (2008) is the closest in spirit to our nearest neighbor (and $k$NN) approach for predictive modeling of structured data. It uses a relational representation of the structured data, which for a single layer of structure is equivalent to a set-based representation. It implements many of the distances on sets discussed in Chapter 2.

The approach of Woznica has been evaluated on a number of benchmark datasets. The basic strategy taken is to first select and then use an appropriate distance (on sets of tuples) from the arsenal at hand. The selection of the distance (as well as the number of neighbors $k$ within the $k$NN algorithm is done by internal cross-validation.

We take exactly the same approach as Woznica within our framework and compare the performance of the two approaches. We evaluate them on four datasets used by Woznica: All of these concern the classification of chemical compounds.

Table 4.1: Comparison of our approach to other approaches in terms of accuracy using 10-fold cross-validation

| Dataset | Our approach | Woznica's approach |
|---|---:|---:|
| Musk 1 | 84.78% | 80.43% |
| Musk 2 | 73.53% | 70.59% |
| Diterpenes | 89.79% | 97.41% |
| Mutagenesis188 | 85.64% | 87.23% |
| Mutagenesis42 | 88.10% | N/A |

Two datasets concern musk compounds (Musk 1 and Musk 2 datasets), one concerns diterpenes and one concern potentially mutagenic compounds (Mutagenesis188).

The results in terms of estimated classification accuracy on unseen cases are given in Table 4.1. The accuracy was estimated by 10-fold cross-validation for both approaches. However, we could not use the same folds for both approaches (since we take the results of Woznica from his PhD thesis).

We can notice that our approach performs better on two datasets and worse on the other two datasets. The largest difference in performance is observed for the diterpene dataset, where Woznica's approach outperforms ours by 7 percentage points.

Finally, let us outline the major difference between the two approaches. The approach of Woznica is limited to classification of structured data, which it considers only on the input side. However, our approach allows for structured data on the output side as well. It can be used for predicting structured outputs (such as hierarchies), as illustrated in the next section with the task(s) of predicting gene functions.

### 4.1.2 Comparing complete and greedy matching for set distances

The matching distance for sets, proposed by Ramon and Bruynooghe (2001) has been shown to perform very well in the context of instance-based learning from structured data. Its calculation is based on solving the minimum-weight maximum-flow problem in flow networks. This unfortunately means that it is computationally expensive.

In Chapter 2, we proposed a greedy approximation of the matching distance, called greedy matching (GM) distance. This approximation has a much lower computational complexity that state-of-the-art algorithms for the complete matching distance. Computational complexity of both distances is discussed in more details in Chapter 2. In this section, we compare the performance of the two distances used within our approach in terms of their predictive performance.

The two distances are used on sets of tuples. We use the Euclidean distance between tuples. The selection of the number of neighbors $k$ within the $k$NN algorithm is done by internal cross-validation.

We compare the performance of the two distances on five datasets, including the four datasets used by Woznica. Again, all of these concern the classification of chemical compounds. Two datasets concern musk compounds (Musk 1 and Musk 2 datasets), one concerns diterpenes and two concern potentially mutagenic compounds (Mutagenesis188 and Mutagenesis42).

The results in terms of estimated classification accuracy on unseen cases are

Table 4.2: Comparison of *GM* and *Matching* distances for sets using 10-fold cross-validation

| Dataset | GM | Matching |
|---|---:|---:|
| Musk 1 | 86.96% | 88.04% |
| Musk 2 | 67.65% | 67.65% |
| Diterpenes | 88.29% | 89.16% |
| Mutagenesis188 | 74.57% | 84.57% |
| Mutagenesis42 | 76.19% | 76.19% |

given in Table 4.2. The accuracy was estimated by 10-fold cross-validation for both approaches. This time, we could use the same folds for both approaches.

We can see that the matching distance performs slightly better than its greedy approximation for all datasets. However, the differences in performance are very small. On the other hand, calculating the greedy approximation is much faster, strongly motivating its use. For example, on the Mutagenesis188 dataset, the algorithm takes 3616 seconds when using *Matching* distance, and it takes 520 seconds when using *GM* distance to classify all of the examples with cross validation. In this case *GM* is about 7 times faster than *Matching* distance, and as discussed in Chapter 2 this difference will be even more notable for larger datasets.

## 4.2 Predictive modelling applications

Let us first focus on the predictive modelling applications. In predictive modelling, based on the input and output space we differentiate datasets (or tasks) that have structured input, and datasets that have structured output. When the datatype on the learning side of the data is structured, then we are dealing with a structured input dataset. Such datasets can be analyzed using the proposed framework, given that the distance for the structured datatype is either predefined, or distances on the type constructors are defined and combined as described earlier. Moreover, the framework allows for the users to define domain specific distances. In the following, we present four datasets that contain structured datatypes. We first present two structured input datasets (*musk* and *diterpenes*) and the results of the analysis. We then present two datasets that have both structured input and structured output *Yeast* dataset and *E. coli* dataset. In the latter two datasets, our goal is to use the gene expression levels under some stress conditions as a predictor of the gene functions.

### 4.2.1 Musk dataset

The musk dataset (Dietterich et al., 1993) is a well studied dataset that concerns the classification of molecules as musks or non-musks. Each molecule is described with its low-energy conformations. The various molecules have different conformations and different number of conformations, i.e., there is a one-to-many relationship between the molecules and the feature vectors describing the conformations. Each feature vector consists of 166 integer values. A molecule is classified as a musk if at least one of its conformations is a musk. This dataset is well studied in the data mining area of multiple instance learning (Dietterich et al., 1997; Amores, 2013).

The dataset has two versions. In this work, we used Version 2, which contains 102 molecules where 39 are labeled as musks and 63 are labeled as non-musks. The total number of conformations is 6598 (on average 64.69 confirmations per molecule and the median is 12).

The goal is then to predict whether a given molecule is a musk or not, i.e., the class of the dataset (which is a binary attribute). The one-to-many relationship between a molecule and its conformations (vectors) makes the molecules suitable to be represented as a set of tuples. Hence, we can describe the data instances, i.e., structured datatypes, for the *musk* dataset using the notation from the framework proposed here, as follows:

$$Molecule = Tuple[Set[Tuple[Integer_1, Integer_2, ..., Integer_{166}]], Binary] \quad (4.1)$$

We performed experiments using the $k$-NN algorithm for structured data proposed in this thesis. Within the experiments, we have tested various distances on sets and distance on tuples thereof. We have also used various values for $k$ (the neighborhood size), but the best results are obtained by using the values 1 or 3 for neighborhood size. By using internal cross-validation, we have concluded that the best results are obtained when *single linkage* distance for sets and *Chebyshev* distance for tuples are applied. Using these distances, on the musk dataset version 2, with 10-fold cross-validation we have obtained accuracy of 73.53%, while the best accuracy reported in the literature is 89.2% (Uwents & Blockeel, 2008) (see Table 4.3). The best reported results uses an algorithm for multi-instance learning that is based on iterated discrimination using axis-parallel rectangles (APR) (Dietterich et al., 1997). Best result using distance-based learning algorithms reported in the literature achieves an accuracy of 88% (Ramon & Bruynooghe, 2001) by using *Matching* distance for sets and *Euclidean* distance for tuples.

### 4.2.2 Diterpenes dataset

In the diterpene dataset (Džeroski et al., 1996), the task is to identify the skeleton type of diterpenoid compounds, given their $^{13}C$-NMR-Spectrum. Diterpenes are one of a few fundamental classes of natural products, with about 5000 known members. A diterpenes' skeleton is a unique connection of carbon atoms each with a specific atom number and, normalized to a pure skeleton molecule without residues, a certain multiplicity (s, d, t or q). The skeleton of every diterpene contains 20 carbon atoms. Sometimes there are additional groups linked to the diterpene skeleton by an oxygen atom with the possible effect of increasing the carbon atom count to more than 20 per diterpene. About 200 different diterpene skeletons are known so far, but some of them are only represented by one member compound. Most of the diterpenes belong to one of 20 common skeleton types. More precisely, the task is to identify the skeleton (type) of diterpenoid compounds, given their $^{13}C$-NMR-Spectra that include the multiplicities and the frequencies of the skeleton atoms. This task is usually done manually by human experts with specialized background knowledge on peak patterns and chemical structures. In the process, each of the 20 skeletal atoms is assigned an atom number that corresponds to its proper place in the skeleton and the diterpene is classified into one of the possible skeleton types.

The collected data contain information on 1503 diterpenes with known structure, stored in three relations *atom*, *bond*, and *nmr*. The first relation specifies to

Table 4.3: Summary of the results on the datasets with structured input compared with the best results reported in the literature.

| Dataset name | $k$-NN for structured data | Best (distance based) | Best reported |
|---|---|---|---|
| Musk 2 | 73.53% | 88.0% | 89.2% |
| Diterpenes | 89.79% | 97.41% | 97.41% |

which element an atom in a given compound belongs. The second relation specifies which atoms are bound and in what way in a given compound. The *nmr* relation stores the measured $^{13}C$-NMR-Spectra. For each of the 20 carbon atoms in the diterpene skeleton, it contains the atom number, its multiplicity and frequency. Additional unary predicates describe the classes to which each compound belongs (23 classes). Considering that each $^{13}C$-NMR-spectrum can be represented as a tuple of multiplicity and frequency, where the frequency is real attribute and multiplicity is nominal, we can represent each diterpene as a set of tuples. Using the notation from the framework proposed in this dissertation, a diterpene can be represented as follows:

$$Diterpene = Tuple[Set[Tuple[Real, Nominal]], Nominal] \tag{4.2}$$

Same as for the *musk* dataset, we performed experiments using the $k$-NN algorithm for structured data proposed in this thesis. Within the experiments, we have tested various distances on sets and distances on tuples by using internal cross-validation. We have also used various values for $k$ (the neighborhood size), but the best results are obtained by using the values 1 or 3 for neighborhood size. The results have revealed that the best results are obtained when *Matching* distance for sets and *Chebyshev* distance for tuples are applied on a neighborhood with size 1 (i.e., nearest neighbor scenario). Using cross-validation we obtained accuracy of 89.79%, while best reported result in the literature using distance-based learning is 97.41% (Woznica, 2008).

### 4.2.3 Yeast dataset

The yeast datasets were constructed from time series gene expression data from the study conducted by Gasch et al. (2000), which are publicly available. They contain time series of expression levels of yeast (Saccharomyces cerevisiae) genes under several diverse environmental stresses, and include also the data introduced by DeRisi et al. (1997). More specifically, the expression levels were measured as cells responded to temperature shocks (25℃ to 37℃ and 35℃ to 25℃), hydrogen peroxide, the superoxide-generating drug menadione, the sulfhydryl-oxidizing agent diamide, the disulfide-reducing agent dithiothreitol (2 different setups), hyper- and hypo-osmotic shock, amino acid starvation, nitrogen source depletion, and switch from anaerobic growth to aerobic respiration upon depletion of glucose, referred to as the diauxic shift. The gene expression levels of around 5000 genes were measured at different time points using microarrays. The data is log-transformed and normalized based on the time-zero measurement of yeast cells under normal environmental conditions. In addition, for each gene we have its gene functions represented as a structured object that is a hierarchy or a tuple of 3 GO hierarchies: biological processes, molecular functions and cellular components. The task is to predict the gene function using the recorded responses for each gene.

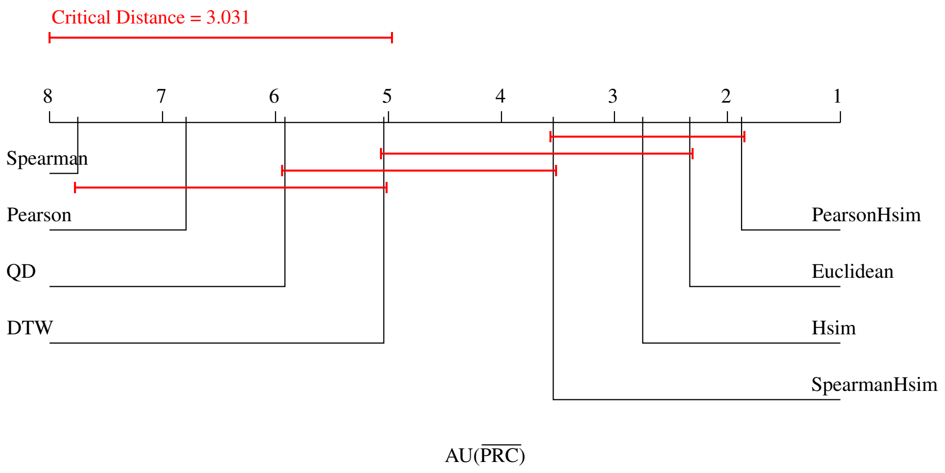
Let us explain how this task will be represented and handled within this framework. On the learning side, we use the response to various stress conditions described above to predict the gene function. We can use either the response to a single environmental stress or several responses at once representing them as a tuple of time series. Also, if available, we can use additional attributes and add them to the tuple, regardless of their structure. We have a total of 13 datasets. Each of the 12 datasets is for a different stress condition described, while the 13th dataset is a combination/tuple of the responses for all 12 stress conditions (it contains 12 time series). In the output space (target space) we have gene function represented as a structured object that is a hierarchy or a tuple of 3 hierarchies. Schietgat et al. (2010) were using various attributes to build a model to predict gene function. With the proposed framework, we can handle the same problem using $k$-NN classification, and using structured data on the left (learning) side as well, which is clearly an advantage of this framework.

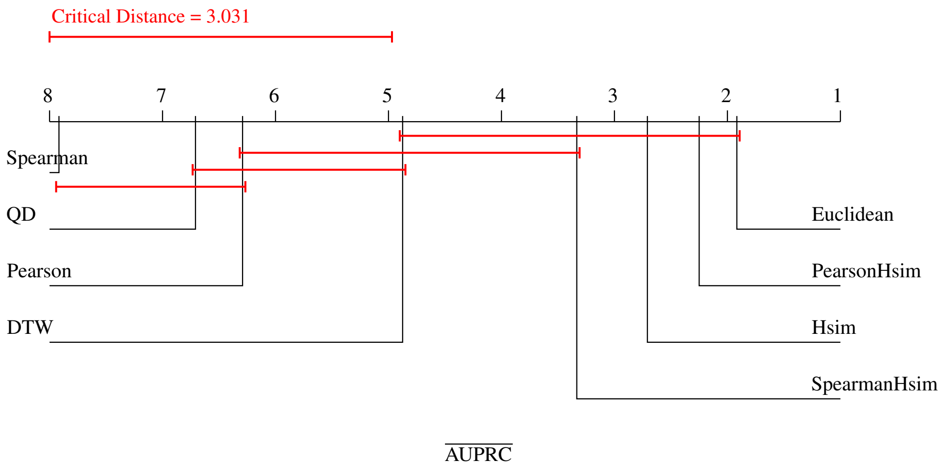
In this dissertation, we present an experimental comparison of 8 distances for time series on the *Yeast* dataset. The evaluated distances include all of the time series distances discussed in Section 2.2.2.3: Dynamic Time Warping, Qualitative distance, Pearson distance (based on the Pearson correlation coefficient), Spearman distance, HSim distance (based on the HSim coefficient), PearsonHsim, SpearmanHsim and Euclidean distance. We use the 1NN algorithm for predicting, since with it we can more accurately measure the influence of the distance function used on the predictive performance (Wang et al., 2013). Using $k$NN, with $k > 1$, is not suitable for such a comparison: it is not clear whether the predictive performance is due to the distance measure used or the aggregation of the $k$ nearest neighbors.

The predictive performance of the 1NN algorithm is estimated using 10-fold cross validation. The performance is assessed by using three error measures based on the precision-recall curve. We use the area under the average PR curve ($AU(\overline{PRC})$), average area under the PR Curves ($\overline{AUPRC}$) and weighted average area under the PR Curves ($\overline{AUPRC_w}$). Each of these measures captures different aspect of the predictions made by the algorithm: $AU(\overline{PRC})$ performs micro-averaging of the performance of the classes, $\overline{AUPRC}$ performs macro-averaging of the classes and $\overline{AUPRC_w}$ performs frequency-weighted averaging of the classes. These measures are typically used in tasks from functional genomics (Vens et al., 2008; Schietgat et al., 2010).

For the statistical evaluation of the results, we employed the corrected Friedman test and the post hoc Nemenyi test as recommended by Demšar (2006). We present the result from the Nemenyi post hoc test with an average ranks diagram). The ranks are depicted on the axis, in such a manner that the best ranking algorithms are at the right-most side of the diagram. The algorithms that do not differ significantly (in performance) for a significance level of 0.05 are connected with a line.

The results of the statistical evaluation of the performance of the distances are depicted in Figure 4.1. We can note that across the three evaluation measures the three best distances are Euclidean, PearsonHsim and Hsim. Additionally, these three distances are statistically significantly better than Sprearman, Pearson and the Qualitative distance. We would also like to note the mediocre performance of the dynamic time warping (DTW) distance – the most widely used distance for time series. This is perhaps due to the fact that the time series under consideration are rather short (the various stress conditions consider less than 10 time points) thus DTW could not obtain optimal results. Next, the inclusion of the Hsim coefficient in





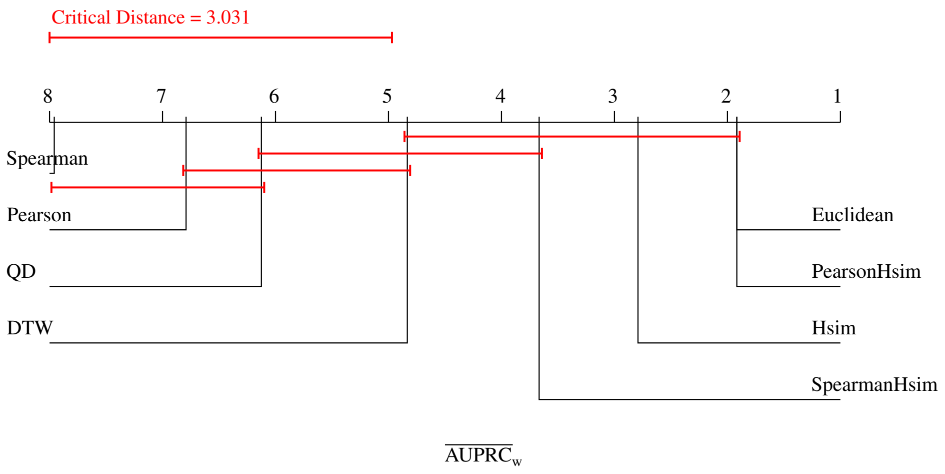


Figure 4.1: A comparison of distances for time series on the task of yeast gene function prediction using the Gene Ontology catalog of gene functions.

the Pearson and Spearman distances improves its predictive performance by a statistically significant margin, i.e., PearsonHsim and SpearmanHsim are statistically significantly better than Pearson and Spearman, respectively. All in all, considering both the computational cost and the predictive performance of the distances, this evaluation reveals that Euclidean distance could be the best choice.

### 4.2.4 E. coli dataset

Partridge et al. (2007) introduced a dataset of time series gene expression data from the bacterium *Escherichia coli*. The bacteria were treated in a controlled environment by lowering the oxygen, and the gene expression levels were recorded under these oxygen-starved conditions, after 5, 10, 15 and 60 minutes. These expression levels were compared to aerobic steady state, and relative changes were calculated. The dataset has 4229 genes and a time series with length 4 for each of the genes.

Similar as above, we can use our framework to exploit the expression data to predict the gene annotations of the unannotated genes, i.e., perform gene function prediction. In the input space of this supervised learning problem, we have a structured object that is a time series and in the output space we have a structured object that is a hierarchy or a tuple of 3 hierarchies, subsets of the GO: biological processes, molecular functions and cellular components. Given the fact that the proposed framework uses the generic $k$-NN algorithm, it can handle time series on the input side, and structured objects such as a tuple of hierarchies on the output side. We are thus able to use it on the E. coli dataset to predict gene functions.

We follow the same experimental procedure as for the Yeast dataset. Namely, we evaluate the performance of 8 distance measures for time series data by using the 1NN algorithm for predicting. The predictive performance is estimated using 10-fold cross-validation. The performance is assessed using the three evaluation measures: $AU(\overline{PRC})$, $\overline{AUPRC}$ and $\overline{AUPRC_w}$.

We present the results of the evaluation using the E. coli dataset in Table 4.4. The results reveal that the two best performing distances are *Hsim* and *Euclidean*. Recall that these two distances were among the best performing also for the Yeast dataset. Next, the inclusion of the Hsim coefficient in the Pearson and Spearman distances improves its predictive performance, i.e., PearsonHsim and SpearmanHsim are better than Pearson and Spearman, respectively.

Table 4.4: A comparison of distances for time series on the task of E. coli gene function prediction using the Gene Ontology catalog of gene functions. The emphasized values in the table are the best values obtained for each of the error measures.

| Distance | $AU(\overline{PRC})$ | $\overline{AUPRC}$ | $\overline{AUPRC_w}$ |
|---|---:|---:|---:|
| DTW | 0.234 | 0.031 | 0.235 |
| QD | 0.237 | 0.018 | 0.227 |
| Euclidean | 0.236 | ***0.033*** | 0.237 |
| Pearson | 0.230 | 0.020 | 0.229 |
| PearsonHsim | 0.234 | 0.029 | 0.236 |
| Hsim | ***0.239*** | 0.032 | ***0.239*** |
| Spearman | 0.214 | 0.015 | 0.222 |
| SpearmanHsim | 0.234 | 0.029 | 0.235 |

## 4.3 Clustering applications

We next present possible applications of the framework in a clustering context. In this case, we do not have an output space (target space). We are dealing only with structured inputs and the task is to find groups (clusters) of these structured objects. These datasets can be analyzed in the proposed framework, given that the distance is either predefined, or already constructed by the use of aggregation functions that correspond to the type constructors are defined and combining distances on component datatypes, as described earlier. Let us describe how the structured data described above can be handled directly with the clustering methods implemented in our framework, without further transformations.

### 4.3.1 Yeast dataset

Let us recall from the previous section that the yeast dataset has both structured input and structured output. For clustering purposes, we focus on the structured input, i.e., the time series response of gene expression to different types of stress. We thus consider a total of 12 datasets: Each of the 12 datasets is for a different stress condition described. The goal is to analyze the time series to find groups of genes that react similarly. To reach the goal we use the techniques for clustering described earlier in this chapter, in particular hierarchical agglomerative clustering. More specifically, we perform hierarchical agglomerative clustering for each of the 12 types of stress response separately by using four different distance measures on time series. We then report the $ICV$ reduction with respect to the number of clusters in the clustering.

In Figure 4.2, we present the reduction of the intra-cluster variance when the number of clusters chosen from a hierarchical agglomerative clustering increases; for hierarchical agglomerative clusterings obtained using 4 different distance measures. Using and comparing various distance measures is also one of the benefits from the proposed framework. The curves of $ICV$ reduction indicate that an appropriate number of clusters would be close to 70.

We then select from all of the clusterings obtained for each stress condition and for each different distance measure the ones with 70 clusters. Each of the clusterings can be visualized as follows. On the horizontal axis, we put the size of the clusters from the given clustering, while on the vertical axis, we put the $ICV$. Next, we visually inspect the obtained graphs and select clusters that are reasonably sized (i.e., that have more than 50 genes) and have relatively low $ICV$. The selected clusters can then be analyzed using tools, such as the STRING database visualization tool (Franceschini et al., 2013). We perform this analysis for all of the clusterings, and in the remainder we illustrate one of the findings of the analysis.

In Figure 4.3, we compare a cluster obtained within the framework (on the right) and a set of randomly selected genes of the same size (on the left). The visualized cluster contains 73 genes and it was obtained using DTW as distance measure and nitrogen source depletion ($NDepletion$) as stress condition. The two groups of genes (the one from our framework and the randomly selected one) are then visualized using STRING showing the known interactions between the genes (proteins) in each group. The genes in the cluster have similar responses as identified by the clustering algorithm using the proposed framework. Moreover, they are connected by a dense network of interactions. In contrast, the genes from the randomly chosen group

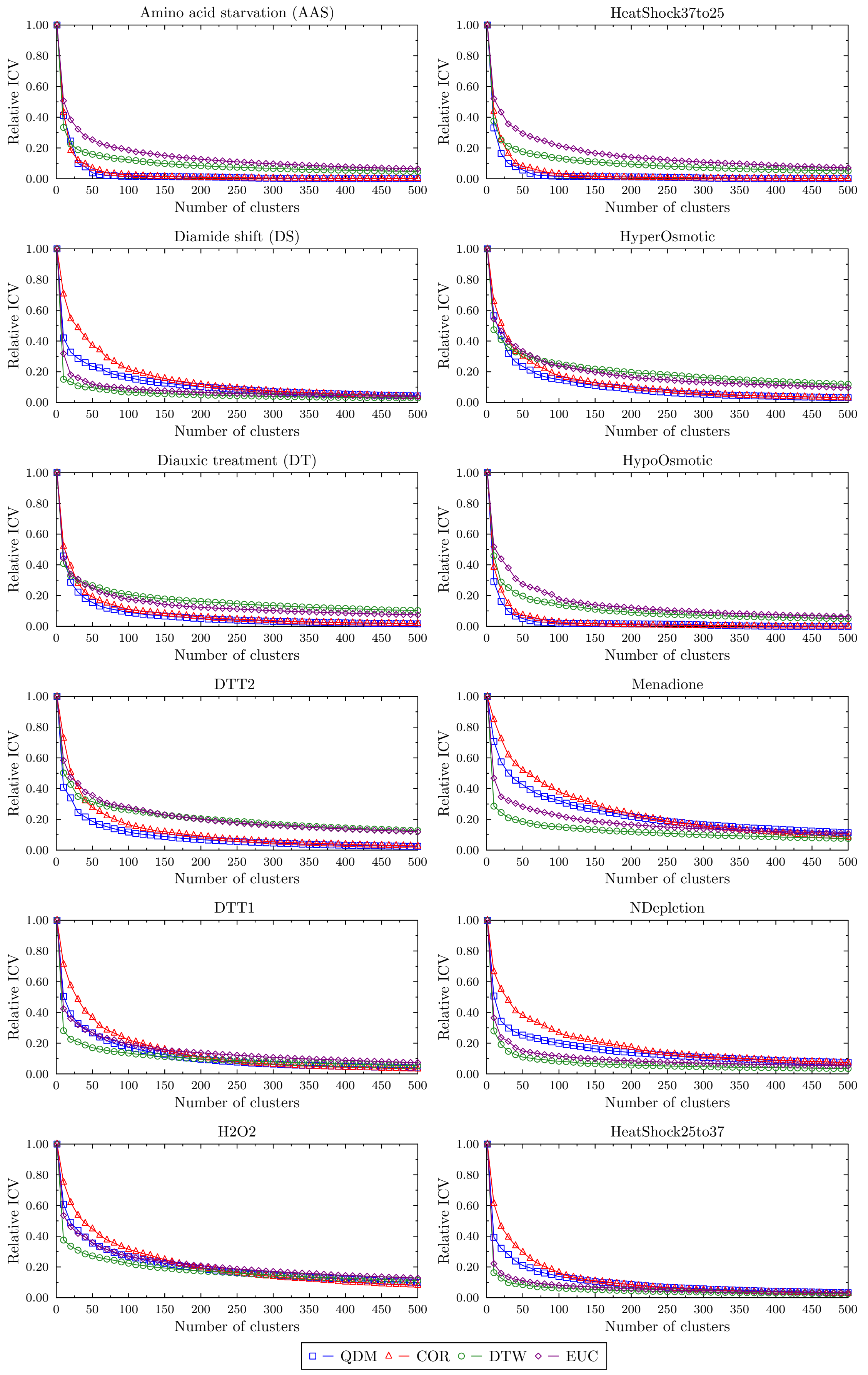


Figure 4.2: ICV decreases with the number of clusters in hierarchical clusterings obtained by using four different distance measures on the yeast datasets.

have only sporadic connections. This clearly demonstrates that the genes from the cluster form a meaningful, functionally connected group.

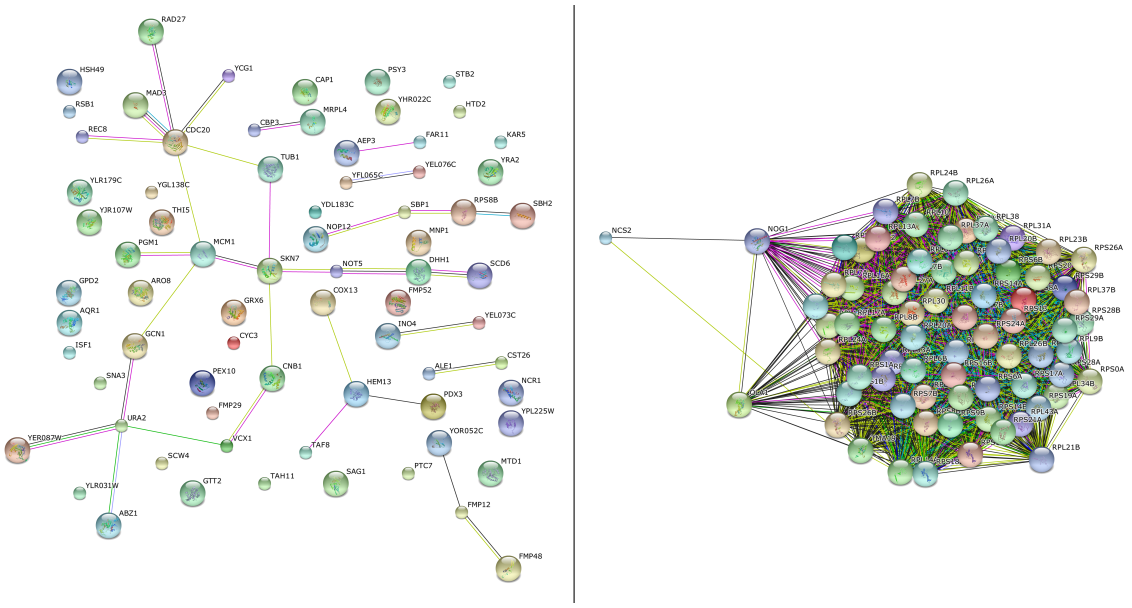


Figure 4.3: A visual representation of two groups of genes and their interactions (taken from the STRING database). On the left, 73 randomly selected yeast genes. On the right, a cluster with 73 yeast genes obtained by clustering their responses (time courses of gene expression levels) to stress.

## 4.3.2 E. coli dataset

The E. coli dataset, described above, has structured input and structured output. Here, for clustering purposes, similar as in the yeast case, we will focus only on the structured input part, the recorded responses in the form of time series. The goal is to analyze the time series to find groups of genes with a similar response to an anoxic shock. We follow the same experimental procedure from the analysis of the yeast dataset.

In Figure 4.4, we present the reduction of the intra-cluster variance when the number of clusters chosen from a hierarchical agglomerative clustering increases; for hierarchical agglomerative clusterings obtained using 4 different distance measures. A smaller number of clusters seems more appropriate than for the yeast datasets. The curves of $ICV$ reduction indicate that an appropriate number of clusters would be close to 30. We then select from all of the clusterings obtained for each different distance measure the ones with 30 clusters.

Next, we visually inspect the reduction of $ICV$ with respect to cluster sizes. From this inspection, one can select several clusters of genes that have a similar response to the stress and relatively low $ICV$. We select an illustrative example for visualization using the STRING tool. This example is presented in Figure 4.5. Similarly, we compare the selected cluster with 144 E. coli genes (on the right) with a set of randomly selected 144 E. coli genes (on the left). Note that the network is much less dense here than in Figure 4.3. Overall, the gene interactions in yeast are much more studies than in E. coli. Hence the difference in density. The genes

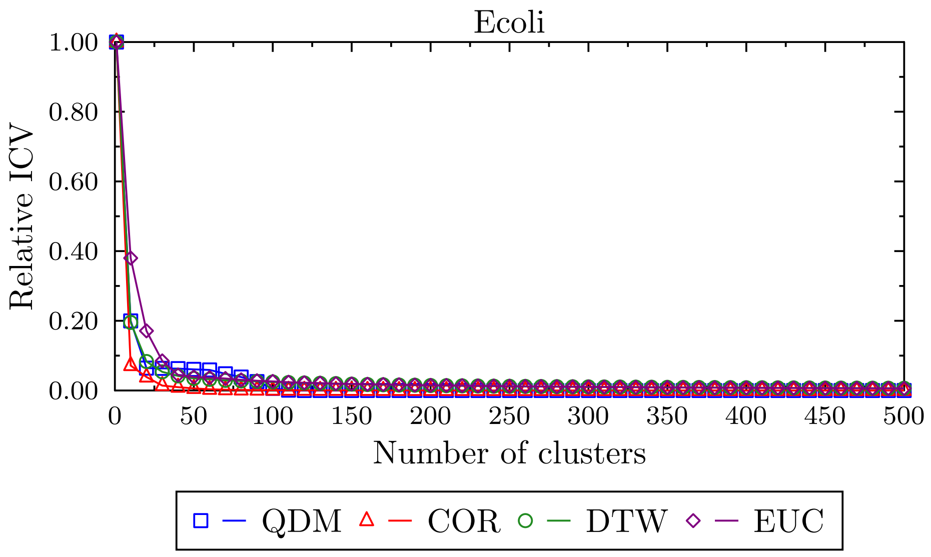


Figure 4.4: ICV decreases with the number of clusters in hierarchical clusterings obtained by using four different distance measures on the E. coli dataset.

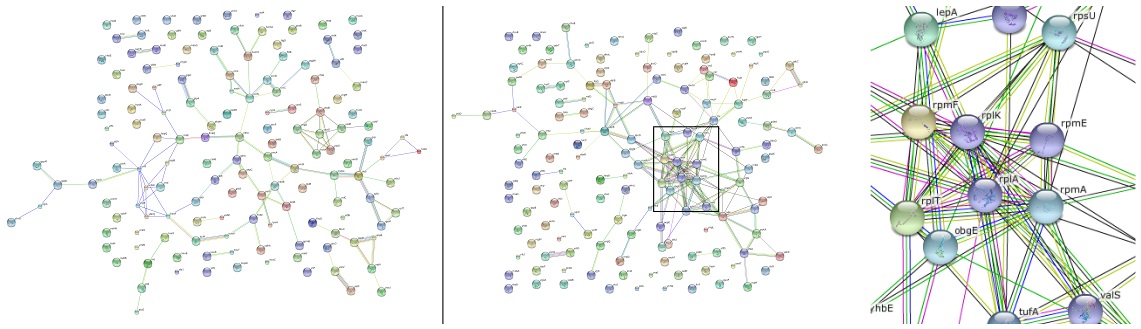


Figure 4.5: A visual representation of two groups of genes and their interactions (taken from the STRING database). On the left, 144 randomly selected E. coli genes. On the right, a cluster with 144 E. coli genes obtained by clustering their responses (time courses of gene expression levels) to stress.

in the cluster have similar responses as identified by the clustering algorithm using the proposed framework. Moreover, they are connected by a denser network of interactions than the random set of genes. This shows that the genes from the cluster form a meaningful group.

## 4.4 Summary

In this chapter we have evaluated the performance and utility of the instance-based class of distance-based methods for mining structured data, presented in the previous chapter. Where possible, we have compared it to existing approaches, examining the performance of our nearest-neighbor approach on benchmark datasets of classification of structured data. We find it performs comparably to the existing approach of Woznica (2008). The greedy approximation of the matching distance on sets gives comparable performance results to its non-greedy original at a lower computational cost.

The major advantage of our approach, however, is that it is applicable to a much larger class of problems than the existing approaches. In the context of predictive modeling, the approach of Woznica is limited to structured data only on the input side. Our approach allows for structured data both on the input and the output side. It can be used for predicting structured outputs (such as hierarchies), as illustrated on the two applications of predicting gene functions.

The clustering part of our proposed framework is also applicable to arbitrarily structured data. We have also considered practical applications illustrating this aspect. We have successfully applied our approach to finding clusters of genes with similar time-course expression profiles: We have showed that the found clusters of genes are much more densely interconnected than randomly selected groups of genes and thus make biological sense.

# Chapter 5

# Predictive Clustering for Structured Data

Predictive clustering is a general framework that unifies clustering and prediction. The alrogithm for learning predictive clustering trees, in its basic form, handles only tuples of primitive datatypes as targets (Blockeel et al., 1998; Struyf & Džeroski, 2006). In this chapter, we first describe how predictive clustering works in its basic form. We then outline the changes necessary to handle arbitrary structured types of data as targets. Next, we describe its extension for predicting time series data, and continue with the extensions for predicting multi-layered datatypes such as: tuples of time series, tuples of hierarchies, and tuples/sets of arbitrarily structured objects.

## 5.1 Predictive clustering algorithm

Predictive clustering is a general framework that combines clustering and prediction (Blockeel et al., 1998). Predictive clustering partitions the data set into a set of clusters such that the instances in a given cluster are similar to each other and dissimilar to the instances in the other clusters. In this sense, predictive clustering is identical to regular clustering (Kaufman & Rousseeuw, 1990). The difference is that predictive clustering associates a predictive model to each cluster. This model assigns instances to clusters and provides predictions for new instances.

Algorithm 5.1 presents the generic induction algorithm for PCTs (Blockeel et al., 1998). It is a variant of the standard greedy recursive top-down decision tree induction algorithm used in many decision tree induction systems, such as C4.5 (Quinlan, 1993). It takes as input a set of instances $I$. The procedure *BestTest* (Algorithm 5.2) searches for the best acceptable test that can be put in a node. If such a test $t^*$ can be found, then the algorithm creates a new internal node labeled $t^*$ and calls itself recursively to construct a sub-tree for each cluster in the partition $\mathcal{P}^*$ induced by $t^*$ on the instances. If no acceptable test can be found, then the algorithm creates a leaf and the recursion terminates. The procedure Acceptable defines the stopping criterion of the algorithm, e.g., specifying maximum tree depth or a minimum number of instances in each leaf.

The description of the induction algorithm, up to this point, is not different from that of a standard decision tree learner. The main difference is the heuristic that is used for selecting the tests and the prototype/centroid calculation function. For PCTs, this heuristic minimizes the average variance in the created clusters (weighted

**Algorithm 5.1:** PCT(I)

```
Data: $I$ - Instances
Result: Predictive clustering tree
(t*, h*, P*) ← BestTest(I);
if $t^* \neq none$ then
    for each $I_k \in \mathcal{P}^*$ do
        $tree_k \leftarrow$ PCT($I_k$);
    end
    return node($t^*$, $\bigcup_k \{tree_k\}$);
else
    return leaf(centroid($I$));
end
```

**Algorithm 5.2:** BestTest(I)

```
Data: $I$ - Instances
Result: BestTest: $(t^*, h^*, \mathcal{P}^*)$
$(t^*, h^*, \mathcal{P}^*) \leftarrow (none, 0, \emptyset)$;
for each possible test $t$ do
    $\mathcal{P} \leftarrow$ partition induced by $t$ on $I$;
    $h \leftarrow Var(I) - \sum_{I_k \in \mathcal{P}} \frac{|I_k|}{|I|} Var(I_k)$;
    if $(h > h^*) \wedge$ Acceptable$(t, \mathcal{P})$ then
        $(t^*, h^*, \mathcal{P}^*) \leftarrow (t, h, \mathcal{P})$;
    end
end
return $(t^*, h^*, \mathcal{P}^*)$;
```

by cluster size, see line 4 of Algorithm 5.2). Minimizing the variance maximizes cluster homogeneity. The next sections discuss how the prototype should be calculated and how the cluster variance can be instantiated for various structured datatypes.

## 5.2 PCTs for arbitrarily structured data

While the original PCT algorithm is very popular and often used to solve various problems, it is limited to handling only data represented as tuples of primitive datatypes, e.g., sets of points in a multi-dimensional Euclidean space. In particular, the targets that can be handled as implemented in CLUS (Struyf & Džeroski, 2006) are tuples of continuous or tuples of discrete variables. In this dissertation, we describe and implement an extension of the algorithm that can handle arbitrary structures in the target/output space.

To be able to do so, we use the principles discussed earlier in Section 2.2. We use these to introduce several changes into the PCT induction algorithm, which we describe in this section. In the following subsections, we describe the changes of the PCT algorithm needed to handle arbitrarily structured data in the target/output space: These involve the calculation of prototypes/centroids and the values of the search heuristic/variance.

### 5.2.1 Calculating the prototype

When the elements of the cluster (or their target parts) are vectors from a vector space, the prototype is well defined and is calculated as the mean of all the vectors in the cluster. It is calculated by summing up all the vectors and multiplying the sum by the scalar $\frac{1}{|C|}$, with $|C|$ being the size of the cluster. The calculation of the prototype is of linear complexity in terms of $|C|$, i.e., has complexity $O(|C|)$ .

For arbitrarily structured data, as in the $k$-medoids and $k$-means algorithm, we suggest using the medoid of the cluster as its prototype. The medoid is computed as $P = \operatorname{argmin}_Q \sum_{X \in C} d(X, Q)^2$. In this case, the prototype can be computed with $|C|^2$ distance computations by trying for $Q$ all objects in the cluster.

### 5.2.2 Calculating the heuristic

To be able to handle arbitrary datatypes as targets, we need to define the heuristic function that is used for selecting the tests (step 4 in the Algorithm 5.2). In this step, the PCT algorithm requires a measure of cluster variance. In the most general case, the variance of a cluster $C$ can be defined based on a given distance $d$ as

$$Var(C) = \frac{1}{|C|} \sum_{X \in C} d(X, P)^2, \tag{5.1}$$

where $P$ is the prototype of $C$. To be able to cluster data of an arbitrarily structured datatype $t$, $d$ should be a distance defined for the structured datatype $t$, as discussed in Chapter 2.

If the prototype $P$ can be calculated in closed form, the variance can be calculated efficiently. This is the case for vector spaces, where $P$ can be calculated in $O(|C|)$ complexity. $Var(C)$ can be calculated with additional $|C|$ computations and the complexity is $O(2|C|) = O(|C|)$.

When the prototype of the cluster is its medoid the complexity of calculating the variance is $O(2|C|^2) = O(|C|^2)$, $|C|^2$ for calculating $P$ and $|C|^2$ for calculating $Var(C)$. In this case, an alternative way to define cluster variance is based on the sum of the squared pairwise distances (SSPD) between the cluster elements, i.e.,

$$Var(C) = \frac{1}{|C|^2} \sum_{X \in C} \sum_{Y \in C} d(X, Y)^2 \tag{5.2}$$

The advantage of this approach is that the prototype is not required in advance: In fact, $P$ can be calculated in the same pass as $Var(C)$ with $|C|^2$ distance computations. This has the same time complexity as just calculating the prototype as the medoid. Hence, using the $Var(C)$ definition based on a prototype is only more efficient if the prototype can be computed in time less than quadratic in the cluster size. For example, this is the case for distances, such as the Euclidean distance, where the prototype can be calculated in closed form as the average of the vectors. For the other distances, with no closed form prototypes, we choose to estimate cluster variance using the SSPD method.

As discussed previously, the algorithm for calculating $SSPD$ has computational cost of $O(N^2)$, where $N$ is the number of elements in the cluster. In some applications, this could be very inefficient (e.g., datasets with tens or hundreds of thousands of examples). We note that in the standard approach (when the elements are elements from the vector space) where the prototype can be calculated in linear time, the algorithm given by Equation 5.1 has linear, $O(N)$ complexity. Therefore, we propose an estimate of the SSPD that could be calculated with sampling, defined with the formula below:

$$Var(C) = \frac{1}{|C|m} \sum_{X \in C} \left( \sum_{Y \in \text{sample}(C,m)} d(X,Y)^2 \right) , \tag{5.3}$$

with sample$(C, m)$ a random sample without replacement of $m$ elements from $C$, and $m \leq |C|$. The computational cost of (5.3) grows only linearly with the cluster size. In the implementation, it is left to the user to choose to use sampling to approximate SSPD or to calculate the exact SSPD. In (Džeroski, Gjorgjioski, et al., 2006), we show that using sampling for estimating the SSPD can be efficiently approximated. In the next sections, we describe the instantiations of $Var(C)$ for specific datatypes. More specifically, we outline their implementations in the framework, describe the supported distances, and outline some real-life applications of the developed methods.

## 5.3 Tuples of reals

Struyf and Džeroski (2006) proposed a modification of the PCT algorithm that is able to cluster and predict several numeric target variables at once (i.e., addresses the task of multi-target prediction). PCTs that are able to predict multiple targets simultaneously are called multi-target predictive clustering trees (MTPCTs). MTPCTs that predict a tuple of continuous variables (regression tasks) are called multi-target regression trees (MTRTs), while MTPCTs that predict a tuple of discrete variables are called multi-target classification trees (MTCTs). The instantiation of the CLUS system that learns multi-target trees is called CLUS-MT (Kocev et al., 2013).

The heuristic used in this algorithm for selecting the attribute tests in the internal nodes is the reduction of variance as defined in step 4 in the Algorithm 5.2. The variance for MTPCTs is defined as $Var(c) = N \cdot \sum_{t=1}^{T} Var\,[y_t]$, with $N$ the number of examples in the cluster, $T$ the number of target variables, and $Var\,[y_t]$ the variance of target variable $t$ in the cluster. For each of the continuous targets $y_t$, the variance $Var(y_t)$ is as defined in its normal sense.

The variances of the target variables are standardized, so that each target variable contributes equally to the overall variance. This is due to the fact that the target variables can have completely different ranges. In addition, CLUS-MT supports weighting of the target variables so that the variance function gives more weight to some variables and less to others. The prototype function (calculated at each leaf) returns as a prediction the tuple with the mean values of the target variables, calculated using the training instances that belong to the given leaf (Kocev et al., 2013). We applied the proposed method on the real-life problem of estimating vegetation height and canopy cover from remotely sensed data (Stojanova et al., 2010). We give the complete study in Chapter 6.

## 5.4 Sets of discrete

In the previous section, we discussed how PCTs are implemented for multi-target regression. PCTs can also handle multi-target classification, and are able to cluster and predict a tuple of discrete variables. In this case, $Var(c) = \sum_{t=1}^{T} Gini\,[y_t]$.

Within this task, PCTs can also be used to predict a tuple of binary variables (i.e., discrete variables with two values that are typically represented as 0 or 1). Each of the binary variables represents the presence or absence of a label and the tuple of binary variables represents a set of labels. The task is also known as multi-label classification.

The main component in the PCTs, as discussed previously, is the variance of the clusters that is used in the heuristic of the algorithm. In the general case, for structured data, the variance is calculated as the sum of the squared distances between each instance and the prototype. To be able to calculate the variance, we need to define the distance between the instances.

When dealing with tuples of discrete/binary values, the tuples can be represented as 0/1 vectors. The variables in this case are typically called 'labels'. The length of the target tuple is the number of all labels in the dataset. In this case, the Euclidean distance between two tuples of labels $C_i$ and $C_j$ can be defined as the Euclidean distance between their vector representations. This measure is directly proportional to the simple count of mismatches between the classes, which is also called *the simple matching coefficient*. We would like to note that calculating the Manhattan distance on the vectors of categorical data, with a 1/0 distance defined between the classes, leads to the same result.

We next focus on defining PCTs for predicting sets of categorical values, i.e., PCTs for addressing the task of multi-label classification. In this instantiation of the PCTs, we consider the output space to consist of sets of discrete values (or a tuple of binary variables). In a similar manner as described above, we define the variance as the sum of the squared pairwise distances between the examples (in this case, sets).

We consider two definitions of the datatype set of discrete. The first one is to represent each set as a vector of zeros and ones, as described earlier, and then to apply the Euclidean distance (this is equal to treating the sets as tuples). The second scenario is to use distance measures specifically designed for sets. The benefits from the second scenario are that the representation is more natural (we do not need to transform the output space from a set to a vector) and it facilitates the use of distances defined for sets. The following distances for the datatype *set* are implemented and can be used in the PCT algorithm: *Hamming distance*, *Jaccard distance* and *Matching distance*. In an empirical study, we performed an evaluation of PCTs for multi-label classification approach and compared the different distances for sets (Gjorgjioski et al., 2011). We present the complete study in Chapter 7.

## 5.5 Sequences of reals: Time series

This section shows how predictive clustering can be applied to predict and cluster time series (Liao, 2005). To cluster time series, as discussed above, we need to be able to calculate the variance of a cluster as defined by Equation 5.1. To this end, we need to define the following: the distance between two time series and the prototype calculation of a cluster.

To begin with, we can use as distance measure here any of the distances described in Chapter 2. More specifically, within the CLUS system, we implemented the following distances on time series datatype: Dynamic time warping distance, Qualitative distance, and Pearson correlation distance. Following a similar discussion as in Section 5.2, we propose to estimate the cluster variance with using SSPD for time series. Since most of the distances on time series do not have closed form prototypes, we propose to use the medoid, where the prototype is one of the time series from the cluster.

We applied the proposed extension of the PCTs on a real-life problem from biology: predicting gene response to stress based on gene functions in *S. cerevisae*. We present the complete study in Chapter 8 with two publications. In (Džeroski, Gjorgjioski, et al., 2006), we presented the methodological part of the algorithm and discussed its implementation. We gave a comparison to other standard data mining methods for clustering time series. This was the first publication that presented the PCTs for predicting time series. In (Slavkov et al., 2010), the focus was more on the application domain and the main discussion was on the biological implications of the results that were produced by applying our method on the given data.

## 5.6 Multi-layered datatypes

In the previous three sections, we discussed the instantiation of PCTs for structured data such as a tuple of reals, a set of categorical data and a time series. A tuple of reals is a datatype that has one type constructor (*Tuple*) applied to primitive datatypes (*real*). A set of discrete datatype is very similar, having *Set* as a type constructor applied to primitive datatype (*discrete*). A time series datatype also has type constructor *sequence* and primitive datatype (*real*). All in all, these datatypes are examples of structured datatypes containing only a single type constructor.

In this section, we focus on the instantiation of the PCT algorithm for predicting more complex datatypes, i.e., multi-layered datatypes. Multi-layered datatypes have multiple type constructors and are discussed in details in Section 2.1.2. Recall that the variance of a cluster is the basic component of the PCT algorithm. Hence, with a proper definition of a distance for a multi-layered/composite datatype, we can instantiate the variance function of the PCTs for multi-layered datatypes. We define the variance function as a sum of the squared pairwise distances between the elements. In the following, we present two multi-layered/composite datatypes that are of practical relevance: tuple of time series and tuple of hierarchies.

### 5.6.1 Tuples of time series

The datatype class *tuple of time series* can be represented using the notation used in this dissertation as $Tuple[TimeSeries_1, TimeSeries_2, ..., TimeSeries_T]$. In Section 2.2.3, we discuss in detail the calculation of the distance for such a datatype and provide examples. We present and evaluate the instantiation of PCTs for predicting a tuple of time series, i.e., multi-dimensional time series on data from ecological modelling. More specifically, we use PCTs for multi-dimensional time series for modelling the forest growing stock in Slovenian forests (Gjorgjioski et al., 2015). We give the complete study in Chapter 9.

### 5.6.2 Tuples of hierarchies

The datatype class *tuple of hierarchies* can be represented using the notation used in this dissertation as

$$Tuple[Tuple_1[v_1, v_2, ..., v_M], Tuple_2[v_1, v_2, ..., v_L], ..., Tuple_H[v_1, v_2, ..., v_K]]$$

where the variables $v_i$ are binary and there are parent-child relationships defined between the variables (i.e., hierarchy constraints). Note that the inner tuples can have variable length. For this multi-layered datatype, the distance can be calculated using the algorithm presented in Section 2.2.3.

Schietgat et al. (2010) perform gene function prediction using hierarchical multi-label decision tree ensembles. They build different ensembles for each hierarchy of the Gene Ontology data: Biological Process hierarchy, Molecular Function hierarchy and Cellular Component hierarchy. With the approach described earlier, we can put the three hierarchies together into one tuple of hierarchies and then we can build a three that will predict this structured datatype. The benefits of such representation of the target would be two-fold: (1) it generates smaller trees, and (2) it groups the genes with similar functions according to all of the three hierarchies.

### 5.6.3 Arbitrarily structured objects

The PCT algorithm with the extensions proposed in this dissertation can be applied to predict arbitrary structured objects. In the previous two subsections, tuples of structured objects were considered. In general, type constructors can be nested arbitrarily to obtain more and more complex structured datatatypes.

To instantiate PCTs to predict arbitrarily complex datatypes, we need to define distances on the target space. These can be composed from distances on simpler (and eventually primitive) datatypes. Once we have a distance on the target space, we can use the approaches from this chapter to build PCTs to predict such targets.

## 5.7 Implementation

In short, the approaches for learning PCTs for predicting structured values presented in this chapter are implemented within the CLUS system. In particular, those that involve type constructors other than tuple, i.e., sequences of real values and sets of discrete values, as well as those concerning multi-layer datatypes were developed within this dissertation. PCTs for short time series are now part of the standard CLUS (http://dtai.cs.kuleuven.be/clus/; http://sourceforge.net/projects/clus/) release, while the other extensions are part of an unreleased branch of CLUS.

In CLUS, distances on the target space play a crucial role. Distances on the structured datatypes of interest (sequences of real values and sets of discrete values, as well as multi-layer datatypes) have been developed and implemented in the context of the instance-based framework from Chapter 3 and its implementation. When implementing the extensions of CLUS, we have re-used code from the implementation of the instance-based framework.

The implementation of the extensions of the predictive clustering trees, i.e., PCTs for arbitrarily structured data, is following the established implementation principles of the CLUS system. It also follows the basic principles described in this chapter, as well as, the principles described in Section 3.4. In both the instance-learning framework and PCTs, the distances are defined similarly, only with the needed adaptations to the CLUS specific data structures. Both systems are implemented in Java, following the object-oriented principles of programming. Therefore, the distance from one system can be easily reused into the other.

Next, we give some implementation details. We added several classes and modified existing classes to extend CLUS for predicting arbitrarily structured data. To begin with, in the package `data.type`, we implement classes for the type constructors explained in this thesis that extend the main generic class *ClusAttrType*,e.g., define attributes from the time series datatype. We extend the *data.io.ClusReader* class to be able to handle reading of structured datatypes from `arff` files. In the `ext` package, we add several new packages, including `sets`, `sets.distances`, `tuples`, `tuples.distances`, etc. In each of the packages, we implement three classes: (1) a class for the type constructor, (2) a class for calculating various statistics that extends *SumPairwiseDistancesStat* class, and (3) a general class that defines distance over the type constructor that extends the *ClusStructuredDistance* class. In each sub-package `distances`, we implement the corresponding distances for each datatype. For example, the instantiation for the data type *Set* contains a class *Set* for type constructor, *SetStatistic* for calculating statistics and *SetDistance* for calculating the distance. It could also include distance definitions, such as *GSMDistance extends SetDistance*. Finally, in the `error` package, we implement error measures used to assess the predictive performance of the models.

All in all, the extension of the PCTs within the CLUS system was performed in a modular and extensible way. In other words, it is relatively easy to add new datatypes and new distances. New distances can be added in a plug-and-play manner. A distance can be plugged into the system by extending the main class that defines the general principles of distance calculation for that datatype. This enables the users of the system to easily implement/develop new distances and to compare them to the existing ones.

# Chapter 6

# Estimating Vegetation Height and Canopy Cover from Remotely Sensed Data with Machine Learning

In Chapter 5, we discussed the instantiation of the predictive clustering paradigm for predicting structured data. The first instantiation concerned predicting multiple continuous variables, i.e., a tuple of reals ($Tuple[Real_1, Real_2, ..., Real_T]$). This task is also known as multi-target regression.

In this chapter, we include a paper concerned with the application of predictive clustering trees for predicting tuples of reals. We address the problem of estimating vegetation height and canopy cover from remotely sensed data. The results of the study can be used to reduce the cost of using expensive remote sensing technologies, such as LiDAR (Light Detection And Ranging), by exploiting cheap and easily accessible remotely sensed data, such as satellite data.

The definition of the problem is as follows. Given are (expensive) LiDAR data and (cheap) satellite data for the same (small) region. We need to learn a model to estimate vegetation height and canopy cover for broader(larger) region from (cheap) satellite data only. From the LiDAR data, we can calculate accurately the values of the target variables, i.e., vegetation height and canopy cover, for the smaller region. For the small region, we can then learn a mapping from the (cheap) satellite data to the forest properties. The obtained model is then used to predict the values of the vegetation height and canopy cover for a broader region from satellite data only, without the need for (expensive) LiDAR data.

We address this problem for the *Kras* area in Slovenia. More specifically, we consider a small portion of the area for learning the predictive models, and the complete Kras region for predicting the values for vegetation height and canopy cover. We have both LiDAR and satellite data for the small portion of Kras, while we have only satelite data for the complete *Kras* region. The goal is then to produce maps of vegetation height and canopy cover for the whole Kras region.

We applied several machine learning methods to the data at hand. From this study, we concluded that ensembles of single and multi-target regression trees perform significantly better than individual trees. Hence, they are further used to generate forestry maps of vegetation height and canopy cover. These can then be used for land-cover and land-use classification, as well as for monitoring and managing ongoing forest processes that affect the stability of forest ecosystems (including spontaneous afforestation, forest reduction and forest fires).

**Paper**:

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., & Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, *5*(4), 256–266. IF=1.351.

**Author's contribution**: For this paper Valentin Gjorgjioski performed some of the experiments in applying PCTs for multi-target regression to estimate forest properties for remotely sensed data. He also prepared some of the corresponding figures. The study was designed and supervised by Sašo Džeroski.

Ecological Informatics 5 (2010) 256–266

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

# Estimating vegetation height and canopy cover from remotely sensed data with machine learning☆

Daniela Stojanova[a], Panče Panov[b,*], Valentin Gjorgjioski[b], Andrej Kobler[a], Sašo Džeroski[b]

[a] Slovenian Forestry Institute, Večna pot 2, SI-1000 Ljubljana, Slovenia
[b] Jožef Stefan Institute, Department of Knowledge Technologies, Jamova cesta 39, SI-1000 Ljubljana, Slovenia

ARTICLE INFO

*Article history:*
Received 14 August 2009
Received in revised form 10 March 2010
Accepted 15 March 2010

*Keywords:*
Remote sensing
LiDAR
Landsat
Vegetation height
Canopy cover machine
Learning

ABSTRACT

High quality information on forest resources is important to forest ecosystem management. Traditional ground measurements are labor and resource intensive and at the same time expensive and time consuming. For most of the Slovenian forests, there is extensive ground-based information on forest properties of selected sample locations. However there is no continuous information of objectively measured vegetation height and canopy cover at appropriate resolution.
Currently, Light Detection And Ranging (LiDAR) technology provides detailed measurements of different forest properties because of its immediate generation of 3D data, its accuracy and acquisition flexibility. However, existing LiDAR sensors have limited spatial coverage and relatively high cost of acquisition. Satellite data, on the other hand, are low-cost and offer broader spatial coverage of generalized forest structure, but are not expected to provide accurate information about vegetation height.
Integration of LiDAR and satellite data promises to improve the measurement, mapping, and monitoring of forest properties. The primary objective of this study is to model the vegetation height and canopy cover in Slovenia by integrating LiDAR data, Landsat satellite data, and the use of machine learning techniques. This kind of integration uses the accuracy and precision of LiDAR data and the wide coverage of satellite data in order to generate cost-effective realistic estimates of the vegetation height and canopy cover, and consequently generate continuous forest vegetation map products to be used in forest management and monitoring.
Several machine learning techniques are applied to this task: they are evaluated and their performance is compared by using statistical significance tests. Ensemble methods perform significantly better than single- and multi-target regression trees and are further used for the generation of forest maps. Such maps are used for land-cover and land-use classification, as well as for monitoring and managing ongoing forest processes (like spontaneous afforestation, forest reduction and forest fires) that affect the stability of forest ecosystems.

## 1. Introduction

In forest management and forestry decision-making there is a continuous need for high quality information on forest resources. The state of forest resources can be monitored by using visualizations of forest properties for a specific spatial region in the form of a map. Forest maps are an effective tool for detecting the state of forest resources and monitoring ongoing spatial processes in forested landscapes. Examples of such processes include the enlargement of forest area by spontaneous afforestation of abandoned agricultural land, and the vertical growth of trees and transitions between developmental stages of existing forest stands. These processes affect the stability of forest ecosystems, an ever more important property due to extreme weather conditions, hydrological stress and the appearance of new diseases and pests.

One of the most important forest properties are: vegetation height and canopy cover. Vegetation height is the height of the vegetation in a stand, relative to the ground. It is a function of the species composition, climate and site quality, and can be used for land-cover classification or in conjunction with vegetation indices. If coupled with species composition and site quality information, vegetation height serves as an estimate of the stand age or the successional stages. Vegetation height is also a useful indicator of forest age and habitat quality. It is an important input variable for ecosystem and forest fire models, and is highly correlated with vegetation biomass and productivity. Biomass is the key component of the carbon circle (Skole and Tucker, 1993) and a surrogate for fuel loading estimation (Finney, 2004).

☆ The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint first authors.
* Corresponding author. Tel.: +386 1 477 3307; fax: +386 1 477 3315.
*E-mail addresses:* daniela.stojanova@gozdis.si (D. Stojanova), pance.panov@ijs.si (P. Panov), valentin.gjorgjioski@ijs.si (V. Gjorgjioski), andrej.kobler@gozdis.si (A. Kobler), saso.dzeroski@ijs.si (S. Džeroski).

1574-9541/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecoinf.2010.03.004

Forest canopy cover is defined as the percent cover of the tree canopy in a stand. It includes the cover from both trees and shrubs, but not herbal vegetation. Canopy cover describes the vertical projection of the tree canopy onto an imaginary horizontal surface representing the ground surface. Forest canopy cover is an ecologically very important forest property because it determines the occurrence and speed of forest regeneration. It is useful for distinguishing different plant and animal habitats, assessing forest floor microclimate, light conditions and estimating other forest variables (e.g., Leaf Area Index). Measurements of canopy cover are essential for silvicultural activities (Jennings et al., 1999).

Traditional ground-based field measurements of forest properties are made by using hand-held equipment. These measurements are expensive, subjective, time consuming and labor intensive, as well as difficult to perform, especially in dense forests (Buckley et al., 1999). Due to these reasons, other methods of estimating forest properties for larger areas are often used, such as remote sensing.

Over the course of the past few decades, remote sensing[1] (RS) has been a valuable source of information in mapping and monitoring forest activities. Remote sensing involves collecting of spatially organized data and information about an area of interest by detecting and measuring signals composed of radiation, particles and fields emanating from objects located beyond the immediate neighborhood of the sensor devices (Franklin, 2001). In this way, it offers a potential for more efficient resource assessment.

Multi-spectral RS is often used to map structural metrics at moderate resolution and broader scale. Multi-spectral satellite imagery is well suited for capturing horizontally distributed (2D) conditions, strictures and changes (Wulder et al., 2008). However, it cannot capture the 3D forest structure directly and is easily influenced by topographical covers and weather conditions.

Light Detection And Ranging (LiDAR) technology, on the other hand, provides horizontal and vertical information (3D) at high spatial resolution and vertical accuracies. It is good for characterizing the vertical structure of vegetation, but has limited spatial coverage mostly due to pricing. By combining remotely sensed data, that describe the horizontal distribution of target phenomena, with LiDAR data, we can improve the measurement, mapping and monitoring of forest properties and provide means of characterizing forest canopy parameters and dynamics.

In this context, many papers have been recently published on the joint use of LiDAR and other active and passive sensors in forest properties estimation problems (Lefsky et al., 1999; Hyde et al., 2006; Maltamo et al., 2006). These studies perform estimation of the forest structure directly from LiDAR measurements and extend them, over limited areas, to spatially homogeneous spectral segments derived from the optical data sets. Medium resolution RS data, such as Landsat images, are relatively inexpensive to acquire over large areas (Franklin and Wulder, 2002), whereas LiDAR covers small areas, at a high cost per unit area (Lim et al., 2003). As a result, these two data types may be combined to generate estimates of vegetation heights and canopy cover over large areas at a reasonable cost (Hudak et al., 2002).

Latest studies (Wulder et al., 2008) of the integration of LiDAR and satellite data point out possible high correlations between different satellite images and forest properties (vegetation height and canopy cover). Hyde et al. (2006) compared the performance of step-wise linear regression models using waveform LiDAR, RaDAR, Landsat, Quickbird and InSAR in a statistical combination of structural information in an attempt to estimate the mean canopy height and biomass. The addition of Landsat ETM+ metrics significantly improved LiDAR estimates of large tree structure — the combination of all sensors is more accurate than using LiDAR alone, but only marginally better than the combination of LiDAR and Landsat ETM+.

Machine learning techniques, such as regression trees, artificial neural networks and support vector machines have been widely used in many remote sensing forestry applications (Lefsky et al., 1999; Moghaddam et al., 2002; Wulder and Seeman, 2003). The typical machine learning task in all these studies is to learn a predictive model that uses a set of remote sensing observations with the aim of predicting the value of forest conditions or properties for unseen cases. The data input to the machine learning system consists of information extracted from different RS data sources, while the output of the system is a predictive model (or a set of predictive models called an ensemble) that describe the forest property.

The main objective of this study is to estimate the vegetation height and canopy cover from an integration of LiDAR and Landsat data in a diverse and unevenly distributed forest. This kind of integration uses the accuracy and precision of LiDAR data and the wide coverage of satellite data in order to generate cost-effective realistic estimation of the forest properties over a geographically large area. The study area is located in the Kras region in western Slovenia, near the border with Italy. The input to the machine learning system are the independent explanatory variables generated from multi-temporal Landsat data and the target variables (representing forest properties that we want to model): The latter are estimated from the 3D LiDAR data and serve as a very good substitute for field-base sample plot measurements. The machine learning system outputs a predictive model of the forest property at hand, which is then used to generate forest vegetation maps that can be used in a variety of forest management applications.

Although forest vegetation maps can be generated with high precision and accuracy purely from LiDAR data, this seems impractical for the nearest future due to the very high cost of high resolution LiDAR data (in our case 4 EUR/ha). On the other hand, the price of Landsat ETM+ data for a multi-temporal coverage is significantly lower (in our case it is free of charge). Using Landsat data as the main data source therefore ensures a very acceptable cost benefit ratio. On the other hand, LiDAR as used here for model calibration seems a very good substitute for field-based sample plot measurements of vegetation height and canopy cover, due to the even higher costs of field measurements which can in some cases also be very difficult and imprecise.

In our preliminary work (Džeroski et al., 2006a,b; Taškova et al., 2006), we introduce the problem of prediction of forest parameters from Landsat and LiDAR data, and present preliminary results using a limited set of machine learning algorithms. The predictive models for estimating the vegetation height and canopy cover from LiDAR and Landsat data, using model and regression trees, pointed out a possible high correlation between satellite data and vegetation properties (Džeroski et al., 2006b). These results were enhanced by using additional machine learning techniques (bagging of model trees) in Taškova et al. (2006).

In this study, we significantly extend and upgrade the work presented in the preliminary work. Here we investigate the performance of a broader set of state-of-the-art machine learning techniques. We confirm the results from our preliminary work by systematically repeating the experiments using the same machine learning techniques. In addition, we apply other state-of-the-art machine learning techniques, i.e., ensemble methods that aim at improving the predictive performance of a given machine learning technique, using single (learning an ensemble for each target variable separately) as well as multi-target setting (learning an ensemble for all target variables together). We use a more carefully chosen experimental methodology that allows extensive comparisons of the predictive performances of all algorithms and perform statistical significance testing. Finally, we use the model with the best predictive power for generation of vegetation height and canopy cover maps of the Kras region of Slovenia and provide a more comprehensive discussion of the experimental results and the use of the map products.

[1] Remote sensing. See also:http://rst.gsfc.nasa.gov (accessed February 11, 2010).

The remainder of the paper is organized as follows. In Section 2, we first describe the data and the methodology used in this study. In Section 3, we then present the results of the modeling process. Next, in Section 4 we present a comparison of the models, discussion on the significance of the results and the map products. Finally, in Section 5 we outline our conclusions and discuss possible directions for further work.

## 2. Materials and methods

### 2.1. Study area

The study area measures 72,226 ha of the Kras region in western Slovenia, in the vicinity of the Adriatic Sea, 5 km from the Gulf of Trieste. The local Gauss–Krueger coordinates of the study area are: $Min.Easting(X) = 389{,}000$, $Max.Easting(X) = 433{,}000$, $Min.Northing(Y) = 37{,}000$ and $Max.Northing(Y) = 86{,}000$.

The relief of the study area is rough with slopes ranging up to 60°, the average slope being 22°. The investigated area covers very diverse and not evenly distributed vegetation. The Kras region has about 40 different types of trees, which includes species such as: *Ostrya carpinifolia* (Hop-hornbeam), *Pinus nigra* (Black pine), *Quercus pubescens* (Downy Oak), *Fraxinus orneus* (South Europea Flowering Ash) and *Fagus syllvatica* (European Beech). In Fig. 1 we present the map of Slovenia on which we mark the area recorded by LiDAR and the Kras region. The study area is encompassed with a black contour line, whereas the study area recorded with LiDAR is covered with black color. The white dots within the LiDAR area present parts not covered with vegetation i.e. denote settlements and were not included in the study.

### 2.2. Data description

#### 2.2.1. Data sources

Passive optical systems such as aerial photography and Landsat, as well as active systems like Radar and LiDAR, provide cost-effective methods of spatial data collection and measurements of forest properties. The suitability of a sensor type for a particular study depends on the scale of study and the nature of the observed objects or processes. In this study, we used the Landsat and LiDAR remote sensing techniques for estimating of the vegetation height and canopy cover.

*2.2.1.1. Landsat.* Landsat 7 Thematic Mapper Plus ETM+[2] is the latest satellite of the Landsat Program designed to collect radiance data in 7 bands (channels) of reflected energy and one band of emitted energy. A well calibrated ETM+ enables one to convert the raw solar energy collected by the sensor to absolute units of radiance. The eight bands of ETM+ data are used to discriminate between Earth surface materials through the development of spectral signatures. Thus, a multi-spectral data set having both high (30 m) and medium to coarse (250 m–1000 m) spatial resolution is acquired on a global basis repetitively and under nearly identical atmospheric and plant physiological conditions. The panchromatic band has spatial resolution of 15 m, while the thermal infrared (TIR) channel has a resolution of 60 m.

*2.2.1.2. LiDAR.* Airborne laser scanning (ALS), also termed airborne LiDAR (Light Detection And Ranging) is an optical remote sensing technology that measures properties of scattered light to find range and/or other information of a distant target. The laser emits a light pulse which is scattered (reflected) from the object back to the sensor. By measuring the round trip time of an emitted laser pulse from the sensor to a reflecting surface and back again, the distance from the sensor to the surface is determined.

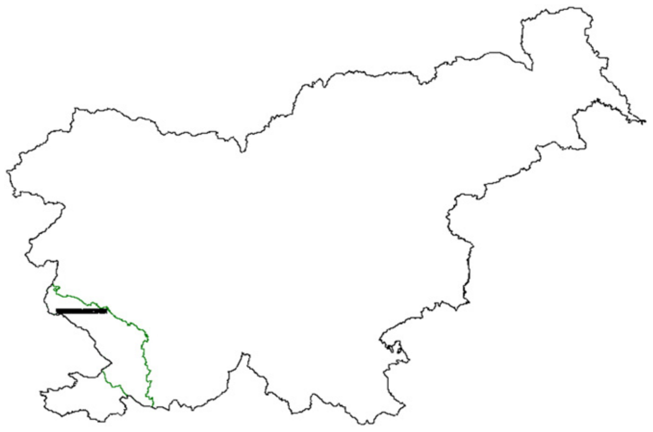


**Fig. 1.** A contour map of Slovenia. The study area is encompassed with a black line whereas the area recorded with LiDAR is presented with black color. The white dots in the LiDAR area present the area not covered with vegetation (e.g., settlements) and these parts were not included in the study.

LiDAR is one of the most promising remote sensing techniques for detailed measurements of forest properties because of its immediate generation of 3D data, self-georeferencing, high spatial resolution (typically 0.5–5 points/m, positional error 10–20 mphcm), accuracy (raging from 15 to 20 cm Root Mean Square Error (RMSE) vertically and 20–30 cm horizontally) and acquisition flexibility.[3] It enables detailed measurements and making of maps with quality comparable to the most passive or active systems. It penetrates through the vegetation layer to the bare ground, enabling structural rendering of vegetation and providing 3D data about objects.

With LiDAR, we can directly define the third dimension of forest layers and the relief under the forest. It is a good source for generation of digital relief models (DEM) and topographical analysis, especially for forested areas, where classical aerophotogrametrical techniques do not give satisfactory accuracy. LiDAR can be used for mapping forest stands, individual tree canopy detection, etc.

#### 2.2.2. Data description and generation of the dataset

The data used in this study consists of multi-spectral multi-temporal Landsat satellite images and 3D LiDAR recordings of the study area. From the Landsat data, we extracted the explanatory variables, while the LiDAR data was used to extract the target variables (forest properties) used in the process of learning the predictive model. The spatial unit of analysis was a 25 m × 25 m square.

*2.2.2.1. Landsat data description.* Multi-spectral Landsat ETM+ data were acquired on August 3rd, 2001, May 18th, 2002, November 10th, 2002, and March 18th, 2003, thus capturing the main phenological stages of forest vegetation in the area. In Fig. 2 we show a part of a Landsat ETM+ band 3′ image, that covers the area recorded with LiDAR, obtained on November 10th, 2002. The Landsat imagery was first geometrically corrected by orthorectification. Image segmentation was then applied. The commercially available eCognition image analysis software, version 2.1 (Definiens Imaging, Munich, Germany) was used for the image segmentation. The software uses a patented procedure for multi-resolution segmentation to extract image objects, exploiting both spatial and spectral information to create objects from image data. The segmentations are typically visually appealing, although the users need to interactively select a useful segmentation level through trial and error (Hay et al., 2003).

[2] Landsat. See also: http://www.trfic.msu.edu/data_portal/Landsat7doc/landsatch5.html (accessed February 11, 2010).

[3] Instrument technical details. See also: http://arsf.nerc.ac.uk/instruments/altm.asp (accessed August 18, 2008).

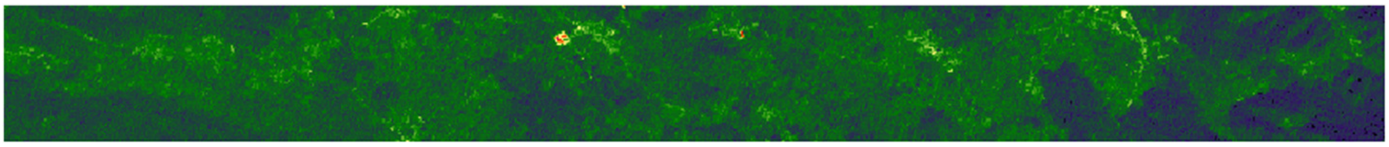


**Fig. 2.** A part of Landsat ETM+ band 3′ image that covers the area recorded with LiDAR acquired on 10.11.2002.

The typical result of image segmentation is extraction of large homogeneous image objects (e.g., meadow), small homogeneity image objects (e.g., forest stand) and small homogeneity objects embedded in a high contrast, especially for data such as Landsat imagery. Each of the four Landsat images was segmented at two levels of spatial detail in order to get realistic object based information that correspond to the real world objects and later serve as information carrier and building block for further analysis. The average image segment sizes were 4 ha for the fine segmentation and 20 ha for the coarse segmentation. Image segmentation is illustrated in Fig. 3 and it represents a segmentation of the Landsat image presented in Fig. 2. It has been derived as a result of fine image segmentation of the third Landsat channel. The objects are given with different colors in order to be distinguishable among each other (the number of objects is around 45,500).

*2.2.2.2. Explanatory variables.* In order to represent and display remote sensed data, we employ basic statistic measures like band mean value, standard deviation and others (Jensen, 2004). The statistic measures can be used further in the analysis of the data directly or indirectly. The link between remote sensing and statistics is strong; clearly, remote sensing can be considered a multivariate problem (Kershaw, 1987) and probabilistic methods constitute one of the most powerful approaches to the analysis of multivariate problems.

Therefore, we generate our explanatory variables from Landsat imagery data based on statistical information for each band. Based on the data within each image segment, four statistic measures (minimum reflectance, maximum reflectance, average reflectance, and standard deviation of reflectance) were computed for each date, for each segmentation level, and for each of the Landsat image channels (2, 3, 4, 5, and 7). Using different segmentation levels, for each example, we take into account two different kinds of neighborhood (narrow and broader). The information about the narrow neighborhood is included with the fine image segmentation level and the broader one is included with the coarse image segmentation level. In this way, we obtain 160 explanatory variables to be used in the predictive modeling. As the borders of individual segments were not identical between dates and segmentation levels, values of the 160 variables were attributed back to individual image pixels, each with dimension 25 m × 25 m.

*2.2.2.3. LiDAR data description.* An east–west transect measuring 2 km × 20 km (highlighted in black in Fig. 1) across a representative part of the Kras region was flown over by LiDAR, in the spring of 2005. The equipment included Optech ALTM 3100 LiDAR flown on a Eurocopter EC-120 B “Colibri” helicopter. The device collects 33,000 laser observations per second in standard operating mode, measuring height, first, intermediate, only and last returns, angle, radian and intensity data. From an operating altitude of 1000 m, the resulting height data has an absolute root mean squared error better than ± 15 cm. The average point cloud density of the LiDAR dataset was 7.5 points/m$^2$, thus 4687.5 discrete 3D LiDAR returns were contained on average in each 25 m × 25 m square.

*2.2.2.4. Target variables.* The target variables were computed from the LiDAR data, at the level of 25 m × 25 m squares corresponding to Landsat pixels. The vegetation height ($H$) for each square (or Landsat pixel) was computed by averaging the heights of the LiDAR-based normalized digital surface model (nDSM) within the 25 m × 25 m square. A nDSM is a high resolution raster map showing the relative height of vegetation above the bare ground. Our nDSM had a horizontal resolution of 1 m$^2$ and was computed using the REIN (REpetitive INterpolation) algorithm for calculation of a Digital Terrain Model (DTM) (Kobler et al., 2007). The REIN algorithm was developed for generating DTMs under forest cover in steep terrain using dense LiDAR data (≥5 points/m$^2$): In such conditions, other filtering algorithms typically have problems distinguishing between ground returns and off-ground points reflected in the vegetation. A field validation of the nDSM on a sample of 120 trees confirmed a vertical RMS error of 0.36 m and a vertical bias of −0.71 m.

The canopy cover (CC) within this study is defined as the percentage of bare ground within 25 m × 25 m (or a Landsat pixel), covered by the vertical projection of the overlying vegetation, higher than 1 m. The canopy cover for each Landsat pixel was computed as the ratio of the heights of the LiDAR-based normalized digital surface model (nDSM) that exceeded 1 m relative height difference between the bare ground of the digital terrain model and the surface of the Landsat pixel. The canopy cover for each 25 m square was computed as the percentage of vegetation within a pixel. The values of the canopy cover are in the interval 0–100%.

## 2.3. Machine learning methodology

Predictive modeling is a machine learning task concerned with predicting the value of one or more dependent variables (classes, targets) from the values of independent variables (explanatory variables). If the target variable is continuous, the task at hand is called regression. If the target is discrete (it has a finite set of nominal values), the task at hand is called classification. The tasks of classification and regression are the two most commonly addressed predictive modeling tasks in machine learning.

In predictive modeling, a set of data records is taken as input to a predictive modeling algorithm, and a predictive model (or set of predictive models called an ensemble) is generated as an output. This model can then be used to predict values of the target variable for new data. If we are predicting a value of a single-target variable, then we

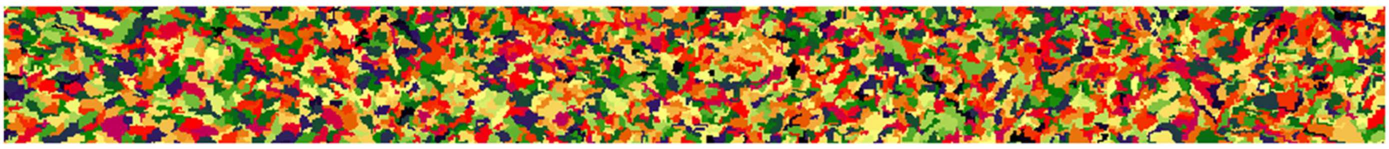


**Fig. 3.** Fine image segmentation of the Landsat ETM+ band 3′ image acquired on 10.11.2002 (presented in Fig. 2).

have a single-target prediction task. In the case when we predict the values of several target variables simultaneously with one model, we have a multi-target prediction task.

In this study, the machine learning task is to learn a predictive model (or a set of models) for predicting vegetation height and canopy cover from an integration of LiDAR and Landsat data. This is a multi-target prediction task. The target variables are derived from the LiDAR data and the explanatory variables are extracted from the Landsat images.

### *2.3.1. Single-target prediction: decision, regression and model trees*

Decision tree learning (Quinlan, 1986) is one of the most widely used methods for inductive learning. A decision tree is a hierarchical structure, where the internal nodes contain tests on the descriptive variables. Each branch of an internal test corresponds to an outcome of the test, and the prediction for the value of the target variable is stored in a leaf. To obtain a prediction for a new data record, the record is sorted down the tree, starting from the root (the top-most node of the tree). For each internal node that is encountered on the path, the test is stored in the applied node. Depending on the outcome of the test, the path continues along the corresponding branch. The resulting prediction of the tree is taken from the leaf at the end of the path.

A decision tree is usually constructed with a recursive partitioning algorithm from a training set of records. The algorithm is known as Top-Down Induction of Decision Trees (TDIDT). The records include measured values of the descriptive and the target attributes. The tests in the internal nodes of the tree refer to the descriptive,while the predicted values in the leaves refer to the target attributes.

The TDIDT algorithm starts by selecting a test for the root node. Based on this test, the training set is partitioned into subsets according to the test outcome. In the case of binary trees, the training set is split into two subsets: one containing the records for which the test succeeds (typically the left subtree) and the other containing the records for which the test fails (typically the right subtree). This procedure is recursively repeated to construct the subtrees.

The partitioning process stops when a stopping criterion is satisfied (e.g., the number of records in the induced subsets is smaller than some predefined value; the length of the path from the root to the current subset exceeds some predefined value, etc.). In that case, the predicted value is calculated and stored in a leaf. The predicted value is the mean value of the target variable calculated over the records that are sorted into the leaf.

One of the most important steps in the tree induction algorithm is the test selection procedure. For each node a test is selected by using a heuristic function computed on the training data. The goal of the heuristic is to guide the algorithm toward smaller trees with good predictive performance.

Regression trees are decision trees that predict the value of a numeric target attribute (Breiman et al., 1984). Each leaf of a regression tree contains a constant value as a prediction for the target variable, as regression trees represent piece-wise constant functions. If the leaf contains a linear regression model that predicts the target value of examples that reach the leaf, the decision tree in question is called a model tree (Quinlan, 1992). Model trees have advantages over regression trees in both compactness and prediction accuracy, and the ability to exploit local linearity in the data. Another advantage over regression trees is that model trees can extrapolate the predicted value outside the range observed in the training cases. In this paper, we use *M5′* regression and model tree algorithm implementation from the WEKA environment (Witten and Frank, 2005).

### *2.3.2. Multi-target prediction: multi-target regression trees*

Multi-target regression trees (Blockeel, 1998; Struyf and Džeroski, 2006) are a generalization of regression trees for the prediction of several numeric target variables simultaneously. The leaves of a multi-target regression tree store a vector of numeric values, instead of storing a single numeric value. Each component of this vector is a prediction for one of the target attributes. The components of the prediction vector are the means of the target variables calculated over the records that are stored in the leaf. The main advantages of multi-target regression trees (over building a separate model for each target) are: (1) a multi-objective model is smaller than the total size of the individual models for all target variables, and (2) such a multi-objective model explicates dependencies between the different target variables.

In this paper, we use the CLUS (Blockeel and Struyf, 2002; Struyf and Džeroski, 2006) system for constructing (multi-target) regression trees. The heuristic used for selecting the attribute tests (that define the internal nodes) in this algorithm is the intra-cluster variance summed over the subsets induced by the test. The variance function is standardized so that the relative contribution of the different targets to the heuristic score is equal. Lower intra-subset variance results in predictions that are more accurate.

### *2.3.3. Ensembles*

An ensemble method constructs a set of predictive models called an ensemble (Dietterich, 2000). An ensemble gives a prediction for a new data record by combining the predictions of the individual models for that data record. For regression tasks, the final prediction can be obtained by averaging the output predictions of the models in the ensemble. The learning of ensembles consists of two steps. In the first step, we have to learn the base models that make up the ensemble. In the second step, we have to figure out how to combine these models (or their predictions) into a single coherent model (or prediction).

When learning base models it makes sense to learn models that are accurate and diverse (Hansen and Salamon, 1990). Accurate models perform better than random guessing on new examples, and diverse models make different prediction errors on new examples. The diversity in an ensemble can be introduced in different ways: by manipulating the training set (e.g., bootstrap sampling, change of weights of the data instances) or by manipulating the learning algorithm used to obtain the base models (e.g., introducing random elements in the algorithm).

Ensemble methods aim at improving the predictive performance of a given machine learning technique. They aim to improve the predictive performance of their base classifier when used in a single-target setting (learn an ensemble for each target attribute separately) (Breiman, 1996, 2001). In Kocev et al. (2007), it is shown that this applies also for the multi-target setting (learn one ensemble for all target attributes). In addition, the ensembles for multi-target prediction should be preferred because they are faster to learn. In this work, we use bagging and random forests, the two most widely used ensemble methods to produce ensembles of regression trees and multi-target regression trees.

*2.3.3.1. Bagging.* Bagging (Breiman, 1996) is an ensemble method that constructs the different base models by making bootstrap replicates of the training set and using them to build the individual models. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set. The bootstrap sample and the training set have an equal number of instances. Bagging can give substantial gains in predictive performance, when applied to an unstable learner (i.e., a learner for which small changes in the training set result in large changes in the predictions), such as classification and regression tree learners.

*2.3.3.2. Random forest.* A random forest (Breiman, 2001) is an ensemble of trees, where the diversity among the individual trees is obtained from two sources: (1) by using bootstrap sampling and (2) randomization of the selection step of the TDIDT algorithm. At each node in the decision tree, a random subset of the input attributes is taken and the best split is selected from this subset. The size of the random subset is given by a function of the number of descriptive attributes. Prediction is made by aggregation (majority vote for classification or averaging for regression) of the predictions of the individual models in the ensemble.

## 3. Results

### 3.1. Experimental design

#### 3.1.1. Dataset

The dataset consists of 160 explanatory variables and 2 target variables. The explanatory variables are derived from Landsat data for two levels of image segmentation, as explained in Section 2. The target variables are: vegetation height (*H*) and canopy cover (CC), derived from LiDAR data. There are 64,000 examples of which 60,607 describe the vegetation outside a settlement and are used in the process of learning.

#### 3.1.2. The learning algorithms

In this study, one of the objectives is to study the predictive performance of state-of-the art machine learning algorithm, for the task of prediction of vegetation height and canopy cover. The problem of prediction of forest properties inherently represents a multi-target learning problem: it can be solved by using algorithms that build a single-target model for each forest property separately or by using algorithms that build a multi-target model for both forest properties at the same time. Another dimension of comparison of the predictive performance is using single models or ensemble of models. In this study, we investigate this dimension by performing experiments for single-model prediction and state-of-the-art ensemble prediction (e.g., bagging and random forests) both in the single-target and multi-target setting.

We use implementations of the state-of-the-art algorithms from two open source machine learning systems: WEKA (Witten and Frank, 2005) and CLUS[4] (Blockeel and Struyf, 2002; Struyf and Džeroski, 2006). In total, we performed experiments using 9 different algorithms. First, we performed experiments using algorithms that have a single model as an output. We used the implementations of regression tree (wRT) and model tree (wMT) algorithm in the WEKA system and single-target (STRT) and multi-target regression trees (MTRT) implemented in the CLUS system. Next, we performed experiments using ensemble learning algorithms that produce a set of models. In this case, we used the implementations of the bagging method from WEKA using model trees as base-level learners (wBagMT), and bagging and random forests of CLUS regression trees (as base learners) in the CLUS system both in the single-target (BagSTRT and RFSTRT) and multi-target setting (BagMTRT and RFMTRT).

The experiments were performed by using the default parameter settings for all the algorithms. Single-target regression trees and multi-target regression trees from the CLUS system are built with the default heuristic (intra-cluster variance) and default pruning method (M5 pruning). The minimal number of examples for the method to form a leaf was 4 examples. The settings for ensembles include the default pruning method, the number of variables in variable selection for random forest was set to 5 variables (calculated using the suggestion by Breiman, 2001), the default ensemble size of 10 and the default voting type for regression (the mean value).

#### 3.1.3. Evaluation and comparison

Evaluation of the models was performed using the standard 10 fold cross-validation evaluation method. All the algorithms were evaluated on the same folds, in order to allow comparison of the results and statistical significance testing. We use two regression evaluation measures to estimate and discuss the predictive performance of the models: correlation and root mean squared error. Correlation (Corr) indicates the strength and direction of a linear relationship between two random variables and is usually expressed through the Pearson correlation coefficient. Root mean squared error (RMSE) is a frequently-used measure of the differences between values predicted by a model of an estimator and the target values actually observed.

To compare the performance of the different algorithms, we use the corrected Friedman test (Friedman, 1940; Iman and Davenport, 1980). The evaluation measure for each fold of the cross-validation represents a data point for the statistical test. The test is performed on each target variable (*H* and CC) separate for each evaluation measure (Corr and RMSE).

The Friedman nonparametric test first ranks the algorithms for each dataset (fold), the best performing algorithm getting the rank of 1. It then compares the average ranks of the algorithms across datasets (folds). The null-hypothesis, which states that all the algorithms are equivalent and so their ranks should be equal.

If the null-hypothesis is rejected, we can proceed with a post-hoc test. The Nemenyi (1963) test is used when in our case, since all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference CD. The results of this test are visualized by using the average rank diagrams on which the critical distance is also depicted (Demšar, 2006). We consider the differences in performance significant if the standard *p*-value is below the threshold of 0.05.

### 3.2. Results — predictive performance

Here, we present the predictive performance of the obtained models in terms of two evaluation measures (Corr and RMSE) for both target variables. The results, presented in Tables 1 and 2, are represented with the corresponding confidence intervals, to show the stability of the used algorithms. We can note that the confidence intervals in both tables are small, due to the size of the dataset (60,607 examples). In Tables 1a and 2a we list the performance for algorithms that produce single models as output, and in Tables 1b and 2b we list the performance of ensemble algorithms.

To check whether the differences in performances are statistically significant, we used the corrected Friedman test for multiple hypothesis testing. To detect which algorithms perform significantly better or worse than others, we used the Nemenyi post hoc test. The results of this procedure are presented in the form of average rank diagrams in Fig. 4, for each target variable and each evaluation measure. The ranks are depicted on the axis in such a manner that the best ranking algorithms are at the right-most side of the diagram. The critical difference (CD) interval, for a significance level of 0.05, is computed by the Nemenyi test and is plotted in the upper left corner; algorithms whose average rank difference is larger than this critical difference can be considered significantly different with 95% probability. The algorithms that do not differ significantly are connected with a line.

The Nemenyi test shows (Fig. 4a and b) that the best performing algorithms are ensemble methods and in particular random forests of multi-target regression trees (RFMTRT), while the worst performing algorithms are single-model algorithms. The test shows that the performance of the ensemble methods, in terms of correlation coefficient, is significantly better than the one of single-model methods. If we compare the multi-target methods, we can see that random forests of multi-target regression trees perform statistically better than individual multi-target regression trees: in the case of bagging, the difference is not statistically significant. Similar conclusions can be drawn if instead of the results for correlation we

[4] The system is available at http://www.cs.kuleuven.be/dtai/clus/ (accessed August 18, 2008).

**Table 1**
Comparison of correlation coefficients of the predictive models for both target variables: *a*) Single model algorithms (wRT — WEKA Regression Tree; wMT — WEKA Model Tree; STRT — CLUS Single Target Regression Tree; MTRT — CLUS Multi-target Regression Tree); *b*) Ensemble algorithms (wBagMT — WEKA Bag of Model Trees; BagSTRT — CLUS Bag of STRTs; RFSTRT — CLUS Random Forest of STRTs; RFMTRT — CLUS Random Forest of MTRTs).

*a*) **Single model algorithms**

| | Single-target | | | Multi-target |
|---|---|---|---|---|
| Target | wRT | wMT | STRT | MTRT |
| *H* | 0.876 ± 0.004 | 0.884 ± 0.004 | 0.874 ± 0.003 | 0.880 ± 0.015 |
| CC | 0.858 ± 0.002 | 0.863 ± 0.004 | 0.851 ± 0.003 | 0.852 ± 0.013 |

*b*) **Ensemble algorithms**

| | Single-target | | | Multi-target | |
|---|---|---|---|---|---|
| Target | wBagMT | BagSTRT | RFSTRT | BagMTRT | RFMTRT |
| *H* | 0.902 ± 0.004 | 0.904 ± 0.003 | 0.906 ± 0.002 | 0.904 ± 0.002 | 0.906 ± 0.002 |
| CC | 0.883 ± 0.002 | 0.880 ± 0.003 | 0.883 ± 0.002 | 0.880 ± 0.002 | 0.883 ± 0.002 |

**Table 2**
Comparison of RMSE of the predictive models for both target variables: *a*) Single model algorithms (wRT — WEKA Regression Tree; wMT — WEKA Model Tree; STRT — CLUS Single Target Regression Tree; MTRT — CLUS Multi-target Regression Tree); *b*) Ensemble algorithms (wBagMT — WEKA Bag of Model Trees; BagSTRT — CLUS Bag of STRTs; RFSTRT — CLUS Random Forest of STRTs; RFMTRT — CLUS Random Forest of MTRTs).

*a*) **Single model algorithms**

| | Single-target | | | Multi-target |
|---|---|---|---|---|
| Target | wRT | wMT | STRT | MTRT |
| *H* [m] | 2.336 ± 0.035 | 2.271 ± 0.038 | 2.361 ± 0.025 | 2.373 ± 0.038 |
| CC [%] | 16.068 ± 0.051 | 15.758 ± 0.129 | 16.481 ± 0.151 | 14.708 ± 0.108 |

*b*) **Ensemble algorithms**

| | Single-target | | | Multi-target | |
|---|---|---|---|---|---|
| Target | wBagMT | BagSTRT | RFSTRT | BagMTRT | RFMTRT |
| *H* [m] | 2.091 ± 0.038 | 2.071 ± 0.029 | 2.056 ± 0.030 | 2.070 ± 0.028 | 2.054 ± 0.029 |
| CC [%] | 14.723 ± 0.079 | 14.868 ± 0.125 | 14.713 ± 0.105 | 14.891 ± 0.109 | 14.708 ± 0.108 |

consider the results for RMSE (see Fig. 4c and d). In general, RFMTRT constructed from the CLUS system perform significantly better than any of the individual trees. The only exception to this is the RMSE for canopy cover, where multi-target regression trees (MTRT) have the same rank as RFMTRT.

### 3.3. Results — maps of vegetation height and canopy cover

The second objective of our work is to produce maps of vegetation height and canopy cover using the predictive models obtained in the study. For that purpose, we used RFMTRT, which is the best performing method according to predictive performance, to generate maps. This model was built using the entire dataset of 60,607 examples, from the representative part of the Kras region (containing variety of different vegetations) for which we have both Landsat and LiDAR data available. Next, we translated the RFMTRT model into functions in the PYTHON[5] programming language, that were later on used in the GIS (Geographical Information System) system to visualize the predictions in the form of a map. Finally, we generated maps of vegetation height (see Fig. 5) and canopy cover (see Fig. 6) by applying the PYTHON functions to the whole Kras region, thus extrapolating the predictions of the model built on the smaller representative part of the region using Landsat data available for the whole region.

[5] http://www.python.org/ (accessed on August 18, 2008).

## 4. Discussion

In this study, we compare several machine learning methods on the task of estimating vegetation height and canopy cover by using LiDAR and Landsat data. To this end, we redesigned the experiments from the first two preliminary studies (Džeroski et al., 2006b; Taškova et al., 2006). We tested additional machine learning methods in order to improve the accuracy of the predictive models. Beside single- and multi-target regression trees used in the previous studies, we also use single- and multi-target ensemble methods.

The best results are obtained using the RFMTRT algorithm, random forests of multi-target regression trees. Ensemble methods improve the accuracy of the predictive models. Moreover, the ensembles for multi-target prediction should be preferred because theyare faster to learn and predict more than one variable at the same time.

All ensemble methods perform better than the single model algorithms (wMT, wRT, STRT and MTRT) used. An exception is the performance in terms of the RMSE for canopy cover where MTRT have the same performance as RFMTRT. The average rank diagram shows that random forests created by CLUS system perform best in all four cases (see Fig. 4). The difference of the performance between ensembles of different types of trees is insignificant.

The results from this study are better than results presented in our preliminary work. Džeroski et al. (2006b) reported a correlation of 0.885 and RMSE = 2.25 m for vegetation height and a correlation of 0.861 and RMSE = 0.17 for canopy cover: These were achieved by using model trees. Taškova et al. (2006) reported a

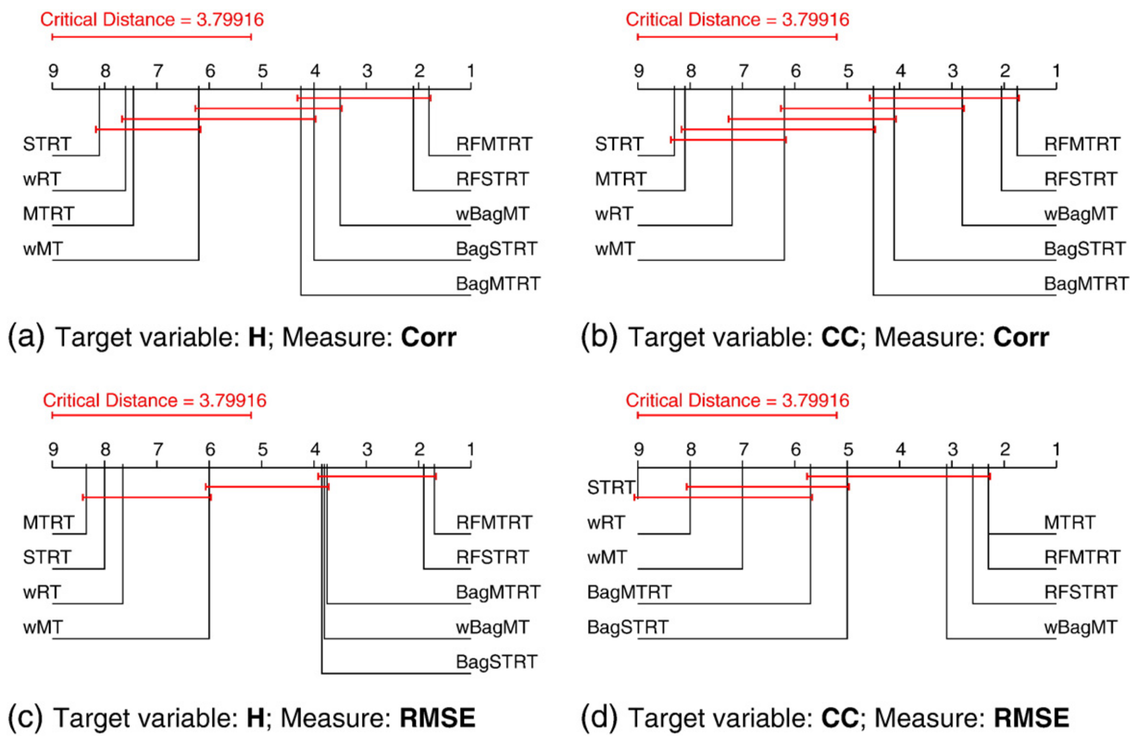


**Fig. 4.** Average ranks diagrams: *a*) target variable — *H* and eval. measure — Corr; *b*) target variable — CC and eval. measure — Corr; *c*) target variable — *H* and eval. measure — RMSE and *d*) target variable — CC and eval. measure — RMSE. Algorithms with lower ranks (far right) perform better. Algorithms whose average rank difference is larger than the critical difference can be considered significantly different with 95% probability. The algorithms that do not differ significantly are connected with a line. Algorithm labels are as follows: wRT — WEKA Regression Tree; wMT — WEKA Model Tree; STRT — CLUS Single-target Regression Tree; MTRT — CLUS Multi-target Regression Tree; wBagMT — WEKA Bag of Model Trees; BagSTRT — CLUS Bag of STRTs; RFSTRT — CLUS Random Forest of STRTs; RFMTRT — CLUS Random forest of MTRTs.

correlation of 0.902 and RMSE = 2.19 m for vegetation height and a correlation of 0.882 and RMSE = 0.238 for canopy cover: These were achieved by using bagging of model trees. The accuracy of the predictive models is improved by using ensemble methods. In this more general study, we obtained higher correlation coefficients and lower error rates. The average error rate (RMSE) of the best models is 2.05 m for the vegetation height and 14% for the canopy cover, whereas the corresponding correlation coefficients are 0.91 and 0.88.

The investigated study area covers very diverse and not evenly distributed vegetation. It was selected by taking into account the diversity and the distribution of the many different vegetation types present in the Kras region. The Kras region has about 40 different types of trees, which includes species such as: *O. carpinifolia* (Hop-hornbeam), *P. nigra* (Black pine), *Q. pubescens* (Downy Oak), *F. orneus* (South Europea Flowering Ash) and *F. syllvatica* (European Beech). The models build using the methodology described in this paper can also serve for estimation of the vegetation height and canopy cover in other study areas with similar vegetation species. The different vegetation types have different influences on the structure and the accuracy of the model. The different combinations of vegetation species will decrease (in most of the cases) the accuracy of the predictions of the model. In case of regions with very diverse vegetation it is preferable to divide the region into smaller subregions and perform modeling in each subregion separately. In addition, special attention when modeling the vegetation properties needs to be focused on the relief of the area.

The generated maps represent a rough, but continuous estimates of the vegetation height and canopy cover over a large spatial area. The precision of the derived maps is lower than the precision of the field measurements done on smaller plots or individual trees within the study area (see field validation of the nDSM in Section 2.2.2). Therefore, these maps cannot be used for determination of the growing stock or other individual tree estimates, but can be useful when coverage of a grater spatial area is required.

Such maps can be used as an input for advanced systems such as GIS to improve their planning, managing and monitoring capabilities, in performing a variety of tasks such as land-cover mapping, land-cover classification, land-use mapping, land-use classification, change detection and many other forestry, ecological, geological and military applications. Moreover, the maps can be used for monitoring and managing a variety of ongoing processes in the forest ecosystems that involve enlargement of forest areas by spontaneous afforestation of abandoned agricultural land, forest area reduction, urban rapprochement, as well as vertical growth and gradual closing of canopy cover of existing forest stands. These maps can be used in the process of monitoring the forest biomass accumulation and $CO_2$ sink in the Kyoto framework.[6] Furthermore they can be used in estimating the risk of forest fire outbreaks.

In addition, these maps can also serve for temporal comparisons. Finally, due to their spatial continuity (unlike the discrete sampling layout of current forest monitoring schemes) potential applications also include the study of forest habitats and transitional agricultural-forest habitats, visual landscape assessments, land-use suitability analysis, visibility analysis for cell phone networks etc. The methodology used in this study integrates remote sensing, forestry and machine learning techniques and can be a powerful tool for diverse mapping and modeling applications in the future.

## 5. Conclusions

In this study, we focus on the estimation of forest properties (forest vegetation height and canopy cover) from remotely sensed data over a large geographical area (the study area measures 72,226 ha of the Kras

[6] Kyoto protocol: http://unfccc.int/resource/docs/convkp/kpeng.html, (accessed August 18, 2008).

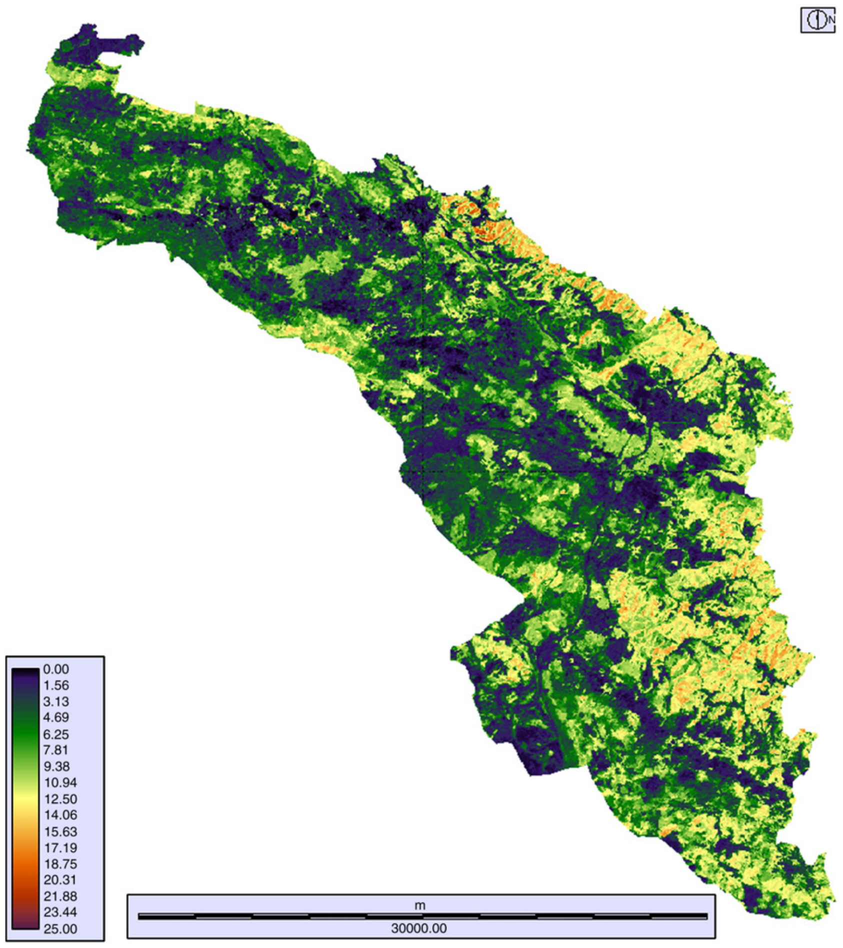


**Fig. 5.** Map of vegetation height for the Kras region generated by using a random forest of multi-target regression trees model. The legend shows the vegetation height in meters.

region in western Slovenia in the vicinity of the Adriatic Sea), by integrating LiDAR and Landsat satellite data and generating predictive models of forest properties. We use machine learning methods for predictive modeling and apply a set of state-of-the-art machine learning techniques. To model the forest properties we focused on two dimensions: modeling the parameters with individual models or ensembles (single model prediction and ensemble prediction) and modeling the target properties separately or simultaneously (single-target and multi-target prediction). The results show the advantages of multi-target over single-target regression, as multi-target models have a smaller size and are faster to learn and apply, and the advantage of ensemble prediction over single model prediction in terms of predictive performance.

Several contributions are presented in this study. First, we use state-of-the-art machine learning methodology to model forest properties, in contrast to the simple statistical methods and linear regression used in similar studies (Hyde et al., 2006). Second, we achieved better results in terms of higher correlation coefficients and lower RMSE errors compared to the results obtained in our preliminary work (Džeroski et al., 2006b; Taškova et al., 2006). Also, we perform modeling of the forest properties in diverse forests, as opposed to modeling of homogeneous forests. Next, we use multi-temporal multi-spectral Landsat data, obtained in different vegetation seasons, instead of mono-temporal data used in similar studies. Finally, we use the accurate and precise LiDAR data to learn models for the representative part of a region and then we extrapolate the predictions on a larger area using less expensive remote sensing Landsat data.

The derived models represent a key piece of infrastructure required in support of sustainable forest management. They serve to generate forest vegetation map products for a large geographical area. Although such maps could be generated with exceeding precision and accuracy purely from LiDAR data, this seems impractical for the foreseeable future due to the very high cost of high resolution LiDAR data. Using Landsat data as the main data source therefore ensures a very acceptable cost benefit ratio. Moreover, using LiDAR for model calibration seems a very good replacement for sample plot field measurements of vegetation height and canopy cover, due to the even higher costs and difficulty or imprecision of the field measurements.

In future work, we first plan to investigate different image segmentation algorithms and to see what is the influence of segmentation on the overall predictive performance. Moreover, we would like to use other preprocessing methods and techniques and combine them with domain-based knowledge (e.g., image clustering, geo-ontologies). Second, we want to incorporate the spatial correlation and the spatial autocorrelation in the predictive models. Finally, we plan to expand the forest maps to broader areas (i.e., country level). We will evaluate the predictions of the machine learning models on different study areas and explore the influence of diverse vegetation and land-cover types on the accuracy of the results.

## Acknowledgment

The acquisition of the LiDAR and satellite data was done within the project “Processing LiDAR data (development and usage of algorithms for mapping and estimation of forest stand biomass and structure using LiDAR and digital multispectral imagery) No. L2-6575, 2004–2007” funded by the Ministry of Education, Science and Sport of Republic of Slovenia.

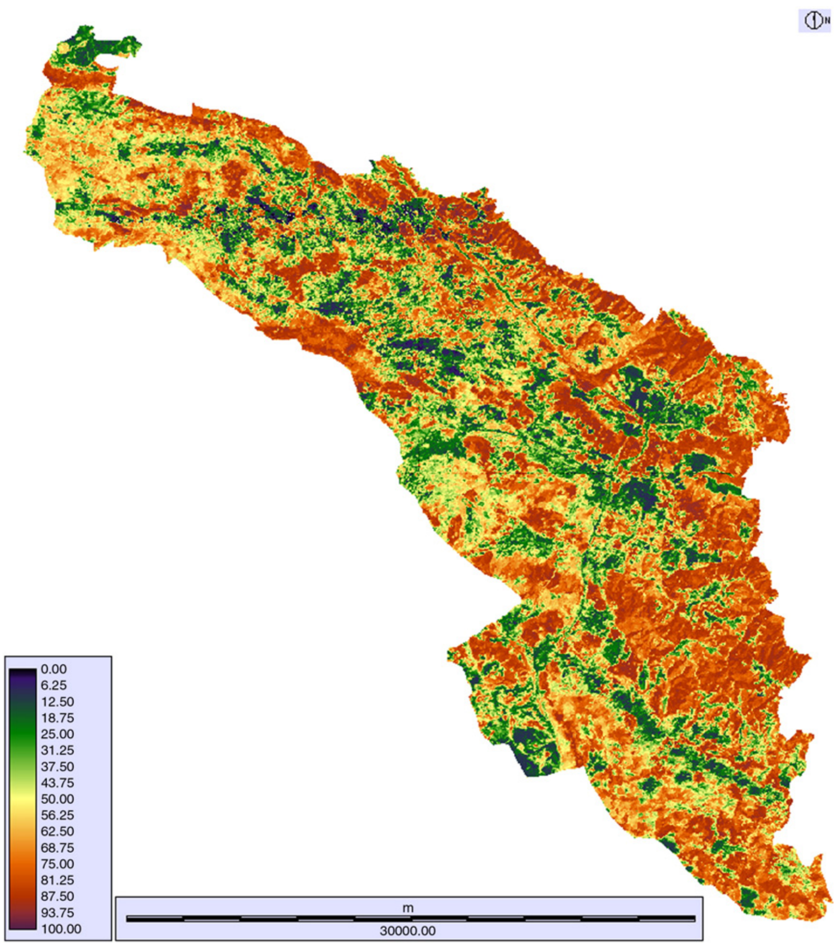


**Fig. 6.** Map of canopy cover for the Kras region generated by using a random forest of multi-target regression trees model. The legend shows the percentage of canopy cover.

## References

Blockeel, H., 1998. Top-down induction of first order logical decision trees. Ph.D. thesis, Katholieke Universiteit Leuven, Belgium.
Blockeel, H., Struyf, J., 2002. Efficient algorithms for decision tree cross-validation. Journal of Machine Learning Research 3, 621–650 Dec.
Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.
Breiman, L., 2001. Random forests. Machine Learning 45 (1), 5–32.
Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984. Classification and Regression Trees. The Wadsworth Statistics/Probability Series. Chapman & Hall/CRC.
Buckley, D.S., Isebrands, J., Sharik, T.L., 1999. Practical field methods of estimating canopy cover, PAR, and LAI in Michigan Oak and pine stands. Northern Journal of Applied Forestry 16 (1), 25–32.
Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.
Dietterich, T.G., 2000. Ensemble methods in machine learning. MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems. Springer-Verlag, London, UK, pp. 1–15.
Džeroski, S., Kobler, A., Gjorgjioski, V., Panov, P., 2006a. Predicting forest stand height and canopy cover from Landsat and Lidar data using data mining techniques. Poster Presentation at Second NASA Data Mining Workshop: Issues and Applications in Earth Science, May 23–24, Pasadena, PA.
Džeroski, S., Kobler, A., Gjorgjioski, V., Panov, P., 2006b. Using decision trees to predict forest stand height and canopy cover from Landsat and LiDAR data. In: Tochtermann, K., Scharl, A. (Eds.), Managing Environmental Knowledge: EnviroInfo 2006: Proceedings of the 20th International Conference on Informatics for Environmental Protection. Aachen: Shaker Verlag, Graz, Austria, pp. 125–133. September.
Finney, M.A., 2004. FARSITE: fire area simulator-model development and evaluation. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station., Res. Pap. RMRS-RP-4, Ogden, UT.
Franklin, S.E., 2001. Remote Sensing for Sustainable Forest Management. Lewis Publishers.
Franklin, S., Wulder, M., 2002. Remote sensing methods in medium spatial resolution satellite data land cover classification of large areas. Progress in Physical Geography 26 (2), 173–205.
Friedman, M., 1940. A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics 11 (1), 86–92.
Hansen, L., Salamon, P., 1990. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (10), 993–1001.
Hay, G.J., Blaschke, T., Marceau, D.J., Bouchard, A., 2003. A comparison of three image-object methods for the multiscale analysis of landscape structure. ISPRS Journal of Photogrammetry and Remote Sensing 57 (5), 327–345 April.
Hudak, A., Lefsky, M., Cohen, W., Berterretche, M., 2002. Integration of LiDAR and Landsat ETM+ data for estimating and mapping forest canopy height. Remote Sensing of Environment 82 (2), 397–416.
Hyde, P., Dubayah, R., Walker, W., Blair, J.B., Hofton, M., Hunsaker, C., 2006. Mapping forest structure for wildlife habitat analysis using multi-sensor (LiDAR, SAR/InSAR, ETM+, Quickbird) synergy. Remote Sensing of Environment 102 (1–2), 63–73.
Iman, R., Davenport, J., 1980. Approximations of the critical region of the Friedman statistic. Communications in Statistics - Theory and Methods A9 (6), 571–595.
Jennings, S., Brown, N., Sheil, D., 1999. Assessing forest canopies and understory illumination: canopy closure, canopy cover and other measures. Forestry 72 (1), 59–74.
Jensen, J.R., 2004. Introductory digital image processing, 3rd Edition. Prentice Hall Series in Geographic Information Science. Prentice Hall.
Kershaw, C., 1987. Discrimination problems for satellite images. International Journal of Remote Sensing 8 (9), 1377–1383.
Kobler, A., Pfeifer, N., Ogrinc, P., Todorovski, L., Oštir, K., Džeroski, S., 2007. Repetitive interpolation: a robust algorithm for DTM generation from Aerial Laser Scanner Data in forested terrain. Remote Sensing of Environment 108 (1), 9–23.
Kocev, D., Vens, C., Struyf, J., Džeroski, S., 2007. Ensembles of multi-objective decision trees. Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Proceedings. : Lecture Notes in Computer Science, Vol. 4701. Springer, pp. 624–631.
Lefsky, M.A., Cohen, W.B., Hudak, A., Acker, S.A., Ohmann, J.L., 1999. Integration of LIDAR, Landsat ETM+ and forest inventory data for regional forest mapping. Proceedings of the ISPRS Workshop Mapping Surface structure and topography by airborne and spaceborne lasers: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXII-3/W14.
Lim, K., Treitz, P., Wulder, M., St-Onge, B., Flood, M., 2003. LiDAR remote sensing of forest structure. Progress in Physical geography 27 (1), 88–106.

Maltamo, M., Malinen, J., Packalén, P., Suvanto, A., Kangas, J., 2006. Nonparametric estimation of stem volume using airborne laser scanning, aerial photography, and stand-register data. Canadian Journal of Remote Sensing 36 (2), 426–436.
Moghaddam, M., Dungan, J., Acker, S., 2002. Forest variable estimation from fusion of SAR and multispectral optical data. IEEE Transactions on Geoscience and Remote Sensing 40 (10), 2176–2187.
Nemenyi, P., 1963. Distribution-free multiple comparisons. Ph.D. thesis, Princeton University, Princeton, NY, USA.
Quinlan, J.R., 1986. Induction of decision trees. Machine Learning 1 (1), 81–106.
Quinlan, J.R., 1992. Learning with continuous classes. Proceedings of the 5th Australian Joint Conference on Artificial Intelligence. World Scientific, pp. 343–348.
Skole, D., Tucker, C., 1993. Tropical deforestation and habitat fragmentation in the Amazon: satellite data from 1978 to 1988. Science 260 (5116), 1905–1910.
Struyf, J., Džeroski, S., 2006. Constraint based induction of multi-objective regression trees. Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID'05, Revised, Selected and Inductive Papers. : Lecture Notes in Computer Science, Vol. 3933. Springer, pp. 222–233.
Taškova, K., Panov, P., Kobler, A., Džeroski, S., Stojanova, D., 2006. Predicting forest stand properties from satellite images with different data mining techniques. Proceedings of the 9th International Multiconference Information Society IS 2006, 9th–14th October 2006, Ljubljana, Slovenia, pp. 259–262.
Witten, I., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition. Morgan Kaufmann, San Francisco.
Wulder, A., Seeman, D., 2003. Forest inventory height update through the integration of Lidar data with segmented Landsat imagery. Canadian Journal of Remote Sensing 29 (5), 536–543.
Wulder, M., White, J., Hay, G., Castilla, G., 2008. Object-based image analysis. Lecture Notes in Geoinformation and Cartography. Springer, Berlin, pp. 345–363. Ch. Pixels to objects to information: Spatial context to aid in forest characterization with remote sensing.

# Chapter 7

# Comparison of Distances for Multi-Label Classification with PCTs

In Section 5.4, we discussed the instantiation of the predictive clustering trees paradigm for predicting sets of discrete values. Using the notation in this dissertation this datatype can be represented as $Set[Dicrete]$.

This datatype appears within the task of multi-label classification. Namely, each instance in multi-label classification is associated with a value of this type, i.e., with a set of labels. The task of multi-label classification has received significant attention in the research community over the past few years and a plethora of various methods have been proposed.

There is a variety of distances that could be applied on the Sets datatype (given in Section 2.2.2.2). We need to select distances that are most relevant for sets of discrete values, i.e., for the task of multi-label classification. In this study, we selected four distances applicable to this task: Euclidean, Jaccard, Hamming and Matching distance. We use these distances to instantiate the variance function of predictive clustering trees.

We evaluate the four distances on 6 benchmark datasets for multi-label classification. The selected datasets have various properties in terms of number of instances, number of labels, number of labels per instance, etc. We used 6 different evaluation measures to assess the predictive performance of the trees with the different distances.

The results reveal that there is no overall best distance measure. Furthermore, the best choice for a distance measure is the one that optimizes a selected evaluation measure. All in all, the Euclidean distance and Hamming loss perform the best when the performance is averaged across all evaluation measures.

**Paper**:

Gjorgjioski, V., Kocev, D., & Džeroski, S. (2011). Comparison of distances for multi-label classification with PCTs. In *Proceedings of the 14th International Multi-conference Information Society* (pp. 121–124). Institut Jožef Stefan, Ljubljana.

**Author's contribution**: Valentin Gjorgjioski implemented the extension of CLUS to deal with multi-label classification as a problem of predicting sets of discrete class values. He also performed all of the experiments on the six datasets. Finally, he wrote the first draft of the paper and took the comments from the co-authors into account. This study was designed and supervised by Sašo Džeroski.

# COMPARISON OF DISTANCES FOR MULTI-LABEL CLASSIFICATION WITH PCTs

*Valentin Gjorgjioski, Dragi Kocev, Sašo Džeroski*
Jozef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia
e-mail: valentin.gjorgjioski@ijs.si

## ABSTRACT

Multi-label classification has received significant attention in the research community over the past few years: this has resulted in the development of a variety of multi-label classification methods. These methods either transform the multi-label dataset to several simpler datasets or adapt the learning algorithm so it can handle the multiple labels. In this paper, we consider the latter approach. Namely, we use predictive clustering trees to perform multi-label classification. Furthermore, we perform an experimental comparison of four distance measures used to select the splits in the nodes of the trees. The experimental evaluation was conducted on 6 benchmark datasets using 6 different evaluation measures. The results show that, averaged overall, the Euclidean distance and the Hamming loss yield the best predictive performance.

## 1 INTRODUCTION

Traditionally, binary classification is concerned with deciding whether a given example has (or doesn't have) a single given target property/class. Multi-class classification involves the labeling of a given example with a single label/class $\lambda_i$ from a finite set of disjoint labels $L = \{ \lambda_1, \lambda_2, \ldots, \lambda_Q\}$, Q>2. In contrast, multi-label classification learns a mapping from an example in the input space ($x \in X$) to a set of labels ($Y \subseteq L$) from the output space $L$. Note that, unlike in multi-class classification, in multi-label classification the labels are not mutually exclusive, i.e., a single example can be labeled with multiple labels. The labels that belong to the output $Y$ are called relevant labels, while those from $L \backslash Y$ are called irrelevant for a given example.

The machine learning task of multi-label classification data has lately received significant attention from the research community [1], which has resulted in development of many methods that tackle this task. The developed methods can be generally divided into two categories: problem transformation and algorithm adaptation. Problem transformation methods transform problem into one or more single-label classification problems. These problems are then solved using a commonly used method for single-label classification and, afterwards, the output is transformed back into a multi-label representation. Algorithm adaptation methods adapt the learning algorithms to handle the multi-label data directly. In this work, we focus on algorithm adaptation methods. Specifically, we use predictive clustering trees (PCTs) [2] as classifiers and extend the distance function used when learning the tree. PCTs are a generalization of decision trees that are capable of predicting structured outputs. Namely, PCTs can handle multiple continuous targets, multiple discrete targets, time-series [3] and hierarchies of classes [4]. In the context of multi-label classification, we employ the PCTs for multiple discrete targets where a weighted Euclidean distance is used to generate the tests in the internal nodes of the tree. Here, we extend the PCTs with three distance measures: Hamming distance, Jaccard distance and a matching distance. These distances will provide additional flexibility for the users when they apply PCTs to different domains.

We compare the predictive performances of the PCTs obtained using different distance measures. The predictive performance was assessed on several benchmark datasets from multi-label classification. The predictive performance was measured with six evaluation measures: Hamming loss, accuracy, precision, recall, F1 score and subset accuracy.

The remainder of this paper is organized as follows. In Section 2, we present the predictive clustering trees for multiple discrete targets. We define the distances that we use in Section 3. We give the experimental design and in Section 4 and the results in Section 5. Section 6 concludes.

## 2 PREDICTIVE CLUSTERING TREES

The Predictive Clustering Trees (PCTs) framework sees a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system, which is available for download at http://www.cs.kuleuven.be/~dtai/clus.

PCTs can be induced with a standard top-down induction of decision trees (TDIDT) algorithm. The algorithm takes as input a set of examples and outputs a tree. The heuristic that is used for selecting the tests is the reduction in variance caused by partitioning the instances. By maximizing the variance reduction the cluster homogeneity is maximized and it improves the predictive performance. If no acceptable test can be found, that is, if the test does not significantly reduces the variance, then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf. The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task. So far, the PCTs have been instantiated for the following tasks: multiple targets prediction [5], hierarchical-multi label classification [4] and prediction of time-series [3].

In this paper, we focus on the first task. PCTs that are able to predict a tuple of discrete variables are called multi-target classification trees (MTCTs). An example of a MTCT is shown in Figure 1. This MTCT presents a habitat model for 14 bioindicator species [6]. The internal nodes of the tree contain tests on the descriptive variables (in this case, chemical parameters of the water samples) and the leaves store the predictions (in this case, which species are encountered and which not in a given water sample).

The variance is calculated as the sum of the squared pairwise distances between the instances, i.e.,

$$Var(E) = \frac{1}{2|E|^2} \sum_{X \in E} \sum_{Y \in E} d^2(X,Y)$$

The function used to calculate the prototype is then $m = \arg\min_q \sum_{X \in E} d^2(X,q)$. In this case, the prototype is an instance from the dataset and is called medoid. Different distances can be used depending on the application domain. By default, PCTs use the Euclidean distance.

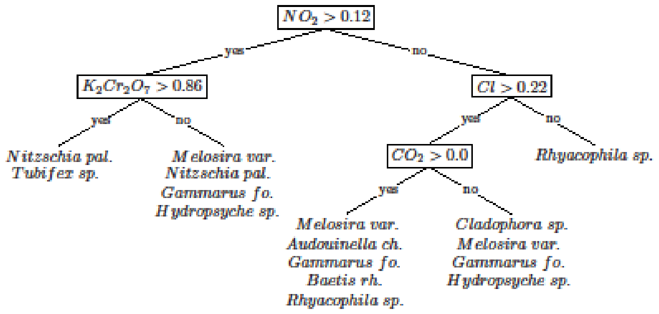


Figure 1: An example of a predictive clustering tree for predicting multiple discrete targets. The leaves predict the presence or absence for each bioindicator species.

## 3 DISTANCES FOR MULTI-LABEL LEARNING

In a multi-label learning setup, the target variable is a set of labels. Therefore, we can readily use distances over sets. Another approach to the problem is to see the multi-label classification problem as a predicting tuples of discrete targets and use distances over tuples. PCTs (and decision trees) have been previously used in the later context [2]. The focus of this study is the former approach to multi-label classification. In the remainder of this section, we present the distances over sets that can be used for extension of PCTs for multi-label classification.

### 3.1 Euclidean distance

The target in multi-label classification can be represented as a tuple of 0/1 values. The length of the target tuple is the number of all labels in the dataset. In this case, the Euclidean distance between two sets of labels $C_i$ and $C_j$ is defined as the Euclidean distance between their vector representations.

### 3.2 Hamming distance

The Hamming distance between two strings (i.e., bit-vectors) of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change the first string into the second. In terms of sets, the Hamming distance between two sets $C_i$ and $C_j$ is defined as:

$$h(C_i, C_j) = |C_i \cup C_j| - |C_i \cap C_j|$$

### 3.3 Jaccard distance

The Jaccard distance measures the dissimilarity between two sets by dividing the difference of the sizes of the union and the intersection of the two sets with the size of the union. The Jaccard distance can be calculated as follows.

$$j(C_i, C_j) = \frac{|C_i \cup C_j| - |C_i \cap C_j|}{|C_i \cup C_j|}$$

### 3.4 Matching distance (MD)

Motivated by a recently introduced distance on sets of structured objects, this distance is based on the matching between object from the sets. The matched objects do not contribute to the distance, which has the value of the unmatched part of the larger dataset, as defined below

$$md(C_i, C_j) = \max(|C_i|, |C_j|) - |C_i \cap C_j|$$

## 4 EXPERIMENTAL DESIGN

We begin by describing the benchmark datasets used in this study. Next, we present the most typically used evaluation measures for multi-label classification. We then give the experimental setup for the data analysis.

### 4.1 Datasets

We use 6 multi-label classification benchmark problems. Parts of the selected problems were used in various studies and evaluations of methods for multi-label learning. In the process of selection of problems, we opted to include benchmark datasets with different scale and from various application domains. Table 1 presents the basic statistics of the datasets. The datasets vary in size: from 391 up to 5318 training examples, from 202 up to 2635 testing examples, from 16 up to 1449 features, from 5 to 53 labels, and from 1.20 to 6.34 average number of labels per example.

|  | domain | N/T | D | Q | $l_c$ |
|---|---|---|---|---|---|
| water quality | ecology | 721/339 | 16 | 14 | 5.07 |
| emotions | music | 391/202 | 72 | 6 | 1.87 |
| mediana | text | 5318/2635 | 79 | 5 | 1.20 |
| soil quality | ecology | 1308/636 | 54 | 39 | 6.34 |
| medical | text | 645/333 | 1449 | 45 | 1.25 |
| enron | text | 1123/579 | 1001 | 53 | 3.38 |

**Table 1.** Description of the datasets in terms of application *domain*, number of training (*N*) and test (*T*) examples, the number of features (*D*), the total number of labels (*Q*) and label cardinality ($l_c$). The problems are ordered by their overall complexity roughly calculated as *N* x *D* x *Q*.

## 4.2 Evaluation measures

The evaluation of the predictive performance for multi-label learning systems differs from that of classical single-label learning systems. In any multi-label experiment, it is essential to include multiple and contrasting measures because of the additional degrees of freedom that the multi-label setting introduces. In our experiments, we used various evaluation measures that have been suggested by Tsoumakas et al [1]. In particular, we used six example-based evaluation measures: Hamming loss, accuracy, precision, recall, F1 score and subset accuracy.

In the definitions below, *Yi* denotes the set of true labels of example xi and $h(x_i)$ denotes the set of predicted labels for the same examples. All definitions refer to the multi-label setting.

**Hamming loss** evaluates how many times an example-label pair is misclassified, i.e., label not belonging to the example is predicted or a label belonging to the example is not predicted. The smaller the value of *hamming_loss*(*h*), the better the performance. The performance is perfect when *hamming_loss*(*h*) = 0. This metric is defined as:

$$hamming\_loss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q}\left|h(x_i)\Delta y_i\right|$$

where Δ stands for the symmetric difference between the two sets, *N* is the number of examples and *Q* is the total number of possible class labels.

**Accuracy** for a single example $x_i$ is defined by the Jaccard similarity coefficients between the label sets $h(x_i)$ and $y_i$. Accuracy is micro-averaged across all examples.

$$accuracy(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|h(x_i)\cap y_i\right|}{\left|h(x_i)\cup y_i\right|}$$

**Precision** is defined as:

$$precision(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|h(x_i)\cap y_i\right|}{\left|y_i\right|}$$

**Recall** is defined as:

$$recall(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|h(x_i)\cap y_i\right|}{\left|h(x_i)\right|}$$

**F1 score** is the harmonic mean between precision and recall and is defined as:

$$F_1(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{2\times\left|h(x_i)\cap y_i\right|}{\left|h(x_i)\right|+\left|y_i\right|}$$

$F_1$ score is an example based metric and its value is an average over all examples in the dataset. $F_1$ score reaches its best value at 1 and worst at 0.

**Subset Accuracy** is defined as follows:

$$subset\_accuracy(h) = \frac{1}{N}\sum_{i=1}^{N} I\left(h(x_i) = y_i\right)$$

where *I*(*true*) = 1 and *I*(*false*) = 0. This is a very strict evaluation measure as it requires the predicted set of labels to be an exact match of the true set of labels.

## 4.3 Experimental setup

We used the predictive clustering framework implemented in the CLUS system to investigate the performance of the different distance measures. To this end, we constructed single PCTs.

The PCTs were pruned with the F-test pruning method. This method checks whether a given test statistically significantly reduces the intra-cluster variance at a given significance level. An optimal significance level was selected by using internal 3-fold cross validation, from the following values: 0.01, 0.02, 0.03, 0.04 and 0.05.

## 5 RESULTS

Tables 2, 3, 4, 5, 6 and 7 show the results from the experimental evaluation of the distance measures. In the following, we briefly discuss the results for each evaluation measure. The Hamming distance has best predictive performance according to the Hamming loss measure. This is expected, since the trees with this distance are set to optimize that measure. Furthermore, since the Euclidean and Hamming distance are quite similar for vectors with 1/0 values, the Euclidean distance also has good predictive performance. On average, the Jaccard distance has the lowest predictive performance.

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | 0.314 | **0.309** | 0.528 | 0.312 |
| emotions | **0.249** | 0.272 | 0.274 | 0.253 |
| mediana | **0.157** | 0.165 | 0.355 | 0.203 |
| soil quality | 0.106 | **0.099** | 0.169 | 0.100 |
| medical | **0.013** | **0.013** | 0.014 | **0.013** |
| enron | 0.058 | **0.055** | 0.062 | 0.057 |

**Table 2.** The Hamming loss measure for different distances

In terms of accuracy, the Euclidean, Hamming and MD distance have similar predictive performance on average, while the Euclidean distance has the best performance on three datasets. The Jaccard distance, on the other hand, has the worst performance on average.

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | 0.298 | 0.315 | **0.370** | 0.317 |
| emotions | **0.496** | 0.469 | 0.488 | 0.493 |
| mediana | **0.589** | 0.588 | 0.302 | 0.505 |
| soil quality | 0.481 | 0.502 | 0.347 | **0.504** |
| medical | **0.733** | 0.731 | 0.718 | 0.727 |
| enron | 0.413 | **0.435** | 0.427 | 0.425 |

**Table 3.** The accuracy for the different distances

The precision and recall have inverted values. In the case of precision, Jaccard distance is the best performing, while for recall it is the worst performing. The distance to the other methods is large in the both cases. This means that the labels produced with Jaccard distance are reliable (low false positive rate); however, they do not cover all relevant labels for a given example (high false negative rate). The other three distances have similar performances to each other.

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | 0.352 | 0.382 | **0.860** | 0.390 |
| emotions | 0.583 | 0.561 | **0.635** | 0.580 |
| mediana | 0.605 | **0.641** | 0.465 | 0.602 |
| soil quality | 0.595 | 0.606 | 0.556 | **0.618** |
| medical | 0.755 | **0.761** | 0.746 | 0.755 |
| enron | 0.502 | 0.524 | **0.558** | 0.523 |

**Table 4.** The precision for the different distances

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | **0.625** | 0.623 | 0.397 | 0.614 |
| emotions | **0.613** | 0.592 | 0.571 | 0.600 |
| mediana | **0.722** | 0.704 | 0.359 | 0.595 |
| soil quality | 0.719 | **0.730** | 0.492 | 0.712 |
| medical | 0.779 | **0.787** | 0.771 | 0.776 |
| enron | 0.568 | **0.600** | 0.552 | 0.572 |

**Table 5.** The recall for the different distances

The $F_1$ score balances the performance measured by the precision and the recall. On average, the Jaccard distance has the lowest performance (because of the weak results for recall). The Hamming distance is slightly better than the remaining two distances.

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | 0.423 | 0.441 | **0.523** | 0.444 |
| emotions | 0.574 | 0.551 | **0.575** | 0.568 |
| mediana | 0.634 | **0.642** | 0.385 | 0.567 |
| soil quality | 0.617 | 0.634 | 0.491 | **0.635** |
| medical | **0.757** | 0.760 | 0.746 | 0.753 |
| enron | 0.515 | **0.543** | 0.535 | 0.530 |

**Table 6.** The $F_1$ scores for the different distances

The subset accuracy measures the fraction of the complete and accurate predictions. In this regard, the Euclidean distance has the best average performance, while MD is the best performing distance on four datasets. The worst performing distance is the Jaccard distance.

|  | Euc. | Ham. | Jac. | MD |
|---|---|---|---|---|
| water quality | 0.009 | 0.012 | 0.000 | **0.018** |
| emotions | 0.262 | 0.233 | 0.223 | **0.272** |
| mediana | **0.468** | 0.440 | 0.063 | 0.327 |
| soil quality | 0.036 | 0.041 | 0.003 | **0.044** |
| medical | **0.661** | 0.640 | 0.631 | 0.646 |
| enron | 0.145 | 0.149 | 0.149 | **0.150** |

**Table 7.** The subset accuracy for the different distances

## 6 CONCLUSIONS

In this paper, we have presented an experimental evaluation of four distance measures for multi-label classification. The evaluation was performed on 6 benchmark datasets using 6 evaluation measures.

The results show that there is no overall best distance measure. The best choice for a distance measure is the one that optimizes a selected evaluation measure. For example, the Hamming distance works the best when optimizing the Hamming loss, while the best according to precision is the Jaccard distance (since there is a strong connection between precision and the Jaccard coefficient). All in all, the Euclidean distance and Hamming loss perform the best averaged across all evaluation measures.

## References

1. G. Tsoumakas, I. Katakis. Multi Label Classification: An Overview. International Journal of Data Warehouse and Mining 3(3). 2007. pp. 1–13.
2. H. Blockeel, L. D. Raedt, J. Ramon. Top-down induction of clustering trees. In Proceedings of the 15th International Conference on Machine Learning. 1998. pp. 55–63.
3. I. Slavkov, V. Gjorgjioski, J. Struyf, and S. Dzeroski. Finding explained groups of time-course gene expression profiles with predictive clustering trees. Molecular Biosystems 6. 2010. pp. 729-740.
4. C. Vens, J. Struyf, L. Schietgat, S. Dˇzeroski, H. Blockeel. Decision trees for hierarchical multi-label classification. Machine Learning 73 (2). 2008. pp. 185–214.
5. J. Struyf and S. Dzeroski. Constraint based induction of multi-objective regression trees. In proceedings of the 4th International Workshop on Knowledge Discovery in Inductive Databases. 2005. pp. 110-121.
6. H. Blockeel, S. Dzeroski, J. Grbovic, Simultaneous prediction of multiple chemical parameters of river water quality with TILDE. In Proceedings of the 3rd European Conference on PKDD - LNAI 1704. 1999. pp. 32-40. Springer.

# Chapter 8

# Analysis of Time Series Data with Predictive Clustering Trees

In this chapter, we present the instantiation of predictive clustering trees (PCTs) for the task of predicting time series, i.e., for tasks where the output datatype is *TimeSeries* or *Sequence*[*Real*]. Time series prediction is a highly relevant task that can be encountered in many areas, including biology (e.g., predicting gene responses to stress) and environmental sciences (e.g., predicting forests response to forestry management, soil response to agricultural practices, etc.). Thus far, this task has been addressed by first performing some clustering over the time series and then, *post facto*, trying to associate some predictive model to the obtained clusters. Moreover, some authors have considered this task outside of the time series modelling domain, by completely ignoring the temporal/sequential nature of the data and treating the problem within the task of regression (Soon & Lee, 2007; Ralanamahatana et al., 2005; Keogh, 2006).

We include here two publications that present the instantiation of PCTs for time series prediction. The first publication appeared at a workshop and mainly focused on the computational aspects of the extension of the PCT framework for time series prediction. It proposed a method that constructs PCTs by partitioning a given set of time series into homogeneous clusters. In addition, PCTs provide a symbolic description of the clusters.

We evaluate the PCTs for time series prediction on several time series datasets from microarray experiments. Each dataset records the change over time in the expression level of yeast genes as a response to a change in environmental conditions. We consider 12 different changes/stresses in the environmental conditions. Furthermore, we describe each of the genes by its functional annotation (i.e., its functions as taken from the Gene Ontology catalogue of gene functions). The evaluation reveals that PCTs for time series prediction are able to successfully cluster genes with similar responses, and to predict the time series based on the description of a gene.

The second publication focuses more on the application of predictive clustering trees for predicting time series data to the analysis of microarray data. In biology, analyzing time course data is usually a two-step process that clusters similar temporal profiles and then derives descriptions of the clusters (e.g., gene ontology terms that are enriched in the clusters) by using expert knowledge. In this work, we propose to use predictive clustering trees for the task at hand. Their advantage over typical clustering approaches is that they partition the time course data into homogeneous clusters, while at the same time providing symbolic descriptions of the clusters. We use the same 12 yeast micro-array time series datasets as above. We demonstrate that PCTs are able to cluster genes with similar temporal profiles, yield a predictive model of the temporal profiles of genes based on a cluster prototype, and provide cluster descriptions, all in a single step. Finally, we discuss the knowledge that is produced by the different models we constructed.

**Papers**:

Džeroski, S., Gjorgjioski, V., Slavkov, I., & Struyf, J. (2006). Analysis of time series data with predictive clustering trees. In *Proceedings of the Fifth International Workshop on Knowledge Discovery in Inductive Databases* (Vol. 4747, pp. 63–80). LNCS. Springer, Berlin.

Slavkov, I., Gjorgjioski, V., Struyf, J., & Džeroski, S. (2010). Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, *6*(4), 729–740. IF=3.825.

**Author's contribution**: For both papers, Valentin Gjorgjioski designed and implemented the algorithms for predictive clustering of time series and performed the experiments with the software on the yeast data. He wrote some parts of the papers (describing the details of the implemented algorithms). The studies were designed and supervised by Sašo Džeroski.

# Analysis of Time Series Data with Predictive Clustering Trees

Sašo Džeroski[1], Valentin Gjorgjioski[1], Ivica Slavkov[1], and Jan Struyf[2]

[1] Dept. of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{Saso.Dzeroski, Valentin.Gjorgjioski, Ivica.Slavkov}@ijs.si
[2] Dept. of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
Jan.Struyf@cs.kuleuven.be

**Abstract.** Predictive clustering is a general framework that unifies clustering and prediction. This paper investigates how to apply this framework to cluster time series data. The resulting system, Clus-TS, constructs predictive clustering trees (PCTs) that partition a given set of time series into homogeneous clusters. In addition, PCTs provide a symbolic description of the clusters. We evaluate Clus-TS on time series data from microarray experiments. Each data set records the change over time in the expression level of yeast genes as a response to a change in environmental conditions. Our evaluation shows that Clus-TS is able to cluster genes with similar responses, and to predict the time series based on the description of a gene. Clus-TS is part of a larger project where the goal is to investigate how global models can be combined with inductive databases.

## 1 Introduction

Predictive clustering is a general framework that combines clustering and prediction [1]. Predictive clustering partitions a given data set into a set of clusters such that the instances in a given cluster are similar to each other and dissimilar to the instances in other clusters. In this sense, predictive clustering is identical to regular clustering [11]. The difference is that predictive clustering associates a predictive model to each cluster. This model assigns instances to clusters and provides predictions for new instances. So far, decision trees [1,22] and rule sets [25] have been used in the context of predictive clustering.

This paper investigates how predictive clustering can be applied to cluster time series [13]. A time series is an ordered sequence of measurements of a continuous variable that changes over time. Fig. 1.a presents an example of eight time series partitioned into three clusters: cluster $C_1$ contains time series that increase and subsequently decrease, $C_2$ has mainly decreasing time series and $C_3$ mainly increasing ones. Fig. 1.b shows a so-called predictive clustering tree (PCT) for this set of clusters. This is the predictive model associated with the clustering.

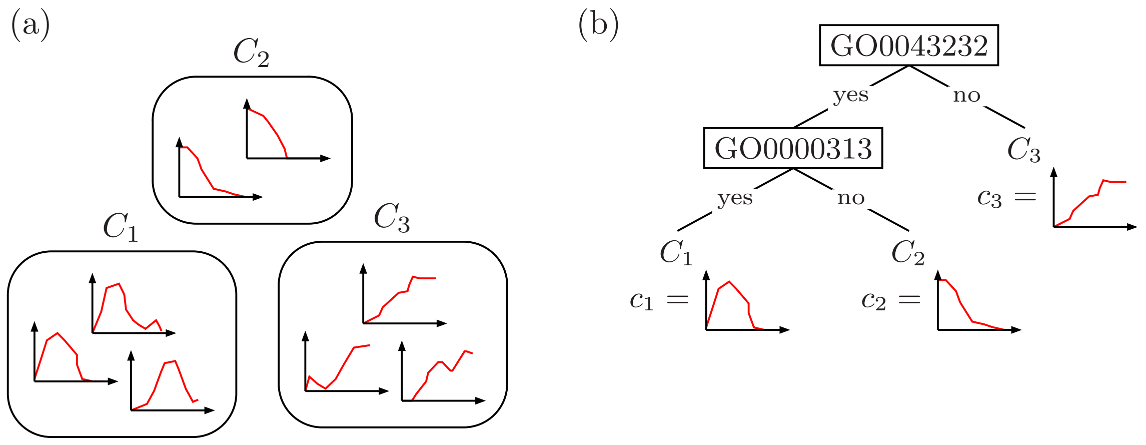


**Fig. 1.** (a) A set of time series clustered into three clusters. (b) A predictive clustering tree associated with this clustering. Each leaf of the tree corresponds to one cluster.

We propose a new algorithm called Clus-TS (Clustering-Time Series) that constructs trees such as the one shown in Fig. 1.b. Clus-TS instantiates the general PCT induction algorithm proposed by Blockeel et al. [1] to the task of time series clustering. This is non-trivial because the general algorithm requires computing a centroid for each cluster and for most distance measures suitable for time series clustering, no closed algebraic form centroid is known.

We evaluate Clus-TS on time series data from microarray experiments [9]. Each data set records the change over time in the expression level of yeast genes in response to a change in environmental conditions. A lot of work has been done on clustering this type of short time series gene expression data [4]. Our main motivation is to use alternative distance measures (that mainly take the shape of the time series into account) and to construct clusters that can be explained by a given set of features. Besides the time series, various other data about each gene are available. Here, we consider motifs and terms from the Gene Ontology (GO) [5]. The motifs are subsequences that occur in the amino acid sequence of many genes. The motivation for using motifs as features is due to Curk et al. [2], who use motifs in a similar analysis. The motifs or GO terms appear in the internal nodes of the PCT (Fig. 1.b) and provide a symbolic description of the clusters. $C_1$ includes, for example, all genes that have terms "GO:0043232" and "GO:0000313" in their description. This is related to itemset constrained clustering [20], which clusters vectors of numeric values and constrains each cluster by means of an itemset.

So far, most research on inductive databases (IDBs) [10,3] has focused on local models (i.e., models that apply to only a subset of the examples), such as frequent item sets and association rules. Clus-TS is part of a larger project [7,22,25] were the goal is to investigate how IDBs can be extended to global models, such as decision trees (for prediction) and mixture models (for clustering). Predictive clustering has been argued to provide a general framework unifying clustering and prediction, two of the most basic data mining tasks, and is therefore an excellent starting point for extending IDBs to global models [25].

In particular, we are interested in developing a system that is applicable to clustering and prediction in many application domains, including bioinformatics. Extending PCTs to time series clustering is a step in this direction.

## 2 Predictive Clustering Trees

### 2.1 Prediction, Clustering, and Predictive Clustering Trees

Predictive modeling aims at constructing models that can predict a target property of an object from a description of the object. Predictive models are learned from sets of examples, where each example has the form $(D, T)$, with $D$ being an object description and $T$ a target property value. While a variety of representations ranging from propositional to first order logic have been used for $D$, $T$ is almost always considered to consist of a single target attribute called the class, which is either discrete (classification problem) or continuous (regression problem).

Clustering [11], on the other hand, is concerned with grouping objects into subsets of objects (called clusters) that are similar w.r.t. their description $D$. There is no target property defined in clustering tasks. In conventional clustering, the notion of a distance (or conversely, similarity) is crucial: examples are considered to be points in a metric space and clusters are constructed such that examples in the same cluster are close according to a particular distance metric. A centroid (or prototypical example) may be used as a representative for a cluster. The centroid is the point with the lowest average (squared) distance to all the examples in the cluster, i.e., the mean or medoid of the examples. Hierarchical clustering and $k$-means clustering are the most commonly used algorithms for this type of clustering (see Section 4.4).

Predictive clustering [1] combines elements from both prediction and clustering. As in clustering, we seek clusters of examples that are similar to each other, but in general taking both the descriptive part and the target property into account (the distance measure is defined on $D \cup T$). In addition, a predictive model must be associated to each cluster. The predictive model assigns new instances to clusters based on their description $D$ and provides a prediction for the target property $T$. A well-known type of model that can be used to this end is a decision tree [17]. A decision tree that is used for predictive clustering is called a predictive clustering tree (PCT, Fig. 1.b). Each node of a PCT represents a cluster. The conjunction of conditions on the path from the root to that node gives a description of the cluster. Essentially, each cluster has a symbolic description in the form of a rule (IF conjunction of conditions THEN cluster)[1], while a tree structure represents the hierarchy of clusters. Clusters that are not on the same branch of a tree do not overlap.

In Fig. 1, the description $D$ of a gene consists of GO terms with which the gene is annotated, and the target property $T$ is the time series recorded for the gene. In general, we could include both $D$ and $T$ in the distance measure. We are, however, most interested in the time series part. Therefore, we define the distance measure

[1] This idea was first used in conceptual clustering [15].

only on $T$. We consider various distance measures in Section 3.1. The resulting PCT (Fig. 1.b) represents a clustering that is homogeneous w.r.t. $T$ and the nodes of the tree provide a symbolic description of the clusters. Note that a PCT can also be used for prediction: use the tree to assign a new instance to a leaf and take the centroid (denoted with $c_i$ in Fig. 1.b) of the corresponding cluster as prediction.

### 2.2 Building Predictive Clustering Trees

Table 1 presents the generic induction algorithm for PCTs [1]. It is a variant of the standard greedy recursive top-down decision tree induction algorithm used, e.g., in C4.5 [17]. It takes as input a set of instances $I$ (in our case genes described by motifs or GO terms and their associated time series). The procedure BestTest (Table 1, right) searches for the best acceptable test (motif or GO term) that can be put in a node. If such a test $t^*$ can be found then the algorithm creates a new internal node labeled $t^*$ and calls itself recursively to construct a subtree for each cluster in the partition $\mathcal{P}^*$ induced by $t^*$ on the instances. If no acceptable test can be found, then the algorithm creates a leaf, and the recursion terminates. (The procedure Acceptable defines the stopping criterion of the algorithm, e.g., specifying maximum tree depth or a minimum number of instances in each leaf).

**Table 1.** The generic PCT induction algorithm Clus

```
procedure PCT($I$) returns tree
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(I)$
2: if $t^* \neq none$ then
3:     for each $I_k \in \mathcal{P}^*$ do
4:         $tree_k = \text{PCT}(I_k)$
5:     return node($t^*$, $\bigcup_k \{tree_k\}$)
6: else
7:     return leaf(centroid($I$))
```

```
procedure BestTest($I$)
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each possible test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $I$
4:     $h = Var(I) - \sum_{I_k \in \mathcal{P}} \frac{|I_k|}{|I|} Var(I_k)$
5:     if $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

Up till here, the algorithm is identical to a standard decision tree learner. The main difference is in the heuristic that is used for selecting the tests. For PCTs, this heuristic is the reduction in variance (weighted by cluster size, see line 4 of BestTest). Maximizing variance reduction maximizes cluster homogeneity. The next section discusses how cluster variance can be defined for time series.

An implementation of the PCT induction algorithm is available in the Clus system, which can be obtained at `http://www.cs.kuleuven.be/~dtai/clus`.

## 3 PCTs for Time Series Clustering

### 3.1 Distance Measures

In this section, we discuss a number of distance measures for time series, which will be used in the definition of cluster variance later on. Some measures require that all time series in the data set have the same length. This property holds true for the data that we consider in the experimental evaluation (Section 4).

If all time series have the same length then one can represent them as real valued vectors and use standard vector distance measures such as the Euclidean or Manhattan distance. These measures are, however, not always appropriate for time series because they assume that the time series are synchronized, and mainly capture the difference in scale and baseline. Below, we discuss three distance measures that have been proposed to alleviate these shortcomings.

**Dynamic Time Warping.** (DTW) [19] can capture a non-linear distortion along the time axis. It accomplishes this by assigning multiple values of one of the time series to a single value of the other. As a result, DTW is suitable if the time series are not properly synchronized, e.g., if one is delayed, or if the two time series are not of the same length. Fig. 2.a illustrates DTW and compares it to the Euclidean distance.

$d_{\text{DTW}}(X, Y)$ with $X = \alpha_1, \alpha_2, \ldots, \alpha_I$, $Y = \beta_1, \beta_2, \ldots, \beta_J$ is defined based on the notion of a warping path between $X$ and $Y$. A warping path is a sequence of grid points $F = f_1, f_2, \ldots, f_K$ on the $I \times J$ plane (Fig. 2.b). Let the distance between two values $\alpha_{i_k}$ and $\beta_{j_k}$ be $d(f_k) = |\alpha_{i_k} - \beta_{j_k}|$, then an evaluation function $\Delta(F)$ is given by $\Delta(F) = 1/(I+J)\sum_{k=1}^{K} d(f_k)w_k$. The weights $w_k$ are as follows: $w_k = (i_k - i_{k-1}) + (j_k - j_{k-1}), i_0 = j_0 = 0$. The smaller the value of $\Delta(F)$, the more similar $X$ and $Y$ are. In order to prevent excessive distortion, we assume an adjustment window ($|i_k - j_k| \leq r$). $d_{\text{DTW}}(X, Y)$ is the minimum of $\Delta(F)$. $d_{\text{DTW}}$ can be computed with dynamic programming in time $O(IJ)$.

Both the Euclidean distance and DTW take into account differences in scale and baseline. If a given time series is identical to a second time series, but scaled by a certain factor or offset by some constant, then the two time series will be distant. For many applications, these differences are, however, not important;

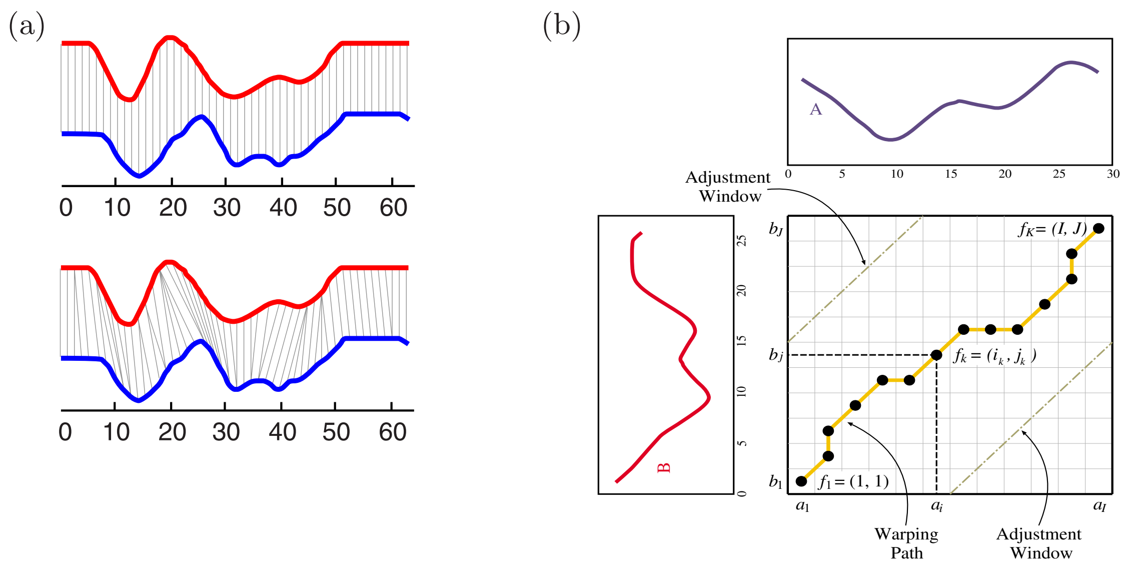


**Fig. 2.** (a) Euclidean distance (top) compared to DTW (bottom). (b) A warping path. (Artwork courtesy of Eamonn Keogh).

only the shape of the time series matters. The next two measures are more appropriate for such applications.

**Correlation.** The correlation coefficient $r(X, Y)$ between two time series $X$ and $Y$ is calculated as

$$r(X,Y) = \frac{E[(X - E[X]) \cdot (Y - E[Y])]}{E[(X - E[X])^2] \cdot E[(Y - E[Y])^2]} \,, \tag{1}$$

where $E[V]$ denotes expectation (i.e., mean value) of $V$. $r(X, Y)$ measures the degree of linear dependence between $X$ and $Y$. It has the following intuitive meaning in terms of the shapes of $X$ and $Y$: $r$ close to 1 means that the shapes are similar. If there is a linear relation between $X$ and $Y$ then the time series are identical but might have a different scale or baseline. $r$ close to -1 means that $X$ and $Y$ have "mirrored" shapes, and $r$ close to 0 means that the shapes are unrelated (and consequently dissimilar). Based on this intuitive interpretation, we can define the distance between two time series as $d_r(X, Y) = \sqrt{0.5 \cdot (1 - r(X, Y))}$. $d_r$ has, however, two drawbacks. First, it is difficult to properly estimate correlation if the number of observations is small (i.e., a short time series). Second, $d_r$ can only capture the linear dependencies between the time series. Two time series that are non-linearly related will be distant. Fig. 3 illustrates this effect.

**A Qualitative Distance.** A third distance measure is the qualitative distance proposed by Todorovski et al. [23]. It is based on a qualitative comparison of the shape of the time series. Consider two time series $X$ and $Y$ (Fig. 3). Then choose a pair of time points $i$ and $j$ and observe the qualitative change of the value of $X$ and $Y$ at these points. There are three possibilities: increase ($X_i > X_j$), no-change ($X_i \approx X_j$), and decrease ($X_i < X_j$). $d_{\text{qual}}$ is obtained by summing the difference in qualitative change observed for $X$ and $Y$ for all pairs of time points, i.e.,

$$d_{\text{qual}}(X,Y) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \mathit{Diff}(q(X_i, X_j), q(Y_i, Y_j))}{N \cdot (N-1)} \,, \tag{2}$$

with $\mathit{Diff}(q_1, q_2)$ a function that defines the difference between different qualitative changes (Fig. 2). Roughly speaking, $d_{\text{qual}}$ counts the number of disagreements in change of $X$ and $Y$.

$d_{\text{qual}}$ does not have the drawbacks of the correlation based measure. First, it can be computed for very short time series, without decreasing the quality of the estimate. Second, it captures the similarity in shape of the time series, regardless

**Table 2.** The definition of $\mathit{Diff}(q_1, q_2)$

| $\mathit{Diff}(q_1, q_2)$ | increase | no-change | decrease |
|---|---|---|---|
| increase | 0 | 0.5 | 1 |
| no-change | 0.5 | 0 | 0.5 |
| decrease | 1 | 0.5 | 0 |

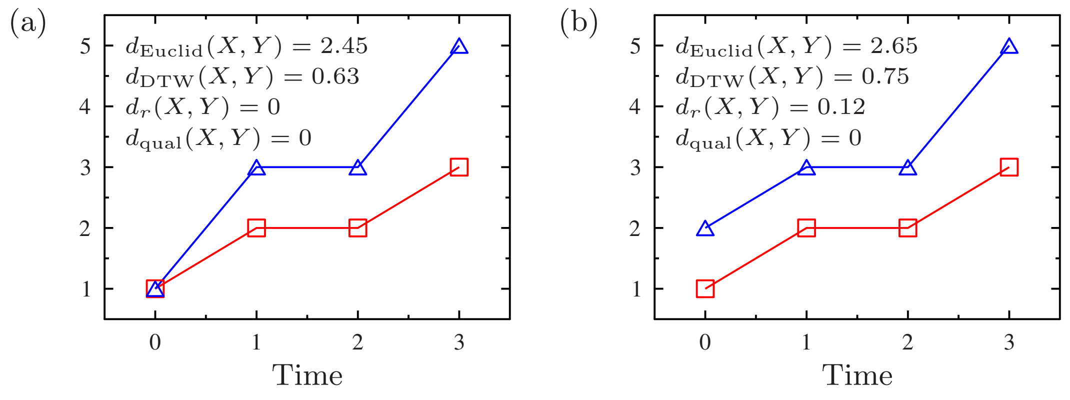


**Fig. 3.** Comparison of four distance measures for time series. Time series (a) are linearly related resulting in $d_r(X,Y) = 0$. Time series (b) are non-linearly related, but still have a similar shape, resulting in $d_{\text{qual}}(X,Y) = 0$.

of whether their dependence is linear or non-linear (Fig. 3). In the experimental evaluation, we will use $d_{\text{qual}}$ (Section 4).

## 3.2 Computing Cluster Variance

Recall from Section 2.2 that the PCT induction algorithm requires a measure of cluster variance in its heuristics. The variance of a cluster $C$ can be defined based on a distance measure as

$$Var(C) = \frac{1}{|C|} \sum_{X \in C} d^2(X,c)\,, \tag{3}$$

with $c$ the cluster centroid of $C$. To cluster time series, $d$ should be a distance measure defined on time series, such as the ones discussed in the previous section.

The centroid $c$ can be computed as $\operatorname{argmin}_q \sum_{X \in C} d^2(X,q)$. We consider two possible representations for $c$: (a) the centroid is an arbitrary time series, and (b) the centroid is one of the time series from the cluster (the cluster prototype). In representation (b), the centroid can be computed with $|C|^2$ distance computations by substituting $q$ with each time series in the cluster. In representation (a), the space of candidate centroids is infinite. This means that either a closed algebraic form for the centroid is required or that one should resort to approximative algorithms to compute the centroid. No closed form for the centroid is known in representation (a) for the distance measure $d_{\text{qual}}$. To the best of our knowledge, the same holds for $d_{\text{DTW}}$ and $d_r$.

An alternative way to define cluster variance is based on the sum of the squared pairwise distances (SSPD) between the cluster elements, i.e.,

$$Var(C) = \frac{1}{2|C|^2} \sum_{X \in C} \sum_{Y \in C} d^2(X,Y)\,. \tag{4}$$

(The factor 2 in the denominator of (4) ensures that (4) is identical to (3) for the Euclidean distance.) The advantage of this approach is that no centroid is required. It also requires $|C|^2$ distance computations. This is the same time complexity as the approach with the centroid in representation (b). Hence, using the definition based on a centroid is only more efficient if the centroid can be computed in time linear in the cluster size. This is the case for the Euclidean distance in combination with using the pointwise average of the time series as centroid. For the other distance measures, no such centroids are known. Therefore, we choose to estimate cluster variance using the SSPD.

A second advantage is that (4) can be easily approximated by means of sampling, e.g., by using,

$$Var(C) = \frac{1}{2|C|m} \sum_{X \in C} \left( \sum_{Y \in \text{sample}(C,m)} d^2(X,Y) \right) , \qquad (5)$$

with sample$(C, m)$ a random sample without replacement of $m$ elements from $C$, instead of (4) if $|C| \geq m$. The computational cost of (5) grows only linearly with the cluster size. In the experimental evaluation, we compare (4) to (5).

### 3.3 Cluster Centroids for the Tree Leaves

The PCT induction algorithm places cluster centroids in its leaves, which can be inspected by the domain expert and used as a prediction. For these centroids, we use representation (b) as discussed above.

## 4 Analyzing Gene Expression Time Series with PCTs

### 4.1 The Problem

DNA microarray analysis is an interesting application area for short time series clustering. Clustering genes by their time expression pattern makes sense because genes which are co-regulated or have a similar function, under certain conditions, will have a similar temporal profile. Instead of simply clustering the expression time series with, e.g., HAC, and later on elucidating the characteristics of the obtained clusters (as done in e.g., [4]), we perform constrained clustering with PCTs. This yields the clusters and symbolic descriptions of the clusters in one step.

We use the data from a study conducted by Gasch et al. [9]. The purpose of the study is to explore the changes in expression levels of yeast (*Saccharomyces cerevisiae*) genes under diverse environmental stresses. Various sudden changes in the environmental conditions are tested, ranging from heat shock to amino acid starvation for a prolonged period of time. The gene expression levels of around 5000 genes are measured at different time points using microarrays. The data is log-transformed and normalized based on the time-zero measurement of yeast cells under normal environmental conditions. We use three sets of experiments from Gasch et al. [9]: amino acid starvation (AAS), diauxic shift (DS), and diamide treatment (DT).

### 4.2 The Mining Scenario

Our mining scenario consists of two steps. In a first step, we use a local pattern mining algorithm to construct patterns based on the description of the yeast genes. In a second step, we use these local patterns as features to construct PCTs. We use two types of features: motifs and GO terms [5].

For the first set of features, we mine frequent subsequences (motifs) occurring in the DNA sequences of the yeast genes, which we obtain from the Stanford database. We use the constraint based mining algorithm FAVST [12,16]. FAVST supports three types of constraints: minimum frequency, and minimum and maximum motif length. We query FAVST for sequences that appear in 25% of the genes and consist of at least 8 nucleotides. In this way, we obtain approximately 300 motifs ranging from 8 to 10 nucleotides. These motifs are passed to Clus-TS to build PCTs with the motifs in the internal nodes.

In the second set of features, each feature is a GO term. We obtain the GO term annotations for each yeast gene from the Gene Ontology [5] (version April, 2006). Note that the GO terms are structured in a hierarchy. We use both the `part_of` and `is_a` relations to include for each gene all relevant GO terms. To limit the number of features, we set a minimum frequency threshold: each GO term must appear for at least 50 of the genes.

### 4.3 Predicting Time Series with PCTs

Recall that PCTs can be used both for prediction and clustering. PCTs predict values just like regular decision trees. They sort each test instance into a leaf and assign as prediction the label of that leaf. PCTs label their leaves with the training set centroids of the corresponding clusters. In this section, we evaluate PCTs in a predictive setting and in Section 4.5 we assess their clustering performance.

To evaluate predictive performance, we need an error metric. An obvious candidate is the root mean squared error (RMSE), which is defined as

$$\mathrm{RMSE}(I, T) = \sqrt{\frac{1}{|I|} \sum_{X \in I} d^2(T(X), \mathrm{series}(X))}\,, \tag{6}$$

with $I$ the set of test instances, $T$ the PCT that is being tested, $T(X)$ the time series predicted by $T$ for instance $X$, and series($X$) the actual series of $X$.

We compare the PCT built by Clus-TS to a default predictor DEF that always predicts the overall training set centroid. We measure predictive RMSE using 10 fold cross-validation. We set the minimum number of time series in each cluster to 10 and all other parameters of Clus-TS to their default values. Clus supports size constraints by means of the post pruning method proposed by Garofalakis et al. [8], which employs dynamic programming to find the most accurate subtree no larger than a given number of leaves. Here, accuracy is estimated as training set RMSE (see also [22]). Fig. 4 presents the results for different values of the size constraint.

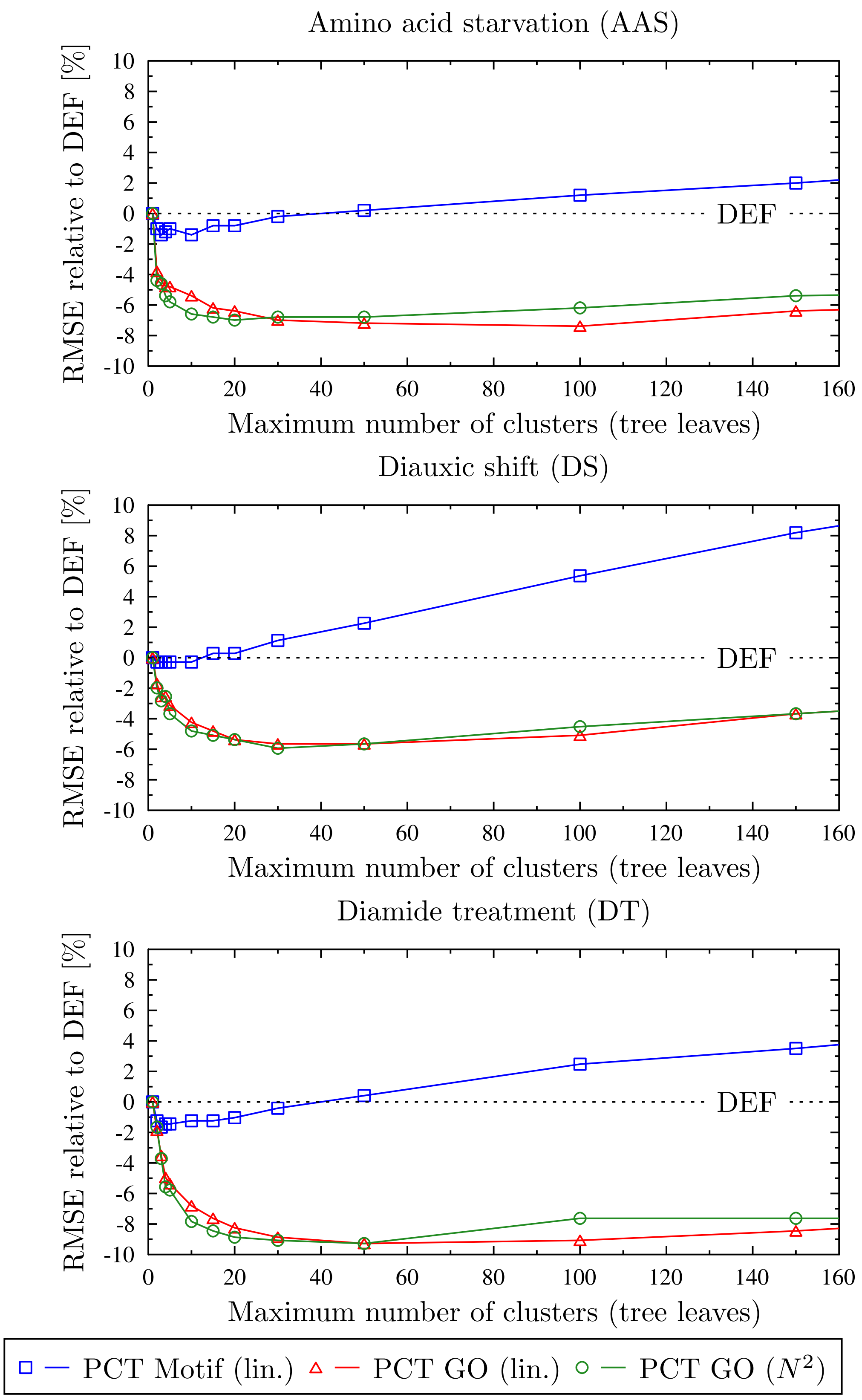


**Fig. 4.** RMS error versus maximum number of clusters

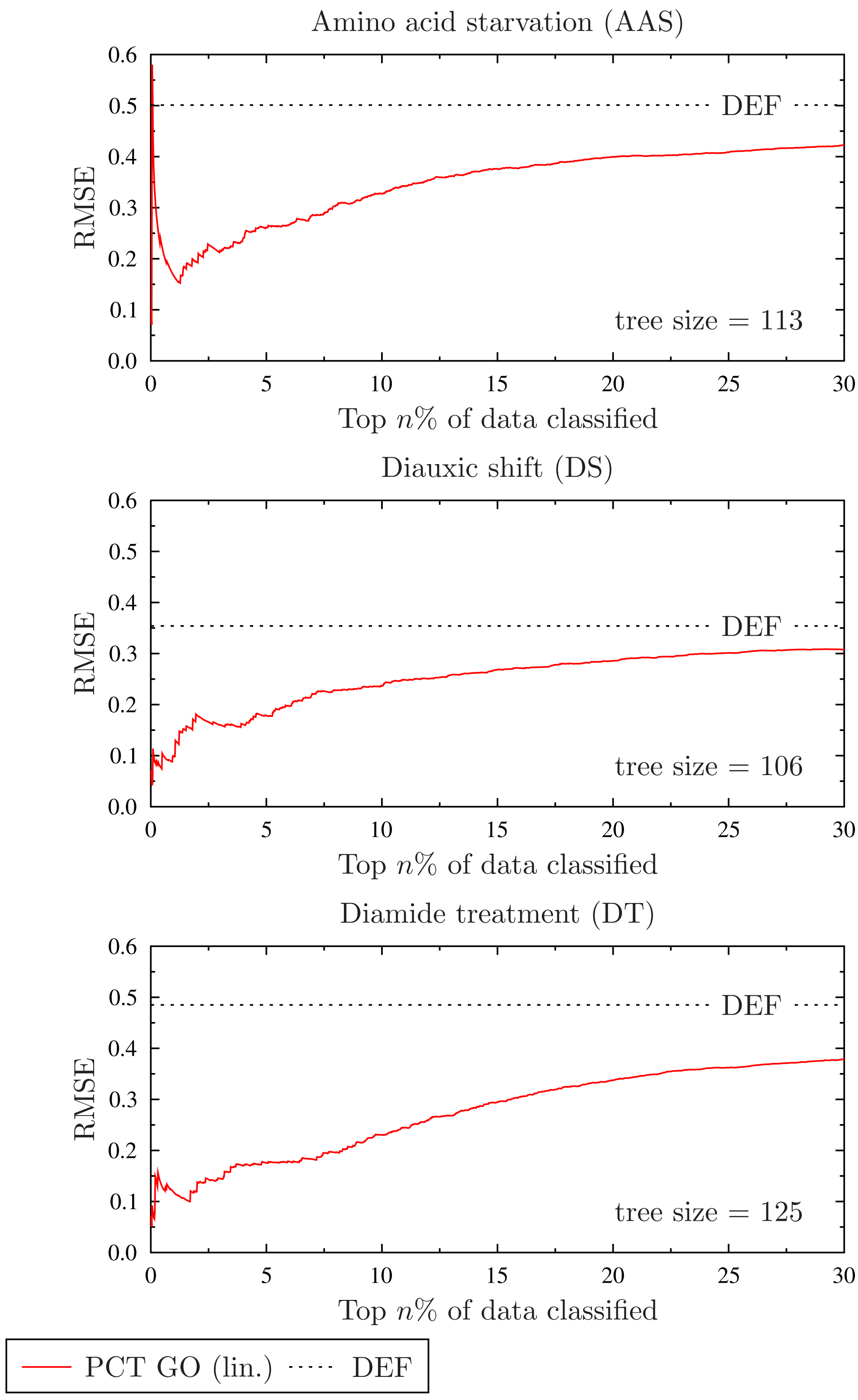


**Fig. 5.** RMS error versus percentage of data classified

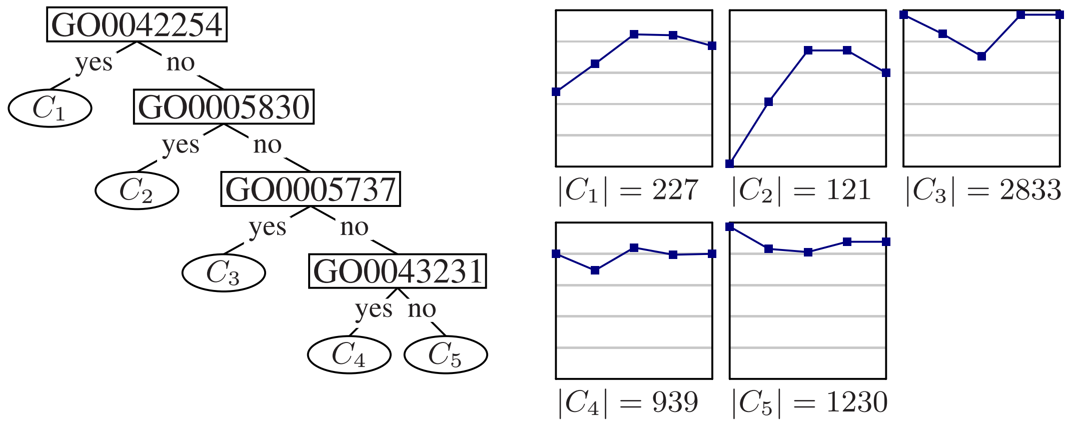


**Fig. 6.** Size 5 PCT for amino acid starvation (AAS)

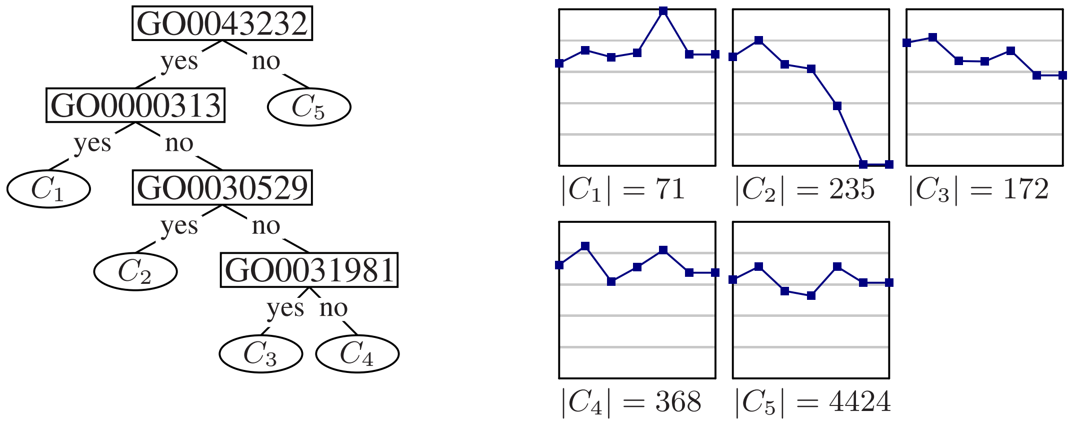


**Fig. 7.** Size 5 PCT for diauxic shift (DS)

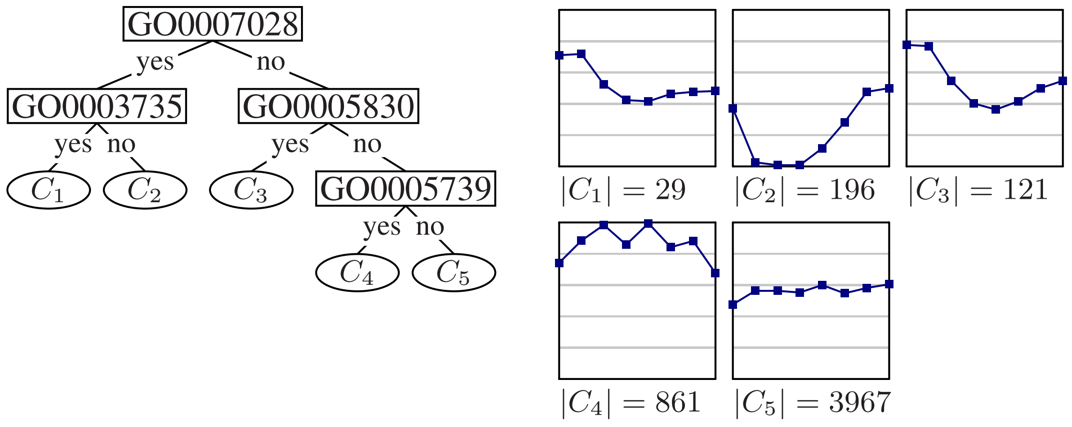


**Fig. 8.** Size 5 PCT for diamide treatment (DT)

The PCTs with motifs as features do not perform well (RMSE close to that of DEF) and quickly overfit for larger size trees. The optimal tree size for the PCTs with GO terms seems to be around 30 nodes. The PCTs with GO terms perform better, but still have a relatively high RMSE. Fig. 4 also compares Clus-TS with the SSPD variance estimate with quadratic time complexity (PCT $N^2$) to the linear approximation with sample size $m = 10$ (PCT lin.). Both estimates yield a comparable predictive performance. PCT $N^2$ performs slightly better for small trees, but becomes worse for larger trees. PCT $N^2$ is a factor 6.8 to 12.6 slower than PCT lin.

From a biological viewpoint, the PCTs cluster genes that have a similar function (GO terms) and a similar response in expression level to a certain change in environmental conditions. One problem is that, as noted by Gasch et al. [9], only a subset of the genes (about 900) have a stereotypical response to the environmental stress. That is, only a subset of the genes can be accurately clustered, whereas the other genes have an uncorrelated response. As a result, we hypothesize that the PCTs are able to more accurately predict the time series of a subset of the genes. We therefore perform the following experiment. Besides recording the predicted time series for each test set gene, we also record a confidence value for each prediction. We then sort the genes by confidence value and compute the RMSE of the top $n$ percent most confident predictions. We use the training set RMSE of the leaf that made the prediction as confidence estimate. This is similar to the approach used for generating a ROC curve for a decision tree [6]. Fig. 5 presents the results[2]. It shows that more accurate predictions are obtained if we restrict the test set based on the confidence of the predictions. For example, if time series are predicted for the top 5%, then the RMSE decreases to about 50% of that of DEF.

Fig. 6, 7, and 8 show as an illustration the PCT for each data set obtained with the maximum size set to 5 leaves. They also show the cluster centroids for each of the leaves.

## 4.4 Hierarchical Agglomerative Clustering

In this section, we briefly discuss Hierarchical Agglomerative Clustering (HAC) (see, e.g., [14]). We use HAC as a baseline to compare Clus-TS to. HAC is one of the most widely used clustering approaches. It produces a nested hierarchy of groups of similar objects, based on a matrix containing the pairwise distances between all objects. HAC repeats the following three steps until all objects are in the same cluster:

1. Search the distance matrix for the two closest objects or clusters.
2. Join the two objects (clusters) to produce a new cluster.
3. Update the distance matrix to include the distances between the new cluster and all other clusters (objects).

[2] PCTs are obtained with the same parameters as before, except that we use validation set based pruning instead of specifying a size constraint. Clus-TS uses here 1000 genes of the original training set for pruning and the rest for the tree construction (suggested by [24]). Simply selecting a PCT from Fig. 4 is unfair; it corresponds to optimizing the size parameter on the test set.

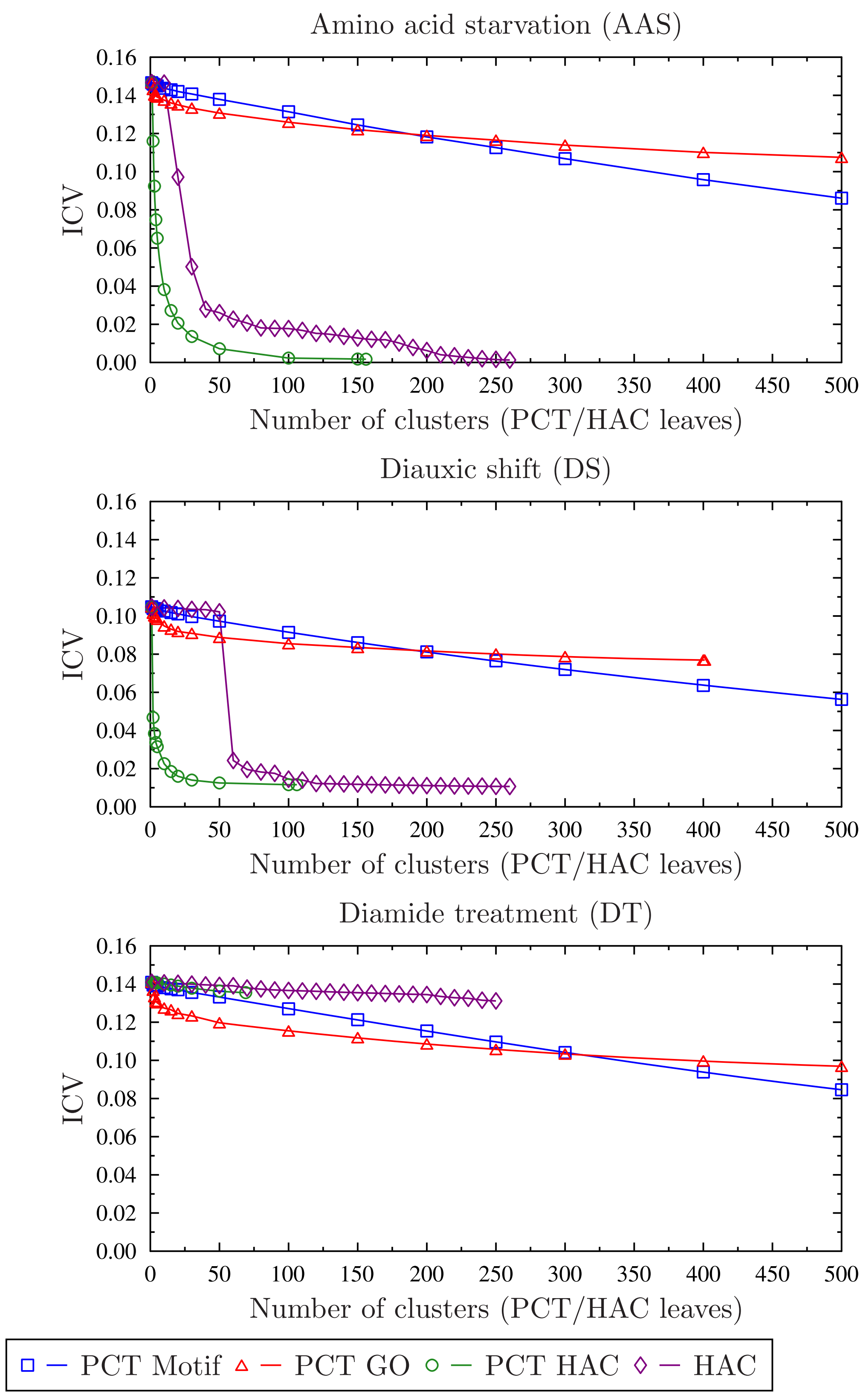


**Fig. 9.** ICV versus number of clusters

There are four well known HAC algorithms: single-link, complete-link, group-average, and centroid clustering, which differ in the cluster similarity measure they employ. We decided to use single-link HAC because it is usually considered to be the simplest approach and it has the lowest time complexity. Furthermore, it yields a better intra cluster variation than the PCTs. Therefore, we did not consider more elaborate approaches. Single-link HAC computes the distance between two clusters as the distance between the closest pair of objects. The HAC implementation that we use has a computational cost of $O(N^2)$, with $N$ the number of time series, and for efficiency it uses a next-best-merge array [14].

An important drawback of single-link HAC is that it suffers from the chaining effect [14], which in some cases may result in undesirable elongated clusters. Because the merge criterion is strictly local (it only takes the two closest objects into account), a chain of points can be extended over a long distance without regard to the overall shape of the emerging cluster.

### 4.5 Clustering Time Series with PCTs

In this section, we compare PCTs to HAC (Section 4.4). The evaluation metric that we use is intra cluster variation (ICV) defined as

$$\mathrm{ICV}(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|C|} \mathrm{Var}(C_i) \, , \tag{7}$$

with $\mathcal{C}$ the set of clusters (PCT or HAC leaves), $|C|$ the data set size, and $\mathrm{Var}(C_i)$ the variance of cluster $C_i$ (Equation 4).

We measure ICV for different values of the size constraint (Section 4.3). The minimum cluster size is set to 5. For HAC, we cut the hierarchy of clusters at different levels to obtain measurements for a varying number of clusters. Fig. 9 presents the results. For the data sets AAC and DS, HAC is able to decrease ICV much faster than PCTs. The reason is that PCTs constrain the clusters based on the given features. If the ICV-wise best split at a given point in the cluster hierarchy can not be described in terms of the features, then Clus-TS will select a suboptimal split. It is therefore important to have good descriptive features when performing predictive clustering.

To test the impact of the features, we constructed artificial data sets with the same time series, but with as features the characteristic vector of the clustering produced by HAC, that is, one Boolean feature for each cluster (internal nodes and leaves) indicating for each example if it belongs to that particular cluster or not. Fig. 9 shows that, given these features, Clus-TS even outperforms HAC.

On the DT data set, HAC performs worse compared to Clus-TS. Note that this may happen because HAC is also heuristic (e.g., because of the chaining effect, cf. Section 4.4).

## 5 Future Work

We plan to extend the experimental evaluation. This includes testing more data sets (e.g., all changes in environmental conditions studied in [9], or other types of short time series data), working with domain experts to interpret the clusterings, and using more types of descriptive features. Our experiments show that having appropriate features is very important for predictive clustering. It would be interesting to try experiments with more features, possibly automatically constructed using feature construction methods.

In Section 3.2, we considered two representations for the cluster centroid that are both time series. The centroid, however, does not need to be in the same domain as the objects that are being clustered. It would be interesting to investigate more expressive representations of the cluster centroid, such as a parametric representation of the distribution of the time series. The advantage of such an approach, while it can be computationally more expensive, is that it captures more information about the cluster. This is akin to classification with the Gini index or information gain heuristics [18], which summarize a set of examples by means of its class distribution instead of the majority class.

We plan to incorporate different types of constraints in our models. This is important in the context of inductive databases because the inductive queries might include various types of constraints on the resulting PCTs. Our current system already includes accuracy and size constraints [22]. In further work, we wish to investigate constraints more specific to clustering [26] and in particular clustering of time series.

Another direction of research is investigating how PCTs, and in particular PCTs for clustering time series, can be integrated tightly with inductive databases. Fromont and Blockeel [7] and Slavkov et al. [21] present ongoing work in this direction.

## 6 Conclusion

This paper proposes predictive clustering trees (PCTs) to cluster time series data. The main advantage of using PCTs over other clustering algorithms, such as hierarchical agglomerative clustering and $k$-means, is that PCTs cluster the time series and provide a description of the clusters at the same time. This allows one to relate various heterogeneous data types and to draw conclusions about their relations.

Using PCTs for time series data is non-trivial because for many appropriate distance measures (correlation based, dynamic time warping, and a qualitative distance), no closed algebraic form for the centroid is known. Therefore, we propose to compute cluster variance based on the sum of squared pairwise distances (SSPD) between the cluster elements. This method has not been used previously in predictive clustering and is one of the contributions of the paper. Our experiments show that the SSPD can be efficiently approximated by means of sampling.

Our approach combines local models (motifs of DNA) with global models (PCTs). The local models are used to describe clusters and can be used to predict cluster membership. Such a combination of models is a typical feature required from an inductive database: a first query is used to mine the local models and a second query returns global models based on these local models.

The experimental evaluation shows that PCTs can be used for predicting the expression response of yeast genes to different changes in environmental conditions. This, however, proved to be a hard task and more research is required, e.g., to find more predictive features.

**Acknowledgments.** This work was supported by the IQ project (IST-FET FP6-516169). Jan Struyf is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

## References

1. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: 15th Int'l Conf. on Machine Learning, pp. 55–63 (1998)
2. Curk, T., Zupan, B., Petrovič, U., Shaulsky, G.: Računalniško odkrivanje mehanizmov uravnavanja istražanja genov. In: Prvo srečanje slovenskih bioinformatikov, pp. 56–58 (2005)
3. De Raedt, L.: A perspective on inductive databases. SIGKDD Explorations 4(2), 69–77 (2002)
4. Ernst, J., Nau, G.J., Bar-Joseph, Z.: Clustering short time series gene expression data. Bioinformatics 21(Suppl. 1), 159–168 (2005)
5. Ashburner, M., et al.: Gene Ontology: Tool for the unification of biology. The Gene Ontology Consortium. Nature Genet. 25(1), 25–29 (2000)
6. Ferri, C., Flach, P.A., Hernández-Orallo, J.: Learning decision trees using the area under the ROC curve. In: 19th Int'l Conf. on Machine Learning, pp. 139–146 (2002)
7. Fromont, E., Blockeel, H., Struyf, J.: Integrating decision tree learning into inductive databases. In: KDID 2006. LNCS, vol. 4747, pp. 81–96. Springer, Heidelberg (2007)
8. Garofalakis, M., Hyun, D., Rastogi, R., Shim, K.: Building decision trees with constraints. Data Mining and Knowledge Discovery 7(2), 187–214 (2003)
9. Gasch, A., Spellman, P., Kao, C., Carmel-Harel, O., Eisen, M., Storz, G., Botstein, D., Brown, P.: Genomic expression program in the response of yeast cells to environmental changes. Mol. Biol. Cell. 11, 4241–4257 (2000)
10. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM 39(11), 58–64 (1996)
11. Kaufman, L., Rousseeuw, P.J. (eds.): Finding groups in data: An introduction to cluster analysis. Wiley, Chichester (1990)
12. Lee, S.D., De Raedt, L.: An efficient algorithm for mining string data-bases under constraints. In: Goethals, B., Siebes, A. (eds.) KDID 2004. LNCS, vol. 3377, pp. 108–129. Springer, Heidelberg (2005)
13. Liao, T.W.: Clustering of time series data – a survey. Pattern Recognition 38, 1857–1874 (2005)
14. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2007)
15. Michalski, R.S., Stepp, R.E.: Learning from observation: conceptual clustering. In: Machine Learning: an Artificial Intelligence Approach, vol. 1, Tioga Publishing Company (1983)
16. Mitasiunaité, I., Boulicaut, J.-F.: Looking for monotonicity properties of a similarity constraint on sequences. In: ACM Symposium of Applied Computing SAC'2006, Special Track on Data Mining, pp. 546–552. ACM Press, New York (2006)

17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, San Francisco (1993)
18. Raileanu, L.E., Stoffel, K.: Theoretical comparison between the Gini index and information gain criteria. Annals of Mathematics and Artificial Intelligence 41(1), 77–93 (2004)
19. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken-word recognition. In: IEEE Transaction on Acoustics, Speech, and Signal Processing. LNAI, vol. ASSP-26, pp. 43–49. IEEE Computer Society Press, Los Alamitos (1978)
20. Sese, J., Kurokawa, Y., Monden, M., Kato, K., Morishita, S.: Constrained clusters of gene expression profiles with pathological features. Bioinformatics 20, 3137–3145 (2004)
21. Slavkov, I., Džeroski, S., Struyf, J., Loskovska, S.: Constrained clustering of gene expression profiles. In: Conf. on Data Mining and Data Warehouses (SiKDD 2005) at the 7th Int'l Multi-Conference on Information Society 2005, pp. 212–215 (2005)
22. Struyf, J., Džeroski, S.: Constraint based induction of multi-objective regression trees. In: Bonchi, F., Boulicaut, J.-F. (eds.) KDID 2005. LNCS, vol. 3933, pp. 222–233. Springer, Heidelberg (2006)
23. Todorovski, L., Cestnik, B., Kline, M., Lavrač, N., Džeroski, S.: Qualitative clustering of short time-series: A case study of firms reputation data. In: ECML/PKDD 2002 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning, pp. 141–149 (2002)
24. Torgo, L.: A comparative study of reliable error estimators for pruning regression trees. In: Coelho, H. (ed.) IBERAMIA 1998. LNCS (LNAI), vol. 1484, Springer, Heidelberg (1998)
25. Ženko, B., Džeroski, S., Struyf, J.: Learning predictive clustering rules. In: Bonchi, F., Boulicaut, J.-F. (eds.) KDID 2005. LNCS, vol. 3933, pp. 234–250. Springer, Heidelberg (2006)
26. Wagstaff, K.L.: Value, cost, and sharing: Open issues in constrained clustering. In: KDID 2006. LNCS, vol. 4747, pp. 24–41. Springer, Heidelberg (2007)

# Finding explained groups of time-course gene expression profiles with predictive clustering trees†

Ivica Slavkov,[a] Valentin Gjorgjioski,[a] Jan Struyf[b] and Sašo Džeroski*[a]

*Received 9th July 2009, Accepted 8th January 2010*
*First published as an Advance Article on the web 19th February 2010*
**DOI: 10.1039/b913690h**

In biology, analyzing time course data is usually a two-step process, beginning with clustering of similar temporal profiles. After the initial clustering, depending on the expert's knowledge, descriptions of the clusters are elucidated (*e.g.*, Gene Ontology terms that are enriched in the clusters). In this paper, we investigate the application of so-called predictive clustering trees (PCTs) for the analysis of time series data. PCTs are a part of a more general framework of predictive clustering, which unifies clustering and prediction. Their advantage over usual clustering approaches is that they partition the time course data into homogeneous clusters while at the same time providing symbolic descriptions of the clusters. We evaluate our approach on multiple yeast microarray time series datasets. Each dataset records the change over time in the expression level of yeast genes as a response to a specific change in environmental conditions. We demonstrate that PCTs are able to cluster genes with similar temporal profiles, yield a predictive model of the temporal profiles of genes based on a cluster prototype, and provide cluster descriptions, all in a single step.

## 1. Introduction

Gene expression is a temporal process that is highly regulated. Much work in bioinformatics studies this process in order to better understand the function of individual genes and to gain insight into complete biological systems. The task most commonly addressed in this context is the task of clustering time series of gene expression data, where the aim is to discover groups of genes with similar temporal profiles of expression and to find common characteristics of the genes in each group. Clustering genes by their time expression pattern is important, because genes that are co-regulated or have a similar function will have similar temporal profiles under certain conditions.

The purpose of our research is to develop a clustering approach that is well suited for analyzing short time series, and to demonstrate its usefulness on time series expression data. Besides finding clusters, *e.g.*, groups of genes, we also aim to find descriptions/explanations for the clusters. Instead of first clustering the expression time series and elucidating the characteristics of the obtained clusters later on (as done in, *e.g.*, ref. 1), we perform so-called constrained clustering, which yields both the clusters and their symbolic descriptions all in one step.

The constrained clustering is performed by using predictive clustering trees (PCTs), which are a part of a more general framework, namely predictive clustering. This general framework of predictive clustering combines clustering and prediction.[2] Predictive clustering partitions a given dataset into a set of clusters such that the instances in a given cluster are similar to each other and dissimilar to the instances in other clusters. In this sense, predictive clustering is identical to regular clustering.[3] The difference is that predictive clustering associates a predictive model to each cluster. This model assigns instances to clusters and provides predictions for new instances. So far, decision trees[2,4] and rule sets[5] have been used in the context of predictive clustering.

This paper investigates how predictive clustering can be applied to cluster time series,[6] *i.e.*, sequences of measurements of a continuous variable that changes over time. For example, Fig. 1a shows eight time series partitioned into three clusters: cluster $C_1$ contains time series that increase and subsequently decrease, $C_2$ has mainly decreasing time series and $C_3$ mainly increasing ones. Fig. 1b shows a so-called predictive clustering tree (PCT) for this set of clusters. The tree represents a hierarchical clustering of the time series, where each leaf corresponds to one of the three clusters. At each leaf, a prototype is given for the cluster. This is the predictive model associated with the cluster. Finally, each cluster is described by a set of conditions. For example, cluster $C_1$ includes all genes that are annotated with the Gene Ontology terms "GO:0043232" and "GO:0000313".

We first propose an extension of the general PCT induction algorithm[2] to the task of time series clustering. We use the name "Clus-TS" (Clustering-Time Series) for this extension. From a computational viewpoint, applying the PCT induction algorithm to time series clustering is non-trivial because the general algorithm requires computing a centroid for each cluster and for most distance measures suitable for time series clustering, no closed algebraic form centroid is known.

[a] *Dept. of Knowledge Technologies, Jožef Stefan Institute, Slovenia. E-mail: Ivica.Slavkov@ijs.si, Valentin.Gjorgjioski@ijs.si, Saso.Dzeroski@ijs.si*
[b] *Dept. of Computer Science, Katholieke Universiteit Leuven, Belgium. E-mail: Jan.Struyf@cs.kuleuven.be*

† This article is part of a Molecular BioSystems themed issue on Computational and Systems Biology.

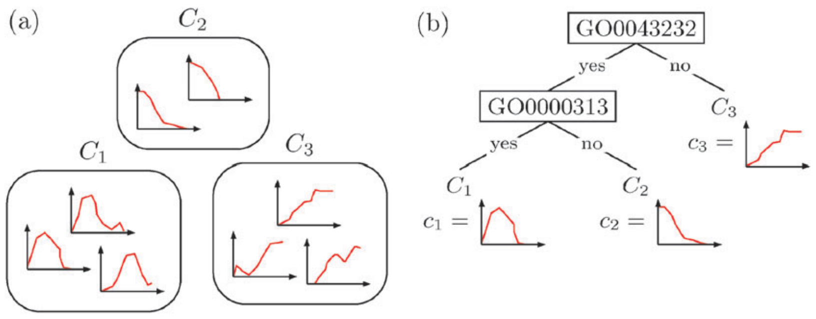


**Fig. 1** (a) A set of time series clustered into three clusters. (b) A predictive clustering tree associated with this clustering. Each leaf of the tree corresponds to one cluster and stores the cluster's prototype which is used for prediction.

We also demonstrate the usefulness of Clus-TS on several time series datasets generated by microarray expression profiling.[7] Each dataset records the change over time in the expression level of yeast genes in response to a different type of change in environmental conditions. There has been significant research related to clustering this type of short time series gene expression data,[1,8–13] using several different distance measures. Our approach uses an alternative distance measure (that mainly takes the shape of the time series into account) and constructs clusters together with their explanations in terms of a given set of descriptive features. Here, as descriptive features, we consider terms from the Gene Ontology (GO),[14] but this can be extended to any other type of gene descriptions (*e.g.*, KEGG pathways,[15] regulatory motifs). The GO terms appear in the internal nodes of the PCT (Fig. 1b) and provide a symbolic description of the clusters.

In the remainder of the paper, we first give an overview of related work. We next present our methodology in more detail: this includes a description of the predictive clustering framework, the PCT induction algorithm, the distance measure used for clustering, and the methodology for evaluating predictive error. We then present the results of our analysis of the yeast gene expression time profiles, where we evaluate our approach in terms of predictive error and the usefulness of the descriptions derived from the PCTs. We conclude the paper with a discussion in light of the presented results and methodology.

## 2. Related work

A large body of work has been devoted to the task of analyzing expression time series data. Bar-Joseph[16] presents an overview of the most important aspects that are relevant when analyzing expression time series data. This includes experimental design, data preprocessing (dealing with differences in sampling rates, missing values, and noise), finding significant genes, modeling gene interaction, and clustering expression time series.

Many different clustering algorithms[3] have been used to cluster expression time series data. The most well-known algorithm is probably UPGMA, which was proposed by Eisen *et al.* in 1998[17] and performs hierarchical clustering based on correlation. More recently, several advanced time series clustering methods have been presented. These model the time series, for example, using spline curves,[8,10] an autoregressive model,[11,13] or a mixture of hidden Markov models.[12]

Datta and Datta[9] compare six clustering algorithms for expression time series data experimentally. Their comparison includes two hierarchical clustering algorithms (among which UPGMA), divisive clustering (Diana), fuzzy clustering (Fanny), a model based clustering method, and *k*-means. They found Diana to be a solid and robust performer across different evaluation measures. A review of the most common evaluation measures for clustering is provided by Handl *et al.*[18]

Often, the clustering methods are not applied to all genes, but only to genes that do respond to the change in environmental conditions or treatment. A gene responds to the treatment if the null hypothesis stating that its expression over time is constant can be rejected.[19] Several methods have been proposed to identify such genes and a comparison can be found in Mutarelli *et al.*[20] An advantage of our method is that it detects such genes during the clustering process itself by assigning confidence values to genes.

Due to the cost of microarray analysis and of obtaining samples, most expression time series are relatively short ($\leq$8 points). Ernst *et al.*[1] propose a clustering method designed for such short time series. Their method creates all possible expression profiles under the constraint that the maximal expression change between subsequent time points is bounded by a fixed number of units. It then assigns time series to the closest profile (in terms of correlation) thereby forming clusters. Our method is also tailored to short time series, but instead of using correlation, we opt for a qualitative distance measure that can be reliably estimated from short time series.

After clustering, Ernst *et al.*[1] label the clusters by finding GO categories that are significantly enriched in the clusters. Our method also provides a description for each cluster in terms of GO categories, but finds these during the constrained clustering process itself. As a result, all genes in a cluster are guaranteed to belong to the GO categories from the description. This is closely related to the constrained clustering method by Sese *et al.*[21] The main difference is that their method deals with static gene expression data and not with time series, and that their cluster descriptions are restricted to item-sets.

## 3. Methodology

### 3.1 Prediction, clustering, and predictive clustering trees

Predictive modeling aims at constructing models that can predict a target property of an object from a description of the object. Predictive models are learned from sets of examples, where each example has the form (*D*,*T*), with *D* being an object description and *T* a target property value. For example, *D* can be the measured gene expression levels of a certain sample, and *T* whether the corresponding tissue is cancerous or healthy. While a variety of representations, ranging from propositional to first order logic, have been used for *D*, *T* is almost always a single target attribute called the class, which is discrete for classification problems or continuous for regression problems.

Clustering,[3] on the other hand, is concerned with grouping objects into subsets of objects (called clusters) that are similar with respect to their description *D*: this is called distance based clustering. There is no target property defined in clustering tasks. In conventional clustering, the notion of a distance (or conversely, similarity) is crucial: examples are considered to be points in a metric space and clusters are constructed such that examples in the same cluster are close according to a particular distance defined on the descriptive space *D*. A centroid (or prototypical example) may be used as a representative for a cluster. The centroid is the point with the lowest average (squared) distance to all the examples in the cluster, *i.e.*, the mean or medoid of the examples. Hierarchical clustering and *k*-means clustering are the most commonly used algorithms for this type of clustering.[3]

Predictive clustering[2] combines elements from both prediction and clustering. As in clustering, we seek clusters of examples that are similar to each other. The distance measure is defined on $D \cup T$, taking both the descriptive part and the target property into account. In addition, a predictive model must be associated to each cluster. The predictive model assigns new instances to clusters based on their description *D* and provides a prediction for the target property *T*. A well-known type of model that can be used to this end is the decision tree.[22] A decision tree that is used for predictive clustering is called a predictive clustering tree (PCT, Fig. 1b). Each node of a PCT represents a cluster. The conjunction of conditions on the path from the root to that node gives a description of the cluster. Essentially, each cluster has a symbolic description in the form of a rule (IF conjunction of conditions THEN cluster)‡, while a tree structure represents the hierarchy of clusters. Clusters that are not on the same branch of a tree do not overlap.

In Fig. 1, the description *D* of a gene consists of GO terms with which the gene is annotated, and the target property *T* is the time course expression recorded for that gene. In general, we could include both *D* and *T* in the distance measure. We are, however, most interested in the time course part. Therefore, we define the distance measure only on *T*. We consider the so-called qualitative distance measure (QDM),[24] described in section 3.5. The resulting PCT (Fig. 1b) represents a clustering that is homogeneous w.r.t. *T* and the internal nodes of the tree provide a symbolic description of the clusters. Note that a PCT can also be used for prediction: we can use the tree to assign a new instance to a leaf and take the centroid (denoted with $c_i$ in Fig. 1b) of the corresponding cluster as a prediction.

‡ This idea was first used in conceptual clustering.[23]

## 3.2 Building predictive clustering trees

The generic algorithm for constructing PCTs[2] is presented in Table 1. It is a variant of the standard greedy recursive top-down decision tree induction algorithm used in ref. 22. It takes as input a set of instances *I*; in our case these are genes described by GO terms and their associated time course measurements. The algorithm calls the procedure BestTest (Table 1, right) to search for the best acceptable test (GO term) that can be put in a node. If such a test *t** can be found then the algorithm creates a new internal node labeled *t**, splits the instances into several subsets (partition *P**) according to the outcome of the test for each instance, and calls itself recursively to construct a tree for each of the subsets in *P**. If no acceptable test can be found, then the algorithm creates a leaf, and the recursion terminates. The procedure "Acceptable" defines the stopping criterion of the algorithm, *e.g.*, specifying maximum tree depth or a minimum number of instances in each leaf. We enforce different constraints on the size of the tree by means of the post pruning method proposed by Garofalakis *et al.*,[25] which employs dynamic programming to find the most accurate subtree no larger than a given number of leaves.

Up till here, the algorithm is identical to a standard decision tree learner. The main difference is in the heuristic that is used for selecting the tests. For PCTs, this heuristic is the reduction in variance (weighted by cluster size, see line 6 of BestTest). Maximizing variance reduction maximizes cluster homogeneity. The next section discusses how cluster variance can be defined for time series.

An implementation of the PCT induction algorithm is available in the Clus system, which can be obtained at http://www.cs.kuleuven.be/˜dtai/clus.

## 3.3 Computing cluster variance

The PCT induction algorithm requires a measure of cluster variance in its heuristics. The variance of a cluster *C* can be defined based on a distance measure as

$$Var(C) = \frac{1}{|C|}\sum_{X\in C} d^2(X,c), \qquad (1)$$

**Table 1** Pseudo-code for the algorithm Clus that induces predictive clustering trees (PCTs). The two key subroutines of the algorithm are BestTest(*I*) and Centroid(*I*). The first selects the best test *t** among the possible tests, according to the heuristic *h*, which for each test *t* measures the reduction of variance between the dataset *I* and the partition $P = I_1, I_2$ produced by the test. The second procedure calculates the cluster centroid

```
procedure PCT($I$)
 1: $t^* = \text{BestTest}(I)$
 2: if $t^* \neq none$ then
 3:     for each $I_k \in \mathcal{P}^*$ do
 4:         $tree_k = \text{PCT}(I_k)$
 5:     return node($t^*$, $\bigcup_k \{tree_k\}$)
 6: else
 7:     return leaf(Centroid($I$))

procedure Centroid($I$)
    return $argmin_q \sum_{x\in I} d^2(x,q)$

procedure BestTest($I$)
 1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
 2: for each possible test $t$ do
 3:     $I_1 = \{e | e \in I \wedge t(e) = true\}$
 4:     $I_2 = I \setminus I_1$
 5:     $\mathcal{P} = \{I_1, I_2\}$
 6:     $h = Var(I) - \sum_{I_k \in \mathcal{P}} \frac{|I_k|}{|I|} Var(I_k)$
 7:     if $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$
        then $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
 8: return $t^*$
```

with $c$ the cluster centroid of $C$. To cluster time series, $d$ should be a distance measure defined on time series, such as the QDM defined in section 3.5.

The centroid $c$ can be computed as $\operatorname{argmin}_q \sum_{X \in C} d^2(X, q)$. We consider two possible representations for $c$: (a) the centroid is an arbitrary time series, and (b) the centroid is one of the time series from the cluster (the cluster prototype). In representation (b), the centroid can be computed with $|C|^2$ distance computations by substituting $q$ with each time series in the cluster. In representation (a), the space of candidate centroids is infinite. This means that either a closed algebraic form for the centroid is required or that one should resort to approximative algorithms to compute the centroid. No closed form for the centroid is known in representation (a) for the QDM distance.

An alternative way to define cluster variance is based on the sum of the squared pairwise distances (SSPD) between the cluster elements, *i.e.*,§

$$Var(C) = \frac{1}{2|C|^2} \sum_{X \in C} \sum_{Y \in C} d^2(X, Y). \qquad (2)$$

The advantage of this approach is that no centroid is required. It also requires $|C|^2$ distance computations. This is the same time complexity as the approach with the centroid in representation (b). Hence, using the definition based on a centroid is only more efficient if the centroid can be computed in time linear in the cluster size. This is the case for the Euclidean distance in combination with using the pointwise average of the time series as centroid. For QDM no such centroids are known. Therefore, we choose to estimate cluster variance using the SSPD.

A second advantage is that (2) can be easily approximated by means of sampling, *e.g.*, by using,

$$Var(C) = \frac{1}{2|C|m} \sum_{X \in C} \left( \sum_{Y \in \text{sample}(C,m)} d^2(X, Y) \right), \qquad (3)$$

with sample($C$, $m$) a random sample without replacement of $m$ elements from $C$, instead of (2) if $|C| \geq m$. The computational cost of (3) grows only linearly with the cluster size. In the experimental evaluation, we only use (3), as a previous experimental comparison shows only small differences between (2) and (3) (results not shown).

The PCT induction algorithm places cluster centroids in its leaves, which can be inspected by the domain expert and used as a prediction. For these centroids, we use representation (b) as discussed above.

### 3.4 Estimating the predictive error of PCTs

PCTs make predictions just like regular decision trees.[22] They sort each test instance into a leaf and assign as prediction the label of that leaf. PCTs label their leaves with the training set centroids of the corresponding clusters.

To evaluate the predictive performance of PCTs, we first need an error measure and also a method to estimate it. For an error measure we use the root mean squared error (RMSE), which is defined as:

$$\text{RMSE}(I, T) = \sqrt{\frac{1}{|I|} \sum_{X \in I} d^2(T(X), \text{series}(X))}, \qquad (4)$$

with $I$ the set of test instances, $T$ the PCT that is being tested, $T(X)$ the time series predicted by $T$ for instance $X$, series($X$) the actual series of $X$, and $d$ the qualitative time course distance measure (described in section 3.5).

For estimating the predictive performance of the PCTs we use $k$ fold cross-validation. In cross-validation the dataset $D$ is first split into $k$ random subsets $\{D_1, D_2, \ldots D_k\}$. We then use $k - 1$ subsets to build the predictive model (in this case the PCT) and we record its error (*i.e.*, RMSE) on the left-out subset(fold). We repeat this $k$ times, each time leaving out a different subset for testing the error. We obtain the final error estimate by averaging the errors obtained for all of the $n$ instances of the dataset $D$.

$$\text{err} = \frac{1}{n} \sum_{i \in D} err(PCT(D_{-i}), D_i) \qquad (5)$$

### 3.5 Qualitative distance measure

Several distance measures have been defined for time series. If all time series have the same length then one can represent them as real valued vectors and use standard vector distance measures such as the Euclidean or Manhattan distance. It is also possible to use a correlation based measure to determine the degree of linear dependence between two time-series.[17] Dynamic Time Warping (DTW)[26] is appropriate to capture non-linear distortion along the time axis and it is suitable if the time series are not properly synchronized (this is useful if one is delayed, or if the two time series are not of the same length).

These measures are, however, not always appropriate for time course clustering, and in particular not for analyzing the short time courses of expression data. The simple Euclidean or the DTW distance mainly capture the difference in scale and baseline. If a given time series is identical to a second time series, but scaled by a certain factor or offset by some constant, then the two time series will be distant (Fig. 2). Correlation is difficult to properly estimate if the number of observations is small (*i.e.*, short time course data) and it only captures the linear dependencies between the time series.

For our application (*i.e.*, clustering short time course gene expression data), the differences in scale and size are not of great importance; only the shape of the time series matters. Namely, we are interested in grouping together time-course profiles of genes that react in the same way to a given condition, regardless of the intensity of the up- or down-regulation.

For that reason, we use the qualitative distance measure proposed by Todorovski *et al.*[24] It is based on a qualitative comparison of the shape of the time series. Consider two time series $X$ and $Y$ (Fig. 2). Then choose a pair of time points $i$ and $j$ and observe the qualitative change in the value of $X$ and $Y$ at these points. There are three possibilities: increase ($X_i > X_j$), no-change ($X_i \approx X_j$), and decrease ($X_i < X_j$). $d_{\text{qual}}$ is obtained

§ The factor 2 in the denominator of (2) ensures that (2) is identical to (1) for the Euclidean distance.

by summing the difference in qualitative change observed for $X$ and $Y$ for all pairs of time points, *i.e.*,

$$d_{\text{qual}}(X, Y) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \mathit{Diff}(q(X_i, X_j), q(Y_i, Y_j))}{N \cdot (N-1)}, \quad (6)$$

with $\mathit{Diff}(q_1,q_2)$ a function that defines the difference between different qualitative changes (Table 2, Fig. 2). Roughly speaking, $d_{\text{qual}}$ counts the number of disagreements in change of $X$ and $Y$.

QDM does not have the drawbacks of correlation based measures. First, it can be computed for very short time series, without decreasing the quality of the estimate. Second, it captures the similarity in shape of the time series, regardless of whether their dependence is linear or non-linear (Fig. 2).

**Table 2** The definition of $\mathit{Diff}(q_1,q_2)$

| $\mathit{Diff}(q_1,q_2)$ | Increase | No-change | Decrease |
|---|---|---|---|
| Increase | 0 | 0.5 | 1 |
| No-change | 0.5 | 0 | 0.5 |
| Decrease | 1 | 0.5 | 0 |

## 4. Results

In this section, we present and evaluate the results of the analysis of time course gene expression data with PCTs. The expression data measures the response of yeast genes to different types of environmental stress and we first give a brief description of it. We then show how the produced PCT models can be interpreted in order to obtain biologically meaningful knowledge. We also discuss the similarity of the biological processes that are involved in the response to different types of stress. We finally present the results of experiments performed for assessing the predictive performance of the constructed PCTs.

### 4.1 Dataset description

For our experiments, we use the time-series expression data from the study conducted by Gasch *et al.*,[7] which are publicly available. The purpose of the study is to explore the changes in expression levels of yeast (*Saccharomyces cerevisiae*) genes under diverse environmental stresses. The gene expression levels of around 5000 genes are measured at different time points using microarrays. The data is log-transformed and

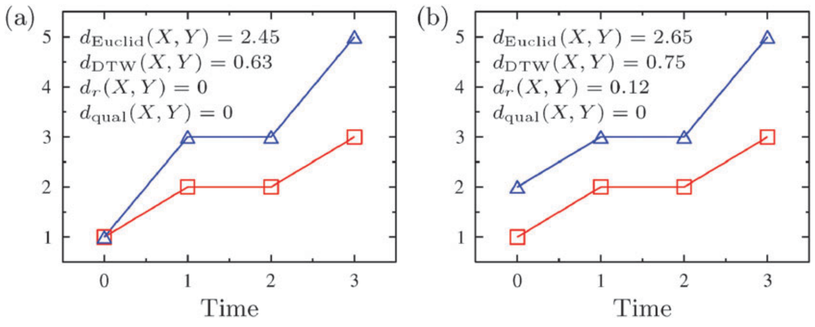


**Fig. 2** Comparison of four distance measures for time series. Time series (a) are linearly related resulting in $d_r(X, Y) = 0$. Time series (b) are non-linearly related, but still have a similar shape, resulting in $d_{\text{qual}}(X, Y) = 0$.

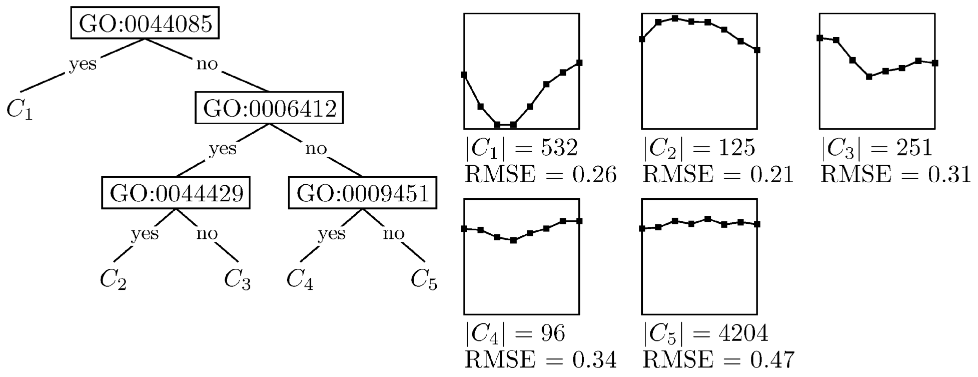


**Fig. 3** On the left-hand side, we show a sample PCT with 5 leaves, produced for the diamide treatment dataset. The GO terms that appear in the nodes are used as descriptions for clusters $C_1$ to $C_5$, found at the leaves of the tree. On the right-hand side, we show each predicted cluster prototype, and its related cluster size and RMSE. Clusters $C_1$ to $C_3$ show significant temporal changes in gene expression and have a relatively low error. Cluster $C_1$ includes genes that have an immediate and very significant down-regulation during diamide exposure. $C_3$ shows the same tendency, except the genes are less down-regulated and there is a short time-lag in their response. Cluster $C_2$ contains genes that are up-regulated during stress. All three cluster prototypes show that changes in gene expression levels are transient. If we just follow the "no" branch of the tree we reach the cluster $C_5$. Its size indicates that the bulk of genes fall into this cluster. We believe that most of the genes that do not have a coordinated stress response fall into this cluster. Indicative of this is the cluster prototype, which shows no major changes in gene expression and has a large error.

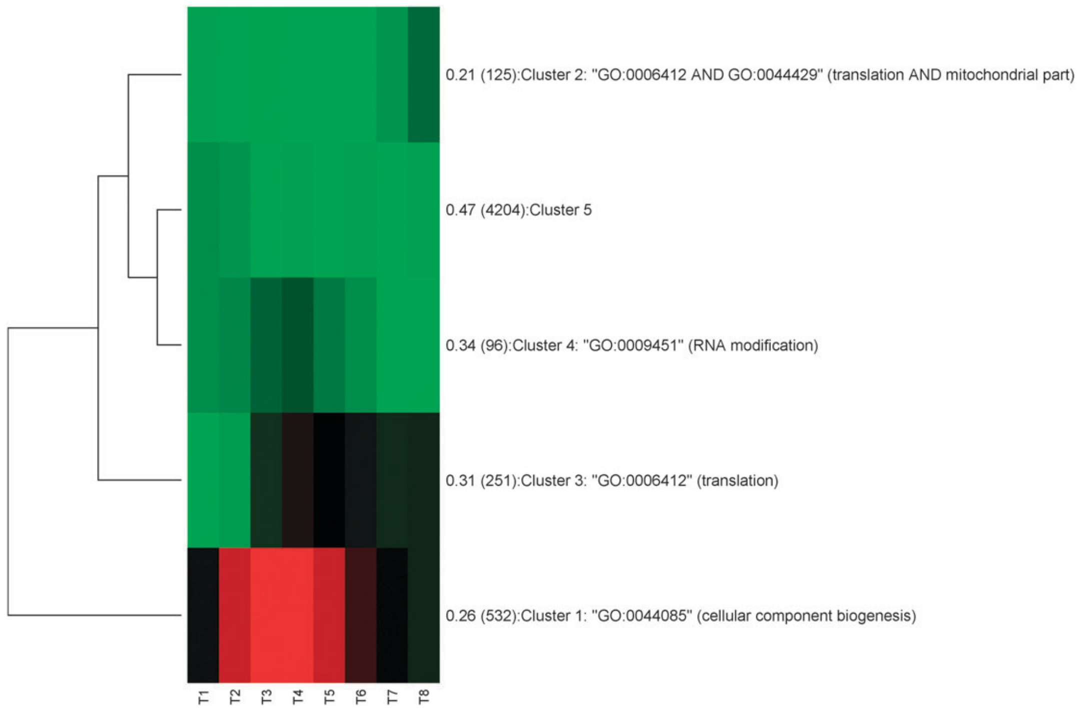


**Fig. 4** Heatmap of the cluster prototypes and their accompanying descriptions from the PCT in Fig. 3. The first number on the right-hand side of the heatmap is the cluster's RMS error, the number in brackets is the cluster's size, the cluster's descriptions follow after the colon. "Cluster 2" contains genes that are involved in translation and whose protein products are a part of the mitochondria. These genes are significantly up-regulated. Cellular component biogenesis is strongly repressed, as evident on "Cluster 1". All of the clusters show a transient response to diamide, except "Cluster 5" which shows almost a constant temporal expression profile.

normalized based on the time-zero measurement of yeast cells under normal environmental conditions.

Various sudden changes in the environmental conditions are tested, ranging from heat shock to amino acid starvation for a prolonged period of time. We used a total of 10 datasets (different stress conditions) for our analysis. We perform a comparative analysis of the obtained descriptions from all of the datasets in section 4.4. For a more detailed discussion of the obtained descriptions (section 4.3) we considered four representative datasets, for different types of stressful conditions (temperature, chemical and starvation). Namely, we consider heat shock (from 25 to 37 °C), diamide treatment, DTT (dithiothreitol) exposure and nitrogen starvation.

From these original time series datasets, we construct extended datasets by including gene descriptions. We obtained the GO term annotations for each yeast gene from the Gene Ontology[14] (version June, 2009). As the GO terms are structured in a hierarchy, we use both the part_of and is_a relations to include all relevant GO terms for each gene. To limit the number of features, we set a minimum frequency threshold: each included GO term must appear in the annotations for at least 50 of the 5000 genes.

## 4.2 Interpretation of PCTs for time course profiles

As explained in section 1, a PCT represents a hierarchical clustering of the time course data, where each leaf corresponds to one cluster. In Fig. 3, we present a sample PCT. For practical purposes, we show a small tree with just 5 leaves, obtained when yeast is exposed to diamide. We also show the cluster centroids for each of the leaves. By following the path from the root of the tree to a leaf, we can obtain the description for each of the clusters.

For example, if we want to derive the description of cluster $C_2$, we begin from the root GO term "GO:0044085", we follow the "no" branch, obtaining the description "GO:0044085 = no". We then add the "GO:0006412 = yes" and "GO:0044429 = yes" by following the "yes" branches ending up at cluster $C_2$. So, the final description of cluster $C_2$ is the following conjunction: "GO:0044085 = no AND GO:0006412 = yes AND GO:0044429 = yes". This can be interpreted as follows: genes that are annotated by both "GO:0006412" and "GO:0044429", but not by "GO:0044085" are contained in cluster $C_2$ and have a temporal profile represented by the prototype of cluster $C_2$.

It should be noted here that for our application only the positive branches of the tree are semantically meaningful. In a biological context, the description "GO:0044085 = no" is not very meaningful because it simply tells us that the genes in cluster $C_2$ are not annotated by that term. Therefore, to describe a cluster we only take the positive "yes" terms, which means that for describing $C_2$ we would only use "GO:0006412 = yes AND GO:0044429 = yes".

After deriving the descriptions from all of the clusters (except for cluster $C_5$), we represent them using a heatmap (Fig. 4). Each row in the heatmap represents a cluster prototype, the more intense the colours, the larger the up- or down-regulation of the genes contained in that cluster. Accompanying the rows, on the right-hand side, is the error of

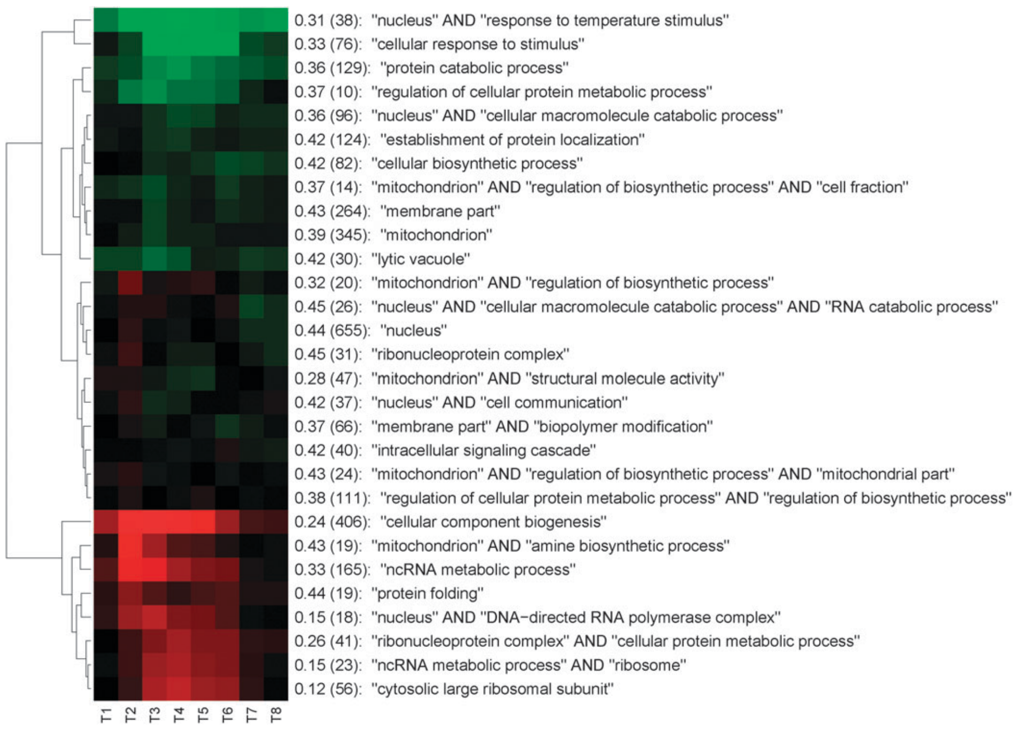


**Fig. 5** When yeast is exposed to heat shock, several clusters of genes show significant, but transient changes in expression levels. According to the heatmap intensity, genes involved in response to temperature stimulus are most strongly induced. Down-regulated are genes involved in biosynthesis processes and genes that code for ribosomal proteins.

each cluster (RMSE, described in section 3.4), the cluster size and the cluster description. Note that the heatmap ordering of the cluster prototypes does not match the ordering produced by the PCTs, but it is a permutation of it. This is for visualization purposes, in order to have all of the up- and all of the down-regulated cluster prototypes grouped together.

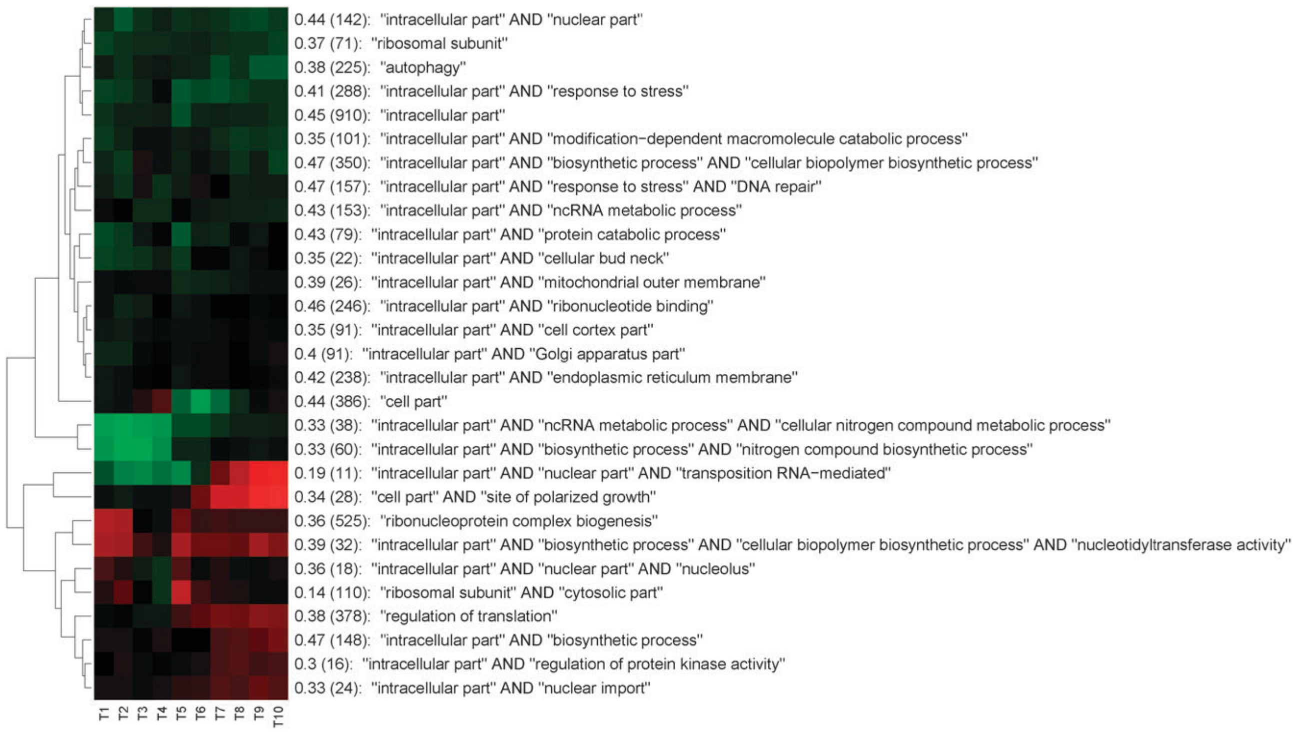


**Fig. 6** Under nitrogen starvation conditions, there is more of a steady down-regulation of genes, rather than a transient pattern. Genes involved in nitrogen metabolism are slowly down-regulated as well as genes coding for ribosomal proteins.

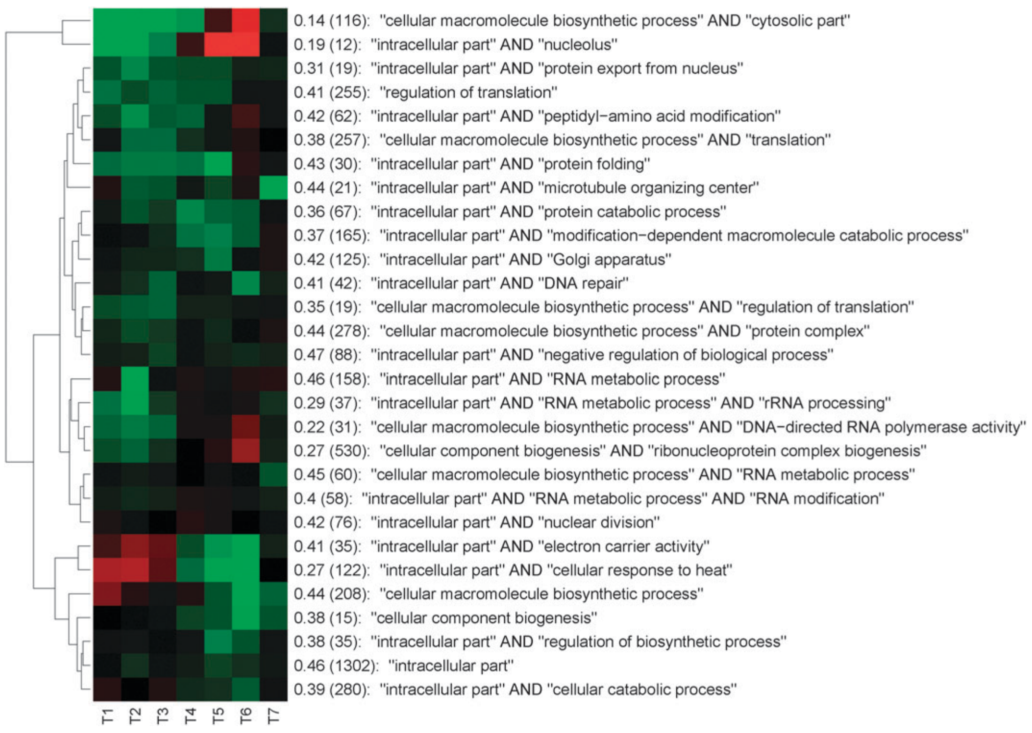


**Fig. 7** DTT treatment of yeast interferes with proper protein folding and changes the cellular redox state. Therefore, an up-regulation of genes involved in heat response and electron carrier activity is evident. More general biosynthetic processes and ribosomal proteins synthesis (nucleolus) are inhibited, *i.e.*, these genes are down-regulated.

### 4.3 Descriptions of yeast stress response clusters

We apply the procedure for deriving cluster descriptions from the previous section, on PCTs constructed for several datasets taken from a study of yeast stress response.[7] While PCTs were applied on all 10 datasets we only present in detail the results for four different stress conditions (heat shock, nitrogen starvation, diamide and DTT exposure). We present the final descriptions by using heatmaps given in Fig. 5–8.

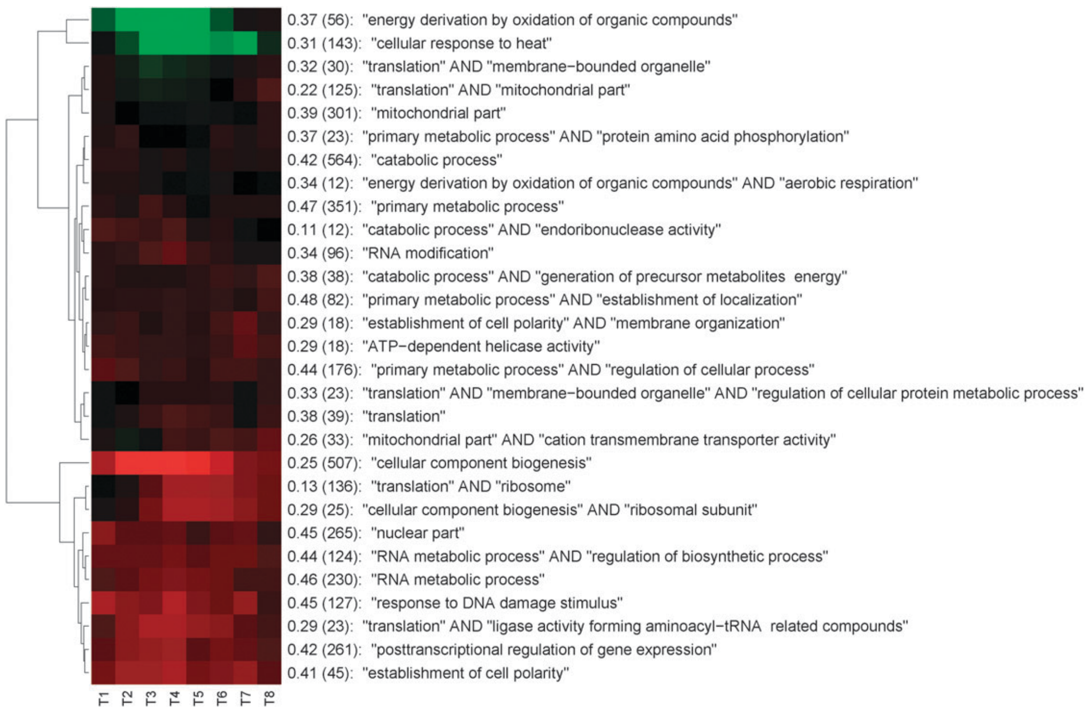


**Fig. 8** Exposure to diamide causes a response similar to DTT treatment and heat shock, in terms of response to protein folding inhibition. Also oxidation of organic compounds is strongly up-regulated, while the down-regulation of biogenesis and ribosomal genes is also apparent.

We first consider the heatmap for the Heat Shock dataset, presented in Fig. 5. One can quickly identify two groups of temporal profiles in this figure: one that shows significant up-regulation and another one that shows significant down-regulation of genes. These significant changes are only transient in nature, meaning that genes first quickly react to the heat shock and then after an adaptation period go back to normal expression levels. This is an expected behavior, also noted in ref. 7, which shows that the predicted cluster prototypes are consistent with the biological reality.

From the induced genes, those involved in cellular response to stimulus, specifically temperature stimulus, show the most notable changes. In the repressed genes group, we can notice two slightly different groups with respect to the delay of response to heat shock. The first group that is quick to react, consists of genes involved in biogenesis and different biosynthetic and metabolitic processes. A slight delay in down-regulation is exhibited by genes coding for ribosomal proteins, which is consistent with general stress response (ref. 7).

In contrast to heat shock, when yeast is subjected to nitrogen starvation, there is no transient temporal pattern present but more of a steady down-regulation of genes, as evident in Fig. 6. Genes involved in nitrogen metabolism slowly decrease their activity, while genes involved in cell growth are most significantly repressed. There is also a slight increase in the activity of autophagy genes.

The elicited response of yeast to DTT (dithiothreitol) is presented in Fig. 7. There is a small group of genes that is repressed over time. These are involved in general biosynthetic processes and genes that code for ribosomal proteins found in the nucleolus. Genes that were most induced are involved in electron carrier activity and genes that are a part of the general response of heat. This is the cell's response to the changed cellular redox state and to the inhibition of protein folding caused by DTT.[7]

Diamide exposure caused a response that can be seen as a combination between the response to heat shock and DTT. Genes involved in the heat shock response were induced (due to protein folding inhibition) as were genes involved in oxidation of organic compounds. As part of the general stress response, genes involved in cell component biogenesis were strongly repressed, as well as (with a small time-lag) genes coding for ribosomal proteins.

Note that we present descriptions derived from PCTs (in Fig. 5–8) from trees of size 60 (with 30 leaves/clusters). These usually consist of GO terms referring to general cell processes or locations. We chose this size as an optimal tree size, appropriate for viewing and with an acceptably low error (RMSE) (Fig. 10(a), (c), (e) and (g)). For obtaining clusters with more specific descriptions, one might consider larger trees.

## 4.4 Semantic similarity of biological processes involved in different types of stress

In the previous section, we presented the GO descriptions of the clusters of gene expression time profiles for four different stress conditions. We briefly discussed their similarities and differences in terms of the kind of cellular processes involved in stress response. Here, we focus on a more quantitative analysis of the derived GO descriptions, where we also include a whole range of stress conditions.

To quantitatively compare the GO descriptions of the different clusterings, we use the semantic similarity measure between GO terms proposed by Wang *et al.*[27] Given two GO terms, this measure quantifies their functional similarity, by considering their common ancestors information from the Gene Ontology. By using the semantic similarity measure, we first determine the similarity between pairs of groups of GO terms, corresponding to pairs of descriptions of PCTs/clusterings of yeast genes for different stressful conditions. We proceed by performing hierarchical clustering of the different stress types, in order to determine to which stressful conditions the yeast genes respond in the most functionally similar way (Fig. 9).

In Fig. 9, we can see that the cell response to heat shock is most similar to the response to DTT exposure and to the response when yeast is undergoing diauxic shift. This is due primarily to the response of genes to protein unfolding, which is the initial response to heat shock and DTT exposure. It is also due to the induction of genes involved in alternative carbon source utilization, which happens during diauxic shift and also as an aftermath of the heat shock.

Hyper and hypo osmotic shock are also grouped together, which is expected because they involve the response of the same set of biological processes, but they respond in an temporally inverse manner.[7]

Diamide is grouped with AA starvation and at a later stage with $H_2O_2$ exposure, which is expected due to its response being very similar to the response of these.[7] Overall, we can

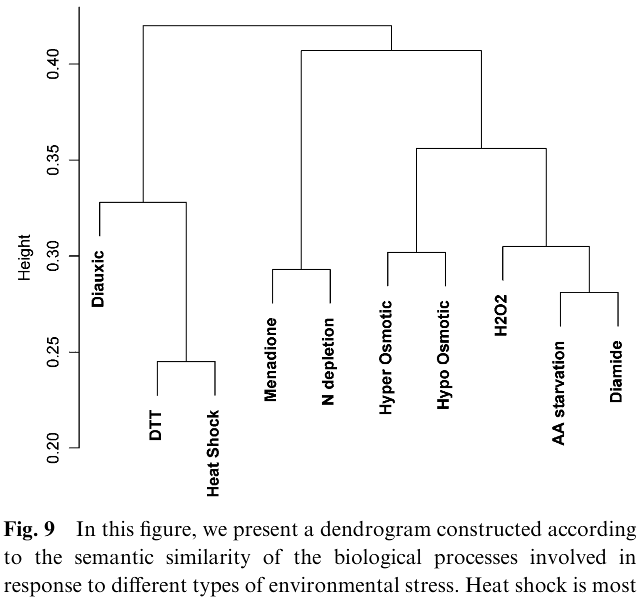


**Fig. 9** In this figure, we present a dendrogram constructed according to the semantic similarity of the biological processes involved in response to different types of environmental stress. Heat shock is most similar to DTT exposure, which can be attributed to the protein unfolding which initially occurs in both types of stress. The other similarity to diauxic shift appears as a result of activation of processes for utilizing alternative carbon sources in the aftermath of heat shock. Hyper and hypo osmotic conditions are grouped together as they involve the same processes in response to the shock. Diamide is most similar to AA starvation and then to $H_2O_2$ exposure. Overall there is high similarity of all biological processes involved in different stress responses, which is indicative of the existence of a general stress response mechanism.

notice that the similarity (*i.e.*, distance) between the different stress conditions is relatively high (low), which implies that there is a commonality of cell responses to different types of stress, *i.e.*, a general stress response mechanism.[7]

## 4.5 Predicting time series with PCTs

We compare the PCTs built by Clus-TS (section 3.2) to a default predictor DEF that always predicts the overall training set centroid. We estimate the RMSE of these predictions by using 10 fold cross-validation, as described in section 3.4. This means that when estimating the error for each fold, the training set contains approximately 4500 genes and the testing fold approximately 500 genes.

We first perform experiments for different maximum PCT sizes and we measure the respective RMSE of the corresponding PCTs. In Fig. 10((a), (c), (e) and (g)), we present the results for different values of the size upper bound. From the results,

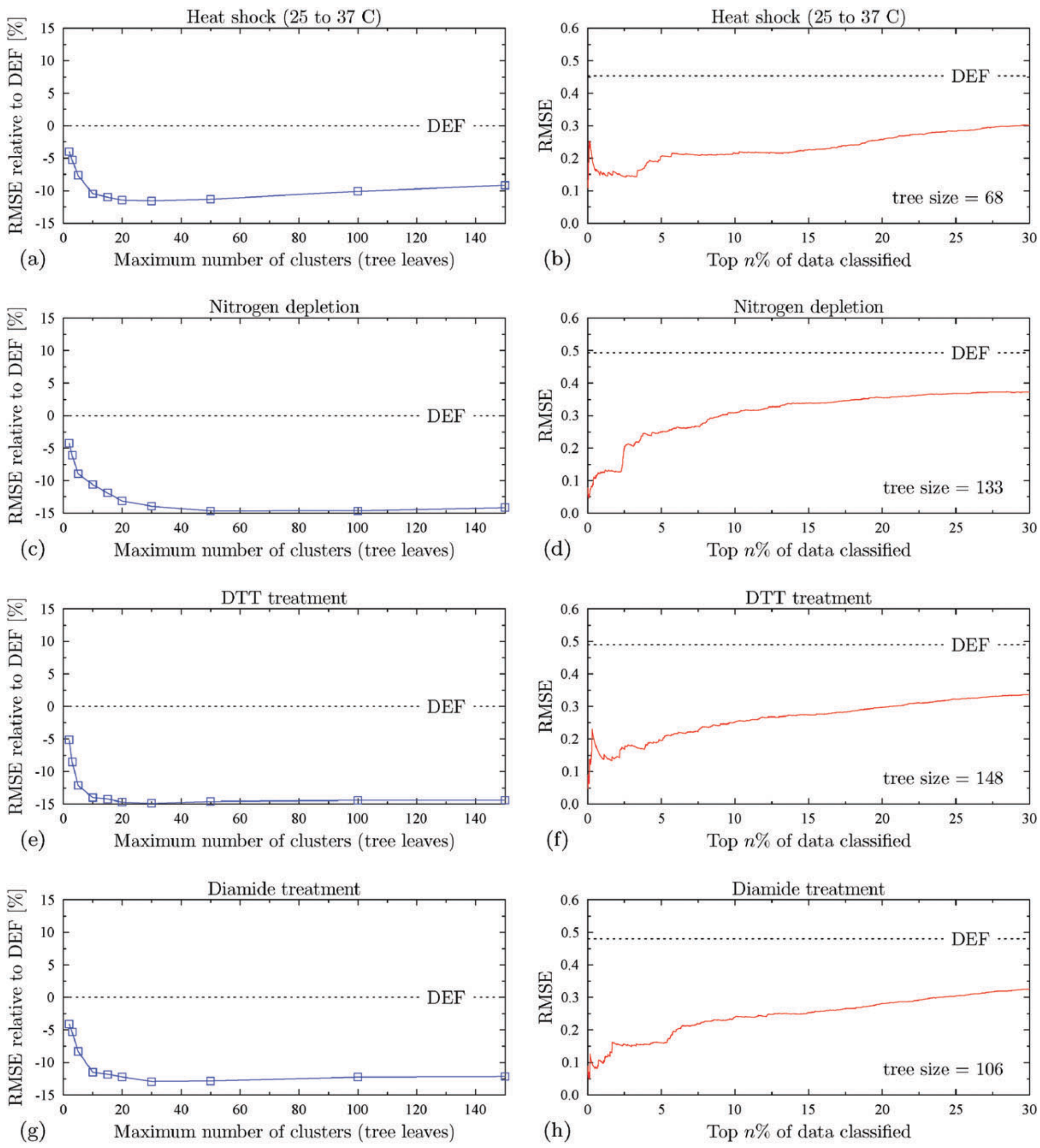


**Fig. 10** A comparison of predictive error (RMSE) of PCTs for different number of clusters ((a), (c), (e) and (g)) and percentage of data classified ((b), (d), (f) and (h)). When increasing the maximum tree size (number of leaves) the RMSE decreases until the size of the tree reaches 20–30 leaves (*i.e.*, clusters). The maximal improvement in the overall RMSE, as compared to the default (DEF) error, is about 15%. This small decrease in the error is problem specific, *i.e.*, has a biological background: not all genes have a coordinated response to the different stresses. Therefore, the PCTs are only able to correctly predict the time-course profile of a limited number of genes. This is evident in (b), (d), (f) and (h). For about 5% of the genes, PCTs are able to correctly predict their time-course profile with a relatively low RMSE as compared to the default (DEF). DEF is the default predictor that always predicts the overall training set centroid.

we can see that the optimal tree size for the PCTs is around 30 leaves. But, as one can notice, the overall RMSE is still relatively high. We hypothesize that the overall high error (RMSE) is domain specific, *i.e.*, there is a biological explanation for it.

Namely, the PCTs cluster genes that are annotated by similar GO terms and have a similar response in expression level to a certain change in environmental conditions. One problem is that, as noted by Gasch *et al.*,[7] only a subset of the genes (about 900) have a stereotypical response to environmental stress. That is, only a subset of the genes can be accurately clustered, whereas the other genes have an uncorrelated response. As a result, we hypothesize that the PCTs are able to accurately predict the time series of only a subset of the genes. We therefore perform the following experiment. Besides recording the predicted time series for each test set gene, we also record a confidence value for each prediction. We then sort the genes by confidence value and compute the RMSE of the top $n$ percent most confident predictions. We use the training set RMSE of the leaf that made the prediction as a confidence estimate. This is similar to the approach used for generating a ROC curve for a decision tree.[28] We present the results in Fig. 10((b), (d), (f) and (h)). PCTs are obtained with the same parameters as before, except that we use validation set based pruning instead of specifying a size constraint on the PCTs. Clus-TS now uses 1000 genes of the original training set for pruning and the rest for the tree construction (as suggested by ref. 29). Simply selecting a PCT from Fig. 10((a), (c), (e) or (g)) is unfair; it corresponds to optimizing the size parameter on the test set. The results show (Fig. 10) that more accurate predictions are obtained if we restrict the test set based on the confidence of the predictions. For example, if time course profiles are predicted for the 5% of genes with highest confidence then the RMSE decreases to about 50% of that of DEF. This is also shown in Fig. 3.

## 5. Conclusions

The typical approach to analyzing time-course expression data is to first group together genes with similar temporal profiles into clusters, which are then subsequently explained in terms of gene properties (such as GO annotations). We present a novel methodology for clustering time course profiles of gene expression data, which unifies the two steps of clustering and inferring a cluster description. The methodology produces a hierarchical clustering, called a predictive clustering tree, where each cluster is described by a conjunction of gene properties (such as GO terms).

There are several advantages of our approach over other analysis methods. First, we perform clustering and provide cluster explanations in a single step. The descriptions can use practically any gene-related information, although for our experiments we only included Gene Ontology terms. Second, in contrast to the usual distance measure used for clustering (typically correlation based), our approach uses a qualitative distance measure (QDM), which was specifically designed to deal with short time course data. This measure explicitly takes into account the temporal nature of the gene expression profiles, and captures mostly the similarity in the shape of the time course data, which is very important for the application at hand. Third, the PCTs also enable the prediction of gene expression time profiles for genes based on their annotations (functions), which is usually not possible with other mainstream clustering approaches.

We apply the proposed methodology to cluster time course data representing yeast gene response to environmental stress. This is repeated for different types of stress producing different PCTs, thus producing different clusters and cluster explanations in terms of GO annotations. Upon close inspection, the explanations of the clusters were consistent with previously published biological results.[7] Furthermore, clusters with similar descriptions under different stress conditions were identified, mainly related to biosynthesis and ribosomal proteins. The results demonstrate the usefulness of our method for analyzing time-course expression data.

Several directions for further work remain to be explored. We consider first and foremost extending our approach to a so-called multi-target approach. Instead of considering a single time course at a time, for different (stress) conditions, we can consider the responses to different kinds of environmental conditions simultaneously. The application of this would be, for example, discovering a common stress response pattern.[30] Instead of producing a separate PCT for each condition, we would obtain just one PCT model for all. Another direction of further research includes the identification of groups of genes with coordinated response. Namely, the hierarchical nature of the PCTs, besides producing compact clusters of "stress" response genes, also produces some clusters that contain genes without a coordinated response. To focus on clusters of genes with coordinated response, we plan to further investigate the use of the so-called predictive clustering rules[5] for analyzing short time course data. Finally, we would like to apply the proposed approach to other time course gene expression data from different biological domains.

## References

1 J. Ernst, G. J. Nau and Z. Bar-Joseph, Clustering short time series gene expression data, *Bioinformatics*, 2005, **21**(Suppl. 1), i159–168.
2 H. Blockeel, L. De Raedt and J. Ramon, Top-down induction of clustering trees, in *15th Int'l Conf. on Machine Learning*, 1998, pp. 55–63.
3 L. Kaufmanand P. J. Rousseeuw*Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.
4 J. Struyf and S. Džeroski, Constraint based induction of multi-objective regression trees, *Lect. Notes Comput. Sci.*, 2006, **3933**, 222–233.
5 B. Ženko, S. Džeroski and J. Struyf, Learning predictive clustering rules, *Lect. Notes Comput. Sci.*, 2005, **3933**, 234–250.
6 T. W. Liao, Clustering of time series data—a survey, *Pattern Recognit.*, 2005, **38**, 1857–1874.
7 A. Gasch, P. Spellman, C. Kao, O. Carmel-Harel, M. Eisen, G. Storz, D. Botstein and P. Brown, Genomic expression program in the response of yeast cells to environmental changes, *Mol. Biol. Cell.*, 2000, **11**, 4241–4257.
8 Z. Bar-Joseph, G. Gerber, T. Jaakkola, D. Gifford and I. Simon, Continuous representations of time series gene expression data, *J. Comput. Biol.*, 2003, **10**, 341–356.
9 S. Datta and S. Datta, Comparisons and validation of statistical clustering techniques for microarray gene expression data, *Bioinformatics*, 2003, **19**(4), 459–466.
10 N. Heard, C. Holmes and D. Stephen, A quantitative study of gene regulation involved in the immune response of anopheline

mosquitoes: An application of bayesian hierarchical clustering of curves, *J. Am. Stat. Soc.*, 2006, **101**, 18–29.
11 M. Ramoni, P. Sebastiani and I. S. Kohane, Cluster analysis of gene expression dynamics, *Proc. Natl. Acad. Sci. U. S. A.*, 2002, **99**(14), 9121–9126.
12 A. Schliep, A. Schonhuth and C. Steinhoff, Using hidden Markov models to analyze gene expression time course data, *Bioinformatics*, 2003, **19**, 255i–I272.
13 L. Wang, M. F. Ramoni and P. Sebastiani, Clustering short gene expression expression profiles, *Lect. Notes Comput. Sci.*, 2006, **3909**, 60–68.
14 M. Ashburner, *et al.*, Gene Ontology: Tool for the unification of biology The Gene Ontology Consortium, *Nat. Genet.*, 2000, **25**(1), 25–29.
15 M. Kanehisa and S. Goto, Kegg: Kyoto encyclopedia of genes and genomes, *Nucleic Acids Res.*, 2000, **28**(1), 27–30.
16 Z. Bar-Joseph, Analyzing time series gene expression data, *Bioinformatics*, 2004, **20**(16), 2493–2503.
17 M. B. Eisen, P. T. Spellman, P. O. Brown and D. Botstein, Cluster analysis and display of genome-wide expression patterns, *Proc. Natl. Acad. Sci. U. S. A.*, 1998, **95**, 14863–14868.
18 J. Handl, J. Knowles and D. B. Kell, Computational cluster validation in post-genomic data analysis, *Bioinformatics*, 2005, **21**(15), 3201–3212.
19 J. D. Storey, W. Xiao, J. T. Leek, R. G. Tompkins and R. W. Davis, Significance analysis of time course microarray experiments, *Proc. Natl. Acad. Sci. U. S. A.*, 2005, **102**, 12837–12842.
20 M. Mutarelli, L. Cicatiello, L. Ferraro, O. Grober, M. Ravo, A. Facchiano, C. Angelini and A. Weisz, Time-course analysis of genome-wide gene expression data from hormone-responsive human breast cancer cells, *BMC Bioinformatics*, 2008, **9**(Suppl 2), S12.
21 J. Sese, Y. Kurokawa, M. Monden, K. Kato and S. Morishita, Constrained clusters of gene expression profiles with pathological-features, *Bioinformatics*, 2004, **20**, 3137–3145.
22 J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann series in Machine Learning, Morgan Kaufmann, 1993.
23 R. S. Michalski and R. E. Stepp, Learning from observation: Conceptual clustering, in *Machine Learning: an Artificial Intelligence Approach*, ed. R. S. Michalski, J. G. Carbonell, T. M. Mitchell, Tioga Publishing Company, 1983, vol. 1, pp. 331–363.
24 L. Todorovski, B. Cestnik, M. Kline, N. Lavrač and S. Džeroski, Qualitative clustering of short time-series: A case study of firms reputation data, in *ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, 2002, pp. 141–149.
25 M. Garofalakis, D. Hyun, R. Rastogi and K. Shim, Building decision trees with constraints, *Data Min. Knowl. Discovery*, 2003, **7**(2), 187–214.
26 H. Sakoe and S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, *IEEE Trans. Acoust., Speech., Signal Process.*, 1978, **26**(1), 43–49.
27 J. Z. Wang, Z. Du, R. Payattakool, P. S. Yu and C. Chen, A new method to measure the semantic similarity of go terms, *Bioinformatics*, 2007, **23**(10), 1274–1281.
28 C. Ferri, P. A. Flach and J. Hernández-Orallo, Learning decision trees using the area under the ROC curve, in *19th Int'l Conf. on Machine Learning*, 2002, pp. 139–146.
29 L. Torgo, A comparative study of reliable error estimators for pruning regression trees, *Lect. Notes Comput. Sci.*, 1998, **1484**.
30 L. Wang, M. Montano, M. Rarick and P. Sebastiani, Conditional clustering of temporal expression profiles, *BMC Bioinformatics*, 2008, **9**, 147.

# Chapter 9

# Predictive Clustering of Multi-dimensional Time Series: Modeling Forest Growing Stock

In this chapter, we present the instantiation of predictive clustering trees (PCTs) for the task of modelling multi-dimensional time series. We use the distance definitions provided in Section 2.2.3 to instantiate the variance and prototype functions for constructing PCTs. We demonstrate the utility of the developed method on data concerning forest growing stock in Slovenia. The growing stock of forest stands is considered as the basic attribute for the description of spatio-temporal dynamics of the forest ecosystem response to natural and anthropogenic impacts and provides a way to follow the structural, functional and compositional changes of forest ecosystems.

The growing stock is typically modelled as an aggregated value from the volumes of individual trees and represents the accumulation of wood production of forest trees through the production period of forest stands. With the aggregation of individual tree volumes, a lot of information on its composition is lost. In order to avoid loss of information due to aggregation, we present the forest growing stock by its distribution into three DBH classes: A $10 - 29cm$, B $30 - 49cm$, and C $50cm$ and more. We consider Slovenian forest inventory data, following a number of forest compartments over a period of four decades: 1970 until 2010. The forest compartments are described with phytogeographical properties and the forest growing stock by tree-size (measured in terms of DBH – diameter at breast size) classes. For each forest compartment, we have information on the dynamics (across the four decades) of the forest growing stock across the three tree-size classes: This dynamics is described by three-dimensional time series. Such complex forestry data prompts for methods for modelling of multi-dimensional time series.

In this study, we have used two scenarios to analyze the data at hand: quantitative and qualitative. In the former scenario, we use the Euclidean distance as a distance between time-series, while in the latter scenario, we use a qualitative distance measure. The complementary nature of the two different distance measures in predictive clustering trees for modelling multi-dimensional time series allows for two orthogonal interpretations of the forest growing stock data. The results reveal that the growing stock in Slovenian forests has been increasing in the last 40 years and that it has progressive dynamics, which indicates that Slovenian forests have balanced structure.

**Paper**:

Gjorgjioski, V., Kocev, D., Bončina, A., Debeljak, M., & Džeroski, S. (2015). Predictive clustering of multi-dimensional time series for modelling forest growing stock. *Ecological Modelling*. Under review.

**Author's contribution**: Valentin Gjorgjioski designed and implemented the software for predictive clustering of multi-dimensional time series, performed the experiments on the forest data and wrote the larger part of this manuscript. The study was designed and supervised by Marko Debeljak and Sašo Džeroski.

# Predictive clustering of multi-dimensional time series applied to forest growing stock data for different tree sizes

Valentin Gjorgjioski[a,b], Dragi Kocev[a], Andrej Bončina[c], Sašo Džeroski[a,b], Marko Debeljak[a,b]

[a]*Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia*
[b]*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
[c]*Biotechnical faculty, University of Ljubljana, Slovenia*

## Abstract

In this paper, we propose a new algorithm for clustering multi-dimensional time series. It is based on the predictive clustering paradigm, which combines elements of predictive modelling and clustering. It builds upon the algorithm for predictive clustering trees for modelling time series, and extends it to model multi-dimensional time series. We also propose adequate distance functions for modelling multi-dimensional time series.

We apply the newly developed approach to the task of analyzing data on forest growing stock in state-owned forests in Slovenia. This task of high importance, since the growing stock of forest stands is a key feature describing the spatio-temporal dynamics of the forest ecosystem response to natural and anthropogenic impacts. It can be thus used to follow the structural, functional and compositional changes of forest ecosystems, which are of increasing importance as the forest area in Europe has been growing steadily for the last 20 years.

We consider Slovenian forest inventory data, following a number of forest compartments over four decades: 1970 until 2010. The compartments are described with phytogeographical properties and the growing stock is divided into tree-size classes (in terms of DBH – diameter at breast size). For each compartment, we have information on the dynamics (across the four decades) of the growing stock in the three tree-size classes: This dynamics is described by three-dimensional time series.

We have used two scenarios (quantitative and qualitative) to analyze the data at hand with predictive clustering trees for modelling multi-dimensional time series. In the former, we use a Euclidean distance between time series, and in the latter, we use a qualitative distance. The complementary nature of the distances allows for two orthogonal interpretations of the forest growing stock data. Overall, the growing stock in Slovenian forests has been increasing in the last

40 years. More specifically, the growing stock of the three tree-size/DBH classes has progressive dynamics, which indicates that Slovenian forests have balanced structure.

*Keywords:* time series modelling, multi-dimensional time series, forest growing stock, forest inventory, predictive clustering trees

## 1. Introduction

### *1.1. European forests and their growing stock*

Explanatory analysis of time series data is an important topic in ecological studies. It is used for understanding and predicting the temporal response of ecosystems to variations in ecological, environmental and management factors. The response of ecosystems can be structural, functional and compositional, where the former refers to the spatial arrangement of various components of an ecosystem (e.g., height of the vegetation, biomass, spatial distribution), the second refers to various ecological processes (e.g., production of organic matter, evapotranspiration) and the latter to the variety of ecosystem components (e.g., species richness and abundance) [1]. Structural, functional and compositional responses are interdependent, so changes of one ecosystem property trigger changes of other ecosystem properties. Scientific evidence shows different responses of ecosystems to global environment changes, such as pollution and climate change. Many ecosystems show signs of damage, such as bleaching of coral reefs and biodiversity loss of arable ecosystems [2], but the results of scientific research on the response of forests in the temperate and boreal climatic zones of Europe show progressive responses over the last 20 years.

The forest area in Europe (including the European part of the Russian Federation) has increased by 17 million *ha* and the growing stock by 8.600 million $m^3$ in the last 20 years [3]. The observed changes could be the result of long-term effects such as suitable biological and environmental conditions and short-term effects (a few decades) such as applied forest management practices [4, 5]. To follow the structural, functional and compositional changes of forest ecosystems, the growing stock of forest stands is considered as the basic attribute. It can be used to describe the spatio-temporal dynamics of the forest ecosystem response to natural and anthropogenic impacts, because it aggregates state and stress indicators (i.e., quality of the soil,

*Email address:* Saso.Dzeroski@ijs.si (Sašo Džeroski)

rainfall, temperature, growing stock increment, annual cut). Its systemic and integrative features can be used to assess the conditions of the forest, to diagnose the cause of structural or functional changes and to predict future trends of growing stock. Therefore, it is used as one of the basic statistics in forest inventories [6].

The dynamic changes of forest growing stock, as well as the predictions of their future development, at both European and national levels, are usually estimated from the data gathered by national forest inventories [7, 8, 9] or, more rarely, from archival forestry data [10, 11, 12, 5]. The European Forest Sector Outlook Study II [13] analyses the dynamic changes of forest resources at European and national levels using several different modelling approaches. In particular, it takes into account outputs from econometric projections for production and consumption of forest products [14, 15], analyses of wood resources balance [16], the results of the European Forest Information Scenario Model (EFISCEN) [17], and the results of the European Forest Institute-Global Forest Sector Model (EFI-GTM) [18]. However, the majority of European countries have also developed their own tools for analyzing dynamic forest changes at their national levels (e.g., [19]). Common approaches to estimating and predicting the future development of forest resources employ various methods, ranging from static inventory projections, to complex techniques of modelling. Such induced models usually project future forest development by using national forest inventory and yield data under some strong assumptions, e.g., that the uncertainty of future growth is small and insignificant, or that the forest management does not change in time [20]. In some countries, detailed inventory and yield data are not available for sufficiently long periods of time (i.e., a few decades), and therefore, such models cannot be constructed.

### *1.2. Modelling overall forest growing stock*

The most frequently used approach for investigating dynamic changes of forest ecosystem structure is based on the analysis, clustering and modelling of time series data, among which the growing stock data often is the most central. This modelling task is usually based on a mechanistic modelling approach, where the field data are used for calibration and validation of manually constructed models, whose structure is based on existing theoretical knowledge. Such an approach is informative, but includes many parameters, some of which are difficult to set or estimate. It often requires many different types of data that are not always possible to obtain (i.e., crown ratio, regeneration). Due to the large number of parameters that have to be fitted in mechanistic models, it is hard to achieve stability of their output accuracy, and their predictive power is lower [21].

Dynamic changes of forest variables (such as growing stock) are modeled using mechanistic modelling approaches that are based on: (i) patch (gap) models that operate at a different level of physiological detail of structural and compositional dynamics of forest ecosystems and simulate forest succession, and species distribution (e.g., PICUS [22]), under a wide range of environmental and management conditions (e.g., FORCLIM [23]); (ii) individual-based models that explore various management and environmental effects on the ecological, structural and spatial dynamics of forests (e.g., MOSES [24], CAPSIS [25], iLand [26]); (iii) spatially explicit forest landscape models that simulate spatial projections of forest derived ecosystem services, and evaluate how these services will be impacted under future climate disturbance and management scenarios (e.g., LAND-CLIM [27], LANDIS II [28, 29]) and iv) data mining models that are developed with the application of inductive machine learning techniques to available forest inventory data, in order to find explanations for temporal changes of growing stock [30].

Even though most attributes describing structural, functional and compositional dynamic changes of (forest) ecosystems are intertwined, all above mentioned modeling methods deal with modeling dynamic changes of only a single attribute at once. Furthermore, ecosystem properties are not in linear relations only but complex, nonlinear interactions between them are predominant. Therefore, methods describing the dynamics of a single property (e.g., total growing stock) provide a limited interpretation of ecosystem responses to changing environmental conditions. In order to overcome this obstacle, a new methodological approach to analyse, cluster and model time series is needed, which would allow for simultaneous modeling of time series for a larger number of ecosystems' structural, functional and compositional properties.

### *1.3. Modelling forest growing stock per tree class*

The growing stock is typically modelled as the aggregated value of the volumes of the individual trees and represents the accumulation of wood production of forest trees through the production period of forest stands. With the aggregation of individual tree volumes, a lot of information on its composition is lost. In order to avoid loss of information due to aggregation, growing stock could be described in detail with its distribution to growing stock classes or into groups of tree species. In such a way, enough information for better understanding of the time changes of growing stock are retained in the data.

Most often, the growing stock distribution into age classes is given. The age structure of forest stands is an important factor, because it could provide insights into harvesting potential, carbon stocks, biodiversity and performance of various ecosystem services [31]. However, age structure of forest stands can be estimated for even-aged forest stands only.

Because of various environmental and management reasons, larger forested areas consist from both even and uneven-aged forests, therefore age classes can't be used to extract additional information about time dynamics of growing stock at larger spatial scales (e.g., region, country). Furthermore, the distribution of even and uneven-aged forests at larger areas can't be estimated at a certainty level suitable for objective scientific studies, therefore the age structure of growing stock has a limited potential for analysis of its dynamics at larger areas. An additional problem of using age classes are the unpredictable changes of even-aged forests into uneven-aged forests due to management and environmental reasons.

To avoid the limitations of using age structure for larger and more diverse forest areas, the growing stock can be described in more detail by its distribution into DBH classes (e.g., A $10-29cm$, B $30-49cm$, and C $50cm$ and more [32], where DBH stands for diameter at breast height). DBH classes are a good substitute for age classes, which can be used for describing growing stock dynamics in even and uneven-aged forests. The task of modelling aggregated forest growing stock can thus be considered as modelling of single dimensional time series data [30], while modelling growing stock distributed to DBH classes can be considered as modelling multi-dimensional time series data, where the multiple dimensions represent the growing stock values of DBH growing classes A, B and C as defined above.

## *1.4. Motivation*

The developments described in this paper were motivated by the needs of data analysis of forest growing stock data aggregated per tree class. There are several methods that analyse and cluster time series data from the domain of environmental sciences [33, 34]. To model time series data, these approaches typically use hidden Markov models, neural networks, genetic programming, regression-based approaches (e.g., autoregressive integrated moving average) or analyse the data in the frequency domain.

Such methods are often used to forecast weather conditions (e.g., rainfall), predict river water levels (or flood protection), analyse temporal remotely sensed data about land use and land cover [35, 36, 37, 38], etc. For example, Li et al. [39] perform clustering on time series data using hidden Markov models. The data they analyze concern the ecological effects of mosquito control produced by changing the drainage patterns in an area south of Brisbane, Australia.

The aforementioned approaches have a number of limitations: They limit the type of variables that can be used, they make assumptions about prior distributions or missing values, and they offer limited interpretability of the learned models. To overcome these limitations, we propose to use predictive clustering trees [40] that do not make such prior assumptions and are readily interpretable. In the past [41], we have used predictive clustering trees to model time series data on interdependent types of agroecosystem vegetation that support different functions in a managed ecosystem: For optimal management, plants that provide economic output (e.g., crops) and those that support ecological functions (e.g., wild plants or 'weeds' should coexist in an agroecosystem.

### *1.5. Outline of the paper*

The task of modelling forest growing stock data aggregated per tree class is a task of multi-dimensional time series modelling. To address this task, we extend the approach of learning predictive clustering trees (PCTs) to handle multi-dimensional time series. Predictive clustering produces both clusters (of multi-dimensional time series) and explanations of the clusters simultaneously: To the best of our knowledge, there is no other method that handles these two aspects simultaneously. The cluster explanations in the form of predictive clustering trees are readily interpretable, which is a very important factor: Interpretable models can be easily used by both domain experts and decision makers.

We apply the proposed method of PCTs for multi-dimensional time series to Slovenian forest inventory data, in order to find explanations for structural changes in Slovenian forests over the period from 1970 to 2010. This is a new methodological approach for explaining time dynamics of growing stock at the level of DBH classes. This is much more informative then modelling time dynamics of overall growing stock (at an aggregated level), and as such complements the classic mechanistic modelling methodologies and single dimensional modelling of time series data.

The reminder of this paper is organized as follows. The basic approaches and distances for clustering time series that are related to the method we propose are described in Section 2. Section 3 presents the approach of predictive clustering trees that we extend to cluster multi-dimensional time series. The task of clustering forest compartments based on the corresponding dynamics of forest growing stock is outlined in Section 4. Section 5 presents and discusses the results obtained by applying our method to the data at hand. Finally, Section 6 concludes and outlines directions for further work.

## 2. Background

### *2.1. Clustering in a nutshell*

Clustering in general is concerned with grouping objects into classes of similar objects [42]. Given a set of examples (object descriptions), the task of clustering is to partition these examples into subsets, called clusters. Clustering is known as cluster analysis in statistics and as unsupervised learning in machine learning, where supervised learning denotes the task of learning to predict a target property of an object. In clustering, the objects do not contain a target property to be predicted, but only an object description.

Conventional clustering focuses on distance-based cluster analysis. The notion of a distance (or conversely, similarity) is crucial here: examples are considered to be points in a metric space (a space with a distance metric). The goal of clustering is to achieve high similarity between objects within individual clusters (intra-cluster similarity) and low similarity between objects that belong to different clusters (inter-cluster similarity).

To represent a cluster, a prototype is typically used, such as the mean/ centroid or the medoid of the objects in the cluster. Compact clusters are desirable, where a compact cluster has a low (intra-cluster) variance. The (per-object) variance within a given cluster is defined as

$$Var(C) = \frac{1}{|C|} \sum_{X \in C} d^2(X, c) \,, \tag{1}$$

where $c$ is the centroid of $C$. Methods like k-means clustering or hierarchical agglomerative clustering can be used to find sets of clusters with low intra-cluster variance and low inter-cluster similarity.

In conceptual clustering [43], a symbolic representation of the resulting clusters is produced in addition to the partition into clusters. We can thus consider each cluster to be a concept (much like a class in classification). In rule-based clustering, the description of a concept is a conjunction of logical conditions, called a rule. Flat clusterings (with possibly overlapping clusters) can be described as sets of rules, while hierarchical clusterings (with non-overlapping clusters) can be represented as tree-based structures.

## 2.2. Clustering time series

Distance-based clustering methods rely on the definition of an appropriate distance measure between two examples. When the examples are time series, we need distance measures on time series. Typically, all time series in a data set have the same length. This holds true, for example, for the data that we consider in this paper.

If all time series have the same length, then one can represent them as real valued vectors and use standard distance measures such as the Euclidean distance or the Manhattan distance. These measures are, however, not always appropriate for time series because (1) they assume that the time series are synchronized, and (2) mainly capture the difference in scale and baseline. Below, we discuss three distance measures that have been proposed to alleviate these shortcomings: the correlation coefficient, dynamic time warping [44] and a qualitative distance [45].

### 2.2.1. Correlation coefficient

The correlation coefficient $r(X, Y)$ between two time series $X$ and $Y$ is calculated as

$$r(X, Y) = \frac{E[(X - E[X]) \cdot (Y - E[Y])]}{E[(X - E[X])^2] \cdot E[(Y - E[Y])^2]}, \tag{2}$$

where $E[V]$ denotes the expectation (i.e., the mean value) of $V$. $r(X, Y)$ measures the degree of linear dependence between $X$ and $Y$. It has the following intuitive meaning in terms of the shapes of $X$ and $Y$: $r$ close to 1 means that the shapes are similar. If there is a linear relation between $X$ and $Y$ then the time series are identical, but might have a different scale or baseline. $r$ close to -1 means that $X$ and $Y$ have "mirrored" shapes, and $r$ close to 0 means that the shapes are unrelated (and consequently dissimilar).

Based on the above intuitive interpretation, we can define the distance between two time series as $d_r(X, Y) = \sqrt{0.5 \cdot (1 - r(X, Y))}$. $d_r$ has, however, two drawbacks. First, it is difficult to properly estimate correlation if the number of observations is small (i.e., for short time series). Second, $d_r$ can only capture the linear dependencies between the time series and time series that are non-linearly related will appear to be distant.

### 2.2.2. Dynamic time warping

Dynamic time warping (DTW) [44] can capture non-linear distortion along the time axis. It accomplishes this by assigning multiple values of one of the time series to a single value of the other. As a result, DTW is suitable to use if the time series are not properly synchronized, e.g., if one is delayed, or if the two time series are not of the same length.

$d_{\text{DTW}}(X, Y)$ with $X = \alpha_1, \alpha_2, \ldots, \alpha_I$, $Y = \beta_1, \beta_2, \ldots, \beta_J$ is defined based on the notion of a warping path between $X$ and $Y$. A warping path is a sequence of grid points $F = f_1, f_2, \ldots, f_K$ on the $I \times J$ plane. Let the distance between two values $\alpha_{i_k}$ and $\beta_{j_k}$ be $d(f_k) = |\alpha_{i_k} - \beta_{j_k}|$, then an evaluation function $\Delta(F)$ is given by $\Delta(F) = 1/(I + J) \sum_{k=1}^{K} d(f_k) w_k$. The weights $w_k$ are as follows: $w_k = (i_k - i_{k-1}) + (j_k - j_{k-1}), i_0 = j_0 = 0$. The smaller the value of $\Delta(F)$, the more similar $X$ and $Y$ are. In order to prevent excessive distortion, we assume an adjustment window ($|i_k - j_k| \leq r$). $d_{\text{DTW}}(X, Y)$ is the minimum of $\Delta(F)$.

Both the Euclidean distance and DTW take into account differences in scale and baseline. If a given time series is identical to a second time series, but scaled by a certain factor or offset by some constant, then the two time series will be distant. For many applications, these differences are, however, not important and only the shape of the time series matters. The qualitative distance is more appropriate for such applications.

### *2.2.3. Qualitative Distance*

The qualitative distance (QD) [45] is based on a qualitative comparison of the shape of the time series. Consider two time series $X$ and $Y$. Then choose a pair of time points $i$ and $j$ and observe the qualitative change of the value of $X$ and $Y$ at these points. There are three possibilities: increase ($X_i > X_j$), no-change ($X_i \approx X_j$), and decrease ($X_i < X_j$). $d_{\text{qual}}$ is obtained by summing the difference in qualitative change observed for $X$ and $Y$ for all pairs of time points, i.e.,

$$d_{\text{qual}}(X, Y) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2 \cdot \mathit{Diff}(q(X_i, X_j), q(Y_i, Y_j))}{N \cdot (N-1)} \,, \tag{3}$$

where the $\mathit{Diff}(q_1, q_2)$ function (Table 1) defines the difference between pairs of qualitative changes. Roughly speaking, $d_{\text{qual}}$ counts the number of disagreements in change of $X$ and $Y$.

$d_{\text{qual}}$ does not have the drawbacks of the correlation based measure. First, it can be computed for very short time series, without decreasing the quality of the estimate. Second, it captures the similarity in shape of the time series, regardless of whether their dependence is linear or non-linear.

Table 1: The definition of $\mathit{Diff}(q_1, q_2)$.

| $\mathit{Diff}(q_1, q_2)$ | increase | no-change | decrease |
|---|---|---|---|
| increase | 0 | 0.5 | 1 |
| no-change | 0.5 | 0 | 0.5 |
| decrease | 1 | 0.5 | 0 |

## *2.3. Predictive clustering*

Predictive modelling aims at constructing models that can predict a target property of an object from a description of the object. Predictive models are learned from sets of examples, where each example has the form $(D, T)$, with $D$ being an object description and $T$ a target property value. While a variety of representations ranging from propositional to first order logic have been used for $D$, $T$ is almost always considered to consist of a single target attribute called the class, which is either discrete (for classification problems) or continuous (for regression problems).

Clustering [46], on the other hand, is concerned with grouping objects into subsets of objects (called clusters) that are similar w.r.t. their description $D$. There is no target property defined in clustering tasks. In conventional clustering, the notion of a distance (or conversely, similarity) is crucial: examples are considered to be points in a metric space and clusters are constructed such that examples in the same cluster are close according to a particular distance metric. A centroid (or prototypical example) may be used as a representative for a cluster. The centroid is the point with the lowest average (squared) distance to all the examples in the cluster, i.e., the mean or medoid of the examples. Hierarchical clustering and $k$-means clustering are the most commonly used algorithms for this type of clustering.

Predictive clustering [47] combines elements of both predictive modelling and clustering. As in clustering, we seek clusters of examples that are similar to each other, but in general taking both the descriptive part and the target property into account (the distance measure can be defined on $D \cup T$ or any subset thereof). In addition, a predictive model must be associated to each cluster. The predictive model assigns new instances to clusters based on their description $D$ and provides a prediction for the target property $T$. A well-known type of model that can be used to this end is a decision tree [48]. A decision tree that is used for predictive clustering is called a predictive clustering tree (PCT, Figures 4 and 6). Each node of a PCT represents a cluster. The conjunction of conditions on the path from the root to that node gives a description of the cluster. Essentially, each cluster has a symbolic description in the form of a rule (IF conjunction of conditions THEN cluster), as in conceptual clustering [49], while the tree structure represents the hierarchy of clusters. Clusters that are not on the same branch of a tree do not overlap.

In Figures 4 and 6, the description $D$ of a forest compartment consists of phytogeographic properties, and the target property $T$ is the three dimensional time series of forest growing stock DBH classes. In general, we could include both $D$ and $T$ in the distance measure. We are, however, most interested in the time series part $T$. Therefore, we define the distance measure only on $T$.

We consider various distance measures in Section 3. The resulting PCTs (Figures 4 and 6) represent a clustering that is homogeneous with respect to $T$ and the nodes of the tree provide symbolic descriptions of the clusters. Note that a PCT can also be used for prediction: we use the tree to assign a new instance to a leaf and take the centroid of the corresponding leaf cluster as prediction.

### *2.4. Predictive clustering of time series*

Džeroski et al. [50] have proposed to use predictive clustering trees (PCTs) for clustering time series data. The main advantage of using PCTs over other clustering algorithms, such as hierarchical agglomerative clustering and $k$-means, is that PCTs cluster the time series and provide a description of the clusters at the same time. This allows one to relate various heterogeneous data types and to draw conclusions about their relations.

Table 2 presents the generic induction algorithm for PCTs [47]. It is a variant of the standard greedy recursive top-down decision tree induction algorithm used, e.g., in C4.5 [48]. It takes as input a set of instances $I$ (in our case forest compartments described by phytogeographic properties and their associated overall forest growing stock time series). The procedure *BestTest* (Table 2, right) searches for the best acceptable test (on a phtyogeographic property) that can be put in a node. If such a test $t^*$ can be found, then the algorithm creates a new internal node labeled $t^*$ and calls itself recursively to construct a subtree for each cluster in the partition $\mathcal{P}^*$ induced by $t^*$ on the instances. If no acceptable test can be found, then the algorithm creates a leaf, and the recursion terminates. (The procedure *Acceptable* defines the stopping criterion of the algorithm, e.g., specifying maximum tree depth or a minimum number of instances in each leaf).

Table 2: The generic PCT induction algorithm Clus.

```
procedure PCT($I$) returns tree
1: $(t^*, h^*, \mathcal{P}^*)$ = BestTest($I$)
2: if $t^* \neq none$ then
3:     for each $I_k \in \mathcal{P}^*$ do
4:         $tree_k$ = PCT($I_k$)
5:     return node($t^*$, $\bigcup_k \{tree_k\}$)
6: else
7:     return leaf(centroid($I$))
```

```
procedure BestTest($I$)
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each possible test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $I$
4:     $h = Var(I) - \sum_{I_k \in \mathcal{P}} \frac{|I_k|}{|I|} Var(I_k)$
5:     if $(h > h^*) \wedge$ Acceptable($t, \mathcal{P}$) then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

Up to this point, the algorithm is identical to a standard decision tree learner. The main difference is in the heuristic that is used for selecting the tests and the prototype function. For PCTs, this heuristic is the reduction in variance (weighted by cluster size, see line 4 of *BestTest*). Maximizing variance reduction maximizes cluster homogeneity. The variance and prototype function for performing the clustering of the instances need to be instantiated depending on the prediction task at hand. So far, the predictive clustering framework has been used for the prediction of multiple continuous variables, prediction of multiple discrete variables, hierarchical multi-label classification (HMC) and prediction of time series [51]. An implementation of the PCT induction algorithm is available in the Clus system, which can be obtained at `http://sourceforge.net/projects/clus`.

In this work, we focus on learning PCTs for the prediction and clustering of time series. Learning PCTs for time series data is non-trivial because for many distance measures (the correlation-based, dynamic time warping, and qualitative distances), no closed algebraic form for the centroid is known. Therefore, Džeroski et al. [50] propose to compute cluster variance based on the sum of squared pairwise distances (SSPD) between the cluster elements.

The per-example variance of a cluster $C$ can be defined based on a distance measure as

$$Var(C) = \frac{1}{|C|} \sum_{X \in C} d^2(X, c)\,, \tag{4}$$

with $c$ the cluster centroid of $C$. To cluster time series, $d$ should be a distance measure defined on time series, such as the ones discussed in the previous section. The centroid $c$ of a cluster can be computed as follows

$$c = \text{argmin}_q \sum_{X \in C} d^2(X, q) \tag{5}$$

We consider two possible representations for $c$: (a) the centroid is an arbitrary time series, and (b) the centroid is one of the time series from the cluster (the cluster prototype). In representation (b), the centroid can be computed with $|C|^2$ distance computations by substituting $q$ with each time series in the cluster. In representation (a), the space of candidate centroids is infinite. This means that either a closed algebraic form for the centroid is required or that one should resort to approximative algorithms to compute the centroid. No closed form for the centroid is known in representation (a) for the distance measure $d_{\text{qual}}$. To the best of our knowledge, the same holds for $d_{\text{DTW}}$ and $d_r$.

An alternative way to define cluster variance is based on the sum of the squared pairwise distances (SSPD) between the cluster elements, i.e.,

$$Var(C) = \frac{1}{2|C|^2} \sum_{X \in C} \sum_{Y \in C} d^2(X, Y)\,. \tag{6}$$

The factor 2 in the denominator of Equation 6 ensures that it is identical to Equation 4 for the Euclidean distance. The advantage of this approach is that no centroid needs to be computed to calculate the variance. This approach also requires $|C|^2$ distance computations to calculate the variance, just as the approach with the centroid in representation (b). Hence, using the definition based on a centroid is only more efficient if the centroid can be computed in time linear in the cluster size. This is the case for the Euclidean distance in combination with using the time-point-wise average of the time series as centroid. For the other distance measures, no such centroids are known. Therefore, we choose to estimate cluster variance using the SSPD. The PCT induction algorithm places cluster centroids in its leaves, which can be inspected by the domain expert and used both as predictions and as cluster prototypes. For these centroids, we use representation (b) as discussed above.

### *2.5. Clustering multi-dimensional time series*

Clustering of one-dimensional time series using hierarchical agglomerative clustering and $k$-means clustering is well known and widely used. In addition, there are extensions that address the problem of clustering multi-dimensional or multivariate time series [52], [53], [54]. All of these approaches use the $k$-means or hierarchical agglomerative algorithms for clustering the time series, and then *post-facto* search for descriptions of each cluster. By using PCTs, we perform constrained clustering, which in a single step gives both the clusters and their descriptions.

## 3. Predictive clustering trees for multi-dimensional time series

In the previous section, we presented the algorithm for learning PCTs for modelling (clustering or predicting) time series. More specifically, we described the variance and prototype functions used within the PCT learning algorithm when the target property of the examples consists of a single time series. We now focus on the task where the target is a multi-dimensional time series. We begin by defining distances on structured data types. Next, we present the distance on multi-dimensional time series. We then outline the PCT algorithm for clustering multi-dimensional time series. Finally, we discuss the selection of proper distance function(s) for time series.

### *3.1. Distances on structured data types*

We distinguish between two kinds of data-types: *primitive* and *structured* data types. Primitive data types have no structure and do not contain another data type within. Examples of primitive data types include *nominal*, *ordinal* (i.e., ordered nominal) and *real*. Structured data types, on the other hand, are built from primitive data types. To build structured data types, we first need to define type constructors. Here, we consider the following type constructors: *tuple* ($Tuple\,(T_1, T_2, ...T_n)$), *set* ($Set\,\{T\}$), and *sequence* ($Sequence\,[T]$). *Tuple* is a type constructor that can contain any fixed number of objects, each from an arbitrary, but fixed data type (the objects at different positions in the tuple may be of different data types). *Set* is a type constructor that can contain any number (not fixed) of objects, all of the same (fixed) type. The type of objects in the set can be either primitive or structured. *Sequence* is a type constructor that is similar to the *Set* type constructor as it can contain an arbitrary number of objects of the same type. However, while the elements of a set are unordered, in a sequence the ordering of the elements is important. The *TimeSeries* is a data type constructed by using the sequence constructor where the underlying type is real and the ordering of the elements is along the time dimension ($TimeSeries = Sequence[real]$).

Using the notions on primitive and structured datatypes, we can define multi-layered datatypes, i.e., structured datatypes than can consist of other structured datatypes.

For example, we can define multi-layered objects (or datatypes) as complex as $Tuple\left[TimeSeries, Sequence\left[Nominal\right], Set\left\{TimeSeries\right\}\right]$. This object is then a tuple consisting of three elements: $TimeSeries$, $Sequence$ and $Set$. Furthermore, the sequence is defined as a sequence of *nominal* and the set is defined as a set of *TimeSeries*.

Defining distances has been studied extensively and is reasonably well understood for primitive data types [55]. The basic idea of a unified approach to mining structured data, such as predictive clustering, is to derive the key components of data mining algorithms for a complex data type (built through using type constructors) from information on the structure of that type (what constructors on what simpler data types) and the key components for the simpler data types. For example, a distance function $d$ on tuples of type $Tuple(T_1, \ldots, T_n)$ can be composed from distance functions $d_i$ on types $T_i$ by adding up the distances for each tuple component, i.e., $d(x, y) = d((x_1, ..., x_n), (y_1, ..., y_n)) = \sum\limits_{i=1}^{n} d_i(x_i, y_i)$.

Distances over structured data types can be defined as follows. For each structured data type, we define a distance as an aggregated value of distances of pairs of underlying component objects. For example, if the structured data type is a tuple of time series, an aggregation function will be used to combine the distances over the pairs of elements of the tuples (i.e., pairs of time series). In this manner, we can define the calculation of the distance recursively depending on the structure of the objects.

The pseudo-code given in Table 3 illustrates the recursive calculation of the distance. The *calculateDistance* function takes as input two data objects ($O_1$ and $O_2$), the definition of the datatype of the data objects ($DT$), and the distance definition ($DI$). The distance definition in the case of primitive datatypes is the *distanceID*, e.g., $\delta$ for the nominal datatype, while for structured datatypes it contains the definitions of the distances for the underlying datatypes and the definition of the aggregation function ($DI = (Agg, DI_s)$). The distance function calculation starts by checking whether the input objects are from a primitive datatype. If that holds, then the distance can be calculated instantly using the specified formula. For example, if the objects are from the datatype numeric, then the distance between them can be calculated by using one of the distances for numeric datatypes (e.g., Euclidean, Manhattan, etc). If the objects are from a structured or multi-layered datatype, then the objects are decomposed (using the $Decompose$ function) into components using the datatype definition $DT$. After the decomposition of the two objects, the matching procedure matches the corresponding components from the objects. In the next step, for each pair of components (one component from the first object and the other from the second object) the *calculateDistance* function is called recursively. Based on the different data object components, the calls to the function *calculateDistance* are instantiated differently, using the components of the datatype $DT$ and distance $DI$. Finally, the distances of the underlying components of the data objects are aggregated bottom-up using the definitions provided with $Agg$.

Table 3: The generic recursive algorithm for calculating the distance between two structured objects

```
function calculateDistance ($O_1, O_2, DT, DI$)
Input: $O_1$, $O_2$ - data objects; $DT$ - datatype definition; $DI$ - distance definition
Output: $d$ - distance value
 1: if $DT$ is a primitive datatype then
 2:     return calculateDistPrimitive($O_1$, $O_2$, $DT$, $DI$)
 3: $DT_s$ = Decompose(DT), $DI = (Agg, DI_s)$
 4: $C_1$ = Decompose($O_1, DT$)
 5: $C_2$ = Decompose($O_2, DT$)
 6: $M$ = Matching($C_1$, $C_2$, $DI_s$),
    $M = \{(c_1, c_2, d_t, d_i) | c_1 \in C_1, c_2 \in C_2, d_t \in DT_s, d_i \in DI_s\}$
 7: for each $(c_1, c_2, d_t, d_i) \in M$ do
 8:     $d_k = calculateDistance(c_1, c_2, d_t, d_i)$
 9: $d = Agg(d_1, d_2, ..., d_{|M|})$
10: return $d$
```

### *3.2. Distances on multi-dimensional time series*

Having defined distances on structured objects in general, we will continue to explain how we can define distances between multi-dimensional time series. First, we define the structured datatype as $Tuple\,(T_1, T_2, ...T_n)$, where $T_1, T_2, ...T_n$ are time series. Then, we define the distances that will be used for the components. For example, for the $Tuple$ we can use aggregation employed with the Euclidean distance ($SQRT(SumSQDist)$), and for the component $TimeSeries$ we will use *Euclidean* and *QD* distance measures. We would like to note that our framework allows the use of a different distance for each component time series in the tuple. We will however use the same distance for all of the time series in the tuple.

Distances on structured objects define both the matching of the component objects and the aggregation function. For example, the matching function for the Euclidean distance on time series defines that the respective elements of the time series are matched. The aggregation function for the Euclidean distance is defined as the squared root of the sum of squared distances between the matched pairs of elements. For the QD, the matching function and the aggregation function are explained in detail in Section 2.2.3.

Using the pseudo-code from Table 3, we illustrate the calculation of distances over tuples of three time series (i.e., three-dimensional time series). We illustrate this on two examples. We first outline the calculation of the Euclidean distance for the multi-dimensional time series datatype. We then illustrate the calculation of the distance between multi-dimensional time series that uses the qualitative distance for the individual time series.

For both cases, the underlying datatype is $DT = Tuple(TimeSeries, TimeSeries, TimeSeries)$ and the distance between two objects (three dimensional tuple of time series) is calculated by calling $d = calculateDistance(O_1, O_2, DT, DI)$. For the first case, the distance is defined as follows: $DI = (SQRT(SumSQDist), [EUC, EUC, EUC])$. For the second case, the distance is defined as $DI = (SQRT(SumSQDist), [QD, QD, QD])$. A step-by-step explanation of the distance calculation is given bellow.

We consider two specific data examples with the datatype definition as given above:

$$O_1 = ([73, 66, 54, 41, 54], [225, 231, 193, 159, 147], [65, 122, 120, 185, 223]), \text{and}$$
$$O_2 = ([76, 97, 81, 73, 66], [165, 208, 143, 158, 162], [53, 102, 87, 117, 181])$$

In the first step, the algorithm will decompose the structured tuples to their underlying components: time series. The algorithm will then match the corresponding time series from the two tuples: the first time series from $O_1$ to the first time series from $O_2$, etc. Next, it continues by calculating the distance between the corresponding time series as follows:

$$d_1 = QD([73, 66, 54, 41, 54], [76, 97, 81, 73, 66]) = 0.35,$$
$$d_2 = QD([225, 231, 193, 159, 147], [65, 122, 120, 185, 223]) = 0.85,$$
$$d_3 = QD([65, 122, 120, 185, 223], [53, 102, 87, 117, 181]) = 0.05.$$

In the final step, the algorithm reads the aggregation function from the distance definition ($DI$): in this case, it is the sqare root of the sum of the squared distances ($SQRT(SumSQDist)$). The aggregation function is then called using the distances on the underlying time series applying the formula from the distance definition as follows:

$$d = \sqrt{d_1^2 + d_2^2 + d_3^2} = \sqrt{0.35^2 + 0.85^2 + 0.05^2} = 0.9206$$

At the end, the *calculateDistance* will return the value 0.9206 as the distance between the two data examples $O_1$ and $O_2$ given above.

## *3.3. Predictive clustering trees for multi-dimensional time series*

The algorithm for the induction of PCTs for modelling multi-dimensional time series is essentially the same as presented in Section 2.4, Table 2. The major difference is in the calculation of the variance and prototype for the task of modelling multi-dimensional time series. Both are based on the notion of a distance between multi-dimensional time series, explained above.

The variance for the PCTs is defined as the sum of squared pairwise distances (SSPD) between the examples. In the PCT algorithm for clustering multi-dimensional time series, we use the distances outlined in the previous subsections, especially those presented in Table 3. More specifically, the distance on multi-dimensional time series is used to calculate the variance (Equation 6) for the split selection in the PCT induction algorithm (line 4 in Table 2, right). In line with using the SSPD to calculate the distance, we select the centroid as one of the multi-dimensional time series from the cluster that has the smallest sum of squared distances to the other elements of the cluster.

## *3.4. Selecting appropriate distances on time series*

As mentioned above, different distances on time series can be used for clustering multi-dimensional time series (i.e., tuples of time series). We can either use one distance measure for all time series/ dimensions, or even different distance measure for different dimensions. When choosing which distance to use, we should take into account the application at hand and the properties of the distances, which we discuss below.

The Euclidean distance, when applied to time series, ignores the temporal component. For two time series to be similar according to this distance, they need to have similar magnitudes at each time point. For highly similar series, this also implies similar shapes, i.e., temporal trends.

The DTW distance allows for time to be stretched to achieve better matching between the two time series. This means that a time series that is delayed with respect to another one or otherwise temporally stretched (but maintains the approximate magnitude) will still be considered similar to the original series. With no delays, the DTW distance behaves very much like the Euclidean distance. The DTW distance can be used for time series of different length (unlike most of the other distances).

The correlation measure is quite often used for time series due to its ability to capture similarity in trends rather than absolute values (magnitudes). In fact, it captures linear dependencies between the time series and is invariant to scaling and translation of the measured/observed values. In this sense, it is quite different from the Euclidean distance.

Finally, the qualitative distance (QD) primarily focuses on the shape of the time series, i.e., the qualitative changes (increase/decrease/steady) between pairs of time points. It can capture non-linear dependencies, as well as linear ones. However, it ignores almost completely the magnitudes of the observed values of the time series.

Based on the above properties of the distances on time series, appropriate distances can be selected for each application domain at hand. For example, if we consider time series to be similar despite slight temporal delays or distortions, we should use the DTW distance. If we do not wish to allow for delays, and temporal (linear) trends are important rather than closeness of magnitudes, correlation would be appropriate. Finally, when we want to capture non-linear dependencies between time series and not focus on magnitudes, the QD would be appropriate.

In this work, we will consider both the magnitude and the the dynamics of the forest stock. We argue that for the dynamics, the QD would be the more appropriate distance, and for the magnitude the Euclidean distance. The QD measures how far apart are two time series by taking in account only their dynamics, regardless of the magnitudes, while the Euclidean distance measures only how far apart the magnitudes are. To illustrate this, Figure 1 shows three time series taken from the dataset. The series $TS_1$ and $TS_2$ have the same dynamics and are very similar according to the QD, but are very dissimilar according to the Euclidean distance. On the other hand, the series $TS_1$ and $TS_3$ are of roughly the same magnitude and very similar according to the Euclidean distance, but are very different in their dynamics and very dissimilar according to the QD.

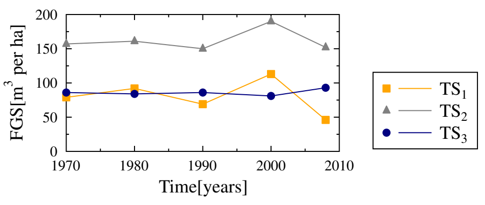


Figure 1: Three time series of forest growing stock, illustrating the differences between the Euclidean distance and the qualitative distance. The Euclidean distance between $TS_1$ and $TS_3$ is small, and between $TS_1$ and $TS_2$ is large. It is the other way around for the qualitative distance, according to which $TS_1$ and $TS_2$ are similar, while $TS_1$ and $TS_3$ are much less similar.

## 4. Clustering forest compartments according to forest growing stock dynamics

### *4.1. Data description*

The data used here come from the spatial information database Silva-SI [4, 56]. The data have been gathered from various sources, mostly forest inventories within Forest Management Plans (FMP), which are typically performed every 10 years (each year approximately 1/10 of FMP are revised). The database currently comprises digitalised data for 21052 forest compartments covering $7452 km^2$ or 64% of the Slovenian forests for the period from the year 2010 back to 1970 (Figure 2). Digitalisation of older data is under-way, but is not yet completed and therefore data from before the year 1970 were not included in this study. In our study, we use only compartments that are 100% owned by the state, which amounts to 5237 compartments.

Forest compartments are permanent and their size and borders have not changed since the first forest inventories. All compartments are described with the same suite of 47 environmental and stand variables, which indicate the state of forest stands. The variables describing the geographic characteristics of compartments were acquired from a digital elevation model with a spatial resolution of $25m \times 25m$. From these variables, based on a previous study [30], we selected a subset of the 6 most important variables: mean inclination (INC), mean elevation (ELV), the type of bedrock (BEDR), prevailing aspect of the compartment (ASP), phyto-geographical region (PHYTOREG) and site productivity (PI).

The last two variables were obtained by classifying each of the compartments into one of the six phytogeographical regions (PHYTOREG) to which Slovenian forests belong: Alpine, Pre-Alpine, Dinaric, Pre-Dinaric, Sub-Panonian and Sub-Mediterranean. Forests within the specific phytogeographical region have uniform climatic conditions and distinctly recognizable type and

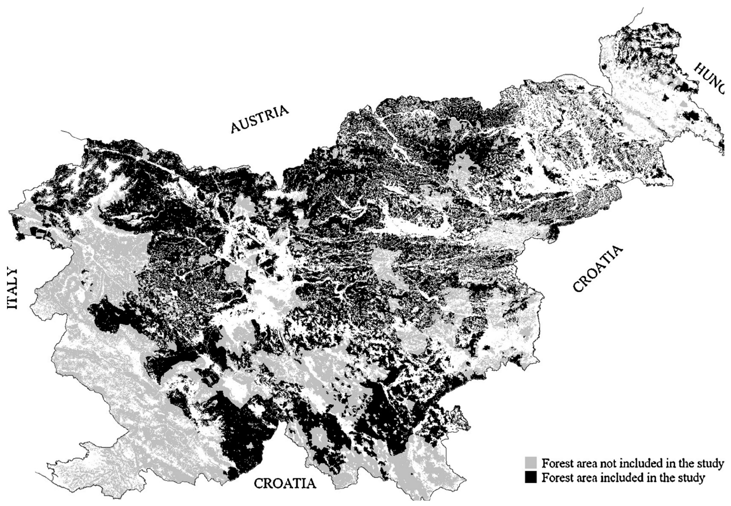

AUSTRIA
HUNG
ITALY
CROATIA
CROATIA
Forest area not included in the study
Forest area included in the study

Figure 2: The location of the forest areas (compartments) included in the study.

structure of vegetation. Next, site productivity (PI) within the compartments was ranked on the scale from 1 to 17 [57], where the most productive compartment gets rank 17 and the least productive one gets rank 1.

The stand variables indicating the state of forest stands within the compartments, such as growing stock (GS) and growing stock DBH classes (A, B, C), were estimated at 10 years' time steps with the official methodology of forest inventories (as approved and applied by the Slovenian Forestry Service). Forest inventories contain a combination of field descriptions of all stands and tree measurements. We distinguish between growing stock (GS), which aggregates individual tree volumes for all trees, and growing stock of DBH classes, which aggregates volumes of the trees with DBH from 10 to $29cm$ into DBH class A, with DBH from 30 to $49cm$ into DBH class B and DBH of $50cm$ and more into DBH class C [32].

The dynamics of growing stock is described by the time course profiles of growing stock values, recorded each 10 years, from 1970 to 2010. At the same 5 time points, we also have the growing stock values per DBH class (A,B,C), representing the growing stock dynamics in more detail. This more detailed view constitutes a multi-dimensional (three-dimensional) time series.

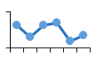
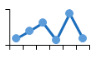
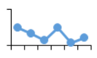
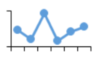
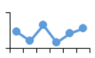
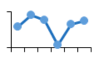
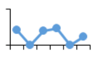
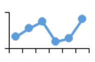
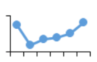
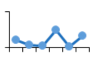
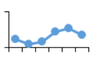
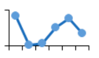
|  | ELV | INC | ASP | BEDR | PI | PHYTOREG | DBH A | DBH B | DBH C |
|---|---|---|---|---|---|---|---|---|---|
| S1 | 1300 | 20 | NW | C | 7 | Alpine |  |  |  |
| S2 | 950 | 19 | NE | NC | 11 | Pre-Dinaric |  |  |  |
| S3 | 1000 | 15 | W | NC | 11 | Dinaric |  |  |  |
| S4 | 700 | 6 | flat | C | 13 | Sub-Panonian |  |  |  |
| ... | ... | | | | | | ... | | |

Figure 3: An excerpt from the dataset assembled for the task of predictive clustering of the dynamics of growing stock for DBH classes A, B and C. The explanatory/independent variables are ELV, INC, ASP, BEDR, PI and PHYTOREG. The target/clustering time series are DBH A, DBH B and DBH C.

### *4.2. Data analysis task and study design*

The data analysis task that we address is to group/cluster forest compartments with similar growing stock dynamics as described by the growing stock DBH classes. In addition, we want to describe the groups/clusters of forest compartments in terms of their geo-physical properties. The goal is thus to relate the dynamics of growing stock for DBH classes A, B and C and the variables describing forest compartments (INC, ELV, BEDR, ASP, PHYTOREG, PI). An excerpt from the dataset assembled for this task is illustrated in Figure 3.

The task at hand is a task of predictive clustering of multi-dimensional time series. The target time series according to which we cluster/group forest compartments are the three time series describing the dynamics of the growing stock for DBH classes A, B and C. The explanatory variables in terms of which we describe the clusters of compartments are INC, ELV, BEDR, ASP, PHYTOREG and PI.

We use two scenarios for data analysis. In the first scenario, we use the Euclidean distance as a distance between the individual time series, since it is more appropriate when analyzing their quantitative aspect (as discussed in Section 3.4). In the second scenario, we use the qualitative distance, since the aim here is to capture the qualitative aspect of the time series, i.e., we are more focused on finding whether the forest growing stock increases or decreases through the years and less focused on how much the forest growing stock has changed. The used scenarios are complementary to each other and give us the opportunity to have different and complementary interpretations of the structural and temporal changes of the forest growing stock.

In both scenarios, we use PCTs for predicting/clustering multi-dimensional time series. After building a tree (and a PCT), it is typical to prune it, in order to deal with noise and other types of imperfection in the data. We employ two pruning algorithms: *F-test* pruning and *MaxSize* pruning. The *F-test* pruning algorithm checks whether the addition of a split at a given leaf of the tree significantly reduces the intra-cluster variance for the examples in that leaf: The significance level is specified by the user. The *MaxSize* pruning algorithm [58] takes as input a given maximum tree size $k$ and computes a subtree of the given tree of size at most $k$ with minimum error. These pruning algorithms increase the interpretability of PCTs, while maintaining (or increasing) their predictive performance (on unseen cases). In our experiments, we set the significance level for the *F-test* pruning to 0.01 and the maximal tree size to 6 leaves (i.e., clusters).

## 5. Results

In this section, we present the results obtained by applying predictive clustering trees for multi-dimensional time series modelling to data on forest growing stock in Slovenia collected over a period of 40 years. We present the results from the two modelling scenarios outlined in the previous section. We first discuss the results of the quantitative scenario and give the obtained predictive clustering tree (Figure 4) and a map of Slovenia illustrating the distribution of the locations that share similar forest growing stock dynamics (Figure 5). Next, in a similar way, we present the results for the qualitative scenario (Figure 6 gives the predictive clustering tree and Figure 7 gives the distribution of locations with similar dynamics).

The structure of the model describing the quantitative aspect of temporal multi-dimensional dynamics of growing stock shows that the most important variable (in state owned forests) related to the dynamics is the phyto-geographic region ($PHYTOREG$), followed by the site productivity index ($PI$), altitude ($ELV$), inclination ($INC$) and bedrock ($BEDR$) (Figure 4). While all forest compartments (all clusters) show permanent increase of the total growing stock (e.g., due to the impact of forest policies, the increasing importance of nature conservation, protection of large trees ...), the dynamics of DBH classes A, B and C of forest stands is still different. In most cases, DBH class C shows the largest accumulation of growing stock, which has the lowest increase at the most extreme growing productive site due to slower forest growth. However, some groups (e.g., group four) show relatively even increase of all three DBH classes, while in other groups (e.g., group six and partly one, two and three) the changes are uneven.

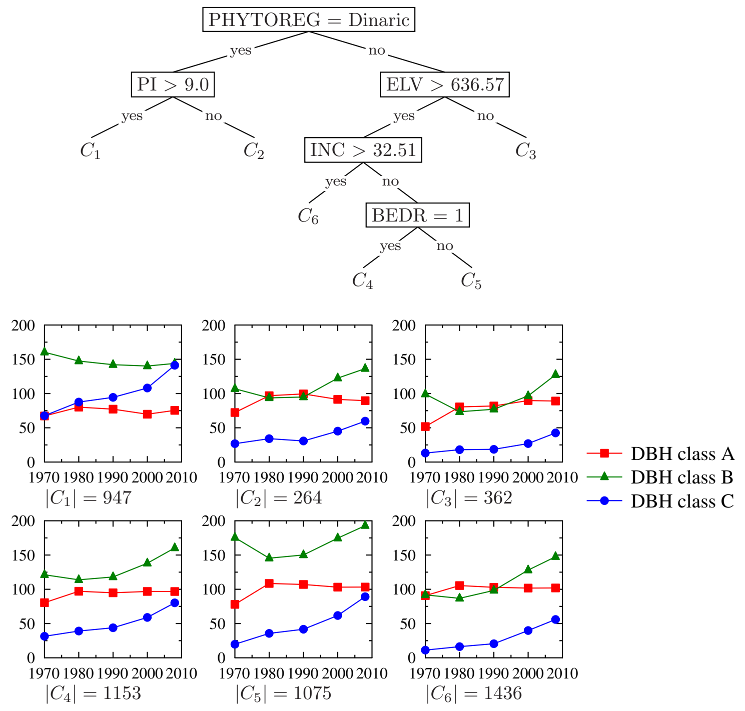


Figure 4: The predictive clustering tree (top) for modelling multi-dimensional time series of growing stock DBH classes A, B and C constructed by using the Euclidean distance measure between time series. In the graphs representing the centroids/prototypes of the clusters (bottom), the x-axis denotes the time in years and the y-axis the forest growing stock of DBH classes A, B, and C, in $m^3$ per *ha*.

A more detailed examination of the model shows that forests from the Dinaric region located at high productivity sites ($C_1$) have the largest change of growing stock in DBH class C. For these forests, the growing stock in DBH classes B declines slowly over the entire period, while the growing stock in Class A remains more or less the at the same level. The different types of change of growing stock in DBH classes B and C show that the growing stock moves from DBH class B to DBH class C, which means that growing stock in these forests continuously increases, while the stability of the growing stock in DBH class A does not indicate aging of these forests.

Such a pattern of change is present also in all other groups, but with a lower intensity of

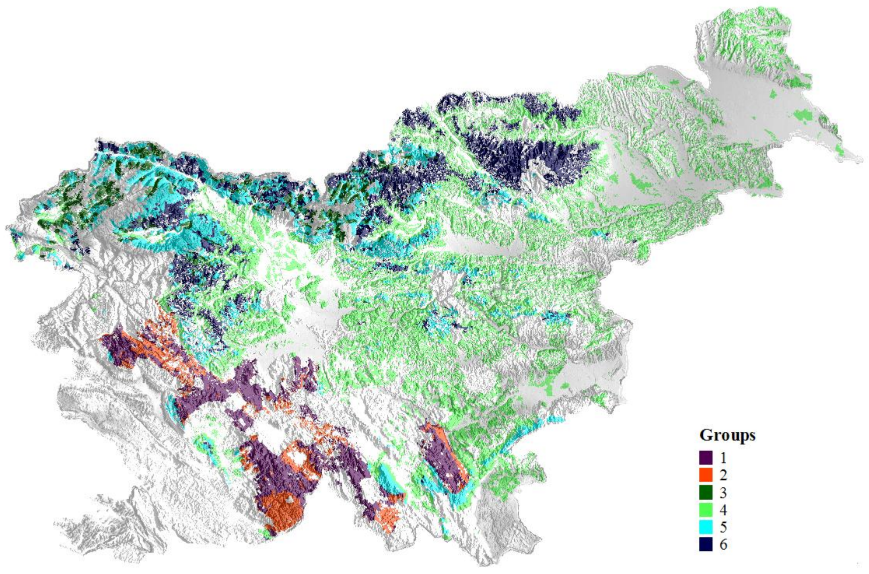


Figure 5: A map of Slovenia that illustrates the distribution of the forest compartments belonging to the different clusters identified with the predictive clustering tree from Figure 4.

accumulation of growing stock in DBH class C. However, growing stock in DBH class C is continually increasing, which indicates the accumulation of growing stock in larger trees. In contrast to group one ($C_1$), growing stock in DBH class B in increasing from year 1980, while DBH class A stays stable. If we compare the absolute values of growing stock for DBH classes A and B between the six groups, we can notice that forests with lower growing stock are located either in the areas with non-favourable conditions for forest growth (e.g., low site productivity index, high inclination, less suitable carbonate bedrock) or at lower altitudes, which were in the past more exposed to human exploitation, due to their vicinity to more densely populated regions.

The temporal pattern of structural changes for growing stock in DBH classes A, B, and C is similar for all six groups: We can thus conclude that the studied state-owned forests are increasing their growing stock mostly in DBH class B, and also in DBH class C, while growing stock in DBH class A stays at the same level. This indicates structural straightening of the studied forests because their growing stock (biomass) is increasing and its structure indicates that they are well developed, but they haven't yet reached the optimal level of growing stock. The only exception are forests from group one, which are probably already in the optimal development stage. It is expected that under the same management measures (e.g., planning and logging), the structure of the growing stock in the other groups will became similar to the structure in the first group.

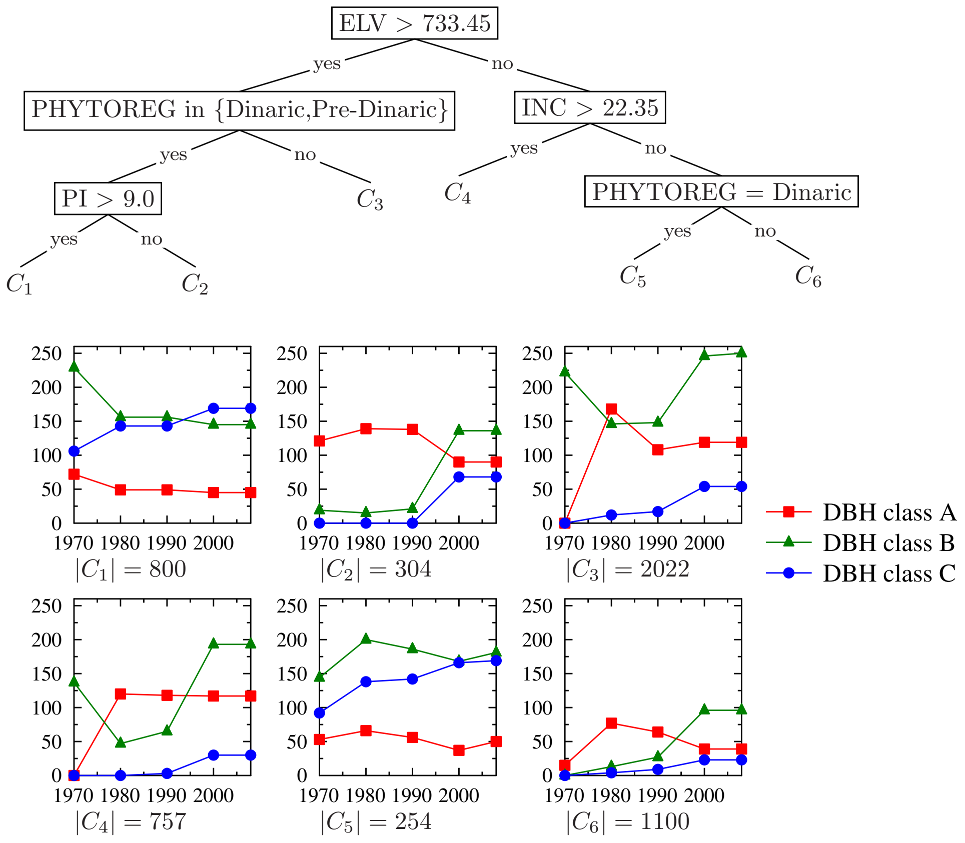


Figure 6: The predictive clustering tree (top) for modelling multi-dimensional time series of growing stock DBH classes A, B, and C, constructed by using the qualitative distance measure between time series. In the graphs representing the centroids/prototypes of the clusters (bottom), the x-axis denotes the time in years and the y-axis the forest growing stock in of DBH classes A, B and C in $m^3$ per *ha*.

The model constructed in the second scenario, where we use the qualitative distance between the growing stock DBH classes A, B and C time series is given in Figure 6. It shows that the most important variable distinguishing between different patterns of growing stock dynamics is the elevation (ELV), followed by phyto-geographic region (PHYTOREG), inclination (INC), site productivity index (PI) and phyto-geographic region (PHYTOREG) again.

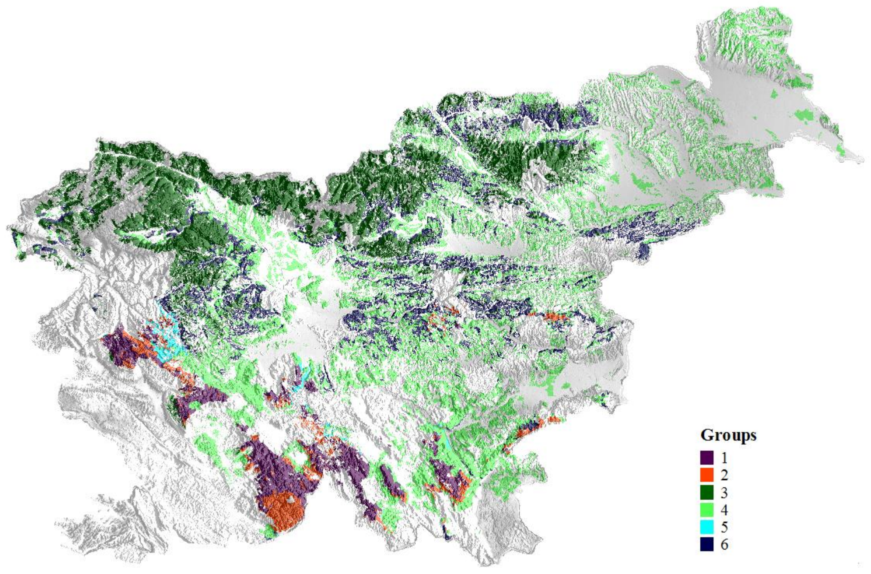


Figure 7: A map of Slovenia that illustrates the distribution of the forest compartments belonging to the different clusters identified with the predictive clustering tree from Figure 6.

The model shows significant structural changes of growing stock in some forests (e.g., groups two, four and six) in the last 40 years. In general, the model confirms the expected interchange of growing stock between DBH classes, where the increase of growing stock in classes B and/or C is matched by a decrease of growing stock in class A. However, a detailed inspection of the model reveals positive temporal dynamics of growing stock in classes B and C, but no significant change of the growing stock of class A after the year 1990. The changes in growing stock in classes B and C after 1990 can be attributed to several factors, such as changes in the Slovenian forest policy due to the adoption of the new Law on Forests, lower logging intensity because of demographic and economic reasons (e.g., decreasing importance of the economic functions of forests and increasing importance of nature conservation).

The continuous change of forest stock also indicates that most of the forests are not yet in a steady state balance with the growing conditions despite the very high and well structured growing stock. This can be seen from the positive trends of growing stock in classes B and C, while the growing stock of class A does not change significantly. Besides organizational changes in 1990, the National Forest Program was adopted in 1996 which set the allowable cut limit to a maximum of 60 % of the total increment. This is reflected in the increasing trend of growing stock in DBH classes B and C and has resulted in an increase of the total growing stock in the period 1990-2000.

However, the growth of growing stock in DBH classes B and C has stopped in the last ten-year period from 2000 to 2010. The reason is probably in the change of economic perception of trees with large DBH (and tree volume), because their quality is decreasing with size, mostly due to senescence effects (e.g, rotting inside the trunk). Because of timber market requirements for medium sized timber, the harvest of these trees was higher than in the periods before the year 2000.

Growing stock dynamics for all DBH classes shows large variability of growing stock, except for the first group, where growing stock in all DBH classes does not change much, but rather stays moderately stable. This finding concurs with the observation for group one in the model constructed by using the Euclidean distance (Figure 4). In both cases, the forests are from the Dinaric region and are located at high productivity sites, showing development stability for almost the entire study period. It seems that these forests have achieved steady state balance between natural growing conditions and the applied forest management practices. It is expected that other forests will follow this scenario and the forests with more suitable growing conditions will achieve their development stability faster.

## 6. Discussion and conclusions

This paper describes an extension of predictive clustering trees for the analysis of multi-dimensional time series and its successful application to a complex forestry data set. The analysis of multi-dimensional time series has been thus far treated by using standard clustering algorithms to detect the potential groups/clusters of examples. The obtained clusters have been then typically described by post facto search for cluster descriptions. The proposed algorithm provides both the clusters of multi-dimensional time series and their symbolic descriptions simultaneously, and to the best of our knowledge, it is the first algorithm of this kind.

We used the proposed algorithm to model Slovenian forest inventory data consisting of phytogeographical properties and forest growing stock of forest compartments. The inventory data spans over four decades, from 1970 until 2010. The task concerns the modelling of the dynamics of the forest growing stock distributed into DBH classes. The forest growing stock is the basic attribute for describing the forest ecosystem response to natural and anthropogenic impacts and allows us to follow the structural, functional and compositional changes of forest ecosystems.

We use two scenarios to analyze the data at hand: quantitative and qualitative. In the former scenario, we use the Euclidean distance as a distance between time series, while in the latter scenario, we use a qualitative distance. Analyzing the quantitative and the qualitative aspect of growing stock enables better understanding of structural dynamic changes of forests as compared to using traditional qualitative studies of growing stock presented either as an aggregated indicator or its distribution into age or DBH classes. The complementarity of the two distance measures that we used within predictive clustering trees for modelling multi-dimensional time series allows two orthogonal interpretations of forest growing stock data. If growing stock is given with its distribution into DBH classes (A: $10 - 29cm$; B: $30 - 49cm$; and C: $50cm$ and more), then the proposed methodology can be used for describing growing stock dynamics in even and uneven-aged forests on a large spatial scale (e.g., regional or national level).

Most often, the changes of growing stock given with DBH classes are considered at stand level and smaller changes are expected at landscape/regional level. However, change of growing stock of forest stands can occur at landscape/regional level too, and our study has identified quite large changes. Since most of the analyzed data are from nature based/uneven-aged forest stands, we expected that the growing stock would remain relatively stable but our results have revealed that this was not the case.

Based on the results, we can make two important conclusions concerning the changes of growing stock structure and its temporal dynamics. First, the quantitative model shows that growing stock in Slovenian forests has continuously increased in the last 40 years. At the same time, the growing stock DBH classes show intensive dynamics, which indicates that despite the large total growing stock, Slovenian forests have not yet reached their steady state of structural development. This is confirmed also with the second conclusion based on the qualitative model, which shows steady and progressive temporal dynamics of growing stock in classes B and C, while the dynamics of class A does not yet show any degressive dynamics. Thus, Slovenian forests are increasing their growing stock while their structure is balanced which, gives them large structural stability.

**Acknowledgement**

We would like to acknowledge the support of the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944).

## References

[1] C. McElhinny, P. Gibbons, C. Brack, J. Bauhus, Forest and woodland stand structural complexity: Its definition and measurement, Forest Ecology and Management 218 (1–3) (2005) 1–24.

[2] M. McCrink-Goode, Pollution: A global threat, Environment International 68 (2014) 162–170.

[3] SOEF, State of Europe's forests – Status and trends in sustainable forest management in Europe, Tech. rep., Forest Europe, UNECE and FAO, Oslo, Norway (2011).

[4] A. Poljanec, Changes in forest stand structure in Slovenia in period 1970–2005, Ph.D. thesis, Biotechnical Faculty, University of Ljubljana, Ljubljana, Slovenia (2008).

[5] M. Klopčič, A. Bončina, Stand dynamics of silver fir (Abies alba Mill.) – European beech (Fagus sylvatica L.) forests during the past century: a decline of silver fir?, Forestry 84 (3) (2011) 259–271.

[6] E. Tomppo, T. Gschwantner, M. Lawrence, R. E. McRoberts, National Forest Inventories: Pathways for Common Reporting, 1st Edition, Springer, 2010.

[7] P. Brassel, U.-B. Brändli, Schweizerische Landesforstinventar. Ergebnisse der Zweitaufnahme 1993-1995, Bern, Verlag Paul Haupt Birmensdorf (1999).

[8] C. Küchli, M. Bolliger, W. Rüsch, La forêt Suisse – un bilan. in: Une analyse politique du deuxième inventaire forestier national (1999).

[9] P. Risser, Oregon state of the environmental report 2000–statewide summary, Oregon Economic and Community Development Compartment, Oregon, US (2000).

[10] P. Linder, L. Östlund, Structural changes in three mid-boreal Swedish forest landscapes, 1885–1996, Biological Conservation 85 (1–2) (1998) 9–19.

[11] A.-L. Axelsson, L. Östlund, E. Hellberg, Changes in mixed deciduous forests of boreal Sweden 1866–1999 based on interpretation of historical records, Landscape Ecology 17 (5) (2002) 403–418.

[12] A. Ficko, A. Poljanec, A. Bončina, Do changes in spatial distribution, structure and abundance of silver fir (Abies alba mill.) indicate its decline?, Forest Ecology and Management 261 (4) (2011) 844–854.

[13] J. Heap, F. Hirsch, D. Ellul, The European Forest Sector Outlook Study II 2010–2030, United Nations Economic Commission for Europe & Food and Agriculture Organization of the United Nations, United Nations, Geneva (2011).

[14] T. J. Postma, F. Liebl, How to improve scenario analysis as a strategic management tool?, Technological Forecasting and Social Change 72 (2) (2005) 161–173.

[15] E. Arets, T. Palosuo, A. Moiseyev, G. Nabuurs, D. Slimani, C. Olsmat, J. Laurijssen, B. Mason, D. McGowan, D. Vötter, Reference futures and scenarios for the European FWC database, EFI Technical Report 85, European Forest Institute, Joensuu (2011).

[16] U. Mantau, F. Steierer, H. Verkerk, M. Lindner, P. Anttila, A. Asikainen, J. Oldenburger, N. Leek, U. Saal, K. Prins, EU wood – real potential for changes in growth and use of EU forests, final report, Hamburg, Germany (2010).

[17] A. Pussinen, M. Schelhaas, E. Verkaik, E. Heikkinen, J. Liski, T. Karjalainen, R. Päivinen, G. Nabuurs, Manual for the European Forest Information Scenario Model (EFISCEN 2.0), EFI Internal Report 5. European Forest Institute, Joensuu (2001).

[18] A. M. I. Kallio, A. Moiseyev, B. Solberg, Economic impacts of increased forest conservation in Europe: a forest sector model analysis, Environmental Science & Policy 9 (5) (2006) 457–465.

[19] G. Nabuurs, H. Pajuoja, K. Kuusela, R. Päivinen, Forest resource scenario methodologies for Europe (EFI discussion paper, 5), European Forest Institute, Joensuu (1998).

[20] G. Mohren, Large-scale scenario analysis in forest ecology and forest management, Forest Policy and Economics 5 (2) (2003) 103–110.

[21] S. E. Jørgensen, G. Bendoricchio, Fundamentals of Ecological Modelling, Elsevier, 2001.

[22] D. Jäger, W. Rammer, M. Lexer, Picus v2.0 a 3-dimensional physiology-based forest patch model, in: H. Hasenauer, A. Mäkelä (Eds.), International Conference on Modeling Forest Production, 2004, pp. 176–183.

[23] L. Rasche, L. Fahse, A. Zingg, H. Bugmann, Enhancing gap model accuracy by modeling dynamic height growth and dynamic maximum tree height, Ecological Modelling 232 (0) (2012) 133–143.

[24] H. Hasenauer, Ein Einzelbaumwachstumssimulator für ungleichaltrige Fichten-Kiefern- und Buchen-Fichtenmischbestände, Österreichische Gesellschaft für Waldökosystemforschung und Experimentelle Baumforschung, Vienna (1994).

[25] S. Dufour-Kowalski, B. Courbaud, P. Dreyfus, C. Meredieu, F. Coligny, Capsis: an open software framework and community for forest growth modelling, Annals of Forest Science 69 (2) (2012) 221–233.

[26] R. Seidl, W. Rammer, R. M. Scheller, T. A. Spies, An individual-based process model to simulate landscape-scale forest ecosystem dynamics, Ecological Modelling 231 (0) (2012) 87–100.

[27] S. Schumacher, H. Bugmann, D. J. Mladenoff, Improving the formulation of tree growth and succession in a spatially explicit landscape model, Ecological Modelling 180 (1) (2004) 175–194.

[28] R. M. Scheller, J. B. Domingo, B. R. Sturtevant, J. S. Williams, A. Rudy, E. J. Gustafson, D. J. Mladenoff, Design, development, and application of LANDIS-II, a spatial landscape simulation model with flexible temporal and spatial resolution, Ecological Modelling 201 (3–4) (2007) 409 – 419.

[29] R. Scheller, D. Mladenoff, An ecological classification of forest landscape simulation models: tools and strategies for understanding broad-scale forested ecosystems, Landscape Ecology 22 (4) (2007) 491–505.

[30] M. Debeljak, A. Poljanec, B. Ženko, Modelling forest growing stock from inventory data: A data mining approach, Ecological Indicators 41 (2014) 30–39.

[31] T. Vilén, K. Gunia, P. Verkerk, R. Seidl, M.-J. Schelhaas, M. Lindner, V. Bellassen, Reconstructed forest age structure in Europe 1950–2010, Forest Ecology and Management 286 (2012) 203–218.

[32] Rules on the forest management and silviculture plans, Official Gazette of the Republic of Slovenia, no. 5/1998 of 23-1-1998 (1998).

[33] T. W. Liao, Clustering of time series data–a survey, Pattern Recognition 38 (11) (2005) 1857–1874.

[34] R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics), 2nd Edition, Springer, 2006.

[35] J.-F. Mari, F. L. Ber, Temporal and spatial data mining with second-order hidden markov models, Soft Comput. 10 (5) (2006) 406–414.

[36] C. Potter, V. Genovese, P. Gross, S. Boriah, M. Steinbach, V. Kumar, Revealing land cover change in California with satellite data, EOS, Transactions American Geophysical Union 88 (26) (2007) 269–274.

[37] N. Viovy, Automatic classification of time series (acts): A new clustering method for remote sensing time series, International Journal of Remote Sensing 21 (6-7) (2000) 1537–1560.

[38] X. Zhou, N. Persaud, H. Wang, Scale invariance of daily runoff time series in agricultural watersheds, Hydrology and Earth System Sciences Discussions 2 (4) (2005) 1757–1786.

[39] C. Li, G. Biswas, M. Dale, P. Dale, Building models of ecological dynamics using hmm based temporal data clustering  a preliminary study, in: F. Hoffmann, D. Hand, N. Adams, D. Fisher, G. Guimaraes (Eds.), Advances in Intelligent Data Analysis, Vol. 2189 of Lecture Notes in Computer Science, 2001, pp. 53–62.

[40] H. Blockeel, Top-down induction of first order logical decision trees, Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1998).

[41] M. Debeljak, G. R. Squire, D. Kocev, C. Hawes, M. W. Young, S. Džeroski, Analysis of time series data on agroecosystem vegetation using predictive clustering trees, Ecological Modelling 222 (14) (2011) 2524–2529.

[42] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, 1st Edition, Wiley-Interscience, 1990.

[43] R. Michalski, Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts, Intl. Jrnl. of Policy Analysis and Information Systems 4 (1980) 219–244.

[44] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spokenword recognition., in: IEEE Transaction on Acoustics, Speech, and Signal Processing, Vol. ASSP-26 of LNAI, 1978, pp. 43–49.

[45] L. Todorovski, B. Cestnik, M. Kline, N. Lavrač, S. Džeroski, Qualitative clustering of short time-series: A case study of firms reputation data, in: ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning, 2002, pp. 141–149.

[46] L. Kaufman, P. Rousseeuw (Eds.), Finding groups in data: An introduction to cluster analysis, John Wiley & Sons, 1990.

[47] H. Blockeel, L. D. Raedt, J. Ramon, Top-down induction of clustering trees, in: Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 55–63.

[48] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann series in Machine Learning, Morgan Kaufmann, 1993.

[49] R. Michalski, R. Stepp, Learning from observation: conceptual clustering, in: Machine Learning: an Artificial Intelligence Approach, Vol. 1, Tioga Publishing Company, 1983.

[50] S. Džeroski, V. Gjorgjioski, I. Slavkov, J. Struyf, Analysis of time series data with predictive clustering trees., in: S. Džeroski, J. Struyf (Eds.), KDID, Vol. 4747 of Lecture Notes in Computer Science, Springer, 2006, pp. 63–80.

[51] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Tree ensembles for predicting structured outputs, Pattern Recognition 46 (3) (2013) 817–833.

[52] M. Vlachos, J. Lin, E. Keogh, D. Gunopulos, A wavelet-based anytime algorithm for k-means clustering of time series, in: In Proc. Workshop on Clustering High Dimensionality Data and Its Applications, 2003, pp. 23–30.

[53] A. Singhal, D. E. Seborg, Clustering multivariate time-series data, Journal of Chemometrics 19 (8) (2005) 427–438.

[54] Q. Wang, V. Megalooikonomou, C. Faloutsos, Time series analysis with multiple resolutions, Information Systems 35 (1) (2010) 56–74.

[55] S. Džeroski, Towards a general framework for data mining., in: S. Džeroski, J. Struyf (Eds.), KDID, Vol. 4747 of Lecture Notes in Computer Science, Springer, 2006, pp. 259–300.

[56] SFS, Slovenian Forest Service – National forest inventory data, http://www.gozdis.si (2009).

[57] Ž. Košir, Comparison of relative fertility of forest sites determined by site coefficient with calculated or estimated site productivity, Gozdarski Vestnik (in Slovene) 60 (1) (2002) 3–23.

[58] J. Struyf, S. Džeroski, Constraint based induction of multi-objective regression trees, in: Proc. of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933, Springer, 2006, pp. 222–233.

# Chapter 10

# Conclusions and Further Work

## 10.1 Scientific contributions of the thesis

This thesis is concerned with the use of distances for learning from structured data. Its main hypothesis is that it is possible to develop distance-based learning methods for structured data of arbitrarily structured datatypes by existing or developing new methods from the areas of instance-based learning and predictive clustering. The thesis explores in detail these two paradigms and proposes several approaches for predictive modeling and clustering within each of the two paradigms.

In this way, it confirms the main hypothesis and its more specific statements given in the introduction. In particular, the developed methods for distance-based learning can handle:

- arbitrary structures on the input side as well as the output side of instance-based learning (Chapter 3, Section 3.2),
- clustering methods for arbitrarily structured types of data in the context of distance-based learning (Chapter 3, Section 3.3), and
- methods for handling arbitrarily structured target datatypes in the context of predictive clustering (Chapter 5).

The developed approaches have been implemented in appropriate software environments. The implementations are described in Chapter 3 (instance-based learning), as well as Chapters 5, 7, 8 and 9 (predictive clustering). Their use has been illustrated, respectively demonstrated, by applying them to datasets coming from practically relevant problems (Chapters 6, 8, 9).

The scientific contributions of the thesis can be summarized as follows:

- *A generic framework and a software environment for instance-based learning from structured data.* The framework allows for the implementation of generic algorithms for instance-based prediction and clustering, with arbitrarily structured datatypes on the input and the output side. Besides the arbitrary datatypes, the framework supports the use of different distances on these structured datatypes, which can be composed from distances on simpler datatypes by using aggregation functions appropriate for the type constructor applied to the simple datatypes.

  Within the framework, the nearest neighbor and $k$-nearest neighbor ($k$-NN) prediction methods have been implemented. Two clustering algorithms have been implemented as well, namely the $k$-means algorithm and its generalization, the $k$-medoids algorithm. The generalized $k$-NN and $k$-medoids algorithm work for arbitrary types of structured data.

- *Predictive clustering trees for arbitrary types of structured targets.* The basic implementation of predictive clustering trees (as well as rules) considers the simultaneous prediction of multiple outputs of primitive data types. The prototypical example is the task of multi-target regression, where the target is a tuple of a real valued variables. We have implemented an extension of the predictive clustering framework which generalizes the basic predictive clustering approach in several directions.

  First, besides using Euclidean distances on tuples of real variables, we can use other distances on such tuples (e.g., Manhattan distance). Furthermore, we can use other type constructors applied to primitive datatypes within the structured output. For example, we can consider sets of discrete/nominal values or sequences/time series of real values.

  More importantly, we have developed approaches for predictive clustering where the target can be of a multi-layered datatype. The multi-layer datatypes can include sets of tuples, tuples of time series, or alternatively, time series of tuples, tuples of hierarchies or sequences of hierarchies. These have been implemented within the software platform of the CLUS system.

- *Applications of predictive clustering trees to practically relevant problems.* The predictive clustering tree approaches for different types of structured outputs were evaluated and their utility demonstrated on a number of practically relevant problems. For the basic case of tuples of real values, the application consisted of predicting the state of the forests in Slovenia (and in particular forest stand height and canopy cover) from remotely sensed data, i.e., satellite images. For the case of predicting time series, the application considered was finding explained groups of genes with similar time course profiles of gene expression under different stressful conditions. Finally, for the case of tuples of time series, the clustering of profiles of forest growth stock in Slovenian forests was considered.

## 10.2 Further work

The approaches presented in the dissertation are very general in the sense that many different types of data can be analyzed with them. Therefore there are many directions for further work that can be followed. Bellow we present a selection of directions for further work, addressing first the directions for further development of instance-based learning, then predictive clustering and finally directions that are relevant for both instance-based learning and predictive clustering.

Concerning distance-based learning, the proposed paradigm and implemented software environment have not been evaluated extensively. Therefore more applications in different domains should be explored, focusing on such domains where structured data appears both on the input and the output side. At present, many different approaches exist for analyzing data which is structured on the input side, with scalar output, as well as many different approaches that allow for structured output prediction with a tuple of primitives as input. An example of an application where we have structured data both on the input and the output side is gene function prediction, where on the input side we could have a tuple of time series of gene expression profiles under different experimental conditions whereas on the output side we have tuples of hierarchies, namely the three hierarchies of the gene ontology.

Within distance-based learning, it would be interesting to explore the use of different distance measures for same structured datatype. These can be obtained by different choices of aggregation functions and distance measure for the simpler datatypes at each intermediate type constructor within the datatype. This can be done in the context of different learning tasks, for example, predicting gene function from a tuple of time series, where different distances on the time series can be chosen for the time series resulting from different stressful conditions. For example, for one time series the dynamic time warping distance measure can be chosen, while for another the qualitative distance measure can be chosen.

The use of different distance measures can also be explored in the context of different learning algorithms including algorithms for both clustering and prediction. For example in the context of hierarchical agglomerative clustering, the distances used on sets have been based on aggregating distances between pairs of elements of the two sets by minimum, maximum and average aggregation functions. However, it is possible to use distances such as the matching distance within this context and derive new hierarchical agglomerative clustering algorithms based on this. It would also be interesting to implement other distance-based algorithms within the framework for distance-based learning. This could include for example online streaming versions of algorithms for nearest neighbor classification. This has been so far considered for classification and regression, but not for structured output prediction.

Finally, a new type of metric learning can be considered within our framework. An appropriate or most appropriate distance can be selected from a space of possible distances on the input space. The selection can be made based on the performance of say the nearest neighbor algorithm for predicting the values of the target, which may be structured as well.

Many directions are also possible for further development of predictive clustering for different types of structured output data. At present, we have focused on two level structures, namely on tuples with one layer types as arguments of the tuple type constructor. It would be interesting to explore different applications with the same datatypes considered so far. For example, the prediction of tuples of time series can be used to relate gene functions to gene expression profiles under multiple stressful conditions.

In terms of algorithm development, we can consider the handling of even more complex multi-layer datatypes that could go beyond two layers. While the implementation of handling multi-layer datatypes in predictive clustering within the CLUS software environment is general and can in principle handle arbitrary datatypes, we have not in fact explored specific applications with datatypes with more than two layers. As the complexity of the outputs increases, it is more and more difficult to get completely annotated examples. In this context handling the missing values of the target, for example missing one argument in a tuple, would be an interesting direction for further development. Another interesting direction for further development is structured output prediction with increasingly complex datatypes in a streaming context. Here the major challenge would be the representation of summaries of sets of structured values and their efficient calculation which is essentially the same as the calculation of prototypes.

The final direction for further work is relevant for both instance-based learning and predictive clustering. At present, different types of structured data as well as distances on structured datatype are not represented declaratively as input to the algorithms for predictive clustering or instance-based learning but rather coded in a modular way and this code is used by the two frameworks of instance-based learning and predictive clustering. It would be desirable that different types of data, on the one hand, and the different distances on the other hand are specified declaratively as an input to the two frameworks.

In this context it would be of interest to have catalogs of datatypes and distances that can be browsed and from which appropriate datatypes and distances on these can be selected for use in instance-based learning or predictive clustering. A possible source for information on datatypes and distances could be the ontologies on datatypes, OntoDT and on data mining OntoDM (Panov, 2012). These ontologies need to be further populated with additional instances of datatypes and distances, but they do provide the necessary general framework for describing, recording and using different datatypes and distances.

# References

Ahuja, R. K., Goldberg, A. V., Orlin, J. B., & Tarjan, R. E. (1992). Finding minimum-cost flows by double scaling. *Mathematical Programming*, *53*, 243–266.

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175–185.

Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, *201*, 81–105.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., ... Sherlock, G. (2000). Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, *25*, 25–29.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*(6), 503–515.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees* (Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium).

Blockeel, H., Raedt, L. D., & Ramong, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 55–63). Morgan Kaufmann.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, *13*(1), 21–27.

Debnath, A., Lopez de Compadre, R., Debnath, G., Shusterman, A., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, *34*(2), 786–797.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

DeRisi, J. L., Iyer, V. R., & Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, *278*(5338), 680–686.

Dietterich, T. G., Domingos, P., Getoor, L., Muggleton, S., & Tadepalli, P. (2008). Structured machine learning: the next ten years. *Machine Learning*, *73*(1), 3–23.

Dietterich, T. G., Jain, A. N., Lathrop, R. H., & Lozano-Pérez, T. (1993). A Comparison of Dynamic Reposing and Tangent Distance for Drug Activity Prediction. In *Advances in Neural Information Processing Systems 6* (p. 216).

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*(1-2), 31–71.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, *3*(3), 32–57.

Džeroski, S. (2006). Towards a general framework for data mining. In *Proceedings of the Fifth International Workshop on Knowledge Discovery in Inductive Databases* (Vol. 4747, pp. 259–300). LNCS. Springer.

Džeroski, S., Gjorgjioski, V., Slavkov, I., & Struyf, J. (2006). Analysis of time series data with predictive clustering trees. In *Proceedings of the Fifth International Workshop on Knowledge Discovery in Inductive Databases* (Vol. 4747, pp. 63–80). LNCS. Springer, Berlin.

Džeroski, S., Kobler, A., Gjorgjioski, V., & Panov, P. (2006). Using decision trees to predict forest stand height and canopy cover from LANSAT and LIDAR data. In *Proceedings of the 20th International Conference on Informatics for Environmental Protection* (pp. 125–133). Shaker, Aachen.

Džeroski, S. & Lavrač, N. (2001). *Relational data mining.* Springer.

Džeroski, S., Schulze-Kremer, S., Heidtke, K. R., Siems, K., & Wettschereck, D. (1996). Applying ILP to Diterpene Structure Elucidation from 13C NMR Spectra. In *Inductive logic programming workshop* (Vol. 1314, pp. 41–54). LNCS. Springer.

Emde, W. & Wettschereck, D. (1996). Relational instance based learning. In *Proceedings of The 13th International Conference on Machine Learning* (pp. 122–130). Morgan-Kaufman Publishers, San Francisco, CA, USA.

Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., ... Jensen, L. J. (2013). STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Research*, *41*(Database-Issue), 808–815.

Gasch, A. P., Spellman, P. T., Kao, C. M., Carmel-Harel, O., Eisen, M. B., Storz, G., ... Silver, P. A. (2000). Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, *11*, 4241–4257.

Gjorgjioski, V., Kocev, D., Bončina, A., Debeljak, M., & Džeroski, S. (2015). Predictive clustering of multi-dimensional time series for modelling forest growing stock. *Ecological Modelling.* Under review.

Gjorgjioski, V., Kocev, D., & Džeroski, S. (2011). Comparison of distances for multi-label classification with PCTs. In *Proceedings of the 14th International Multi-conference Information Society* (pp. 121–124). Institut Jožef Stefan, Ljubljana.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning.* Springer.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, (37), 547–579.

Kaufman, L. & Rousseeuw, P. (1987). Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, 405–416.

Kaufman, L. & Rousseeuw, P. (1990). *Finding groups in data: an introduction to cluster analysis.* New York: John Wiley and Sons.

Keogh, E. (2006). A decade of progress in indexing and mining large time series databases. In *Proceedings of the 32nd International Conference on Very Large Data Bases* (pp. 1268–1268). VLDB '06. VLDB Endowment.

Kocev, D. (2011). *Ensembles for predicting structured outputs* (Doctoral dissertation, IPS Jožef Stefan, Ljubljana, Slovenia).

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.

Langley, P. (1996). *Elements of machine learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions and Insertions and Reversals. *Soviet Physics Doklady*, *10*(8), 707–710.

Li, G.-G. & Wang, Z.-Z. (2009). Evaluation of similarity measures for gene expression data and their correspondent combined measures. *Interdisciplinary Sciences: Computational Life Sciences*, *1*(1), 72–80.

Liao, T. (2005). Clustering of time series data—a survey. *Pattern Recognition*, *38*(11), 1857–1874.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam & J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297). University of California Press.

Panov, P. (2012). *A Modular Ontology of Data Mining* (Doctoral dissertation, Jožef Stefan Iternational Postgraduate School, Ljubljana, Slovenia).

Partridge, J. D., Sanguinetti, G., Dibden, D. P., Roberts, R. E., Poole, R. K., & Green, J. (2007). Transition of Escherichia coli from aerobic to micro-aerobic conditions involves fast and slow reacting regulatory components. *The Journal of Biological Chemistry*, *282*(15), 11230–11237.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Ralanamahatana, C., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., & Das, G. (2005). Mining time series data. In *Data Mining and Knowledge Discovery Handbook* (pp. 1069–1103). Springer US.

Ramon, J. & Bruynooghe, M. (2001). A polynomial time computable metric between point sets. *Acta Informatica*, *37*(10), 765–780.

Sakoe, H. & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken-word recognition. In *IEEE Transaction on Acoustics, Speech, and Signal Processing* (Vol. ASSP-26, pp. 43–49). LNAI.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., & Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, *11*, 2.

Shaker, A. & Hüllermeier, E. (2012). IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, *3*(4), 235–249.

Slavkov, I., Gjorgjioski, V., Struyf, J., & Džeroski, S. (2010). Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, *6*(4), 729–740. IF=3.825.

Soon, L.-K. & Lee, S. H. (2007). An empirical study of similarity search in stock data. In *Proceedings of the 2nd International Workshop on Integrating Artificial Intelligence and Data Mining* (pp. 31–38). Australian Computer Society, Inc.

Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bulletin del'Académie Polonaise des Sciences*, *1*, 801–804.

Stevens, S. S. (1946). On the theory of scales of measurement. *Science*, *103*, 677–680.

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., & Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, *5*(4), 256–266. IF=1.351.

Struyf, J. & Džeroski, S. (2006). Constraint based induction of multi-objective regression trees. In *Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID'05, Revised, Selected and Inductive Papers* (Vol. 3933, pp. 222–233). LNCS. Springer.

Székely, G. J. & Rizzo, M. L. (2005). Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method. *Journal of Classification*, *22*, 151–183.

Todorovski, L., Cestnik, B., Kline, M., Lavrač, N., & Džeroski, S. (2002). Qualitative clustering of short time-series: a case study of firms reputation data. In *ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning* (pp. 141–149).

Uwents, W. & Blockeel, H. (2008). A comparison between neural network methods for learning aggregate functions. In *International Conference on Discovery Science* (pp. 88–99). LNCS. Springer.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185–214.

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., & Keogh, E. J. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, *26*(2), 275–309.

Wettschereck, D. (1994). *A study of distance-based machine learning algorithms* (Doctoral dissertation, Corvallis, OR, USA).

Woznica, A. (2008). *Distance and kernel based learning over composite representations* (Doctoral dissertation).

Ženko, B. (2007). *Learning predictive clustering rules* (Doctoral dissertation, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia).

# Bibliography

## Publications Related to the Thesis

### Journal articles

Slavkov, I., Gjorgjioski, V., Struyf, J., & Džeroski, S. (2010). Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, *6*(4), 729–740. IF=3.825.

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., & Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, *5*(4), 256–266. IF=1.351.

### Conference papers

Džeroski, S., Kobler, A., Gjorgjioski, V., & Panov, P. (2006). Using decision trees to predict forest stand height and canopy cover from LANSAT and LIDAR data. In *Proceedings of the 20th International Conference on Informatics for Environmental Protection* (pp. 125–133). Shaker, Aachen.

Džeroski, S., Gjorgjioski, V., Slavkov, I., & Struyf, J. (2006). Analysis of time series data with predictive clustering trees. In *Proceedings of the Fifth International Workshop on Knowledge Discovery in Inductive Databases* (Vol. 4747, pp. 63–80). LNCS. Springer, Berlin.

Gjorgjioski, V., Kocev, D., & Džeroski, S. (2011). Comparison of distances for multi-label classification with PCTs. In *Proceedings of the 14th International Multi-conference Information Society* (pp. 121–124). Institut Jožef Stefan, Ljubljana.

### Journal article submitted for publication

Gjorgjioski, V., Kocev, D., Bončina, A., Debeljak, M., & Džeroski, S. (2015). Predictive clustering of multi-dimensional time series for modelling forest growing stock. *Ecological Modelling*. Under review.

# Biography

Valentin Gjorgjioski was born on October 28$^{th}$, 1981 in Prilep, Macedonia. He completed primary and secondary school in Prilep in 2000. During this period, he attended numerous competitions in mathematics, informatics and physics achieving an outstanding success. His most notable achievements are: winning for three times the first place on Macedonian National Mathematics Competitions, winning three silver medals on National Olympiads in Mathematics, winning a bronze medal on the Balkan Olympiad in Informatics, and a gold medal on the National Olympiad in Informatics.

In 2000, he started his studies at the Institute of Informatics, Faculty of Natural Sciences and Mathematics, Ss. Cyril and Methodius University in Skopje. He was enrolled in a BSc program in the area of Computer Science and Software Engineering. He graduated in 2005 with the highest grade point average in the class, and among the top five students in the history of the Institute. He defended his BSc thesis under the supervision of Professor Margita Kon-Popovska. During both the secondary school and later the undergraduate studies, he held a state scholarship awarded to talented students by the Ministry of Education of the Republic of Macedonia.

In the fall of 2005, he started his graduate studies at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, under the supervision of Professor Sašo Džeroski. At the same time, he started working as a researcher at the Jožef Stefan Institute, and was funded by the Slovenian Research Agency under the "young researcher" program. He has collaborated on several EU funded projects, including IQ (Inductive Queries for Mining Patterns and Models) and PHAGOSYS (Systems biology of phagosome formation and maturation modulation by intracellular pathogens).

His research is in the field of data mining and includes the study, development and application of different data mining algorithms. He has applied the developed algorithms to problems in ecological modelling and bioinformatics. He has published his work in several journal papers and has presented it at several international conferences and workshops. Recently, he is spending time on research connected to real-time analytics and cloud-based machine learning services. In addition, he is exploring technologies such as: Storm, Kafka, Hadoop, Node.js, NoSQL, etc.

In May 2011, he left the Jožef Stefan Institute and later in June 2012 joined the SAS Institute in Ljubljana, where he works in the analytics department and develops custom-made solutions for real-life data mining problems for clients operating in retail (grocery), banking and telecommunication industries.

He is active as an entrepreneur since 2009. In April 2009, together with Kiril Ivanoski and Darko Todoroski he co-founded GrabIT, an IT company specialized for web application development, and in January 2014, together with Nikola Keskinov he co-founded E2E, an IT company specialized for mobile application development.